

INDICE

INTRODUZIONE [ACG]	34
Uso della documentazione di AutoCAD [ACG]	34
Uso del manuale [ACG]	34
Convenzioni tipografiche [ACG]	34
Modifiche e miglioramenti della Release 14 [ACG]	35
Parte I Personalizzazione [ACG]	36
Nozioni fondamentali sulla personalizzazione [ACG]	37
Requisiti di base [ACG]	37
L'ambiente AutoCAD [ACG]	37
Percorso di ricerca per le librerie [ACG]	37
Struttura della directory di esempio [ACG]	38
Procedura per la ricerca dei comandi [ACG]	39
File di supporto personalizzabili [ACG]	40
File dei parametri di programma: acad.pgp [ACG]	41
Definizione di comandi esterni [ACG]	41
Alias dei comandi [ACG]	43
Personalizzazione della documentazione in linea [ACG]	44
File della Guida di Windows [ACG]	44
Componenti del sistema di guida di Windows [ACG]	44
Uso dei file della Guida di Windows con AutoCAD [ACG]	45
File HTML [ACG]	47
Uso dei file HTML con AutoCAD [ACG]	47
File di Guida di AutoCAD [ACG]	48
Uso di un file di Guida di AutoCAD in una directory di sola lettura [ACG]	49
Formato del file di Guida di AutoCAD [ACG]	49
Conversione dei file di Guida R12 [ACG]	52

Tipi di linea e modelli di tratteggio [ACG]	53
File di definizione dei tipi di linea [ACG]	53
Tipi di linea semplici [ACG]	54
Tipi di linea complessi [ACG]	55
Forme nei tipi di linea complessi [ACG]	55
Testo nei tipi di linea complessi [ACG]	57
Creazione di modelli di tratteggio [ACG]	58
File dei modelli di AutoCAD: acad.pat [ACG]	58
Come sono costruiti i modelli di tratteggio [ACG]	58
Modelli con linee tratteggiate [ACG]	59
Forme, font e supporto PostScript [ACG]	60
Compilazione dei file di forma e di font [ACG]	61
Compilazione dei font PostScript [ACG]	61
File di definizione delle forme [ACG]	61
Descrizione di forme [ACG]	62
Codice di lunghezza e di direzione dei vettori [ACG]	62
Codici speciali [ACG]	63
Descrizione dei font di testo [ACG]	67
Descrizione dei big font [ACG]	68
Definizione di un file big font [ACG]	68
Definizione di un file big font esteso [ACG]	69
Uso di un big font [ACG]	73
Descrizione dei font Unicode [ACG]	74
Formato dei file [ACG]	74
Tecnica avanzata per la definizione delle forme [ACG]	75
Supporto PostScript [ACG]	76
Esportazione di immagini PostScript [ACG]	76
File di supporto PostScript di AutoCAD: acad.psf [ACG]	76

Elaborazione PostScript avanzata [ACG]	79
Importazione di immagini PostScript [ACG]	80
Interprete PostScript di AutoCAD: acadps [ACG]	81
File della mappa dei font di AutoCAD: fontmap.ps [ACG]	81
Errori nelle immagini PostScript importate [ACG]	82
Riferimenti a PostScript [ACG]	82
Menu personalizzati [ACG]	82
Uso dei file di menu [ACG]	83
Tipi di file di menu [ACG]	83
File DLL delle risorse [ACG]	84
Caricamento dei file di menu [ACG]	84
Menu di base e parziali [ACG]	85
Verifica dei menu parziali con AutoLISP [ACG]	85
Struttura dei file di menu [ACG]	86
Sintassi delle voci di menu [ACG]	87
Contrassegni di nome [ACG]	87
Etichette [ACG]	87
Macro di menu [ACG]	88
Sintassi delle macro di menu [ACG]	88
Terminazione delle macro [ACG]	89
Attesa dell'input utente [ACG]	90
Supporto per lingue straniere nelle macro di menu [ACG]	90
Annullamento di un comando [ACG]	91
Visualizzazioni e messaggi di richiesta [ACG]	91
Caratteri di controllo nelle voci di menu [ACG]	91
Lunghezza delle macro di menu [ACG]	92
Ripetizione delle macro di menu [ACG]	92
Uso della modalità a selezione singola di oggetti [ACG]	93

Scambio tra menu [ACG]	93
Espressioni macro condizionali [ACG]	94
Uso di AutoLISP nelle macro di menu [ACG]	95
Gruppi di menu. [ACG]	96
Menu di pulsanti ed ausiliari [ACG]	96
Creazione di menu di pulsanti ed ausiliari [ACG]	96
Scambio dei menu di pulsanti ed ausiliari [ACG]	97
Uso speciale della barra rovesciata [ACG]	98
Menu a discesa ed a cursore [ACG]	99
Sintassi delle etichette di menu a discesa ed a cursore [ACG]	100
Titoli delle barre dei menu a discesa [ACG]	100
Sottomenu a discesa [ACG]	101
Separazione di etichette di voci di menu [ACG]	101
Controllo della visualizzazione di etichette di voci di menu [ACG]	101
Contrassegno di etichette [ACG]	102
Disattivazione e contrassegno simultanei [ACG]	103
Riferimento di menu a discesa ed a cursore [ACG]	103
Accesso di AutoLISP allo stato delle etichette [ACG]	104
Scambio ed inserimento in menu a discesa [ACG]	105
Scambio tra menu a discesa [ACG]	105
Inserimento e rimozione di me nu a discesa [ACG]	106
Barre degli strumenti [ACG]	106
Creazione di barre degli strumenti [ACG]	106
Definizioni delle barre degli strumenti. [ACG]	107
Definizioni dei pulsanti [ACG]	108
Definizioni delle icone a comparsa [ACG]	108
Definizioni degli elementi di controllo [ACG]	109
Bitmap definite dall'utente [ACG]	109

Riferimento di barre degli strumenti [ACG]	110
Menu a gruppi di immagini [ACG]	110
Voci dei menu a gruppi di immagini [ACG]	110
Etichette di menu a gruppi di immagini [ACG]	110
Macro dei menu a gruppi di immagini [ACG]	111
Visualizzazione dei menu a gruppi di immagini [ACG]	111
Menu a gruppi di immagini di esempio [ACG]	112
Preparazione di diapositive per menu a gruppi di immagini [ACG]	113
Menu di schermo [ACG]	114
Creazione di menu di schermo [ACG]	114
Sottomenu di schermo [ACG]	115
Etichette di voci [ACG]	117
Variabile di sistema MENUCTL [ACG]	117
Menu di tavoletta [ACG]	118
Creazione di menu di tavoletta [ACG]	118
Guida specifica di menu [ACG]	119
Tasti di scelta [ACG]	119
Linguaggio DIESEL e configurazione della riga di stato [ACG]	121
Configurazione della riga di stato -- MODEMACRO [ACG]	121
Variabile MODEMACRO [ACG]	121
Definizioni di MODEMACRO [ACG]	122
Definizioni di MODEMACRO con AutoLISP [ACG]	123
Espressioni DIESEL nei menu [ACG]	124
Espressioni DIESEL in AutoLISP [ACG]	126
Catalogo delle funzioni delle stringhe DIESEL [ACG]	126
+ (addizione) [ACG]	126
- (sottrazione) [ACG]	127
* (moltiplicazione) [ACG]	127

/ (divisione) [ACG]	127
= (uguale a) [ACG]	127
< (minore di) [ACG]	127
> (maggiore di) [ACG]	128
!= (diverso da) [ACG]	128
<= (minore di o uguale a) [ACG]	128
>= (maggiore di o uguale a) [ACG]	128
and [ACG]	128
angtos [ACG]	129
edtime [ACG]	129
eq [ACG]	130
eval [ACG]	130
fix [ACG]	130
getenv [ACG]	130
getvar [ACG]	131
if [ACG]	131
index [ACG]	131
linelen [ACG]	131
nth [ACG]	132
or [ACG]	132
rtos [ACG]	132
strlen [ACG]	132
substr [ACG]	133
upper [ACG]	133
xor [ACG]	133
Messaggi di errore [ACG]	133
Interfacce di programmazione [ACG]	134
Comandi di tipo script [ACG]	134

Come richiamare uno script durante il caricamento di AutoCAD [ACG]	135
Creazione di dimostrazioni con diapositive [ACG]	135
Automazione ActiveX [ACG]	136
Uso di applicazioni di Automazione [ACG]	136
Lancio di un'applicazione dalla riga di comando [ACG]	136
Lancio di un'applicazione da un menu [ACG]	137
AutoLISP [ACG]	138
Uso di applicazioni AutoLISP [ACG]	138
Caricamento ed esecuzione automatici [ACG]	139
Caricamento automatico del file acad.lsp [ACG]	139
Caricamento automatico del file .mnl [ACG]	140
Caricamento automatico dei comandi [ACG]	140
ARX (AutoCAD Runtime Extension) [ACG]	141
Uso di applicazioni ARX [ACG]	141
Caricamento automatico delle applicazioni ARX [ACG]	142
ADSRX [ACG]	142
ADS [ACG]	142
Uso di applicazioni ADS [ACG]	142
Caricamento automatico delle applicazioni ADS [ACG]	143
Parte II AutoLISP [ACG]	143
Parte II AutoLISP [ACG]	144
Nozioni fondamentali sull'uso di AutoLISP [ACG]	144
Espressioni AutoLISP [ACG]	145
Tipi di dati in AutoLISP [ACG]	145
Subroutine e Subroutine esterne [ACG]	146
Numeri interi [ACG]	146
Numeri reali [ACG]	146
Stringhe [ACG]	147

Elenchi [ACG]	147
Gruppi di selezione [ACG]	147
Nomi di entità [ACG]	147
Descrittori di file [ACG]	148
Simboli e variabili [ACG]	148
Sintassi delle funzioni AutoLISP [ACG]	148
File di programma AutoLISP [ACG]	149
Commenti [ACG]	149
Indentazione e allineamento [ACG]	150
Variabili AutoLISP [ACG]	150
Uso delle variabili alla riga di comando [ACG]	151
Variabili predefinite [ACG]	151
Calcoli numerici [ACG]	152
Gestione delle stringhe [ACG]	153
Caratteri di controllo nelle stringhe [ACG]	154
Corrispondenza con caratteri jolly [ACG]	155
Uguaglianza e condizionale [ACG]	156
Gestione degli elenchi [ACG]	156
Elenchi punto [ACG]	157
Coppie puntate [ACG]	158
Gestione di simboli e funzioni [ACG]	159
C:FunzioniXXX [ACG]	160
Aggiunta di comandi [ACG]	160
Ridefinizione dei comandi di AutoCAD [ACG]	161
Simboli locali nelle funzioni [ACG]	162
Funzioni con argomenti [ACG]	162
Gestione degli errori [ACG]	163
Gestione delle applicazioni [ACG]	164

Caricamento di applicazioni AutoLISP [ACG]	164
Funzione S::STARTUP: esecuzione automatica [ACG]	165
Tecniche di caricamento [ACG]	165
Caricamento di applicazioni ADS e ARX [ACG]	166
Funzioni di utilità generali [ACG]	166
Funzioni di richiesta e di comando [ACG]	167
Sottomissione di comandi [ACG]	167
Supporto per altre lingue [ACG]	168
Pausa per input dell'utente [ACG]	168
Trasmissione di punti di selezione a comandi di AutoCAD [ACG]	168
Variabili di sistema e d'ambiente [ACG]	169
Controllo della configurazione [ACG]	169
Controllo della visualizzazione [ACG]	169
Visualizzazione di messaggi sulla riga di comando [ACG]	170
Controllo dei menu [ACG]	170
Controllo delle finestre di disegno e di testo [ACG]	172
Controllo della grafica a basso livello [ACG]	172
Input dell'utente [ACG]	173
Funzioni getxxx [ACG]	173
Controllo delle condizioni delle funzioni di input utente [ACG]	174
Opzioni di input per funzioni di input utente [ACG]	175
Opzioni per parole chiave [ACG]	175
Input arbitrario da tastiera [ACG]	176
Convalida dell'input [ACG]	176
Utilità geometriche [ACG]	177
Snap ad oggetto [ACG]	177
Estensioni di testo [ACG]	178
Conversioni [ACG]	181

Conversioni di stringhe [ACG]	181
Conversione angolare [ACG]	182
Conversione di codici ASCII [ACG]	183
Conversione di unità [ACG]	184
Conversione da pollici in metri [ACG]	184
File di definizione delle unità [ACG]	185
Trasformazioni del sistema di coordinate [ACG]	186
Trasformazioni di punti [ACG]	187
Gestione di file [ACG]	188
Ricerca di file [ACG]	188
Accesso ai file della Guida [ACG]	189
Controllo ed accesso a dispositivi [ACG]	189
Accesso all'input dell'utente [ACG]	190
Graduazione delle tavolette [ACG]	190
Interfaccia ASE AutoLISP [ACG]	191
Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli [ACG]	191
Gestione dei gruppi di selezione [ACG]	192
Elenchi dei filtri dei gruppi di selezione [ACG]	193
Modelli di caratteri jolly negli elenchi dei filtri [ACG]	194
Filtraggio di dati estesi [ACG]	195
Verifiche relazionali [ACG]	195
Raggruppamento logico delle verifiche dei filtri [ACG]	195
Gestione dei gruppi di selezione [ACG]	196
Passaggio di gruppi di selezione tra applicazioni AutoLISP e ADSRX [ACG]	197
Gestione degli oggetti [ACG]	197
Funzioni riguardanti i nomi delle entità [ACG]	198
Gestori di entità e loro uso [ACG]	199
Contesto dell'entità e dati per la trasformazione delle coordinate [ACG]	199

Funzioni per l'accesso all'entità [ACG]	202
Funzioni dei dati dell'entità [ACG]	202
Uso dei blocchi [ACG]	204
Blocchi anonimi [ACG]	204
Creazione di entità complesse [ACG]	205
Funzioni riguardanti i dati dell'entità e schermo di disegno [ACG]	206
Polilinee e polilinee ottimizzate [ACG]	206
Elaborazione di polilinee curve e spline [ACG]	207
Gestione degli oggetti non grafici [ACG]	207
Oggetti di tabelle di simboli [ACG]	207
Oggetti dizionario [ACG]	208
Dati estesi [ACG]	208
Organizzazione dei dati estesi [ACG]	209
Registrazione di un'applicazione [ACG]	211
Recupero dei dati estesi [ACG]	211
Unione di dati estesi ad una entità [ACG]	212
Gestione dell'utilizzo della memoria da parte dei dati estesi [ACG]	213
Gestori nei dati estesi [ACG]	213
Oggetti xrecord [ACG]	213
Accesso alle tabelle di simboli e ai dizionari [ACG]	214
Tabelle di simboli [ACG]	214
Dizionari [ACG]	215
Accesso ai gruppi AutoCAD [ACG]	216
Gestione delle finestre di dialogo [ACG]	216
Apertura e chiusura delle finestre di dialogo [ACG]	216
Gestione di caselle ed attributi [ACG]	218
Espressioni delle azioni e funzioni di richiamo [ACG]	218
Espressioni delle azioni [ACG]	219

Cause di funzioni di richiamo [ACG]	220
Azioni di default e DCL [ACG]	220
Gestione delle caselle [ACG]	221
Inizializzazione di modalità e valori [ACG]	221
Modifica di modalità e valori al momento dell'esecuzione di una funzione di richiamo [ACG]	222
Gestione di cluster di scelta [ACG]	222
Gestione dei dispositivi di scorrimento [ACG]	223
Gestione delle caselle di modifica [ACG]	224
Finestre di dialogo nidificate [ACG]	224
Come nascondere le finestre di dialogo [ACG]	224
Richiesta della parola d'ordine [ACG]	226
Caselle di riepilogo ed elenchi a comparsa [ACG]	227
Pulsanti e gruppi di immagini [ACG]	228
Creazione di immagini [ACG]	229
Gestione dei pulsanti immagine [ACG]	230
Dati specifici dell'applicazione [ACG]	230
Riepilogo delle funzioni delle finestre di dialogo [ACG]	231
Sequenza di funzioni [ACG]	231
Finestra di dialogo di definizione dei blocchi di esempio [ACG]	232
Corso interattivo di AutoLISP [ACG]	232
Obiettivo [ACG]	232
Introduzione [ACG]	233
Come ottenere input [ACG]	234
Orientamento [ACG]	236
Come disegnare le piastrelle [ACG]	236
Come aggiungere il comando ad AutoCAD [ACG]	238
Come aggiungere un'interfaccia a finestra di dialogo [ACG]	239
Il file DCL ddgp.dcl [ACG]	239

Funzioni di finestra di dialogo nel file AutoLISP ddgp.lsp [ACG]	240
Sommario delle funzioni di AutoLISP [ACG]	243
Funzioni di base [ACG]	244
Funzioni aritmetiche [ACG]	244
Funzioni di gestione delle stringhe [ACG]	245
Funzioni di uguaglianza e funzioni condizionali [ACG]	246
Funzioni di manipolazione delle liste [ACG]	247
Funzioni di gestione dei simboli [ACG]	248
Funzioni di gestione delle funzioni [ACG]	248
Funzioni di gestione degli errori [ACG]	249
Funzioni di gestione dell'applicazione [ACG]	249
Funzioni di utilità [ACG]	250
Funzioni di richiesta e di comando [ACG]	250
Funzioni di controllo della visualizzazione [ACG]	251
Funzioni di input utente [ACG]	252
Funzioni geometriche [ACG]	253
Funzioni di conversione [ACG]	253
Funzioni di gestione dei file [ACG]	254
Funzioni di accesso ai dispositivi [ACG]	254
Funzioni di gruppi di selezione, di oggetti e di tabelle di simboli [ACG]	255
Funzioni per la manipolazione dei gruppi di selezione [ACG]	255
Funzioni di gestione degli oggetti [ACG]	255
Funzioni di gestione dei dati estesi [ACG]	256
Funzioni di gestione delle tabelle di simboli e di dizionari [ACG]	256
Funzioni di programmazione delle finestre di dialogo [ACG]	257
Funzioni di apertura e chiusura delle finestre di dialogo [ACG]	257
Funzioni di gestione di caselle ed attributi [ACG]	258
Funzioni di gestione di caselle di riepilogo ed elenchi a comparsa [ACG]	258

Funzioni di gestione dei gruppi di immagini [ACG]	259
Funzioni di gestione di dati specifici dell'applicazione [ACG]	259
Funzioni di gestione della memoria [ACG]	259
AutoLISP: catalogo delle funzioni [ACG]	260
+ (addizione) [ACG]	267
- (sottrazione) [ACG]	267
* (moltiplicazione) [ACG]	267
/ (divisione) [ACG]	268
= (uguale a) [ACG]	268
/= (diverso da) [ACG]	268
< (minore di) [ACG]	269
<= (minore di o uguale a) [ACG]	269
> (maggiore di) [ACG]	269
>= (maggiore di o uguale a) [ACG]	269
~ (NOT a livello bit) [ACG]	270
1+ (incremento) [ACG]	270
1- (decremento) [ACG]	270
abs [ACG]	270
acad_colordlg [ACG]	271
acad_helpdlg [ACG]	271
acad_strlsort [ACG]	271
action_tile [ACG]	272
add_list [ACG]	272
ads [ACG]	273
alert [ACG]	273
alloc [ACG]	273
and [ACG]	273
angle [ACG]	274

angtof [ACG]	274
angtos [ACG]	275
append [ACG]	275
apply [ACG]	275
arx [ACG]	276
arxload [ACG]	276
arxunload [ACG]	276
ascii [ACG]	277
assoc [ACG]	277
atan [ACG]	277
atof [ACG]	278
atoi [ACG]	278
atom [ACG]	278
atoms-family [ACG]	279
autoarxload [ACG]	279
autoload [ACG]	280
autoxload [ACG]	280
Boole [ACG]	280
boundp [ACG]	281
car e cdr [ACG]	281
chr [ACG]	282
client_data_tile [ACG]	283
close [ACG]	283
command [ACG]	283
cond [ACG]	285
cons [ACG]	285
cos [ACG]	286
cvunit [ACG]	286

defun [ACG]	286
dictadd [ACG]	287
dictnext [ACG]	288
dictremove [ACG]	288
dictrename [ACG]	288
dictsearch [ACG]	289
dimx_tile e dimy_tile [ACG]	289
distance [ACG]	289
distof [ACG]	290
done_dialog [ACG]	290
end_image [ACG]	291
end_list [ACG]	291
entdel [ACG]	291
entget [ACG]	292
entlast [ACG]	293
entmake [ACG]	293
entmakex [ACG]	295
entmod [ACG]	295
entnext [ACG]	296
entsel [ACG]	296
entupd [ACG]	297
eq [ACG]	297
equal [ACG]	298
error [ACG]	298
eval [ACG]	299
exit [ACG]	299
exp [ACG]	300
expand [ACG]	300

expt [ACG]	300
fill_image [ACG]	300
findfile [ACG]	301
fix [ACG]	301
float [ACG]	302
foreach [ACG]	302
gc [ACG]	302
gcd [ACG]	302
get_attr [ACG]	303
get_tile [ACG]	303
getangle [ACG]	303
getcfig [ACG]	304
getcname [ACG]	304
getcorner [ACG]	304
getdist [ACG]	305
getenv [ACG]	305
getfiled [ACG]	306
getint [ACG]	307
getkeyword [ACG]	308
getorient [ACG]	308
getpoint [ACG]	309
getreal [ACG]	309
getstring [ACG]	309
getvar [ACG]	310
graphscr [ACG]	310
grclear [ACG]	310
grdraw [ACG]	311
grread [ACG]	311

grtext [ACG]	313
grvecs [ACG]	313
handent [ACG]	314
help [ACG]	314
if [ACG]	315
initget [ACG]	315
inters [ACG]	317
itoa [ACG]	318
lambda [ACG]	318
last [ACG]	318
length [ACG]	319
list [ACG]	319
listp [ACG]	319
load_dialog [ACG]	320
load [ACG]	320
log [ACG]	321
logand [ACG]	321
logior [ACG]	321
lsh [ACG]	321
mapcar [ACG]	322
max [ACG]	322
mem [ACG]	322
member [ACG]	323
menucmd [ACG]	323
gruppomenu [ACG]	324
min [ACG]	324
minusp [ACG]	324
mode_tile [ACG]	325

namedobjdict [ACG]	325
nentsel [ACG]	325
nentselp [ACG]	327
new_dialog [ACG]	328
not [ACG]	328
nth [ACG]	328
null [ACG]	329
numberp [ACG]	329
open [ACG]	329
or [ACG]	330
osnap [ACG]	330
polar [ACG]	331
prin1 [ACG]	331
princ [ACG]	332
print [ACG]	332
progn [ACG]	332
prompt [ACG]	333
quit [ACG]	333
quote [ACG]	333
read [ACG]	334
read-char [ACG]	334
read-line [ACG]	334
redraw [ACG]	335
regapp [ACG]	335
rem [ACG]	336
repeat [ACG]	336
reverse [ACG]	336
rtos [ACG]	336

set [ACG]	337
set_tile [ACG]	337
setcfg [ACG]	338
setfunhelp [ACG]	338
setq [ACG]	339
setvar [ACG]	339
sin [ACG]	340
setview [ACG]	340
slide_image [ACG]	340
snvalid [ACG]	340
sqrt [ACG]	341
ssadd [ACG]	341
ssdel [ACG]	341
ssget [ACG]	342
Filtri del gruppo di selezione [ACG]	343
Verifiche relazionali [ACG]	343
Raggruppamento logico delle verifiche dei filtri [ACG]	344
ssgetfirst [ACG]	345
sslenght [ACG]	345
ssmemb [ACG]	345
ssname [ACG]	346
ssnamex [ACG]	346
sssetfirst [ACG]	347
startapp [ACG]	347
start_dialog [ACG]	348
start_image [ACG]	348
start_list [ACG]	348
strcase [ACG]	349

strcat [ACG]	349
strlen [ACG]	349
subst [ACG]	350
substr [ACG]	350
tablet [ACG]	350
tblnext [ACG]	351
tblobjname [ACG]	352
tblsearch [ACG]	352
term_dialog [ACG]	353
terpri [ACG]	353
textbox [ACG]	353
textpage [ACG]	354
textscr [ACG]	354
trace [ACG]	354
trans [ACG]	355
type [ACG]	356
unload_dialog [ACG]	357
untrace [ACG]	357
vector_image [ACG]	357
ver [ACG]	358
vports [ACG]	358
wcmatch [ACG]	358
while [ACG]	360
write-char [ACG]	360
write-line [ACG]	360
xdroom [ACG]	361
xdsiz e [ACG]	361
xload [ACG]	362

xunload [ACG]	362
zerop [ACG]	363
Accesso ai comandi definiti esternamente e alle variabili di sistema [ACG]	363
3DSIN [ACG]	364
3DSOUT [ACG]	365
ALLINEA [ACG]	365
ASEADMIN [ACG]	365
ASEEXPORT [ACG]	366
ASELINKS [ACG]	366
ASEROWS [ACG]	366
ASESELECT [ACG]	366
ASESQLED [ACG]	367
SFONDO [ACG]	367
SOLID [ACG]	367
GRADIENT [ACG]	368
IMAGE [ACG]	368
MERGE [ACG]	369
ENVIRONMENT [ACG]	369
CAL [ACG]	369
NEBBIA [ACG]	370
LUCE [ACG]	370
A--Ambient (circostante) [ACG]	371
D--Delete (cancella) [ACG]	371
L--List (elenca) [ACG]	371
M--Modify (modifica) [ACG]	372
ND-New Distant Light (nuova luce distante) [ACG]	373
NP-New Point Light (nuova luce puntiforme) [ACG]	373
NS-New Spotlight (nuovo riflettore) [ACG]	374

R--Rename (rinomina) [ACG]	375
MODPAES [ACG]	375
LIBPAES [ACG]	376
ADD [ACG]	377
DELETE [ACG]	377
MODIFY [ACG]	377
OPEN [ACG]	378
SAVE [ACG]	378
LIST [ACG]	379
NPAES [ACG]	379
LIBMAT [ACG]	380
SPECCHIO3D [ACG]	380
PSTRASCINA [ACG]	380
PSMOTIVO [ACG]	381
PSIN [ACG]	381
RENDER [ACG]	382
Impostazione del render per le opzioni di file [ACG]	382
TGA [ACG]	383
PCX [ACG]	383
BMP [ACG]	384
PS [ACG]	384
TIFF [ACG]	384
RENDERUPDATE [ACG]	385
REPLAY [ACG]	385
MATERIALE [ACG]	386
A--Attach (unisci) [ACG]	386
C--Copy (copia) [ACG]	387
D--Detach (distacca) [ACG]	387

L--List (elenca) [ACG]	387
M--Modify (modifica) [ACG]	388
N--New (nuovo) [ACG]	388
Standard [ACG]	389
Marble [ACG]	390
Granito [ACG]	390
Wood [ACG]	391
Argomenti bitmap [ACG]	392
RUOTA3D [ACG]	393
RPREF [ACG]	394
DEST--Destinazione [ACG]	394
ICON--Icona [ACG]	395
STYPE--Tipo di rendering [ACG]	395
SELECT--Selezione [ACG]	395
TOGGLE--Interruttore [ACG]	396
SALVAIMM [ACG]	396
SCENA [ACG]	397
D--Delete (cancella) [ACG]	397
L--List (elenca) [ACG]	397
M--Modify (modifica) [ACG]	398
N--New (nuova) [ACG]	398
R--Rename (rinomina) [ACG]	399
S--Set (imposta) [ACG]	399
MAPPAGGIO [ACG]	400
A (Assegna) [ACG]	400
D--Detach (distacca) [ACG]	401
SHOWMAT [ACG]	401
SOLPROF [ACG]	401

STATIST [ACG]	402
Gestione della memoria [ACG]	402
Spazio per i nodi [ACG]	402
Recupero dello spazio per i nodi [ACG]	403
Note tecniche [ACG]	403
Spazio per le stringhe [ACG]	404
Memoria per i simboli [ACG]	404
Allocazione manuale [ACG]	404
Sistema a paginazione virtuale delle funzioni [ACG]	405
Codici e messaggi di errore AutoLISP [ACG]	406
Codici di errore [ACG]	406
Messaggi di errore [ACG]	408
Errori del programma dell'utente [ACG]	408
Errori interni [ACG]	412
Parte III Finestre di dialogo programmabili [ACG]	413
Finestre di dialogo programmabili [ACG]	413
Componenti della finestra di dialogo [ACG]	413
Struttura dei file DCL [ACG]	415
File base.dcl e acad.dcl [ACG]	416
Riferimenti ai file DCL [ACG]	416
Sintassi DCL [ACG]	417
Definizioni di caselle [ACG]	417
Riferimenti alle caselle [ACG]	418
Attributi e valori degli attributi [ACG]	418
Commenti [ACG]	419
Gestione degli errori DCL [ACG]	419
Verifica semantica per i file DCL [ACG]	419
Esempio di finestra di dialogo [ACG]	420

Tecniche DCL [ACG]	421
Distribuzione delle caselle in un cluster [ACG]	422
Spazio tra le caselle [ACG]	422
Spazio superfluo a destra o in basso [ACG]	423
Spazio superfluo intorno ad una riga o colonna incasellata [ACG]	423
Personalizzazione del testo dei pulsanti di uscita [ACG]	424
Direttive di progettazione [ACG]	425
Estetica ed ergonomia [ACG]	425
Coerenza di progettazione e chiarezza di linguaggio [ACG]	425
Controllo dell'utente [ACG]	426
Annullamento degli errori [ACG]	427
Funzione di Guida in linea [ACG]	427
Utenti disabili [ACG]	428
Uso delle lettere maiuscole [ACG]	428
Abbreviazioni [ACG]	429
Layout [ACG]	429
Dimensione e posizionamento [ACG]	429
Caselle disabilitate [ACG]	429
Finestre di dialogo nidificate [ACG]	430
Come nascondere le finestre di dialogo [ACG]	430
Valori di default [ACG]	430
Input della tastiera [ACG]	430
Direttive per caselle e cluster predefiniti [ACG]	431
Direttive per caselle e cluster predefiniti [ACG]	431
Pulsanti [ACG]	431
Cluster [ACG]	432
Caselle di modifica [ACG]	432
Pulsanti immagine e gruppi di immagini [ACG]	433

Caselle di riepilogo [ACG]	433
Pulsanti, righe e colonne di scelta [ACG]	433
Dispositivi di scorrimento: [ACG]	434
Testo [ACG]	434
Interruttori [ACG]	434
Gestione degli errori [ACG]	435
Caselle di errore [ACG]	435
Caselle di allarme [ACG]	435
Documentazione correlata [ACG]	436
Linguaggio DCL [ACG]	436
Attributi [ACG]	436
Tipi di attributi [ACG]	437
Attributi limitati [ACG]	437
Indice degli attributi predefiniti [ACG]	437
Attributi predefiniti [ACG]	438
action [ACG]	439
alignment [ACG]	439
allow_accept [ACG]	439
aspect_ratio [ACG]	440
big_increment [ACG]	440
children_alignment [ACG]	440
children_fixed_height [ACG]	440
children_fixed_width [ACG]	441
color [ACG]	441
edit_limit [ACG]	442
edit_width [ACG]	442
fixed_height [ACG]	442
fixed_width [ACG]	442

fixed_width_font [ACG]	443
height [ACG]	443
initial_focus [ACG]	443
is_bold [ACG]	444
is_cancel [ACG]	444
is_default [ACG]	444
is_enabled [ACG]	444
is_tab_stop [ACG]	445
key [ACG]	445
label [ACG]	445
layout [ACG]	446
list [ACG]	446
max_value [ACG]	446
min_value [ACG]	446
mnemonic [ACG]	447
multiple_select [ACG]	447
password_char [ACG]	447
small_increment [ACG]	448
tabs [ACG]	448
tab_truncate [ACG]	448
value [ACG]	448
width [ACG]	449
Attributi definiti dall'utente [ACG]	449
Sinopsi delle funzioni delle caselle DCL [ACG]	450
Caselle attive predefinite [ACG]	450
Cluster di caselle [ACG]	450
Caselle informative e decorative [ACG]	451
Cluster di testo [ACG]	451

Pulsanti di uscita dalla finestra di dialogo e caselle di errore [ACG]	451
Caselle limitate [ACG]	452
Elenco delle caselle DCL [ACG]	452
boxed_column [ACG]	452
boxed_radio_column [ACG]	452
boxed_radio_row [ACG]	453
boxed_row [ACG]	454
button [ACG]	454
column [ACG]	455
concatenation [ACG]	455
dialog [ACG]	455
edit_box [ACG]	456
errtile [ACG]	456
image [ACG]	457
image_button [ACG]	457
list_box [ACG]	458
ok_only [ACG]	458
ok_cancel [ACG]	459
ok_cancel_help [ACG]	459
ok_cancel_help_errtile [ACG]	459
ok_cancel_help_info [ACG]	460
paragraph [ACG]	460
popup_list [ACG]	460
radio_button [ACG]	461
radio_column [ACG]	462
radio_row [ACG]	462
row [ACG]	463
slider [ACG]	463

text [ACG]	464
text_part [ACG]	464
toggle [ACG]	464
spacer [ACG]	465
spacer_0 [ACG]	465
spacer_1 [ACG]	466
Codici ASCII [ACG]	466
Formato dei file di interscambio dei disegni [ACG]	467
Formato di file DXF ASCII [ACG]	468
Struttura generale dei file [ACG]	468
Struttura dei file DXF [ACG]	468
Codici di gruppo [ACG]	469
Esempio di tabella di simboli [ACG]	469
Scrittura di programmi interfaccia DXF [ACG]	471
Formato dei file DXF binari [ACG]	473
File DXB [ACG]	473
Formato di file DXB [ACG]	474
Formato dei file di diapositiva [ACG]	476
Intestazione diapositive per versioni precedenti [ACG]	477
Formato dei file di libreria di diapositiva [ACG]	478
Codici di gruppo DXF [ACG]	478
Convenzioni DXF generali [ACG]	478
Intervalli dei codici di gruppo [ACG]	479
Codici di gruppo in ordine numerico [ACG]	480
Codici degli oggetti e delle entità [ACG]	482
Sezione HEADER [ACG]	482
Sezione CLASSES [ACG]	487
Sezione TABLES [ACG]	488

Tabelle dei simboli nei file DXF [ACG]	488
Codici di gruppo della tabella dei simboli [ACG]	489
Codici di gruppo comuni per le voci della tabella dei simboli [ACG]	489
APPID [ACG]	490
BLOCK_RECORD [ACG]	491
DIMSTYLE [ACG]	491
LAYER [ACG]	492
LTYPE [ACG]	493
STYLE [ACG]	493
UCS [ACG]	494
VIEW [ACG]	494
VPORT [ACG]	495
Sezione BLOCKS [ACG]	496
Blocchi nei file DXF [ACG]	496
BLOCK [ACG]	497
ENDBLK [ACG]	498
Sezione ENTITIES [ACG]	498
Codici di gruppo comuni per gli oggetti grafici [ACG]	499
3DFACE [ACG]	500
3DSOLID [ACG]	500
ARC [ACG]	500
ATTDEF [ACG]	501
ATTRIB [ACG]	502
BODY [ACG]	503
CIRCLE [ACG]	503
DIMENSION [ACG]	503
Codici di gruppo comuni per le quote [ACG]	504
Codici di gruppo per le quote allineate, lineari e ruotate [ACG]	505

Codici di gruppo per le quote lineari e ruotate [ACG]	505
Codici di gruppo per le quote radiali e di diametro [ACG]	506
Codici di gruppo per quote angolari [ACG]	506
Codici di gruppo per quote di coordinate [ACG]	507
Sostituzioni degli stili di quota [ACG]	508
ELLIPSE [ACG]	508
HATCH [ACG]	509
Dati del tracciato di contorno [ACG]	510
Dati del modello [ACG]	511
IMAGE [ACG]	512
INSERT [ACG]	512
LEADER [ACG]	513
LINE [ACG]	514
LWPOLYLINE [ACG]	514
MLINE [ACG]	515
MTEXT [ACG]	516
OLEFRAME [ACG]	517
OLE2FRAME [ACG]	517
POINT [ACG]	518
POLYLINE [ACG]	519
Mesh poliedriche [ACG]	519
RAY [ACG]	520
REGION [ACG]	520
SEQEND [ACG]	520
SHAPE [ACG]	521
SOLID [ACG]	521
SPLINE [ACG]	522
TEXT [ACG]	522

TOLERANCE [ACG]	523
TRACE [ACG]	524
VERTEX [ACG]	524
VIEWPORT [ACG]	525
XLINE [ACG]	526
ACAD_PROXY_ENTITY [ACG]	527
Sezione OBJECTS [ACG]	527
Dizionario degli oggetti denominati [ACG]	528
Codici di gruppo di oggetti nei file DXF [ACG]	528
Codici di gruppo di oggetti comuni [ACG]	528
DICTIONARY [ACG]	529
Dizionario GROUP: ACAD_GROUP [ACG]	529
Dizionario MLINestyle: ACAD_MLInestyle [ACG]	530
XRECORD [ACG]	531
ACAD_PROXY_OBJECT [ACG]	531
Problemi DXF avanzati [ACG]	531
Oggetti di database [ACG]	531
Gestori di riferimento costanti tra oggetti [ACG]	532
Riferimenti di puntatore e proprietà [ACG]	532
Hard e soft [ACG]	533
Gestori liberi [ACG]	533
Codici di gruppo 1005 [ACG]	533
Contrassegni di sottoclasse [ACG]	533
Dizionario di estensione e reattori costanti [ACG]	534
Dati estesi [ACG]	535
OCS (Sistema di Coordinate Oggetto) [ACG]	537
Algoritmo dell'asse arbitrario [ACG]	538

Introduzione

Panoramica

AutoCAD è un sistema per la creazione di disegni progettato con un'architettura aperta in modo tale da consentire di personalizzare ed estendere le sue funzioni. È quindi possibile espandere ed adattare AutoCAD in base alle proprie necessità. Nel *Manuale di personalizzazione di AutoCAD* sono riportati i dettagli e le procedure per implementare queste ed altre caratteristiche di personalizzazione.

Il rivenditore può offrire applicazioni ed estensioni sviluppate indipendentemente che permettono di personalizzare ulteriormente AutoCAD in base alle proprie necessità.

Introduzione

Uso della documentazione di AutoCAD

Oltre al *Manuale di personalizzazione di AutoCAD*, sono disponibili altri manuali, sia in linea che stampati, contenenti informazioni per agevolare la fase di apprendimento e l'utilizzo di AutoCAD. Gli utenti principianti, prima di iniziare ad usare il corso interattivo *Esercitazioni*, dovrebbero leggere le informazioni relative ai concetti CAD ed alle funzioni di AutoCAD nella *Rapida panoramica* in linea. Se si sta eseguendo l'aggiornamento alla Release 14 da una release precedente, vedere *Novità*. Consultare quindi il *Manuale dell'utente di AutoCAD* e la *Guida di riferimento dei comandi di AutoCAD*. Tutta la documentazione di AutoCAD Release 14 è in linea ed è possibile accedervi dal menu ?.

Nota Per informazioni sulle ultime modifiche e procedure di AutoCAD, vedere il file *readme.hlp*.

Introduzione

Uso del manuale

Per facilitare l'utilizzo delle funzioni di personalizzazione di AutoCAD, il *Manuale di personalizzazione* è suddiviso in tre parti:

Parte I: Personalizzazione in cui sono descritte le caratteristiche generali di personalizzazione di cui AutoCAD dispone.

Parte II: AutoLISP in cui viene descritta l'interfaccia di programmazione AutoLISP per AutoCAD. Con AutoLISP è possibile creare comandi ed applicazioni che possono essere eseguiti all'interno di AutoCAD.

Parte III: Finestre di dialogo programmabili in cui viene descritto come progettare e realizzare finestre di dialogo per fornire interfacce nelle applicazioni definite dall'utente.

Appendice A: Codici ASCII in cui sono elencati i codici ASCII standard; ciò risulta utile nelle applicazioni AutoLISP.

Appendice B: Formati dei file di interscambio in cui vengono descritti i formati di file DXF, DXB e di diapositiva.

Appendice C: Codici di gruppo DXF in cui sono descritti i codici di gruppo inclusi nei file DXF e individuati dalle applicazioni AutoLISP e ARX.

Introduzione

Convenzioni tipografiche

Per individuare le funzioni di AutoCAD e riportarle nello stesso modo in cui appaiono sullo schermo, i termini specifici sono riportati utilizzando tipi di caratteri che consentono di distinguerli dal testo. All'interno della documentazione di AutoCAD vengono utilizzate le convenzioni descritte nella tabella riportata di seguito.

Elemento di testo	Esempio
Tasti che vengono premuti sulla tastiera.	CANC, ESC, INVIO
Tasti che vengono premuti contemporaneamente	CTRL+C
Voci di menu, messaggi di richiesta ed altro testo visualizzato sullo schermo	Scegliere Apri dal menu File. Comando: area
Testo che viene digitato dall'utente	<Primo punto>/Oggetto/Aggiungi/Sottrai:
Nomi di file e di directory e istruzioni che seguono le sequenze di messaggi di richiesta	Alla riga di comando, digitare forma <i>disc.dwg</i> <i>c:\acad\support</i>
Oggetti con nome assegnato da AutoCAD, quali ad esempio layer, tipi di linea e stili, nonché comandi e variabili di sistema di AutoCAD	Selezionare oggetti: <i>Usare uno dei metodi di selezione oggetti</i> DISC-FRONT, DASHDOT, STANDARD SALVA, COPIA, DWGNAME, PUNTINI
Variabili AutoLISP e C, codice di esempio e testo di file ASCII	La variabile <i>pi</i> viene preimpostata sul valore <i>pi</i> . ***POP1
I nomi di funzioni AutoLISP, ARX e DIESEL	command ads_command ()
Argomenti formali specificati nelle definizioni delle funzioni	Gli argomenti <i>stringa</i> e <i>modalità</i> .

Introduzione

Modifiche e miglioramenti della Release 14

Nella Release 14 di AutoCAD, gran parte delle funzioni di personalizzazione di AutoCAD sono state migliorate o modificate. Le modifiche generali apportate alle funzioni di personalizzazione sono le seguenti:

- L'Automazione ActiveX è una nuova interfaccia di programmazione per AutoCAD che consente di creare script, macro e applicazioni di altri produttori tramite ambienti di programmazione di Automazione, quale ad esempio Visual Basic 4.0.
- Con la Release 14 l'ambiente di programmazione ADS è diventato obsoleto. Molte applicazioni ADS scritte per la Release 13 continueranno a funzionare con AutoCAD, ma tutte le nuove applicazioni devono essere create con ADSRX o ARX.
- Nuovi oggetti, quali immagine, polilinea ottimizzata e tratteggio.
- La variabile di sistema ACADCFG (od opzione /c) consente ora di puntare ad un file; non è più necessario utilizzare directory di configurazione distinte.
- È possibile tagliare e incollare testo sulla riga di comando.
- I file *acad.ini* e *acadenv.ini* non sono più utilizzati.
- Per poter utilizzare font PFB in AutoCAD, è necessario compilarli in file SHX.
- La guida personalizzata di AutoCAD è stata migliorata.
- La variabile di ambiente ACADHELP è stata eliminata.
- Per richiamare script all'avvio di AutoCAD, è necessario utilizzare le opzioni della riga di comando. Non è più possibile iniziare un nuovo disegno con un nome specifico; il nome del file viene applicato al disegno al momento del salvataggio.

Le modifiche apportate ai menu sono le seguenti:

- Come conseguenza della nuova funzione di modifica del riga di comando, i seguenti tasti di scelta non possono essere personalizzati: FRECCIA SU, FRECCIA GIÙ, FRECCIA SINISTRA, FRECCIA DESTRA, HOME, FINE (le versioni tasti MAIUSC+ e CTRL+ sono ancora disponibili).
- Non è più consentito creare sottomenu (**xxx) nelle sezioni di menu Buttons, Aux, Pop e Tables. Per i sottomenu è ora necessario utilizzare la sintassi di sezione di menu completa (**xxx) con alias opzionali (**xxx).
- Il nome dell'ultimo menu che è stato caricato viene memorizzato nel registro di sistema. Viene inoltre memorizzato nel disegno per compatibilità con versioni precedenti.

Le modifiche relative ad AutoLIST sono le seguenti:

- Le variabili e le definizioni di AutoLISP vengono mantenute tra i diversi disegni. Ciò è controllato dall'opzione Ricarica AutoLISP

tra i disegni della scheda Compatibilità, nella finestra di dialogo Preferenze, e dalla variabile di sistema LISPINIT.

- Le nuove funzioni AutoLISP sono le seguenti:

`menugroup` verifica che un gruppomenu sia stato caricato.

- Le seguenti funzioni AutoLISP sono state modificate:

`grclear` è diventata obsoleta.

Le modalità 1 e 2 della funzione `redraw` sono state leggermente modificate, mentre la modalità 3 è stata bilanciata con la modalità 4.

Parte I Personalizzazione

Panoramica

Le caratteristiche di personalizzazione della Release 14 di AutoCAD vengono descritte in questa sezione ed includono:

Esecuzione di programmi ed utilità esterni da AutoCAD. Ad esempio, se si desidera copiare un disco o cancellare un file, è possibile eseguire tali operazioni dall'interno di AutoCAD aggiungendo il comando esterno appropriato al file di parametri di programma, `acad.pgp`. È possibile creare semplici abbreviazioni o alias per i comandi AutoCAD usati più frequentemente. Ad esempio, è possibile richiamare il comando PTRATT digitando `ptr`. Gli alias di comandi vengono definiti nel file di parametri di programma, `acad.pgp`.

Creazione della Guida in linea per le proprie applicazioni e le procedure interne creando file di Guida di AutoCAD. I file di Guida di AutoCAD sono accessibili con programmi AutoLISP ed ARX. Tramite la creazione di un file di Guida di AutoCAD, è possibile creare la guida personalizzata per i comandi AutoCAD e per le proprie applicazioni.

Creazione di tipi di linea personalizzati utilizzando modelli di tratteggio, forme e font di testo. È possibile creare i propri modelli di tratteggio, forme e font di testo conformi agli standard ed ai metodi di lavoro della propria azienda.

Creazione di menu e barre degli strumenti personalizzati per modificare l'interfaccia utente AutoCAD. I menu controllano molti aspetti dell'interfaccia AutoCAD. Se i comandi usati più frequentemente vengono inseriti nella posizione più accessibile in un menu, e se vengono combinati gruppi di comandi in un singolo elemento di menu, viene aumentata la produttività dell'utente. Personalizzando i menu, è possibile adeguare l'interfaccia di AutoCAD alle proprie esigenze. Il risultato della selezione di un pulsante con un dispositivo di puntamento, così come la funzionalità e l'aspetto di menu a discesa, menu di schermo, menu di tavoletta e menu a gruppi di immagini, vengono definiti nel file di menu. Anche le barre degli strumenti ed i tasti di scelta sono definiti in questo file.

Personalizzazione della riga di stato. È possibile usare il linguaggio DIESEL e la variabile di sistema MODEMACRO per fornire informazioni aggiuntive nella linea di stato quali la data e l'ora, le impostazioni delle variabili di sistema o altre informazioni recuperabili attraverso AutoLISP.

Automatizzazione di attività ripetitive scrivendo degli script. Uno script è un file di testo ASCII contenente comandi AutoCAD elaborati come un file di comandi al momento dell'esecuzione dello script. Ad esempio, se un insieme di disegni deve essere plottato in un determinato modo, è possibile scrivere uno script che apra il disegno, attivi e disattivi vari layer ed esegua comandi di PLOT. Sarà poi possibile eseguire uno script durante la notte, usare gli script con *diapositive* per creare presentazioni automatizzate o continue come quelle usate per le fiere commerciali. Una diapositiva è uno snapshot non modificabile dell'area grafica di AutoCAD. Le diapositive sono anche usate per includere informazioni grafiche in menu a gruppi di immagini e finestre di dialogo.

Ridefinizione o disabilitazione di comandi AutoCAD selezionati, sia alla riga di comando che come parte di un programma AutoLISP o ARX. Ad esempio, è possibile ridefinire determinati comandi di AutoCAD per eseguire messaggi ed istruzioni supplementari o per creare un sistema di gestione di disegni in cui i comandi ESCI e FINE vengano ridefiniti per scrivere informazioni in un file di registrazione prima di concludere la sessione di modifica.

In questo manuale viene inoltre fornita un'introduzione alle interfacce di programmazione di applicazione (API) di AutoCAD. AutoLISP, una versione ad hoc del linguaggio di programmazione LISP, fa parte del pacchetto di AutoCAD. È possibile adeguare AutoCAD alle proprie esigenze usando AutoLISP per automatizzare operazioni ripetitive e creare nuovi comandi AutoCAD. È possibile scrivere programmi AutoLISP o usare programmi shareware o di altre società. L'Automazione ActiveX, definita in precedenza Automazione OLE, è un'alternativa ad AutoLISP e DCL. È possibile accedere e gestire gli oggetti AutoCAD da qualsiasi applicazione in grado di controllare l'Automazione (ad esempio Visual Basic o qualsiasi applicazione di Microsoft Office 97). Con l'interfaccia di programmazione ARX (AutoCAD Runtime Extension) è possibile utilizzare il linguaggio di programmazione C e C++ per personalizzare AutoCAD. È possibile eseguire programmi applicativi ARX di altre società oppure crearne di nuovi.

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

Panoramica

In questo capitolo vengono descritte le funzioni di base per la personalizzazione di AutoCAD. Se non si conoscono tali funzioni, leggere prima questo capitolo, che descrive come specificare da AutoCAD i comandi del sistema operativo e come creare abbreviazioni per i comandi utilizzati più di frequente. Di seguito viene inoltre descritto come creare la guida in linea per le proprie applicazioni e procedure.

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

Requisiti di base

La maggior parte delle funzioni per la personalizzazione descritte in questo manuale richiede la modifica o la creazione di file di testo ASCII. Pertanto, è necessaria la conoscenza di un editor di testo (ad esempio, il Blocco note di Windows) o di un elaboratore di testi (ad esempio, Microsoft Word) che consenta di salvare file in formato ASCII. È necessario, inoltre, eseguire copie di backup dei file di supporto di AutoCAD prima di modificarli.

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

Requisiti di base

L'ambiente AutoCAD

La struttura di default della directory relativa ai file di programma e di supporto di AutoCAD è impostata in modo tale da organizzare tali file in gruppi logici. Se tale organizzazione non dovesse soddisfare le proprie necessità, sarà possibile apportare delle modifiche. Tuttavia, poiché alcune applicazioni cercheranno determinati file in specifiche posizioni, occorre verificare che tali modifiche non siano in conflitto con i requisiti delle applicazioni stesse. Se non si specificano l'unità e la directory, AutoCAD è in grado di individuare solo i file che si trovano nel percorso di ricerca per le librerie (descritto di seguito).

Struttura della directory di esempio

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

Requisiti di base

L'ambiente AutoCAD

Percorso di ricerca per le librerie

AutoCAD ricerca i file di supporto nell'ordine specificato dal *percorso delle librerie* nel modo riportato di seguito:

- 1 Directory corrente.
- 2 Directory contenente il file di disegno corrente.
- 3 Directory elencate nel percorso di Supporto. Per ulteriori informazioni, vedere "Percorso di ricerca dei file di supporto" in *Manuale di installazione di AutoCAD*.
- 4 Directory contenente i file di programma di AutoCAD.

Nota In base all'ambiente corrente, è possibile che due o più directory tra quelle sopra indicate siano uguali.

Se un file non si trova in questo percorso di ricerca, è necessario specificare sia il nome del percorso che il nome del file affinché AutoCAD sia in grado di trovarlo. Ad esempio, se si desidera inserire il disegno *part5.dwg* nel disegno corrente e questo non si trova nel percorso di ricerca per le librerie, occorre specificare per intero il nome del percorso e del file come mostrato qui di seguito:

Comando: **inser**

Nome del blocco (o ?) **/files2/olddwgs/part5**

Se il disegno si trova nella posizione specificata, AutoCAD richiederà di terminare il comando *INSER* nel modo consueto.

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

Requisiti di base

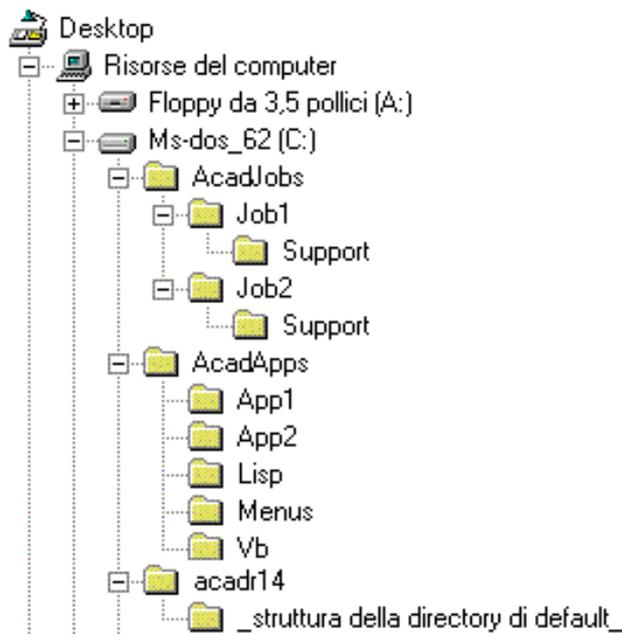
L'ambiente AutoCAD

Struttura della directory di esempio

Questa sezione illustra le opzioni per migliorare la struttura delle directory del sistema operativo in uso e per facilitare la gestione dei file. AutoCAD utilizza directory di file strutturate ad albero.

È opportuno tenere i file supplementari (come le applicazioni AutoLISP ed i file di menu) separati dal programma AutoCAD e dai file di supporto, per facilitare l'individuazione di possibili conflitti e per poter aggiornare singolarmente ogni applicazione senza influire sulle altre.

Le sezioni successive fanno riferimento alla struttura di directory di esempio riportata nella figura che segue come metodo di impostazione delle directory.



Esempio di struttura della directory AutoCAD

È possibile creare una directory per le applicazioni AutoLISP e Visual Basic personalizzate, per i file di menu e per le applicazioni di altre società (riportate nell'esempio come *AcadApps*). Se si desidera gestire più directory di disegno (per file di lavoro separati), occorre creare una directory del tipo *AcadJobs*, contenente delle sottodirectory per ogni lavoro.

Configurazioni multiple

Se si configura AutoCAD per i driver di un dispositivo di puntamento e di un plotter, le informazioni fornite vengono registrate in un file di configurazione. Per default, il file *acad14.cfg* si trova nella directory contenente i file di programma di AutoCAD, ma è possibile specificare un percorso e/o un nome di file diversi.

In genere, è sufficiente una sola configurazione, ma talvolta può rendersi necessaria una configurazione multipla. Ad esempio, se si

utilizza prevalentemente il mouse, ma occasionalmente occorre utilizzare la tavoletta di digitalizzazione, è possibile impostare il proprio sistema in modo che possa gestire configurazioni multiple senza dover rieseguire la configurazione ogni volta che occorre cambiare dispositivo.

Nel file di configurazione sono memorizzati i valori di numerose variabili di sistema di AutoCAD e le opzioni di configurazione della finestra di dialogo Preferenze. Se per tali variabili e parametri operativi si desidera impostare valori diversi, è possibile salvare i valori in file di configurazione diversi. Per un elenco delle variabili di sistema e dei relativi tipi, vedere "Variabili di sistema" nella *Guida di riferimento dei comandi di AutoCAD*.

Per poter usufruire dei vantaggi delle configurazioni multiple, occorre configurare AutoCAD per l'uso di differenti file di configurazione. Utilizzare la riga di comando /c per specificare i file di configurazione alternativi.

Directory di disegno multiple

Avere a disposizione più directory di disegno non è solo conveniente, ma spesso necessario. Tenere i file di disegno ed altri file associati in directory separate rende più semplice la gestione di base dei file. La disposizione descritta in questa sezione si basa sull'esempio della struttura delle directory descritte precedentemente, ma è possibile espanderla o modificarla secondo le proprie necessità.

È possibile impostare la directory /AcadJobs in modo che contenga tutte le sottodirectory di disegno, le quali possono includere altre sottodirectory contenenti i file di supporto relativi a quel tipo di disegno o lavoro. La directory /AcadJobs/Job1/Support può contenere blocchi e file AutoLISP specifici per i file di disegno contenuti nella directory /AcadJobs/Job1. Specificando **support** (senza alcun prefisso di percorso) nel percorso Support, la directory *support* all'interno della directory corrente viene aggiunta al percorso di Support. Da notare che se si utilizza la finestra di dialogo Preferenze per specificare una directory, il percorso di tale directory verrà codificato a livello del codice. Per utilizzare la convezione di denominazione relativa descritta precedentemente, è necessario specificare il percorso di supporto digitando l'opzione /s sulla riga di comando.

Assicurando che la directory di disegno richiesta è quella corrente al momento dell'avvio di AutoCAD, tutti i file e le sottodirectory contenute in quella directory sono facilmente accessibili. Per specificare la directory di lavoro corretta per ogni lavoro, è possibile creare un'icona di programma o aggiungere una voce nel menu Avvio.

In alternativa alle icone e ai menu, è possibile utilizzare programmi batch, i quali consentono la creazione di nuove directory per i lavori in modo automatico. Il programma batch che segue verifica l'esistenza di una determinata directory, la rende corrente ed esegue AutoCAD.

```
@echo off
C:
if exist \AcadJobs\Jobs\%1 goto RUNACAD
echo.
echo *** Creating \AcadJobs\Jobs\%1
echo *** Press Ctrl+C to cancel.
echo.
pause
mkdir \AcadJobs\Jobs\%1
:RUNACAD
cd \AcadJobs\Jobs\%1
start C:\Acad14\acad.exe
```

Utilizzando il Blocco note o un altro editor di testo ASCII, salvare questo programma batch in un file denominato *acad.cmd*. Assicurarsi che il nome della directory e l'unità corrispondano a quelli del sistema in uso. Inserire questo file in una directory che si trova nel percorso di ricerca del sistema in uso (ad esempio C:\WINNT). Questo programma batch può essere eseguito utilizzando il comando Esegui del menu Avvio, in Gestione risorse, in Program Manager ed in altri menu. Se il file è stato salvato come *acad.cmd*, utilizzare la seguente sintassi:

acad *nomelavoro*

Dove *nomelavoro* è il nome della directory di lavoro da rendere corrente.

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

Requisiti di base

Procedura per la ricerca dei comandi

Quando si digita un comando, AutoCAD esegue una serie di operazioni per controllare la validità del nome del comando. Un comando può essere una variabile di sistema o un comando incorporato, un comando esterno o un alias definito nel file *acad.pgp* (descritto nella sezione successiva) oppure un comando AutoLISP definito dall'utente. I comandi possono, inoltre, essere definiti da applicazioni ARX o da un comando per driver di periferica. È possibile digitare un comando sulla riga di comando o sceglierne uno

dal menu appropriato. I comandi possono essere specificati anche da un file di script o da un'applicazione AutoLISP o ARX.

I passaggi di seguito riportati descrivono la procedura di ricerca dei comandi di AutoCAD.

- 1 Se l'input è una risposta nulla (SPACEBAR o ENTER), AutoCAD utilizza il nome dell'ultimo comando inviato. GUIDA è il valore di default.
- 2 AutoCAD controlla il nome del comando in base ad un elenco di comandi incorporati. Se il comando è incluso nell'elenco e non è preceduto da un punto (.), verrà controllato in base ad un elenco di comandi non definiti. Se si tratta di un comando non definito, la ricerca continua. In caso contrario e se non esistono problemi di altro tipo, ad esempio nel caso in cui non risulti possibile un'esecuzione trasparente o in modalità Prospettica, il comando viene eseguito.
- 3 AutoCAD controlla il nome del comando in base ai nomi di comandi definiti da un driver di periferica e poi in base a quelli definiti da un driver video.
- 4 AutoCAD controlla il nome del comando in base ai comandi esterni definiti nel file di parametri del programma (*acad.pgp*). Se tale nome corrisponde ad un comando esterno definito, il comando viene eseguito e la ricerca è terminata.
- 5 AutoCAD controlla il nome del comando in base all'elenco di comandi definito dalle applicazioni AutoLISP o ARX. Questo è il momento in cui vengono caricati i comandi che prevedono il caricamento automatico. Per informazioni sui comandi a caricamento automatico, vedere "Caricamento automatico dei comandi."
- 6 AutoCAD controlla il nome del comando in base all'elenco delle variabili di sistema. Se il nome è incluso nell'elenco, AutoCAD esegue il comando MODIVAR utilizzando l'input come nome di variabile.
- 7 Se il nome del comando corrisponde ad un alias di comando definito nel file dei parametri di programma, AutoCAD utilizza il nome del comando per esteso e continua la ricerca, iniziando una nuova iterazione al passaggio 2.
- 8 Se tutte le operazioni descritte nei precedenti passaggi hanno esito negativo, la ricerca termina con il messaggio di avvertenza che indica l'uso di nomi di comandi non ammessi.

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

Requisiti di base

File di supporto personalizzabili

AutoCAD utilizza i file di supporto per scopi quali la memorizzazione delle definizioni dei menu, il caricamento delle routine AutoLISP ed ARX e la descrizione dei font di testo. Molti file di supporto sono file di testo che è possibile modificare con un editor di testo.

Di seguito viene riportato un elenco dei file di supporto di AutoCAD che possono essere modificati (la maggior parte di essi sono descritti nel manuale). Si raccomanda di eseguire una copia di backup di questi file prima di modificarli.

File di supporto personalizzabili

File	Descrizione
*.ahp	Sono i file di Guida di AutoCAD. I file indice di guida associati hanno estensione .hdx.
acad.ase	Include un elenco dei nomi e delle posizioni dei driver di database ASE.
*.ccp	Sono i file della tavolozza dei colori CalComp. Da utilizzare con le stampanti ed i plotter CalComp.
*.cus	È il file dizionario personalizzato.
*.dcl	Sono le descrizioni delle finestre di dialogo nel linguaggio DCL (Dialog Control Language) di AutoCAD.
acad.dcl	Descrive le finestre di dialogo standard di AutoCAD.
*.dxt	È il file del programma di conversione DXFIX.
*.lin	Sono file di definizione dei tipi di linea di AutoCAD.
acad.lin	È il file della libreria dei tipi di linea standard di AutoCAD.
*.lsp	Sono i file di programma AutoLISP.
acad.lsp	È una routine AutoLISP definita dall'utente che viene caricata ogni volta che si inizia un disegno.
*.mln	È il file di una libreria di multilinee.
*.mnd	È un tipo speciale di file di menu non compilato che contiene le macro. Per compilare questo file viene usato il file mc eseguibile.
*.mnl	Sono le routine AutoLISP usate dai menu di AutoCAD. Un file .mnl deve avere lo stesso nome del file .mnu che supporta.
acad.mnl	Sono le routine AutoLISP usate dal menu standard di AutoCAD.

*.mns	<i>Sono i file sorgenti dei menu generati da AutoCAD. Contiene le stringhe di comando e la sintassi delle macro che definiscono i menu di AutoCAD.</i>
acad.mns	<i>È il file sorgente per il menu standard di AutoCAD.</i>
*.mnu	<i>Sono i file sorgenti dei menu di AutoCAD. Contiene le stringhe di comando e la sintassi delle macro che definiscono i menu di AutoCAD.</i>
acad.mnu	<i>È il file sorgente per il menu standard di AutoCAD.</i>
acadfull.mnu	<i>È il file sorgente per il menu di tipo DOS.</i>
*.pat	<i>Sono i file di definizione dei modelli di tratteggio di AutoCAD.</i>
acad.pat	<i>È il file della libreria dei modelli di tratteggio standard di AutoCAD.</i>
*.pcp	<i>Sono i file dei parametri di configurazione per il plottaggio di AutoCAD. Ogni file .pcp memorizza le informazioni di configurazioni per uno specifico plotter.</i>
acad.pgp	<i>È il file dei parametri di programma di AutoCAD. Contiene le definizioni per i comandi esterni e per gli alias dei comandi.</i>
fontmap.ps	<i>È il file della mappa dei font di AutoCAD. Viene utilizzato dal comando PSIN; è il catalogo (o mappa dei font) di tutti i font conosciuti dall'interprete PostScript di AutoCAD.</i>
acad.psf	<i>È il file di supporto PostScript di AutoCAD; è il file di supporto principale per i comandi PSOUT e PSMOTIVO.</i>
*.rpf	<i>È il file di definizione del riempimento di aree chiuse raster. Da utilizzare con i driver delle stampanti e dei plotter Hewlett-Packard.</i>
acad.rx	<i>Elenchi di applicazioni ARX che vengono caricate all'avvio di AutoCAD.</i>
*.scr	<i>Sono i file di script di AutoCAD. Un file di script contiene un gruppo di comandi AutoCAD elaborati come un batch.</i>
*.shp	<i>Sono file di definizione delle forme e dei font di AutoCAD. I file delle forme e dei font compilati hanno l'estensione .shx.</i>
acad.unt	<i>È il file di definizione dell'unità di AutoCAD. Contiene i dati che consentono di effettuare la conversione da un gruppo di unità ad un altro.</i>

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

File dei parametri di programma: acad.pgp

Il file dei parametri di programma di AutoCAD, *acad.pgp*, è un file di testo ASCII in cui sono memorizzate le definizioni dei comandi. Può essere considerato come un elenco di comandi di AutoCAD personalizzati. Quando si digita un comando non usuale, AutoCAD ricerca tale comando nel file *acad.pgp*. Questo file è diviso in due sezioni. La prima sezione definisce i comandi esterni, mentre la seconda definisce gli alias dei comandi. Il file può contenere righe di commento precedute da un punto e virgola (;).

Ogni volta che si apre un disegno nuovo o esistente, AutoCAD esegue la ricerca nel percorso delle librerie e legge il primo file *acad.pgp* che trova. Utilizzare il comando INIZIALIZZA per inizializzare o caricare di nuovo il file *acad.pgp* durante una sessione di modifica.

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

File dei parametri di programma: acad.pgp

Definizione di comandi esterni

Durante l'esecuzione di AutoCAD, è possibile richiamare altri programmi o utilità, come quelli riportati di seguito:

- Comandi ed utilità di Windows, quali **start**, **type**, **dir** o **copy**.
- Applicazioni come editor di testo o elaboratori di testo.

- Programmi di gestione di database, fogli elettronici e programmi di comunicazione.
- Programmi forniti dall'utente, quali file batch o applicazioni Visual Basic.

Quando si definisce un comando esterno, si specifica il nome di un comando da utilizzare dalla riga di comando di AutoCAD ed una stringa di comando eseguibile che viene inviata al sistema operativo. Ogni riga presente nella sezione relativa ai comandi esterni contiene cinque campi delimitati da virgola, come riportato di seguito.

```
comando, file_eseguibile, memoria[, [*]messaggio_richiesta[, codice_ritorno]
```

comando

Il comando da digitare alla riga di comando. Se questo nome è quello di un comando interno AutoCAD, viene ignorato. Per il nome è possibile utilizzare indifferentemente sia caratteri minuscoli che maiuscoli.

file_eseguibile

Questa stringa costante viene inviata al sistema operativo quando si digita il nome del comando e può rappresentare qualsiasi comando che è possibile eseguire dal messaggio di richiesta del sistema. La stringa può includere opzioni o parametri. La possibilità di utilizzare indifferentemente sia caratteri minuscoli che maiuscoli dipende dal sistema operativo in uso e dall'applicazione in esecuzione.

memoria

Questo campo mantiene la compatibilità con le versioni precedenti di AutoCAD. Sebbene il valore di questo campo *non* venga usato in AutoCAD, è necessario che in esso sia presente un numero (di solito 0).

prompt

Questo campo è opzionale e specifica il messaggio di richiesta da visualizzare alla riga di comando AutoCAD. La risposta a questo messaggio di richiesta viene aggiunta alla stringa fornita nel campo del file eseguibile. Se il primo carattere del campo del messaggio di richiesta è un asterisco (*), la risposta può contenere spazi e l'utente deve premere ENTER per terminarla, altrimenti la risposta può essere terminata premendo SPACEBAR o ENTER. Se non viene specificato alcun messaggio di richiesta, non è richiesto alcun input. Se tuttavia deve essere specificato un codice di ritorno o se per il messaggio di richiesta viene richiesto uno spazio finale, è necessario aggiungere una virgola.

codice_ritorno

Questo è un parametro opzionale codificato in bit. Questi valori interi sono addizionabili tra loro in qualsiasi combinazione per raggiungere il risultato desiderato. Ad esempio, se sono richiesti i valori 1 e 2, utilizzare 3 come codice di ritorno. I valori vengono definiti nel modo riportato di seguito (i codici 0 e 4 sono inutili in un ambiente a finestra di dialogo, quindi non sono stati inclusi):

- 1** Carica il file DXB. Al termine del comando, AutoCAD carica nel disegno il file DXB *\$cmd.dxb*. Dopo che il file DXB è stato caricato, il file *\$cmd.dxb* viene cancellato. Questa azione provoca lo stesso risultato del comando DXBIN.
- 2** Costruisce la definizione di un blocco da un file DXB. AutoCAD crea la definizione di un blocco dal file DXB *\$cmd.dxb*. La risposta al campo del messaggio di richiesta viene usata come nome del blocco. Tale nome deve essere un nome di blocco valido che non sia già presente nel disegno. Di conseguenza, questa modalità non consente di definire di nuovo un blocco definito in precedenza. Dopo che AutoCAD carica il file DXB, il file *\$cmd.dxb* viene cancellato. Il nome di default per il comando INSER viene impostato sul nome del blocco appena definito.

I comandi Windows **start** e **cmd** risultano molto utili nella definizione di comandi esterni. Se si specifica una stringa eseguibile che *non* utilizza questi comandi, AutoCAD non sarà disponibile fino a quando la finestra non viene chiusa.

Il comando **start** visualizza una finestra distinta ed esegue uno specifico programma o comando; se viene utilizzato senza alcun parametro, apre una nuova finestra della riga di comando. Il comando dispone di molte opzioni per la modifica della visualizzazione della nuova finestra. In Windows NT 3.51, con l'opzione /realtime la nuova finestra viene visualizzata sovrapposta alla finestra di disegno di AutoCAD. Ciò non si verifica necessariamente in Windows95 o NT 4.0. Questa disposizione è la più appropriata se si intende digitare comandi al messaggio di richiesta di Windows. Utilizzare **start** senza alcuna opzione per avviare un'applicazione per Windows. Questo comando risulta inoltre utile per l'apertura diretta di documenti associati ad una applicazione, ad esempio un documento creato con un elaboratore di testi o un file HTML.

Il comando **cmd** visualizza una finestra della riga di comando che agisce come shell di AutoCAD; per poter tornare alla riga dei comandi di AutoCAD, è necessario che questa finestra sia chiusa. Le opzioni /c e /k risultano utili per i comandi esterni; la prima esegue il comando specificato e completa l'operazione chiudendo la finestra, la seconda esegue il comando specificato e continua con l'esecuzione lasciando la finestra aperta. Quando si utilizza l'opzione /k, è necessario chiudere la finestra dei comandi (con il comando **exit**).

In genere, **start** può essere utilizzato per lanciare una nuova finestra o applicazione per l'esecuzione di un processo distinto da AutoCAD. Utilizzare **cmd** per eseguire un file batch o script di comandi che non crea una finestra distinta oppure per creare una finestra che deve essere chiusa prima di ritornare ad AutoCAD. Per ulteriori informazioni su questi comandi, vedere la documentazione sui comandi di Windows.

L'esempio che segue, riguardante una sezione di comandi esterni, definisce tre nuovi comandi: RUN, LISTSET e DXB2BLK.

```
RUN, cmd /c,0,*Batch file to run: ,
LISTSET,cmd /k SET,0
DXB2BLK,cmd /c DXBCOPY,0,DXB file: ,2
```

Il comando RUN esegue un file batch o uno script di comando. Il comando **cmd** seguito dall'opzione /c apre una finestra di comando, esegue il file batch e chiude la finestra.

Il comando LISTSET visualizza le impostazioni correnti delle variabili d'ambiente DOS. Poiché questo esempio utilizza **cmd /k** anziché **start**, la finestra di comando deve essere chiusa prima di ritornare ad AutoCAD. Se si desidera mantenerla aperta, utilizzare **start /realtime**.

Il comando DXB2BLK crea la definizione di un blocco dal file DXB specificato. Tale file converte tutti gli oggetti in linee. Un vantaggio di questa procedura è che fornisce un metodo semplice per esplodere oggetti di testo in linee.

DXB2BLK passa il nome del file DXB specificato al file batch *dxbcopy*, che copia tale file assegnandogli il nome *\$cmd.dxb*. AutoCAD crea quindi un blocco dal file DXB specificato. Il nome fornito al messaggio di richiesta per il file DXB viene usato come nuovo nome del blocco. Per creare il file *dxbcopy.cmd*, digitare quanto riportato di seguito alla riga di comando di DOS.

```
echo copy %1.dxb $cmd.dxb > dxbcopy.cmd
```

In questo modo, il file *dxbcopy.cmd* viene creato nella directory corrente. Spostare tale file in una directory del percorso DOS oppure specificarne esplicitamente la posizione nel file *acad.pgp*. Ad esempio, se il file *dxbcopy.cmd* si trova in *D:/cad*, nella sezione dei comandi esterni del file *acad.pgp*, digitare quanto indicato di seguito.

```
DXB2BLK, cmd /c D:\CAD\DXBCOPY,0,DXB file: ,2
```

Per creare un file DXB, scegliere DXB AutoCAD, formati di file di output come stampante corrente e stampare su un file. Per ulteriori informazioni sulla configurazione di stampanti, vedere il capitolo 2, "Impostazione di AutoCAD", nel *Manuale di installazione di AutoCAD*.

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

File dei parametri di programma: acad.pgp

Alias dei comandi

È possibile abbreviare i comandi AutoCAD di uso frequente definendo degli alias nella relativa sezione del file *acad.pgp*. Gli alias possono essere definiti per ogni comando AutoCAD, per ogni comando riguardante i driver di periferica e per ogni comando esterno.

I due campi che seguono, delimitati da una virgola, definiscono l'alias di un comando nel file *acad.pgp*.

```
abbreviazione,*comando
```

abbreviazione

È l'abbreviazione del comando che viene digitata alla riga di comando.

comando

È il comando AutoCAD che è stato abbreviato. Digitare un asterisco (*) prima del nome del comando per identificare la linea come alias di comando.

Se è possibile immettere un comando in modo trasparente, è anche possibile immettere in modo trasparente il corrispondente alias. Quando si digita l'alias di un comando, AutoCAD visualizza il nome completo nella riga di comando ed esegue il comando.

La parte di sezione riportata di seguito riguardante gli alias dei comandi contiene elementi simili a quelli presenti nel file *acad.pgp* standard.

```
AR,*ARCO
C,*CERCHIO
CP,*COPIA
```

La sezione degli alias dei comandi può includere comandi che hanno come prefisso il simbolo speciale meno (-). Ciò consente di creare alias che accedono alla versione della riga di comando di determinati comandi.

```
PTR,*-PTRATT
CON,*-CONTORNI
```

Nota Non è possibile utilizzare gli alias dei comandi negli script di comando e non è consigliabile utilizzare gli alias nel file di menu.

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

Personalizzazione della documentazione in linea

È importante ed utile per l'utente documentare tutte le modifiche o i miglioramenti apportati ad AutoCAD, in quanto una documentazione completa ed accurata consente una migliore utilizzazione del programma.

Mediante AutoCAD è possibile accedere facilmente a tre tipi di documentazione in linea definibile dall'utente: file della Guida di Windows, file HTML e file della Guida di AutoCAD. I file della Guida di Windows forniscono una documentazione in linea sensibile al contesto e personalizzabile, contenuta in un unico file. Questi file sono adatti per le applicazioni che richiedono interfacce o finestre di dialogo sensibili al contesto oppure una documentazione che non deve essere aggiornata di frequente. Esistono molti programmi che semplificano le operazioni di creazione e gestione dei file HTML. I file HTML sono molto facili da aggiornare e possono rivelarsi indispensabili per le società che necessitano di fornire documentazione a siti remoti. I file della Guida di AutoCAD sono file ASCII che possono essere creati e gestiti facilmente con un editor di testo. Il contenuto di un file della Guida di AutoCAD viene visualizzato in una finestra di dialogo nella finestra di disegno di AutoCAD.

AutoCAD fornisce ulteriori funzioni di guida mediante il file di menu (vedere "Guida specifica di menu."

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

Personalizzazione della documentazione in linea

File della Guida di Windows

I file della Guida di Windows (WinHelp) compongono il sistema di documentazione in linea nativo per Windows 95 e Windows NT. Questi file possono essere utilizzati come documentazione indipendente o visualizzati in AutoCAD ed in altre applicazioni per Windows. I file WinHelp possono contenere sia testo che a grafica ed elementi multimediali. La loro funzionalità può essere migliorata mediante le macro WinHelp e le funzioni API; ciò consente di avere a disposizione uno strumento di documentazione molto potente.

Lo scopo di questo manuale non è quello di insegnare a creare un file della Guida di Windows, ma di fornire informazioni generali sul processo ed i componenti, e di descrivere il modo in cui questi file vengono utilizzati con AutoCAD. Sono disponibili molti libri e applicazioni che trattano la creazione di file WinHelp. È tuttavia necessario disporre almeno dell'applicazione Microsoft Help Workshop, la quale include il compilatore della Guida ed un file WinHelp che fornisce tutte le informazioni essenziali necessarie alla creazione di file della Guida.

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

Personalizzazione della documentazione in linea

File della Guida di Windows

Componenti del sistema di guida di Windows

Il compilatore della Guida di Windows crea file di guida da file RTF (Rich Text Format) codificati in modo speciale. Poiché RTF è un formato di output standard di Microsoft Word, Word è l'ambiente più appropriato per la creazione dei file WinHelp. È comunque possibile creare file di guida da file RTF codificati manualmente o dall'output di altre applicazioni.

La compilazione di un file WinHelp viene effettuata da uno o più file .rtf. Il file .hpl (Help Project File) elenca tutti i file .rtf da includere, oltre ad altri parametri, quali la posizione di file grafici e definizioni di macro, dimensioni, posizione e colori della finestra della Guida. Il compilatore della Guida legge il file .hpl e costruisce il file della Guida finale (.hlp). Il file di guida generato dal compilatore può essere visualizzato eseguendolo dal File Manager o da Gestione risorse.

I file della Guida di Windows utilizzano la finestra Guida in linea per visualizzare le schede Sommario, Indice e Trova. La scheda Sommario fornisce un'interfaccia grafica per gli argomenti in uno o più file di guida. Gli elementi ed il layout di questa scheda vengono definiti da un file di sommario (*.cnt). Per permettere all'utente ed agli sviluppatori di espandere il sistema della Guida di AutoCAD standard, viene fornito un metodo per aggiungere ulteriori argomenti alla scheda Sommario.

Se il file *include.cnt* è presente nella directory *help/*, tutti i file di sommario a cui si fa riferimento in quel file vengono aggiunti alla scheda Sommario di AutoCAD. Il file *include.cnt* deve contenere solo istruzioni :Include, come ad esempio:

```
:Include standard.cnt
:Include miaguida.cnt
:Include webhelp.cnt
```

L'esempio precedente aggiunge gli argomenti elencati nei file *standard.cnt*, *miaguida.cnt* e *webhelp.cnt* alla scheda Sommario di AutoCAD. Per informazioni sul contenuto dei file CNT, vedere "Aggiunta di file della Guida di Windows alla scheda Sommario" e "Aggiunta di file HTML alla scheda Sommario."

Nota Le modifiche apportate al file *include.cnt* o a uno dei file CNT a cui è stato fatto riferimento risultano effettive solo dopo aver eliminato il file *acad.gid*.

L'elenco che segue mostra i file che vengono utilizzati o creati dal sistema della Guida di Windows.

File utilizzati dalla Guida di Windows

Estensione file	Descrizione
RTF	File sorgente WinHelp
HPJ	File di progetto WinHelp
HLP	File WinHelp compilato
CNT	File di sommario
GID	File di configurazione di WinHelp File binario creato dall'eseguibile <i>winhlp32</i> . Include informazioni sul file della Guida, tra cui collegamenti al file di sommario, nomi di file di tutti i file di guida inclusi, parole chiave e posizione dei file.
FTS	File di ricerca estesa. File binario creato dall'eseguibile <i>winhlp32</i> la prima volta che si esegue una ricerca estesa.
FTG	File di gruppi di ricerca estesa. File binario creato dall'eseguibile <i>winhlp32</i> .

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

Personalizzazione della documentazione in linea

File della Guida di Windows

Uso dei file della Guida di Windows con AutoCAD

Se si utilizzano i file della Guida di Windows come documentazione per applicazioni personalizzate o standard aziendali, può risultare utile accedere facilmente a tali informazioni da AutoCAD. Le sezioni che seguono descrivono i metodi forniti da AutoCAD per accedere ai file WinHelp.

Aggiunta di file della Guida di Windows alla scheda Sommario

È possibile aggiungere i propri file WinHelp ed i collegamenti con altre applicazioni alla scheda Sommario della Guida di AutoCAD. Il file *include.cnt*, descritto precedentemente nella sezione "Componenti del sistema di guida di Windows", definisce voci aggiuntive visualizzate nella scheda Sommario. Se si desidera includere i file aggiuntivi nelle ricerche di parole chiave e di testo estese, utilizzare l'istruzione :Index come mostrato nell'esempio che segue.

Una possibile aggiunta alla Guida di AutoCAD potrebbero essere gli standard aziendali. L'esempio che segue fornisce accesso agli argomenti del file *standard.hlp* ed utilizza l'istruzione :Index per includere questo file in qualsiasi parola chiave o ricerche di testo estese.

```
:Index Standards=standards.hlp
1 Standard
2 Architettonico
3 Informazioni generali=arch_overview@standard.hlp
3 Progettazione=arch_design@standard.hlp
3 Disegno=eng_drafting@standard.hlp
3 Relazioni=eng_reports@standard.hlp
2 Ingegneristico
```

```

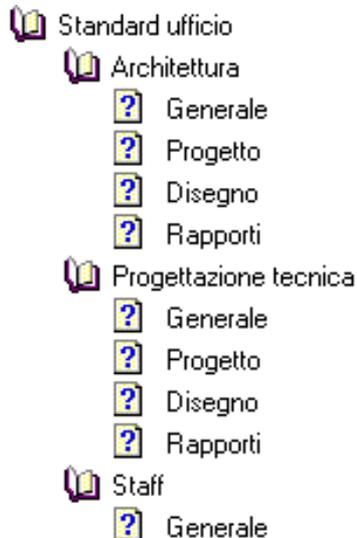
3 Informazioni generali=eng_overview@standards.hlp
3 Progettazione=eng_design@standard.hlp
3 Disegno=eng_drafting@standard.hlp
3 Relazioni=eng_reports@standard.hlp
2 Personale di supporto
3 Informazioni generali=eng_overview@standard.hlp

```

Al file *help/include.cnt* è necessario aggiungere quanto segue:

```
:Include standard.cnt
```

A questo punto, cancellare il file *acad.gid*; se non si esegue questa operazione, è possibile che quanto aggiunto alla scheda Sommario non venga visualizzato. La volta successiva che si esegue il file *acad.hlp* (dal File Manager o da Gestione risorse) o si sceglie Sommario dal menu ? di AutoCAD, le nuove voci appariranno dopo quelle esistenti.



Esecuzione dei file della Guida di Windows dalla riga di comando

Un modo facile e veloce di accedere al file di guida personalizzato dalla riga di comando consiste nell'aggiungere la riga seguente alla sezione dei comandi esterni del file *acad.pgp* file:

```
MIAGUIDA,start c:\acad14\help\miaguida.hlp,0
```

Questo comando esterno definisce il nuovo comando di AutoCAD MIAGUIDA. Ricordarsi di utilizzare il comando INIZIALIZZA per ricaricare il file *acad.pgp*.

Per la definizione di comandi personalizzati che richiamano i file di guida, è anche possibile utilizzare AutoLISP. Per ulteriori informazioni, vedere la sezione successiva.

Esecuzione dei file della Guida di Windows da AutoLISP

Il codice AutoLISP seguente richiama il file WinHelp *miaguida.hlp*:

```
(help "miaguida.hlp")
```

{La funzione **help** accetta argomenti aggiuntivi. Per ulteriori informazioni, vedere "help."}

Quando si preme F1, la funzione **setfunhelp** associa la guida sensibile al contesto ad un comando definito dall'utente. Per ulteriori informazioni, vedere "setfunhelp."

Per eseguire un file WinHelp è possibile utilizzare anche la funzione **startapp**. Per visualizzare un file WinHelp con la funzione **startapp**, è necessario passare il nome dell'applicazione (*winhlp32* per file WinHelp 4.0 e *winhelp* per file WinHelp di versioni precedenti) seguito dal nome del file di guida. Ad esempio, il codice seguente utilizza la funzione **startapp** per visualizzare il file *granhelp.hlp*:

```
(startapp "winhlp32" (findfile "help/granhelp.hlp"))
```

Nota L'argomento relativo al nome del file deve essere definito in modo esplicito; usare pertanto la funzione **findfile** per fornire il percorso completo ed il nome del file.

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

Personalizzazione della documentazione in linea

File HTML

Con una quantità sempre crescente di informazioni disponibili su Internet, è possibile che i file HTML (pagine Web) diventino il sistema di documentazione in linea di riferimento. Uno dei più importanti vantaggi della documentazione HTML è che può essere aggiornata senza dover reinstallare o modificare il sistema locale. Il collegamento con siti remoti si rivela più efficiente se si dispone di una connessione diretta con Internet, mentre può rivelarsi frustrante e controproducente per gli utenti che ne sono sprovvisti. Tuttavia, anche senza accedere ad Internet, è possibile fare buon uso dei file HTML sul sistema locale in uso.

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

Personalizzazione della documentazione in linea

File HTML

Uso dei file HTML con AutoCAD

Se si desidera accedere facilmente a siti Web remoti oppure utilizzare i file HTML come documentazione per applicazioni personalizzate o standard aziendali, è possibile configurare AutoCAD in modo da accedere senza problemi a queste informazioni. Le sezioni che seguono descrivono i metodi forniti da AutoCAD per accedere a questi file.

Aggiunta di file HTML alla scheda Sommario

È possibile aggiungere alla scheda Sommario della Guida di AutoCAD i propri file HTML o collegamenti con pagine Web. Il file *include.cnt* definisce voci aggiuntive visualizzate nella scheda Sommario. Per ulteriori informazioni sui file CNT, vedere "Componenti del sistema di guida di Windows".

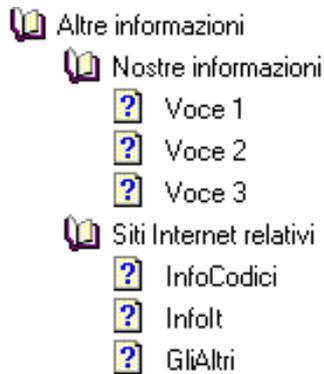
Se ad esempio si vuole accedere senza problemi alla propria documentazione HTML, cioè *varie1.htm*, *varie2.htm* e *varie3.htm*, e ad alcuni importanti siti Web, creare il file *help/piùhelp.cnt* che includa:

```
1 Più help
2 Nostre info
3 Varie 1=!EF("varie1.htm", "", 4)
3 Varie 2=!EF("varie2.htm", "", 4)
3 Varie 3=!EF("varie3.htm", "", 4)
2 Siti web relativi
3 CodInfo=!EF("http://www.CodInfo.com", "", 4)
3 Spazioin=!EF("http://www.Spazioin.com", "", 4)
3 GliAltri=!EF("http://www.GliAltri.com", "", 4)
```

Successivamente, aggiungere al file *help/include.cnt* quanto segue:

```
:Include piùhelp.cnt
```

A questo punto, cancellare il file *acad.gid*; se non si esegue questa operazione, è possibile che quanto aggiunto alla scheda Sommario non venga visualizzato. La volta successiva che si esegue il file *acad.hlp* (dal File Manager o da Gestione risorse) o si sceglie Sommario dal menu ? di AutoCAD, le nuove voci appariranno dopo quelle esistenti.



Esecuzione di file HTML dalla riga di comando

Un modo facile e veloce di lanciare il browser in uso con un determinato file HTML è di aggiungere la riga seguente alla sezione dei comandi esterni del file *acad.pgp*:

```
MIOHTML,start c:\acad14\help\miohtml.html,0
```

Questo comando esterno definisce il nuovo comando di AutoCAD MIOHTML. Ricordarsi di utilizzare il comando INIZIALIZZA per ricaricare il file *acad.pgp*.

Per la definizione dei comandi creati che richiamano file HTML, è anche possibile utilizzare AutoLISP. Per ulteriori informazioni, vedere la sezione successiva.

Esecuzione di file HTML da AutoLISP

Per visualizzare un file HTML con la funzione **startapp**, è necessario passare il nome dell'applicazione seguito da quello del file HTML. Ad esempio, il codice seguente utilizza la funzione **startapp** per visualizzare il file *miaweb.html*:

```
(setq app "\"C:\\Programmi\\Netscape\\Program\\Netscape.exe\"")
(startapp app (findfile "help/miaweb.html"))
```

Nota Il percorso dell'applicazione e i nomi dei file che includono spazi *devono* essere racchiusi tra virgolette. È inoltre necessario definire in modo esplicito l'argomento relativo al nome del file; usare pertanto la funzione **findfile** per fornire il percorso completo ed il nome del file.

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

Personalizzazione della documentazione in linea

File di Guida di AutoCAD

I file di Guida di AutoCAD sono in formato di testo ASCII con estensione *.ahp* e devono essere richiamati da un'applicazione AutoLISP o ARX. Per ogni file di Guida, AutoCAD crea e gestisce un file di indice (*.hdx*) che permette il corretto funzionamento della funzione di guida. Quando si apportano modifiche ad un file di Guida, il relativo indice viene aggiornato alla successiva lettura del file. Dopo che un file di Guida è stato modificato, cancellare il file *.hdx* esistente per essere certi che il file di indice venga aggiornato.

Il formato del file di Guida di AutoCAD è cambiato rispetto alla Release 13. Nella sezione successiva viene descritto il nuovo formato. Per convertire i file di Guida esistenti della Release 12 nel nuovo formato, vedere "Conversione dei file di Guida R12."

Nota I file della Guida della Release 12 *non possono* essere utilizzati dalla Release 14. Se sul sistema in uso vengono mantenuti i file della Guida della Release 12, assicurarsi che i file di Guida aggiornati si trovino nelle directory riportate *prima* dei file della Guida della Release 12 nel percorso Support.

Le applicazioni AutoLISP ed ARX possono visualizzare ulteriori file di Guida chiamando la funzione **help**. Per la descrizione della funzione **help** vedere "help" nel capitolo 13 e "Accesso ai file della Guida". La funzione **ads_help()** viene descritta nella documentazione ARX. È possibile utilizzare la funzione **help** AutoLISP alla riga di comando per visualizzare un file di guida. Il codice riportato di seguito visualizza il file di Guida di AutoCAD *miaguida.ahp*.

Comando: (**help "miaguida.ahp"**)

Se al file *acad.lsp* si aggiunge il codice AutoLISP che segue, viene creato il comando MIAGUIDA, che visualizza il file *miaguida.ahp*.

```
(defun C:MIAGUIDA ( )
  (help "miaguida.ahp")
  (princ)
)
```

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

Personalizzazione della documentazione in linea

File di Guida di AutoCAD

Uso di un file di Guida di AutoCAD in una directory di sola lettura

Per default, AutoCAD crea il file di indice (.hdx) nella stessa directory del file di Guida. Tuttavia, se si utilizza un file di Guida di AutoCAD che si trova in una directory di sola lettura (ad esempio, un CD oppure un'unità di rete con accesso riservato), non sarà possibile creare il file di indice. Inserendo un file di indice fittizio in una directory (con autorizzazione alla scrittura) nel percorso Support si impone la creazione del file di indice in quella directory. Ad esempio, digitando i seguenti comandi al messaggio di richiesta DOS, si crea un file di indice fittizio per il file *miaguida.ahp* nella directory *\acad14*:

```
C:> echo dummy > \acad14\miaguida.hdx
```

Di seguito viene descritto il processo con cui viene installato il file HDX. È possibile forzare l'installazione del file in una posizione qualsiasi del percorso Support inserendo un file HDX fittizio nella directory desiderata.

- 1 Individuare il file di Guida di AutoCAD. Scrivere un file di indice in quella directory se è possibile farlo, altrimenti andare al passo 2.
- 2 Cercare un file di indice con quel nome nel percorso Support. Se viene trovato un file di indice, aprirlo per la scrittura, se è possibile farlo, altrimenti continuare la ricerca nel percorso Support. Se un file di indice scrivibile non viene trovato, andare al passo 3.
- 3 Cercare il file *acad14.cfg* nel percorso Support. Se viene trovato e il file di indice, scrivere il file di indice in tale directory, se è possibile farlo. In caso contrario, AutoCAD non può scrivere alcun file di indice e il file della Guida non viene visualizzato.

La volta successiva, la stessa logica viene utilizzata per trovare il file di indice generato. Nella riga di comando viene visualizzato un messaggio in cui è indicata la posizione in cui il file è stato scritto.

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

Personalizzazione della documentazione in linea

File di Guida di AutoCAD

Formato del file di Guida di AutoCAD

La struttura di un file di Guida di AutoCAD è simile al formato di un file della Guida di Windows. Le informazioni contenute nei file di Guida di AutoCAD sono organizzate in argomenti. Ogni argomento deve avere un identificativo univoco chiamato *ID argomento* (topic ID), denominato anche stringa di contesto (*context string*). Se lo si desidera, è possibile assegnare all'argomento un titolo ed associarvi alcune parole chiave (keyword). L'ID argomento, il titolo e le parole chiave si trovano su linee speciali chiamate *direttive*, che iniziano sempre con una barra inversa (\). Nella tabella riportata di seguito sono elencate le direttive per la Guida di AutoCAD (le prime tre direttive, \#, \\$ e \K, devono apparire nell'ordine mostrato).

Direttive per la Guida di AutoCAD

Direttiva	Uso	Dove viene visualizzata
\#	ID argomento (stringa di contesto)	Non visibile per l'utente, usata internamente, deve essere univoca e non deve contenere spazi.
\\$	Titolo	Elenco Cronologia e Cerca Argomento.
\K	Una o più parole chiave	Elenco ricerca. Le parole chiave sono separate da un punto e virgola (;).
\(spazio)	Commento	Le linee che iniziano con una barra rovesciata, seguite da uno spazio, vengono ignorate.
\E	Fine file	Fine del file di guida; usata una sola volta alla fine del file.

Le regole riportate di seguito si applicano alle direttive presenti nei file di Guida di AutoCAD.

- Ogni argomento deve avere una stringa di contesto; gli argomenti che non presentano tali stringhe vengono ignorati.
- Un argomento termina quando viene riscontrata un'altra direttiva.
- La stringa della direttiva inizia immediatamente dopo \#, \\$, \K nella posizione 3 dei caratteri. Se la direttiva è seguita da uno spazio, tale spazio viene incluso nella stringa della direttiva.

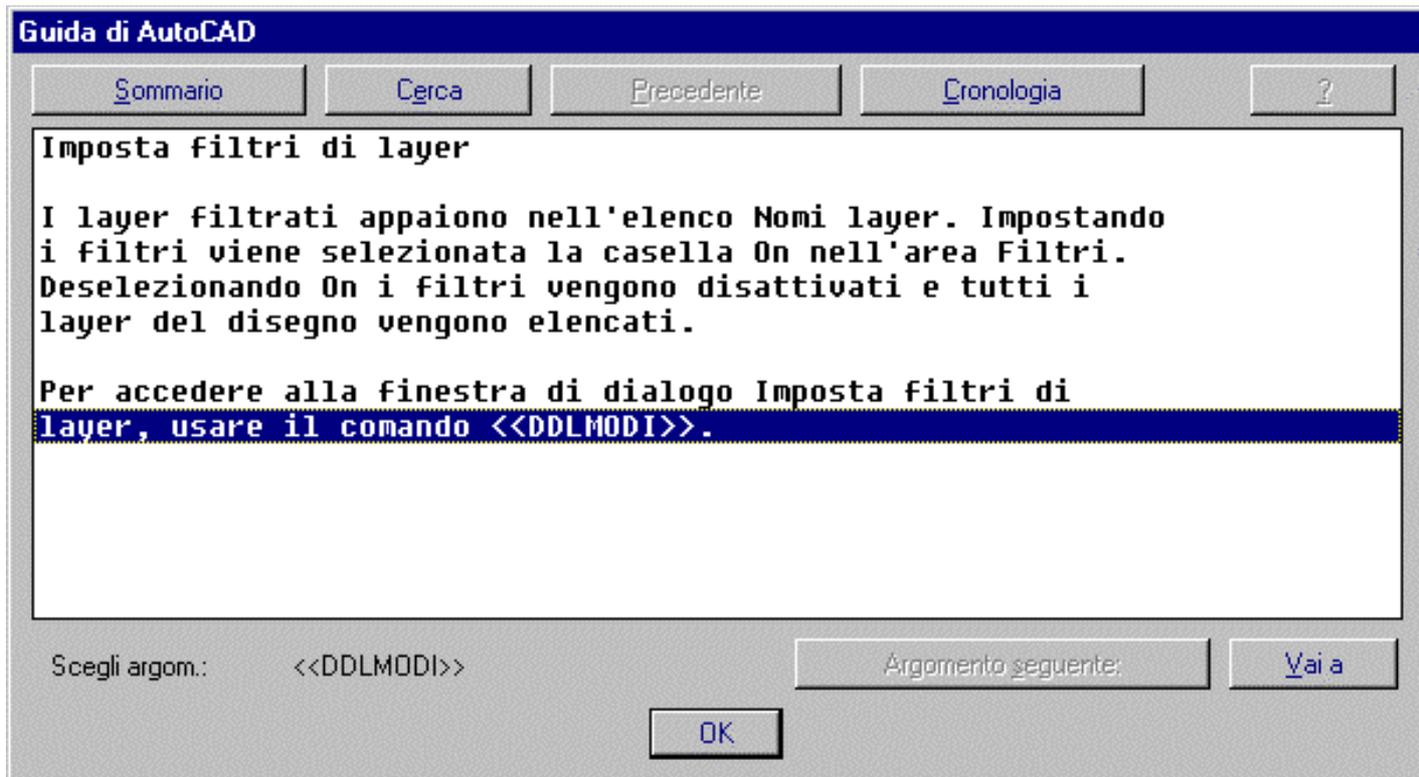
Il programma di visualizzazione della Guida di AutoCAD è realizzato per operare su testo in formato DOS o UNIX EOL, cioè le linee di testo nel file sorgente devono essere chiuse da una sequenza di ritorno a capo (CRLF) (ENTER). I file di guida scritti con altri protocolli EOL potrebbero non operare in modo appropriato. Il file sorgente può essere scritto in formato ASCII a 7 o 8 bit, il che significa che può essere scritto facilmente.

Il programma di visualizzazione della Guida di AutoCAD formatta il testo nello spazio disponibile in una casella di riepilogo a scorrimento ed inserisce una linea vuota tra i paragrafi. La Guida di AutoCAD utilizza come separatore dei paragrafi ENTER (sequenza CRLF). Il programma di visualizzazione della Guida gestisce semplici operazioni di formattazione quali i ritorni a capo senza interlinea, le tabulazioni ed i paragrafi rientrati rispetto al margine (vedere "Formattazione speciale").

L'esempio riportato di seguito mostra un argomento del file di Guida di AutoCAD incluso nel file *acad.ahp*. Poiché è importante vedere la posizione di ENTER (sequenza CRLF), gli esempi riportati in questa sezione mostrano il carattere ¶ anzichéENTER .

```
\#set_layer_filters_dialog¶
\$(Imposta filtri layer, finestra di dialogo¶
\KDDLMODI;finestre di dialogo;Imposta filtri layer, finestra di dialogo¶
Imposta filtri layer¶
I layer filtrati appaiono nell'elenco Nomi layer.
Impostando i filtri viene selezionata la casella On nell'area
Filtri. Deselezionando On i filtri vengono disattivati e tutti
i layer del disegno vengono elencati.¶
Per accedere alla finestra di dialogo Imposta filtri layer, utilizzare
il comando <<DDLMODI>>ddlmodi>.¶
\#select_color_dialog_box¶
```

L'ID univoco di questo argomento è set_layer_filters_dialog. Il titolo è Imposta filtri layer, finestra di dialogo ed è possibile accedervi mediante le parole chiave: DDLMODI, finestre di dialogo e la finestra di dialogo Imposta filtri layer. Il corpo principale dell'argomento include tre sequenze CRLF che lo formattano in tre paragrafi. Il collegamento contiene un collegamento ipertesto con il comando DDLMODI (vedere "Collegamenti ipertesto"). L'argomento viene chiuso dalla direttiva \#select_color_dialog_box che a sua volta inizia l'argomento successivo. L'illustrazione riportata di seguito mostra il modo in cui il primo argomento viene visualizzato nel programma di visualizzazione della Guida di AutoCAD.



Formattazione speciale

Il programma di visualizzazione della Guida di AutoCAD fornisce i metodi per migliorare la disposizione del testo relativo ad un determinato argomento. Per eliminare lo spazio in più tra i paragrafi, la barra inversa (\) viene interpretata come "ritorno senza interlinea". Le tabulazioni vengono riconosciute ed impostate su una tabulazione ogni quattro posizioni. Per adattare i paragrafi rientrati rispetto al margine, il programma di visualizzazione della Guida riporta automaticamente il testo a capo in base all'ultima tabulazione. È possibile formattare elenchi puntati e numerati; tuttavia, quando l'etichetta iniziale è particolarmente lunga, è necessario adottare una tecnica speciale. Nella parte di file di Guida riportata di seguito vengono mostrate queste funzioni speciali di formattazione. Notare che la freccia verso destra (→) viene usata per rappresentare la tabulazione.

```
\#sample_topic¶
\${Argomento di esempio}¶
\KEsempio;Verifica;Argomento¶
Argomento di esempio (segue spazio)¶
Questo primo esempio\ \xa6 mostra l'uso¶
del "ritorno senza interlinea" (senza spazio prima)¶
→ Questo esempio mostra un paragrafo rientrato.
Poiché inizia con una tabulazione, l'intero paragrafo viene
rientrato rispetto al margine.¶
→ 1. → Questo esempio mostra la tecnica per la
creazione di un elenco numerato, nonché come poter utilizzare questa funzione per
generare un ulteriore rientro rispetto al margine.¶
ETICHETTA LUNGA → Questo esempio mostra la tecnica da usare\ \xa6
→ → → se si desiderano etichette composte da un certo numero
di caratteri. Per ottenere il risultato desiderato, potrebbe essere
necessario effettuare più prove. ¶
\E¶
```

Di seguito viene mostrato come la parte di codice sopra riportata verrebbe visualizzata dal programma di visualizzazione della Guida di AutoCAD.

```
Argomento di esempio (segue spazio)
Questo primo esempio
mostra l'uso del "ritorno senza interlinea" (senza spazio prima)
Questo esempio mostra un paragrafo rientrato.
Poiché inizia con una tabulazione, l'intero paragrafo
viene rientrato rispetto al margine.
1. Questo esempio mostra la tecnica per la creazione di un
elenco numerato e come poter utilizzare questa funzione
per generare un ulteriore rientro rispetto al margine.
ETICHETTA LUNGA Questo esempio mostra la tecnica da
usare se si desiderano etichette
lunghe. Prima di ottenere il risultato
desiderato può essere necessario effettuare
più prove.
```

Collegamenti ipertesto

Tra gli argomenti presenti in uno stesso file di guida possono esistere collegamenti ipertesto. Tali collegamenti possono avere due forme: salto all'argomento o argomento concatenato. Di seguito viene riportata la sintassi:

```
<<hot spot>>topic_id>
<<hot spot>>topic_id]
```

Un segno > (simbolo di maggiore di) dopo topic_id crea un argomento concatenato, ed una parentesi quadra chiusa] dopo topic_id crea un salto sull'argomento. In entrambi i casi, <<hot spot>> appare nel programma di visualizzazione della Guida. Le scritte topic_id> e topic_id] non sono visibili per l'utente. Quando l'utente fa doppio clic su una linea di testo nella casella di riepilogo a scorrimento contenente un singolo collegamento all'argomento <<hot spot>>, il visualizzatore mostra l'argomento associato a tale topic_id. Quando l'utente seleziona un argomento concatenato <<hot spot>>, l'argomento associato a tale topic_id viene visualizzato in una finestra di dialogo sovrapposta al visualizzatore della Guida. Quando si esce da tale finestra di dialogo, l'argomento originale risulterà ancora visualizzato.

Se <<hot spot>> va automaticamente a capo su un'altra linea, è possibile selezionare anche tale linea. Se sulla stessa linea sono presenti più <<hot spot>>, l'utente può selezionare l'argomento appropriato utilizzando i pulsanti Successivo e Vai a.

Capitolo 1 -- Nozioni fondamentali sulla personalizzazione

Personalizzazione della documentazione in linea

File di Guida di AutoCAD

Conversione dei file di Guida R12

È possibile che sia necessario usare con la Release 14 i file di Guida creati per la Release 12. Se si stanno apportando miglioramenti ai file di Guida esistenti, potrebbe essere utile aggiungere ulteriori parole chiave e creare collegamenti ipertestuali. Per iniziare, è possibile eseguire la procedura riportata di seguito in cui sono delineati i passaggi necessari per rendere disponibile un file di Guida di AutoCAD per il programma di visualizzazione della Guida di AutoCAD.

- 1 Rinominare il file con un'estensione *.ahp*.
- 2 Copiare ogni argomento nelle due righe seguenti ed inserire i caratteri di direttiva #, \$ e K. Ad esempio, l'argomento \DDOSNAP deve essere modificato nel seguente modo:

```
\#DDOSNAP
\ $DDOSNAP
\KDDOSNAP
```

Se un argomento presenta più etichette di sezione, usare la prima etichetta come ID argomento, titolo e prima parola chiave ed usare le altre etichette come parole chiave aggiuntive (separate da punto e virgola).

- 3 Terminare con una barra inversa tutte le righe del testo di Guida (non le righe di testo con le direttive) tranne l'ultima di ogni paragrafo. Oppure è possibile eliminare i ritorni a capo da ogni paragrafo del testo di Guida, rendendo ogni paragrafo una lunga riga di testo.
- 4 Inserire la direttiva di fine file (\E) dopo l'ultimo argomento.

Nell'esempio riportato di seguito viene mostrato un file di guida della Release 12 seguito dallo stesso file convertito in base alla procedura sopra descritta in modo da poterlo usare dalla Release 13.

File di Guida della Release 12:

```
\#DDOSNAP
\ESECUZIONE SNAP OGGETTO
```

Il comando DDOSNAP attiva le modalità di snap ad oggetti per tutte per tutte le successive selezioni di punti. Quando la modalità osnap è attiva, in corrispondenza dell'intersezione del cursore grafico viene visualizzata una casella di destinazione; la sua dimensione è determinata dal comando APERTURA.

Seleziona impostazioni

```
-----
In questa area, selezionare le caselle di controllo di snap ad oggetti necessarie.
Dimensioni apertura
```

```
-----
In questa area, è possibile impostare la dimensione della casella di destinazione
(con un destinazione (con un massimo di 20 pixel). Spostando il dispositivo di
scorrimento, la dimensione viene di scorrimento, la dimensione viene illustrata
graficamente nel gruppo di immagini a destra.
```

```
\DDRENAME
```

Il comando DDRENAME consente di cambiare il nome di un blocco, di uno stile di quota, di un layer, di un tipo di linea, di uno stile di testo, di una vista contrassegnata da un nome, del sistema di coordinate utente (UCS) o la configurazione dell'utente. Nella casella di riepilogo Oggetti, selezionare l'oggetto al quale si desidera cambiare il nome. Nella casella di riepilogo Elementi sono visualizzati i nomi di tutti gli oggetti che possono essere rinominati. Per cambiare il nome dell'oggetto, digitare il nome corrente nella casella di modifica Vecchio nome, oppure selezionarlo nella casella di riepilogo Elementi. Digitare il nuovo nome nella casella di modifica Rinomina in e selezionare il pulsante button per aggiornare il nome dell'oggetto nella casella di riepilogo Elementi. Per attivare la modifica, selezionare OK.

File di Guida di AutoCAD convertito:

```
\#DDOSNAP
\ $DDOSNAP
\ KDDOSNAP;ESECUZIONE SNAP OGGETTO
```

Il comando DDOSNAP attiva le modalità di snap ad oggetti per tutte le successive selezioni di punti. Quando la modalità osnap è attiva, in corrispondenza dell'intersezione del cursore grafico viene visualizzata una casella di destinazione; la sua dimensione è determinata dal comando APERTURA.

```
Seleziona impostazioni\
-----\
```

In questa area, selezionare le caselle di controllo di snap ad oggetti necessarie.

```
Dimensioni apertura\
-----\
```

In questa area, è possibile impostare la dimensione della casella di destinazione (con un massimo di 20 pixel). Spostando il dispositivo di scorrimento, la dimensione viene illustrata graficamente nel gruppo di immagini a destra.

```
\#DDRENAME
\ $DDRENAME
\KDDRENAME
```

Il comando DDRENAME consente di cambiare il nome di un blocco, di una quota, di un layer, di un tipo di linea, di uno stile di testo, di una vista contrassegnata da un nome, del sistema di coordinate utente (UCS) o la configurazione della finestra.

Nella casella di riepilogo Oggetti, selezionare l'oggetto al quale si desidera cambiare il nome. Nella casella di riepilogo Elementi sono visualizzati i nomi di tutti gli oggetti che possono essere rinominati. Per cambiare il nome dell'oggetto, digitare il nome corrente nella casella di modifica Vecchio nome, oppure selezionarlo nella casella di riepilogo Elementi. edit box. Digitare il nuovo nome nella casella di modifica Rinomina in e selezionare il pulsante Rinomina in per aggiornare il nome dell'oggetto nella casella di riepilogo Elementi. Per attivare la modifica, selezionare OK.

```
\E
```

Capitolo 2 -- Tipi di linea e modelli di tratteggio

Panoramica

AutoCAD contiene librerie di tipi di linea e di modelli di tratteggio standard. È possibile utilizzare tali tipi e modelli così come vengono forniti, modificarli oppure crearne di nuovi.

Utilizzando i riempimenti PostScript è possibile definire i propri modelli; vedere "fill: definizione dei propri modelli di riempimento PostScript."

Capitolo 2 -- Tipi di linea e modelli di tratteggio

File di definizione dei tipi di linea

I tipi di linea di AutoCAD vengono definiti dai file di definizione dei tipi di linea. Tali file hanno come estensione *.lin*. I tipi di linea di AutoCAD sono composti da una serie di punti e linee separate da spazi e possono contenere oggetti di testo e forme incorporate. Il file di default è *acad.lin*. Questo file può essere stampato per capire meglio come costruire i tipi di linea.

I tipi di linea composti solo da punti, linee e spazi sono considerati tipi di linea *semplici*. I tipi di linea che contengono forme incorporate ed oggetti di testo insieme ai punti, alle linee ed agli spazi, sono tipi di linea *complessi*. Sebbene questi due tipi di linea siano gestiti in modo simile da AutoCAD, le loro definizioni sono notevolmente diverse.

Per creare e modificare le definizioni dei tipi di linea sono disponibili due metodi. È possibile modificare il file *.lin* utilizzando un editor di testo o un elaboratore di testi oppure è possibile utilizzare l'opzione Crea del comando -TLINEA. Non è possibile creare o modificare tipi di linea complessi dalla riga di comando.

Un file di tipi di linea può contenere molte definizioni. È possibile aggiungere tipi di linea personalizzati al file *acad.lin* o iniziare la

composizione di un proprio file della libreria dei tipi di linea. In un file *.lin* è possibile includere commenti. Tutto il testo presente su una linea che inizia con un punto e virgola viene ignorato.

Nel file *.lin*, ogni tipo di linea viene definito su due righe. La prima riga definisce il nome del tipo di linea e consente di specificare una descrizione opzionale.

```
*nome-tipo-di-linea [, descrizione]
```

La riga deve iniziare con un asterisco seguito immediatamente dal nome del tipo di linea. Se viene fornita la descrizione, questa deve essere separata dal nome mediante una virgola e non deve contenere più di 47 caratteri. Tale descrizione non viene utilizzata da AutoCAD, ma è destinata ad essere di ausilio per l'utente per prevedere l'aspetto del tipo di linea.

La seconda riga è il codice che descrive il modello effettivo.

```
allineamento, descmod-1, descmod-2, ...
```

La riga inizia con il codice di allineamento (normalmente è disponibile solo A), seguito da un elenco di descrittori di modello separati da virgole (non si possono utilizzare spazi).

Capitolo 2 -- Tipi di linea e modelli di tratteggio

File di definizione dei tipi di linea

Tipi di linea semplici

È possibile avere un tipo di linea semplice chiamato LP1 con il modello ripetitivo riportato di seguito.

- Lineetta, lunga 0.5 unità di disegno
- Spazio, lungo 0.25 unità di disegno
- Punto
- Spazio, lungo 0.25 unità di disegno

Il tipo di linea verrebbe definito come

```
*LP1, _____ . _____ . _____ . _____ .  
A, .5, -.25, 0, -.25
```

LP1 è il nome del tipo di linea ed il campo *descrizione* è la descrizione del tipo di linea visualizzato dalla sequenza di comandi - LAYER Tipolinea ?. In questo caso, la descrizione è una semplice rappresentazione del modello lineetta-punto.

La descrizione è opzionale e può essere una sequenza di punti, spazi e lineette oppure un commento come "Usare questo tipo di linea per linee nascoste". Se la descrizione viene omessa, non digitare una virgola dopo il nome del tipo di linea. Se invece la descrizione viene inclusa, non deve contenere più di 47 caratteri.

Il campo *allineamento* specifica l'azione per l'allineamento del modello alle estremità di linee, cerchi ed archi singoli. Attualmente, AutoCAD prevede una sola azione per l'allineamento. Specificarla digitando **A**. Tale carattere viene inserito automaticamente nella definizione quando si usa l'opzione Crea del comando -TLINEA; tuttavia, se la definizione del tipo di linea viene creata utilizzando un editor di testo, è necessario inserire il carattere **A** in modo manuale. AutoCAD non accetta altri caratteri in questo campo.

Ogni campo *mod-n* specifica la lunghezza dei segmenti che compongono il tipo di linea. Se la lunghezza è positiva, verrà disegnato un segmento ottenuto con la condizione di penna abbassata. mentre una lunghezza negativa genera uno spazio (penna sollevata). Una lunghezza di lineetta uguale a zero disegna un punto. Per ogni tipo di linea è possibile digitare fino a 12 lunghezze diverse per le lineette, sempre che tali descrizioni rientrino in una riga di 80 caratteri.

Con l'allineamento di tipo A, AutoCAD garantisce che i punti finali delle linee e degli archi inizino e terminino con una lineetta. Ad esempio, si supponga di creare un tipo di linea chiamato CENTRALE che visualizzi la sequenza ripetitiva lineetta-punto comunemente usata come linea centrale. AutoCAD regola tale sequenza sulla base di una singola linea in modo che le lineette coincidano con i punti finali della linea. Il modello viene adattato alla linea in modo che almeno metà della prima specifica di lineetta inizi e termini la linea. Se necessario, la prima e l'ultima lineetta vengono allungate. Se una linea è talmente corta da non poter contenere neanche una sola sequenza lineetta-punto, AutoCAD disegna una linea continua tra le due estremità. Per gli archi, il modello viene anche regolato in modo tale che le lineette vengano disegnate in corrispondenza dei punti finali. I cerchi non hanno punti finali, ma AutoCAD regola la sequenza lineetta-punto in modo da ottenere visualizzazioni accettabili.

Per l'allineamento di tipo A, la lunghezza della prima lineetta deve essere uguale o maggiore di 0 (un punto o un segmento ottenuto con la penna abbassata), mentre la lunghezza della seconda lineetta deve essere minore di 0 (segmento ottenuto con la penna sollevata). Per questo allineamento è necessario avere almeno due specifiche di lineetta. Tra la lineetta iniziale e quella finale

vengono disegnate sequenzialmente le specifiche delle linee del modello, cominciando con la specifica della seconda linea e ricominciando il modello, se necessario, con la specifica della prima linea.

Nota Quando si crea un tipo di linea, questo non viene caricato automaticamente nel disegno. Utilizzare l'opzione Richiama del comando -TLINEA.

Capitolo 2 -- Tipi di linea e modelli di tratteggio

File di definizione dei tipi di linea

Tipi di linea complessi

Le descrizioni dei tipi di linea complessi si trovano nei file *.lin* insieme ai tipi di linea semplici. Un tipo di linea complesso specifica un'unica linea, possibilmente spezzata, che incorpora dei simboli. Questo tipo di linea può indicare utilità, contorni, sagome e così via. Come per i tipi di linea semplici, le linee complesse vengono disegnate dinamicamente non appena l'utente specifica i vertici. Le forme e gli oggetti di testo incorporati nelle linee sono sempre visualizzati completamente e non risultano mai interrotti.

La sintassi dei tipi di linea complessi è simile a quella dei tipi semplici, in quanto è costituita da un elenco di descrittori di modello separati da virgole. I tipi di linea complessi possono includere forme ed oggetti di testo come descrittori di modello, nonché i descrittori linea-punto dei tipi di linea semplici.

La sintassi per i descrittori delle forme e degli oggetti di testo presenti nella descrizione di un tipo di linea è quella riportata di seguito.

forma

```
[nome_forma,nomefile_shx] oppure
[nome_forma,nomefile_shx,trasformazione]
```

text

```
["stringa",nome_stile] oppure
["stringa",nome_stile,trasformazione]
```

dove *trasformazione* è opzionale e può essere qualsiasi serie tra quelle riportate di seguito (ognuna preceduta da una virgola):

R= Rotazione relativa

A= Rotazione assoluta

S= Scala

X= Sfalsamento X

Y= Sfalsamento Y

In questa sintassi, ## è un numero decimale con segno (1, -17, 0.01 e così via), la rotazione è espressa in gradi e le opzioni restanti sono le unità di disegno del tipo di linea messe in scala. Le lettere sopra riportate, riguardanti la trasformazione, se usate, devono essere seguite da un segno di uguale e da un numero.

Capitolo 2 -- Tipi di linea e modelli di tratteggio

File di definizione dei tipi di linea

Tipi di linea complessi

Forme nei tipi di linea complessi

Quanto riportato di seguito definisce un tipo di linea chiamato CON1LINE che è composto dal modello ripetitivo di un segmento di linea, di uno spazio e della forma incorporata CON1 del file *es.shx*. Perché questo esempio sia valido, il file *es.shx* deve trovarsi nel percorso di supporto.

```
*CON1LINE, --- [CON1]--- [CON1] --- [CON1]
-A,1.0,-0.25,[CON1,es.shx],-1.0
```

Il codice, ad eccezione di quello specificato tra parentesi quadre, è simile a quello utilizzato per la definizione dei tipi di linea semplici. Questo esempio mostra la definizione minima per un tipo di linea nella quale è incorporata una forma.

Come descritto precedentemente, per definire una forma come parte di un tipo di linea è possibile utilizzare un totale di sei campi. I primi due sono obbligatori e dipendono dalla loro posizione, gli altri quattro sono opzionali e possono essere ordinati nel modo desiderato. I due esempi riportati di seguito mostrano le varie voci dei campi per la definizione delle forme.

```
[CAP,es.shx,S=2,R=10,X=0.5]
```

Questo codice elabora la forma CAP definita nel file di forma *es.shx* con una scala doppia rispetto alla scala unitaria del tipo di linea, una rotazione tangenziale di 10 gradi in senso orario ed uno sfalsamento X di 0.5 unità di disegno prima dell'elaborazione della forma.

```
[DIP8,pc.shx,X=0.5,Y=1,R=0,S=1]
```

Questo codice elabora la forma DIP8 definita nel file di forma *pc.shx* con uno sfalsamento X di 0.5 unità di disegno prima dell'elaborazione della forma, uno sfalsamento Y di una unità di disegno al di sopra del tipo di linea, con una scala uguale alla scala unitaria del tipo di linea e rotazione 0.

La sintassi riportata di seguito definisce una forma come parte di un tipo di linea complesso.

```
[nome_forma,nomefile_forma,scala,rotazione,sfalsamento_x, sfalsamento_y]
```

Di seguito sono riportate le definizioni dei campi presenti nella sintassi.

nome_forma

Il nome della forma che deve essere elaborata. Questo campo è obbligatorio. Se viene omissso, la definizione del tipo di linea non riesce. Se *nome_forma* non esiste nel file di forma specificato, continuare l'elaborazione del tipo di linea senza la forma incorporata.

nomefile_forma

Il nome del file compilato di definizione della forma (*.shx*). Se viene omissso, la definizione del tipo di linea non riesce. Se *nomefile_forma* non risulta qualificato, cioè non è specificato alcun percorso, il file viene cercato nel percorso della libreria. Se *nomefile_forma* risulta invece interamente qualificato, ma non viene trovato nella posizione specificata, togliere il prefisso e cercare il file nel percorso della libreria. Se non viene trovato, continuare l'elaborazione del tipo di linea senza la forma incorporata.

scala

S=valore. La scala della forma viene usata come fattore di scala per il quale viene moltiplicata la scala della forma definita internamente. Se quest'ultima scala è 0, come scala viene usato solamente *S=valore*.

rotazione

R=valore o *A=valore*. *R=* indica la rotazione relativa o tangenziale rispetto all'elaborazione delle linee. *A=* indica la rotazione assoluta della forma in relazione all'origine. Tutte le forme hanno la stessa rotazione indipendentemente dalla loro posizione. Al valore è possibile aggiungere una *d* per indicare i gradi (se omissso, questo è il valore di default), una *r* per i radianti o una *g* per i gradi centesimali. Se la rotazione viene omisssa, viene usata la rotazione relativa 0.

sfalsamento_x

X=valore. Lo sfalsamento X specifica lo spostamento della forma sull'asse X del tipo di linea calcolato dall'estremità del vertice di definizione del tipo di linea. Se *sfalsamento_x* viene omissso, oppure se il suo valore è 0, la forma viene elaborata senza sfalsamento. Includere questo campo se si desidera avere una linea continua con forme. Questo valore non viene messo in scala dal fattore di scala definito da *S=*.

sfalsamento_y

Y=valore. Questo campo è lo spostamento della forma sull'asse Y del tipo di linea calcolato dall'estremità del vertice di definizione del tipo di linea. Se *sfalsamento_y* viene omissso, oppure se il suo valore è 0, la forma viene elaborata senza sfalsamento. Questo valore non viene messo in scala dal fattore di scala definito da *S=*.

Capitolo 2 -- Tipi di linea e modelli di tratteggio

File di definizione dei tipi di linea

Tipi di linea complessi

Testo nei tipi di linea complessi

I tipi di linea complessi che presentano testo vengono utilizzati principalmente per incorporare testo come forma da elaborare. La differenza principale tra l'uso delle forme e l'uso del testo è che il testo è associato ad uno stile di testo nel disegno mentre le forme vengono associate direttamente ad un file di forma. Lo stile associato con il tipo di linea deve esistere prima di effettuare il caricamento del tipo di linea nel disegno.

Quello che segue è un esempio di definizione di un tipo di linea complesso che include uno stile di testo.

```
*MClinea, --- MC --- MC --- MC
-A,1.0,-0.25,["MC",miostile,S=1,R=0,X=0,Y=-0.25],-1.25
```

Dove MClinea è il nome del tipo di linea e "--- MC --- MC --- MC" è la descrizione ASCII. La sintassi della seconda riga della definizione del tipo di linea è la seguente:

```
["stringa",stile,S=scala,R=rotazione,X=sfalsamento_x,Y=sfalsamento_y]
```

Di seguito sono riportate le definizioni dei campi presenti nella sintassi.

stringa

Il testo da utilizzare nel tipo di linea complesso.

stile

Il nome dello stile di testo da elaborare. Lo stile di testo specificato deve essere incluso. Se viene omissso, viene utilizzato lo stile corrente.

scala

S=*valore*. La scala dello stile viene usata come fattore di scala per il quale l'altezza dello stile viene moltiplicata. Se l'altezza dello stile è 0, come scala viene usato solamente S=*valore*.

Poiché l'altezza finale del testo viene definita sia da S=*valore* che dall'altezza assegnata allo stile di testo, se quest'ultima viene impostata su zero si otterranno risultati più prevedibili. Si consiglia inoltre di creare stili di testo distinti per il testo nei tipi di linea complessi in modo da evitare conflitti con altro testo nel disegno.

rotazione

R=*valore* o A=*valore*. R= indica la rotazione relativa o tangenziale rispetto all'elaborazione delle linee. A= indica la rotazione assoluta del testo in relazione all'origine. Tutto il testo ha la stessa rotazione, indipendentemente dalla sua posizione relativa alla linea. Al valore è possibile aggiungere una d per indicare i gradi (se omissso, questo è il valore di default), una r per i radianti o una g per i gradi centesimali. Se la rotazione viene omisssa, viene usata la rotazione relativa 0.

La rotazione è centrata tra la linea di base e la casella altezze estremità nominali.

sfalsamento_x

X=*valore*.Lo sfalsamento X specifica lo spostamento del testo nell'asse X del tipo di linea calcolato dall'estremità del vertice di definizione del tipo di linea. Se *sfalsamento_y* viene omissso, oppure se il suo valore è 0, il testo viene elaborato usando come sfalsamento l'angolo in basso a sinistra. Includere questo campo se si desidera avere una linea continua contenente testo. Questo valore *non* viene messo in scala dal fattore di scala definito da S=.

sfalsamento_y

Y=*valore*.Lo sfalsamento Y è lo spostamento del testo nell'asse Y del tipo di linea calcolato dall'estremità del vertice di definizione del tipo di linea. Se *sfalsamento_y* viene omissso, oppure se il suo valore è 0, il testo viene elaborato usando come sfalsamento il baricentro. Questo valore *non* viene messo in scala dal fattore di scala definito da S=.

Capitolo 2 -- Tipi di linea e modelli di tratteggio

Creazione di modelli di tratteggio

Lo sviluppo delle definizioni dei modelli di tratteggio per AutoCAD richiede conoscenza, pratica, pazienza oltre ad un editor di testo. È possibile aggiungere un modello al file della libreria *acad.pat*, oppure memorizzare il modello da solo in un file, nel qual caso il nome del file dovrà essere uguale al nome del modello. Ad esempio, il nome del file di un modello chiamato PIT sarà *pit.pat*.

Indipendentemente da dove la definizione viene memorizzata, il suo formato è sempre lo stesso. L'intestazione sarà simile a quella riportata di seguito.

```
*nome-modello [, descrizione]
```

Il modello contiene anche uno o più descrittori di linea del tipo riportato di seguito.

```
angolo, origine-x, origine-y, delta-x, delta-y [, lineetta-1, lineetta-2, ...]
```

AutoCAD ignora le righe vuote ed il testo a destra del punto e virgola.

Ad esempio, un modello chiamato L45 che esegue i tratteggi con linee inclinate di 45 gradi separate da uno spazio di 0.5 unità di disegno verrebbe definito nel modo riportato di seguito:

```
*L45, linee a 45 gradi
45, 0,0, 0,0.5
```

Questo modello semplice specifica che deve essere disegnata una linea con un angolo di 45 gradi, che la prima linea del tratteggio deve passare attraverso l'origine del disegno (0,0) e che lo spazio tra le linee del tratteggio deve essere di 0.5 unità di disegno. L45 è il nome del modello ed il campo della descrizione è la descrizione opzionale del modello visualizzata dal comando RETINO ?. Se la descrizione viene omessa, *non* specificare alcuna virgola dopo il nome del modello. Ogni riga del file di definizione del modello può contenere fino a 80 caratteri.

Capitolo 2 -- Tipi di linea e modelli di tratteggio

Creazione di modelli di tratteggio

File dei modelli di AutoCAD: *acad.pat*

I modelli di tratteggio di default utilizzati da AutoCAD sono memorizzati nel file *acad.pat*. È possibile aggiungere definizioni di modelli a questo file oppure creare file propri, come descritto precedentemente.

Nelle finestre di dialogo Tratteggio e Tavolozza dei modelli di tratteggio sono visualizzati i nomi di tutti i modelli di tratteggio definiti nel file *acad.pat*. È possibile aggiungere a tali finestre di dialogo nuovi modelli di tratteggio aggiungendo le loro definizioni al file *acad.pat*. Le diapositive il cui nome nel file *acad.slb* corrisponde al nome di un modello nel file *acad.pat* verranno visualizzate nelle finestre di dialogo. Per ulteriori informazioni, vedere PTRATT nella *Guida di riferimento dei comandi di AutoCAD*.

Capitolo 2 -- Tipi di linea e modelli di tratteggio

Creazione di modelli di tratteggio

Come sono costruiti i modelli di tratteggio

Un modello è composto da una o più *linee di modello* (AutoCAD non impone limiti per il numero di linee). Ogni linea viene considerata il primo membro di una *famiglia di linee* creata applicando sfalsamenti delta in entrambe le direzioni per generare una famiglia infinita di linee parallele. Il valore *delta-y* fornisce la spaziatura tra i membri della famiglia (cioè, la spaziatura viene misurata perpendicolarmente rispetto alle linee). *Delta-x* fornisce l'intervallo tra i membri della famiglia nella direzione della linea, chiaramente solo nel caso di linee tratteggiate. La lunghezza di una linea viene considerata infinita; un modello tratteggiato viene sovrapposto alla linea.

Il processo di tratteggio consiste nell'espandere ciascuna linea presente nella definizione del modello per formare la sua famiglia infinita di linee parallele. Tutti gli oggetti selezionati vengono controllati per verificare l'esistenza di eventuali intersezioni con una qualsiasi di queste linee, in quanto le intersezioni causano l'attivazione o la disattivazione delle linee del tratteggio, a seconda dello

stile di tratteggio. Se la linea del tratteggio è a sua volta tratteggiata, viene disegnata con il modello di tratteggio usato nelle rispettive aree.

Poiché ogni famiglia di linee del tratteggio viene generata da una trasposizione parallela da una linea iniziale con un'origine assoluta, il tratteggio delle aree adiacenti risulterà sicuramente allineato in modo corretto.

Capitolo 2 -- Tipi di linea e modelli di tratteggio

Creazione di modelli di tratteggio

Modelli con linee tratteggiate

Per definire modelli di linee tratteggiate, aggiungere alla fine dell'elemento di definizione della linea elementi aventi la lunghezza delle linee. Ogni elemento avente la lunghezza della lineetta specifica la lunghezza di un segmento che compone la linea. Se la lunghezza è positiva, verrà disegnato un segmento ottenuto con la condizione di penna abbassata. Se invece la lunghezza è negativa, il segmento viene ottenuto con la condizione di penna sollevata, di conseguenza non verrà disegnato. Il modello inizia in corrispondenza del punto di origine con il primo segmento e scorre tra i vari segmenti con direzione circolare. Una lunghezza di lineetta uguale a zero disegna un punto. Per ogni linea del modello è possibile specificare sino a 6 lunghezze di lineetta.

Ad esempio, modificare un modello per linee inclinate di 45 gradi per disegnare linee tratteggiate con una lunghezza di lineetta di 0.5 unità ed una spaziatura fra i tratti di 0.5 unità. Tale modello verrebbe definito nel modo riportato di seguito.

```
*LINEETTA45,Linee tratteggiate a 45 gradi
45, 0,0, 0,.5, .5,-.5
```

Questa definizione è uguale a quella del modello originario con inclinazione a 45 gradi, ma include una lineetta finale. La lunghezza della lineetta ottenuta con la condizione di penna abbassata è di 0.5 unità, come quella ottenuta con la condizione di penna alzata, sempre di 0.5 unità di disegno, raggiungendo così gli obiettivi prefissati. Se si desiderava disegnare una lineetta lunga 0.5 unità, uno spazio di 0.25 unità, un punto, uno spazio di 0.25 unità prima della lineetta successiva, la definizione sarebbe stata quella riportata di seguito.

```
*LPUNTO45,Motivo lineetta punto lineetta: 45 gradi
45, 0,0, 0,.5, .5,-.25,0,-.25
```

Adesso, considerare l'effetto delle specifiche delta-x sulle famiglie di linee tratteggiate. Innanzi tutto, considerare la definizione riportata di seguito.

```
*LSERIE
0, 0,0, 0,.5, .5,-.5
```

Questa definizione disegna una famiglia di linee separate da 0.5 unità di disegno, con ogni linea suddivisa equamente in linee e spazi. Poiché il valore di delta-x è zero, le linee di ogni membro della famiglia risulteranno allineate. Un'area tratteggiata con questo modello apparirà simile a quella riportata di seguito.

```
- - - - -
- - - - -
- - - - -
```

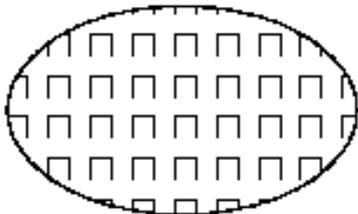
Adesso, cambiare il modello in

```
*SFALSATO
0, 0,0, .5,.5, .5,-.5
```

La definizione è uguale alla precedente, con l'eccezione che delta_x è stato impostato su 0.5. In questo modo, ogni membro della famiglia successiva viene sfalsato di 0.5 unità di disegno nella direzione della linea (in questo caso, parallelamente all'asse X). Poiché le linee sono infinite, il modello viene spostato verso il basso della quantità specificata. L'area tratteggiata sarà come quella riportata di seguito.

```
- - - - -
- - - - -
- - - - -
```

Tutti i modelli descritti fino ad ora utilizzano i punti di origine (0,0); in questo modo, un membro della famiglia di linee passa attraverso l'origine, con il relativo modello tratteggiato che inizia in tal punto. Quando si compongono modelli più complessi, è necessario specificare attentamente il punto iniziale, gli sfalsamenti ed i modelli tratteggiate di ogni famiglia di linee per formare correttamente il modello di tratteggio. Considerare quanto riportato di seguito: si supponga di voler disegnare un modello composto da U squadrate capovolte (una linea verso l'alto, una orizzontale ed una verso il basso). Nella figura riportata di seguito, nel modello viene ripetuta ogni singola unità, ciascuna delle quali ha un'altezza ed una larghezza di 0.5.



Il modello verrebbe definito nel modo riportato di seguito.

```
*UCAP,U capovolte
90, 0,0, 0,1, .5,-.5
0, 0,.5, 0,1, .5,-.5
270, .5,.5, 0,1, .5,-.5
```

La prima linea, quella verso l'alto, è semplicemente una linea tratteggiata con origine (0,0). La seconda linea, quella superiore, deve iniziare alla fine della linea verso l'alto, quindi la sua origine è (0,.5). La terza linea, quella verso il basso, deve iniziare alla fine di quella superiore, al punto (.5,.5) per la prima ricorrenza del modello, quindi la sua origine si trova in tale punto. La specifica della terza linea del modello poteva anche essere quella riportata di seguito.

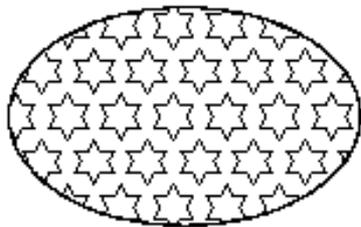
```
90, .5,0, 0,1, .5,-.5
```

oppure

```
270, .5,1, 0,1, -.5,.5
```

Il modello tratteggiato inizia al punto di origine e continua nella direzione del vettore data dalla specifica dell'angolo. Quindi, due famiglie di linee tratteggiate opposte di 180 gradi *non* sono uguali, mentre due famiglie di linee piene sono uguali.

Considerare il modello di stelle a sei punte riportato di seguito.



Questo esempio può essere di ausilio per migliorare la propria abilità nella definizione dei modelli (suggerimento: 0.866 è il seno di 60 gradi). Di seguito viene riportata la definizione di AutoCAD per tale modello.

```
*STELLE,Stella di David
0, 0,0, 0,.866, .5,-.5
60, 0,0, 0,.866, .5,-.5
120, .25,.433, 0,.866, .5,-.5
```

Capitolo 3 -- Forme, font e supporto PostScript

Panoramica

Le forme sono oggetti che possono essere utilizzati in maniera simile ai blocchi. Per utilizzare una forma, è necessario dapprima caricare su disco, mediante il comando CARICA, un file compilato in cui è contenuta la definizione della forma. Quindi, utilizzare il comando FORMA per inserire nel disegno le forme presenti in tale file. È possibile specificare il valore di scala e rotazione da utilizzare per ogni forma nel momento in cui la si aggiunge al disegno. I font di testo e le forme di AutoCAD vengono definiti nello stesso modo.

I blocchi risultano più versatili e semplici da usare e da applicare rispetto alle forme. Tuttavia, AutoCAD riesce a memorizzare e disegnare le forme in modo più efficace. Le forme definite dall'utente sono utili quando si deve inserire più volte un semplice componente del disegno e quando si desidera rendere più veloci le operazioni.

In questo capitolo viene descritto come creare i propri file di forma e di font per disegnare simboli e font di testo. Vengono illustrate inoltre le caratteristiche avanzate PostScript di AutoCAD.

Capitolo 3 -- Forme, font e supporto PostScript

Compilazione dei file di forma e di font

Le forme possono essere descritte in un file di testo formattato in modo particolare con estensione *.shp*. Per creare un file di questo tipo, utilizzare un editor di testo o un elaboratore di testi che consenta di salvare il file in formato ASCII e poi compilare il file ASCII stesso. Mediante la compilazione, i file di descrizione delle forme vengono convertiti in un formato accettato dal comando CARICA o STILE.

Per compilare un file di forma o di font, digitare:

Comando: **compila**

AutoCAD visualizza la finestra di dialogo Seleziona il file di forma o di font dalla quale è possibile selezionare un file di definizione delle forme (*.shp*) oppure un file di font PostScript (*.pfb*). La compilazione ha inizio dopo aver selezionato il nome del file. Se viene rilevato un errore nelle descrizioni delle forme, viene visualizzato un messaggio indicante il tipo di errore ed il numero di riga. Al termine della compilazione, vengono visualizzati i seguenteimessaggi:

Compilazione riuscita.

Il file di output *nome..shx* contiene *nnn* byte.

Il file compilato ha lo stesso nome del file di definizione delle forme, ma con l'estensione *.shx*. Se il file di definizione delle forme definisce un font, occorre utilizzare il comando STILE per definire lo stile di testo e quindi uno dei comandi per il posizionamento del testo, ad esempio (TESTO, TESTODIN o TESTOM), per posizionare i caratteri nel disegno. Se nel file di definizione vengono descritte le forme, è possibile usare il comando CARICA per caricare il file delle forme nel disegno e quindi il comando FORMA per posizionare le singole forme nel disegno; tale procedura è concettualmente simile al comando INSER.

Capitolo 3 -- Forme, font e supporto PostScript

Compilazione dei file di forma e di font

Compilazione dei font PostScript

Per utilizzare un font PostScript Type 1 in AutoCAD, è necessario dapprima compilarlo in un file di forma di AutoCAD. Il comando COMPILA accetta come input sia i file *.shp* che i file *.pfb* e genera un file *.shx*. Poiché le versioni compilate dei font PostScript occupano molto spazio sul disco, è opportuno compilare solo i font usati frequentemente.

Non si ha alcuna garanzia che AutoCAD possa compilare e caricare ogni tipo di font Type 1. Le funzioni dei font PostScript in AutoCAD sono state progettate per l'elaborazione di un sottogruppo di font Adobe. Se durante la compilazione di un font PostScript si verifica un errore, è possibile che il file *.shx*, se generato, non venga caricato in AutoCAD.

Per ulteriori informazioni sul formato di font Adobe Type 1, vedere la pubblicazione *Adobe Type 1 Font Format Version 1.1*. Dopo aver acquistato ed installato questi font, è possibile utilizzarli con AutoCAD.

Nota Assicurarsi di comprendere tutti i copyright relativi ai font PostScript che vengono utilizzati. Le stesse restrizioni di copyright riguardano generalmente il formato *.shx* dei font compilati dall'utente.

Capitolo 3 -- Forme, font e supporto PostScript

File di definizione delle forme

I file di font e di forme di AutoCAD (*.shx*) vengono generati (compilati) dai file di definizione delle forme (*.shp*), che possono essere creati o modificati con un editor di testo o un elaboratore di testi che ne esegua il salvataggio in formato ASCII.

La sintassi della descrizione delle forme è la stessa per ogni forma (o carattere), indipendentemente dall'utilizzo finale (forma o font) di quella determinata descrizione di forma. Se un file di definizione delle forme deve essere usato come un file di font, la prima immissione nel file descrive il font stesso invece che una forma all'interno del file. Se questa immissione iniziale descrive una forma,

il file viene usato come un file delle forme.

AutoCAD viene fornito con due file di forma di esempio: *pc.shx* ed *es.shx*. Il primo viene utilizzato per il layout di circuiti stampati, mentre il secondo per gli schemi di elettronica. Per creare facilmente la definizione delle forme di AutoCAD, si consiglia di prendere visione del contenuto di questi file e di modificare le descrizioni delle forme.

Capitolo 3 -- Forme, font e supporto PostScript

File di definizione delle forme

Descrizione di forme

Ogni riga presente in un file di definizione delle forme può contenere al massimo 128 caratteri. Non è possibile compilare righe più lunghe. AutoCAD ignora le righe vuote ed il testo a destra del punto e virgola. Tale segno consente di incorporare commenti nel file di definizione delle forme.

Ogni descrizione di forma è inclusa una riga di intestazione del tipo riportato di seguito ed è seguita da una o più righe contenenti byte di specifica separati da virgole e con uno 0 finale.

```
*numero_forma,def_bytes,nome_forma
byte_spec1,byte_spec2,byte_spec3,...,0
```

Di seguito vengono descritti i campi relativi alla descrizione di una forma:

numero_forma

Indica un numero univoco per il file, compreso tra 1 e 258 e preceduto da un asterisco (*). Ad ogni forma inclusa in un file di forma deve essere associato un numero (i numeri 256, 257 e 258 si riferiscono agli identificativi simbolici Degree_Sign, Plus_Or_Minus_Sign e Diameter_Symbol). I font di testo, ossia i file che contengono le definizioni delle forme per ogni carattere, richiedono numeri specifici che corrispondono al valore del carattere rappresentato nel codice ASCII; alle altre forme è possibile assegnare un numero qualsiasi.

byte_def

Indica il numero di byte di dati (*byte_spec*) necessari per descrivere la forma, compreso lo zero finale. Il limite massimo è di 2000 byte per forma.

nome_forma

Indica il nome della forma e, per essere riconosciuto, deve essere specificato in lettere maiuscole. Un nome contenente caratteri minuscoli viene ignorato, ma utilizzato di solito per etichettare le definizioni di forma dei font.

byte_spec

Indica il byte di specifica della forma. Ogni byte di specifica rappresenta un codice che definisce la lunghezza e la direzione del vettore oppure uno dei codici speciali. Un byte di specifica può essere espresso nel file di definizione delle forme sia come valore decimale che esadecimale. Negli esempi di questa sezione vengono utilizzati entrambi i valori del byte di specifica, come si verifica per la maggior parte dei file di definizione delle forme. Se il primo carattere di un byte di specifica è 0 (zero), i due caratteri successivi vengono interpretati come valori esadecimali.

Capitolo 3 -- Forme, font e supporto PostScript

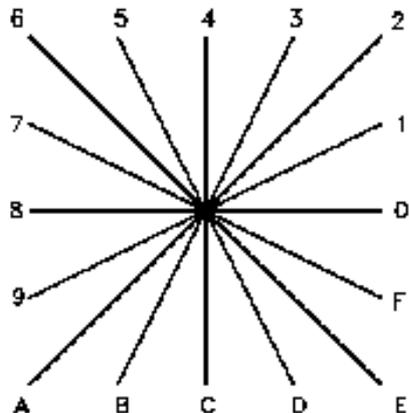
File di definizione delle forme

Descrizione di forme

Codice di lunghezza e di direzione dei vettori

Il byte di specifica della forma contiene la lunghezza e la direzione del vettore codificata in un unico byte (un unico campo *byte_spec*). Ogni codice relativo alla lunghezza ed alla direzione del vettore è una stringa composta da tre caratteri. Il primo carattere deve essere uno 0, che indica ad AutoCAD che i due caratteri seguenti vengono interpretati come valori esadecimali. Il secondo

carattere indica la lunghezza del vettore espressa in unità. I valori esadecimali validi sono compresi tra 1 ed F corrispondenti, ovvero da una a quindici unità di lunghezza. Il terzo carattere specifica la direzione del vettore. Nella figura riportata di seguito vengono illustrati i codici di direzione.



Codici di direzione del vettore

Tutti i vettori riportati nella figura precedente sono stati disegnati specificando la stessa lunghezza. I vettori diagonali si estendono in modo da coincidere con lo spostamento X o Y del vettore ortogonale più vicino. Questa procedura è simile all'azione della griglia snap di AutoCAD.

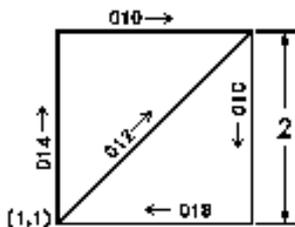
Nell'esempio seguente viene costruita una forma definita DBOX, a cui è stato assegnato il numero di forma arbitrario 230.

```
*230,6,DBOX
014,010,01C,018,012,0
```

La sequenza di byte di specifica precedente definisce un quadrato di altezza e larghezza pari ad una unità, con una diagonale che congiunge l'angolo inferiore sinistro con l'angolo superiore destro. Dopo aver salvato i file con il nome *dbox.shp*, utilizzare il comando COMPILA per generare il file *dbox.shx*. Caricare il file di forma in cui è contenuta questa definizione mediante il comando CARICA, quindi utilizzare il comando FORMA nel modo seguente:

- Comando: **forma**
- Nome della forma (o ?): **dbox**
- Punto iniziale: **1,1**
- Altezza <corrente>: **2**
- Angolo di rotazione <corrente>: **0**

La figura di seguito riportata mostra la forma risultante.



Capitolo 3 -- Forme, font e supporto PostScript

-  **File di definizione delle forme**
-  **Descrizione di forme**
-  **Codici speciali**

I codici speciali di seguito riportati possono essere utilizzati da un byte di specifica, oltre che per definire vettori, anche per creare

forme supplementari e per specificare determinate azioni da intraprendere. Per utilizzare un codice speciale, il secondo carattere della stringa composta da tre caratteri, che indica la lunghezza del vettore, deve essere uguale a 0; in alternativa è possibile specificare soltanto il numero di codice (ad esempio, 008 e 8 sono entrambi valori di specifica validi).

Codici dei byte di specifica

Codice	Descrizione
000	Definizione fine della forma.
001	Attiva la modalità di disegno (penna abbassata).
002	Disattiva la modalità di disegno (penna sollevata).
003	Divide le lunghezze del vettore per il byte successivo.
004	Moltiplica le lunghezze del vettore per il byte successivo.
005	Salva la posizione corrente nella pila.
006	Ripristina la posizione corrente nella pila.
007	Disegna la forma secondaria con il numero fornito dal byte successivo.
008	Indica lo spostamento X-Y fornito da due byte successivi.
009	Indica più spostamenti X-Y che terminano con (0,0).
00A	Indica l'arco ottante definito dai due byte successivi.
00B	Indica l'arco frazionario definito dai cinque byte successivi.
00C	Indica l'arco definito dallo spostamento X-Y e dalla curvatura.
00D	Indica più archi specificati da curvaturei.
00E	Elabora il comando successivo soltanto se il testo è verticale.

Codice 0: fine della forma

Il codice 0 contrassegna la fine della definizione della forma.

Codici 1 e 2: controllo della modalità di disegno

I codici 1 e 2 controllano la modalità di disegno, che viene attivata all'inizio di ogni forma. Se questa modalità è attiva, i vettori consentono il disegno di linee, altrimenti i vettori si spostano su una nuova posizione senza effettuare l'operazione di disegno.

Codici 3 e 4: controllo della dimensione

I codici 3 e 4 controllano la dimensione relativa di ogni vettore. L'altezza specificata mediante il comando FORMA viene inizialmente considerata come lunghezza di un singolo vettore ortogonale (direzione 0, 4, 8 o C). I codici 3 e 4 vengono seguiti da un byte di specifica che contiene un fattore di scala intero compreso tra 1 e 255. Se si desidera che l'altezza della forma indichi la dimensione della forma intera e si utilizzano la lunghezza di vettore 10 per disegnarla, è possibile specificare 3,10 per determinare la scala dell'altezza. All'interno di una forma il fattore di scala è cumulativo; vale a dire, se si moltiplica per 2 e poi per 6 il risultato sarà un fattore di scala pari a 12. Generalmente si desidera annullare l'effetto dei fattori di scala alla fine della forma, in particolar modo per le forme secondarie e per quelle relative al font di testo; tuttavia, AutoCAD non ripristina automaticamente tale fattore.

Codici 5 e 6: salvataggio/ripristino della posizione

I codici 5 e 6 salvano e ripristinano la posizione della coordinata corrente mentre si disegna una forma in modo che sia possibile tornare in tale posizione successivamente. È possibile ripristinare qualsiasi forma salvata. La capacità della pila è di quattro posizioni soltanto. Se si supera tale capacità poiché si eseguono troppi salvataggi o non si effettuano sufficienti ripristini, quando si disegna la forma viene visualizzato il messaggio seguente.

Overflow della pila di posizione nella forma nnn

In modo analogo, se si tenta di ripristinare un numero maggiore di posizioni rispetto a quelle salvate nella pila, quando si disegna la forma viene visualizzato il messaggio seguente.

Underflow della pila di posizione nella forma nnn

Codice 7: forma secondaria

Il codice 7 rappresenta il riferimento di una forma secondaria. Il byte di specifica che segue questo codice è un numero di forma compreso tra 1 e 255. A questo punto, viene disegnata la forma con il numero specificato, presente nello stesso file di forma. Tenere presente che la modalità di disegno non viene ripristinata per la nuova forma. Una volta completata la forma secondaria, riprende il disegno della forma corrente.

Codici 8 e 9: spostamenti X-Y

I byte di specifica dei vettori delle normali disegnano soltanto nelle 16 direzioni predefinite e la lunghezza massima è pari a 15. Tali restrizioni contribuiscono a rendere più efficienti le definizioni delle forme, ma talvolta rappresentano un limite. Mediante i codici 8 e 9 è possibile disegnare vettori non standard utilizzando gli spostamenti X, Y. Il codice 8 deve essere seguito da due byte di specifica nel formato seguente:

8, spostamento_X, spostamento_Y

Gli spostamenti X, Y possono essere compresi tra -128 e +127. Il segno iniziale + è opzionale ed è possibile utilizzare le parentesi

per migliorare la leggibilità. Nell'esempio seguente viene mostrato un vettore che disegna o sposta 10 unità a sinistra e tre unità verso l'alto.

8, (-10, 3)

Dopo aver specificato i due byte di spostamento, la forma torna nella modalità Vettore normale.

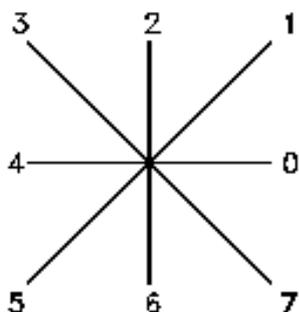
Il codice 9 può essere utilizzato per disegnare una sequenza di vettori non standard. Questo codice può essere seguito da un numero qualsiasi di coppie di spostamenti X, Y e deve terminare con una coppia (0,0). Nell'esempio seguente vengono disegnati tre vettori non standard e riattivata quindi la modalità Vettore normale.

9, (3, 1), (3, 2), (2, -3), (0, 0)

Tenere presente che la sequenza di coppie di spostamento X, Y deve terminare con la coppia (0,0), affinché AutoCAD possa riconoscere i vettori perpendicolari o il codice speciale successivo.

Codice 00A: arco ottante

Il codice speciale 00A (o 10) utilizza i due byte di specifica successivi per definire un arco. Questo arco viene chiamato *arco ottante* in quanto racchiude uno o più *ottanti* di 45 gradi ed inizia e termina alle estremità di un ottante. Gli ottanti vengono numerati in senso antiorario a partire dalla posizione 3 dell'orologio, come illustrato nella figura riportata di seguito.



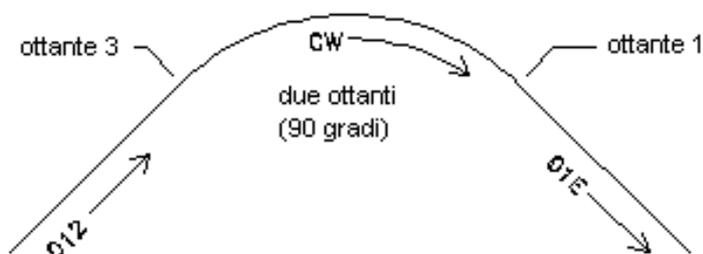
La specifica dell'arco è

10, raggio, (-) 0SC

Il valore del raggio può essere compreso tra 1 e 255. Il secondo byte di specifica indica la direzione dell'arco (in senso antiorario se positiva, in senso orario se negativa), il relativo ottante iniziale (S: un valore compreso tra 0 e 7) ed il numero di ottanti che attraversa (C: un valore compreso tra 0 e 7, in cui 0 significa otto ottanti, cioè un cerchio completo). Per migliorare la leggibilità è possibile utilizzare le parentesi. Si consideri ad esempio la seguente parte di definizione di una forma:

...012, 10, (1, -032), 01E, ...

Questo codice disegna un vettore di una unità verso l'alto e verso destra, un arco in senso orario dall'ottante 3 (con un raggio pari ad una unità per due ottanti) ed un vettore di una unità verso il basso e verso destra, come mostrato nella figura seguente.



Codice 00B: arco frazionario

Il codice speciale 00B (11) disegna un arco che non deve necessariamente iniziare e terminare alle estremità di un ottante. La definizione utilizza cinque byte di specifica.

11, sfalsamento_iniziale, sfalsamento_finale, raggio_alto, raggio, (-) 0SC

Lo *sfalsamento_iniziale* e lo *sfalsamento_finale* rappresentano le distanze fra le estremità dell'arco ed il contorno di un ottante. Il *raggio_alto* rappresenta gli otto bit più significativi del raggio e sarà uguale a 0 a meno che il *raggio* non sia superiore a 255 unità. Per creare il raggio di un arco maggiore di 255 unità, moltiplicare per 256 il valore di *raggio_alto* ed aggiungere il risultato al valore del *raggio*. Il *raggio* ed il byte di specifica finale sono identici a quelli della specifica dell'arco ottante, ossia il codice 00A descritto precedentemente.

Lo sfalsamento iniziale viene determinato calcolando la differenza in gradi tra il limite dell'ottante iniziale (un multiplo di 45 gradi) e l'inizio dell'arco. Quindi è necessario moltiplicare la differenza ottenuta per 256 e dividerla per 45. Se l'arco inizia dal limite di un ottante, lo sfalsamento iniziale è 0.

Lo sfalsamento finale viene calcolato in modo simile, utilizzando però il numero di gradi a partire dal punto finale dell'ottante attraversato fino alla fine dell'arco. Se l'arco termina sul limite di un ottante, lo sfalsamento finale è 0.

Ad esempio, un arco frazionario compreso tra 55 gradi e 95 gradi con un raggio di 3 unità verrebbe codificato nel seguente modo:

```
11, (56,28,0,3,012)
```

Di seguito è riportata la spiegazione dell'esempio sopra citato:

```
sfalsamento_iniziale    = 56 in quanto ((55 - 45) * 256 / 45) = 56
sfalsamento_finale     = 28 in quanto ((95 - 90) * 256 / 45) = 28
raggio_alto             = 0 in quanto (raggio < 255)
raggio                  = 3
ottante_iniziale       = 1 in quanto l'arco inizia dall'ottante a 45 gradi
ottante_finale         = 2 in quanto l'arco termina sull'ottante a 90 gradi
```

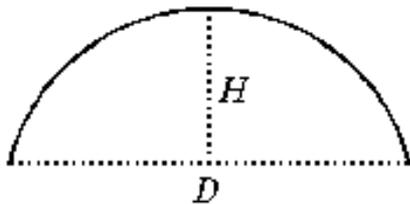
Codici 00C e 00D: archi specificati da curvature

I codici speciali 00C e 00D (12 e 13) offrono un altro mezzo per includere segmenti di arco nelle descrizioni delle forme. Essi sono simili ai codici 8 e 9 in quanto possono essere utilizzati per specificare gli spostamenti X , Y . Tuttavia, i codici 00C e 00D consentono di disegnare gli archi applicando un *fattore di curvatura* al vettore di spostamento. Il codice 00C disegna un segmento di arco, mentre il codice 00D traccia più segmenti di arco (*poliarco*) fino a quando non viene terminato da uno spostamento (0,0).

Il codice 00C deve essere seguito da tre byte che descrivono l'arco:

```
0C, spostamento_X, spostamento_Y, curvatura
```

Gli spostamenti X ed Y e la curvatura dell'arco possono essere compresi tra -127 e +127. Se il segmento di linea specificato dallo spostamento è uguale a D e l'altezza perpendicolare dal punto medio di tale segmento è H , l'ampiezza della curvatura sarà $((2 * H / D) * 127)$. Se l'arco viene tracciato in senso orario dalla posizione corrente alla nuova posizione, il segno sarà negativo.



Un semicerchio ha una curvatura di 127 o -127 ed è l'arco più grande che può essere rappresentato come segmento di arco singolo usando questi codici; per archi maggiori, usare due segmenti di arco consecutivi. Una curvatura uguale a 0 è valida e rappresenta un segmento di linea retta. È necessario tuttavia tenere presente che se si utilizza il codice 8 per un segmento di linea retta si risparmia un byte nella descrizione della forma.

Il codice di poliarco 00D o 13 è seguito da 0 oppure da più segmenti di arco e termina con uno spostamento (0,0). Tenere presente che dopo lo spostamento finale non viene specificata alcuna curvatura. Ad esempio, la lettera S potrebbe essere definita dalla sequenza seguente:

```
13, (0,5,127), (0,5,-127), (0,0)
```

I segmenti con curvatura uguale a zero risultano utili all'interno di poliarchi per rappresentare segmenti retti; terminare un poliarco, inserire un segmento retto ed iniziare un altro poliarco sono infatti operazioni meno efficaci.

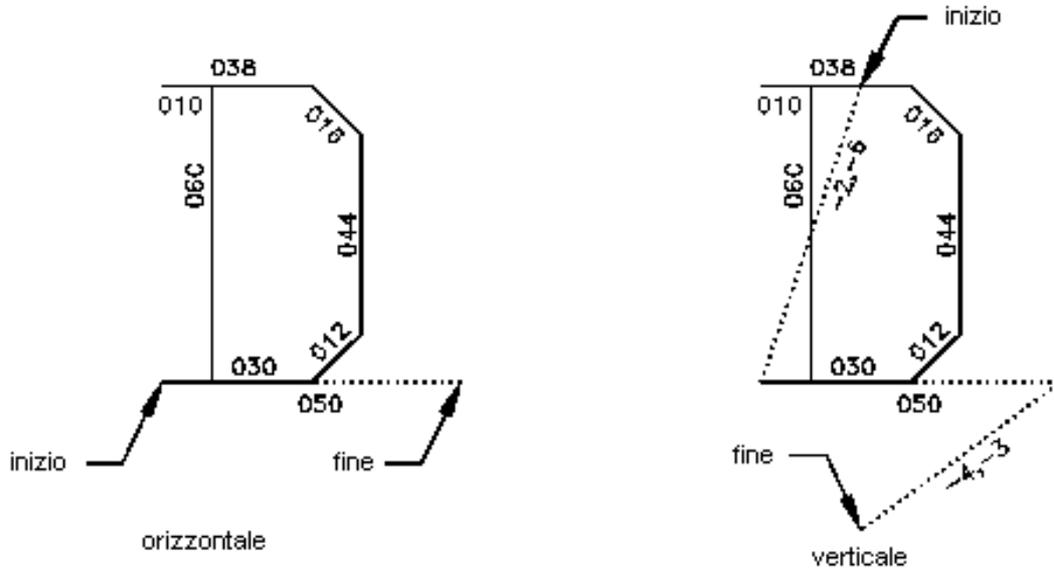
Non è possibile utilizzare il numero -128 nelle definizioni di poliarchi e segmenti di arco.

Codice 00E: comando per il flag relativo al testo verticale

Il codice speciale 00E (14) viene utilizzato soltanto nelle descrizioni di font che possono avere un orientamento orizzontale e verticale. Quando si rileva la presenza di questo codice in una definizione di caratteri, il codice successivo viene elaborato od omesso a seconda dell'orientamento. Se l'orientamento è verticale, il codice successivo viene elaborato; se l'orientamento è orizzontale, tale codice viene omesso.

Nel testo orizzontale il punto iniziale di ogni carattere è rappresentato dall'estremità sinistra della linea di base. Nel testo verticale si presume invece che il punto iniziale sia il punto centrale superiore del carattere. Alla fine di ogni carattere viene di solito tracciato un segmento verso l'alto per raggiungere il punto iniziale del carattere successivo. Per il testo orizzontale la direzione è verso destra, per il testo verticale è verso il basso. Il codice speciale 00E (14) viene usato principalmente per regolare le differenze tra i punti iniziali e quelli finali e consente quindi di utilizzare la stessa definizione di forma di carattere per l'orientamento orizzontale e verticale. Ad esempio, la definizione di seguito riportata relativa alla D maiuscola può essere utilizzata in testi orizzontali o in testi verticali.

```
*68,22,ucd
2,14,8,(-2, 6),1,030,012,044,016,038,2,010,1,06C,2,050,
14,8,(-4,-3),0
```



Capitolo 3 -- Forme, font e supporto PostScript

File di definizione delle forme

Descrizione dei font di testo

AutoCAD dispone di numerosi font di testo. Il comando STILE può essere utilizzato per espandere, comprimere o inclinare questi font in modo da ottenere i caratteri desiderati. Usando questi font è possibile disegnare testi di qualsiasi altezza, con qualsiasi angolazione rispetto alla base della linea e con orientamento orizzontale o verticale.

Se si desidera creare font di testo personalizzati, leggere attentamente il file *txt.shp* e gli altri font consegnati con AutoCAD; essi rappresentano esempi validi di argomenti trattati in questo capitolo. I font di testo di AutoCAD sono file di definizione delle forme in cui i numeri di forma corrispondono al codice ASCII di ogni carattere. Per ottenere un elenco completo dei codici ASCII, vedere l'appendice A "Codici ASCII".

I codici da 1 a 31 sono destinati ai caratteri di controllo, dei quali uno soltanto viene utilizzato nei font di testo di AutoCAD:

10 (LF)

Il codice di nuova riga (LF) esegue il passaggio alla riga successiva senza effettuare alcun disegno. Viene utilizzato per i comandi TESTO ripetuti per inserire ulteriori righe dopo la prima.

```
*10,5,lf
2,8,(0,-10),0
```

È possibile modificare la spaziatura delle righe modificando il movimento verso il basso specificato dalla definizione di forma LF.

I font di testo devono includere un numero di forma speciale 0 che trasmette informazioni relative al font stesso. Il formato avrà la seguente sintassi:

```
*0,4,nome-font
sopra,sotto,modalità,0
```

dove *sopra* specifica di quante lunghezze di vettore le lettere maiuscole si estendono al di sopra della linea di base, mentre *sotto* indica di quanto le lettere minuscole si abbassano al di sotto della linea di base. La linea di base è concettualmente simile alle linee di un foglio di quaderno. Questi valori definiscono la dimensione di base del carattere e vengono utilizzati come fattori di scala per l'altezza specificata nel comando TESTO.

Il byte *modalità* deve essere 0 per un font con orientamento orizzontale e 2 per i font con orientamento sia orizzontale che verticale.

Il codice speciale 00E (14) viene considerato solo quando *modalità* è impostato su 2.

I font standard forniti con AutoCAD comprendono alcuni caratteri supplementari necessari per la funzione di quotatura di AutoCAD.

%%d - Simbolo dei gradi (°)

%%p - Simbolo di tolleranza più/meno (±)

%%c - Simbolo di quotatura per il diametro di un cerchio (∅)

È possibile utilizzare le suddette e le sequenze di controllo %%nnn, come descritto in "TESTO" nella *Guida di riferimento dei comandi di AutoCAD*.

Nota AutoCAD disegna i caratteri di testo mediante i rispettivi codici ASCII (numeri di forma) e non mediante il nome. Per risparmiare memoria, specificare in minuscolo la parte del nome della forma di ogni definizione della forma di testo come riportato nell'esempio seguente. I nomi scritti in minuscolo non vengono salvati in memoria.

```
*65,11,uca
024,043,04d,02c,2,047,1,040,2,02e,0
```

Poiché il nome di forma uca è scritto in minuscolo, esso non viene memorizzato. Tuttavia, è possibile utilizzare tale nome come riferimento quando si modifica il file di definizione dei font. Nell'esempio uca sta per uppercase A (A maiuscola).

Capitolo 3 -- Forme, font e supporto PostScript

File di definizione delle forme

Descrizione dei big font

Alcune lingue, come ad esempio il giapponese, utilizzano font di testo con molti caratteri non ASCII. AutoCAD supporta un tipo speciale di file di definizione di forme chiamato *big font* per fare in modo che il testo di tipo non ASCII possa essere inserito nei disegni.

Capitolo 3 -- Forme, font e supporto PostScript

File di definizione delle forme

Descrizione dei big font

Definizione di un file big font

Un font con centinaia o migliaia di caratteri deve essere gestito in modo diverso da un font contenente la serie di 256 caratteri ASCII. Oltre a dover utilizzare tecniche più complesse per ricercare il file, AutoCAD deve disporre di un mezzo per rappresentare caratteri con codici a due byte e codici composti da un solo -byte. I problemi di questo tipo vengono risolti utilizzando codici speciali all'inizio di un file big font.

La prima riga di un file big font di definizione di forme deve essere la seguente:

```
*BIGFONT n_car,n_interv,b1,e1,b2,e2,...
```

in cui *n_car* è il numero approssimato di definizioni di caratteri presente nella serie utilizzata; se l'approssimazione supera il 10% circa, la velocità di elaborazione o la dimensione del file risulterà notevolmente ridotta. È possibile utilizzare il resto della riga per definire codici di caratteri speciali (codici di escape) che indicano l'inizio di un codice a due byte. Sui computer giapponesi, ad esempio, i caratteri Kanji iniziano con codici esadecimali compresi nell'intervallo 90-AF o E0-FF. Quando il sistema operativo rileva la presenza di uno di questi codici, legge il byte successivo e combina i due byte in un codice per un carattere Kanji. Nella riga *BIGFONT, *n_interv* indica il numero di intervalli contigui utilizzati come codici di escape; *b1*, *e1*, *b2*, *e2* e così via, definiscono l'inizio e la fine dei codici di ogni intervallo. Pertanto, l'intestazione per un file big font giapponese potrebbe essere la seguente:

```
*BIGFONT 4000,2,090,0AF,0E0,0FF
```

La definizione di font che segue la riga *BIGFONT è uguale ad un regolare font di testo di AutoCAD, ad eccezione del fatto che i codici di carattere (numeri di forme) non possono essere maggiori di 65535.

Capitolo 3 -- Forme, font e supporto PostScript

File di definizione delle forme

Descrizione dei big font

Definizione di un file big font esteso

Per ridurre la dimensione dei caratteri Kanji composti, è possibile definire un file big font esteso. I big font estesi utilizzano il codice di forma secondaria seguito da uno 0.

La prima riga di un file big font esteso è identica a quella di un file big font regolare. Le righe seguenti del file avranno il seguente formato:

```
*0,5,nome-font
altezza-carattere,0,modalità,ampiezza-carattere,0
.
.
.
*numero-forma,byte_def,nome-forma
.
codice,0,primitiva#,punto_base-x,punto_base-y,larghezza,altezza,
.
.
codice,0,primitiva#,punto_base-x,punto_base-y,larghezza,altezza,
.
carattere_terminazione
```

altezza-carattere

Viene utilizzato insieme all'ampiezza del carattere per indicare il numero di unità che definiscono il carattere del font.

ampiezza-carattere

Viene utilizzato insieme all'altezza del carattere per indicare il numero di unità che definiscono il carattere del font. I valori altezza-carattere ed ampiezza-carattere vengono utilizzati per mettere in scala le primitive del font. In questo contesto, le primitive sono le stringhe di punti, di poligoni o di caratteri del font geometricamente orientato nello spazio bidimensionale. Un carattere Kanji è composto da varie primitive utilizzate ripetutamente in differenti scale e combinazioni.

modalità

Uguale ai font di testo regolari.

numero-forma

Codice di carattere.

byte_def

Dimensione del byte. Il valore è sempre 2 e consiste in un codice esadecimale o in una combinazione di codici decimali ed esadecimali.

nome-forma

Nome del carattere.

codice

Codice speciale per la descrizione delle forme. Il valore è sempre 7 in modo da utilizzare la funzione di forma secondaria.

primitiva#

Si riferisce al numero della forma secondaria. È sempre uguale a 2.

punto_base-x

Origine X della primitiva.

punto_base-y

Origine Y della primitiva.

larghezza

Fattore di scala della larghezza della primitiva.

altezza

Fattore di scala dell'altezza della primitiva.

carattere_terminazione

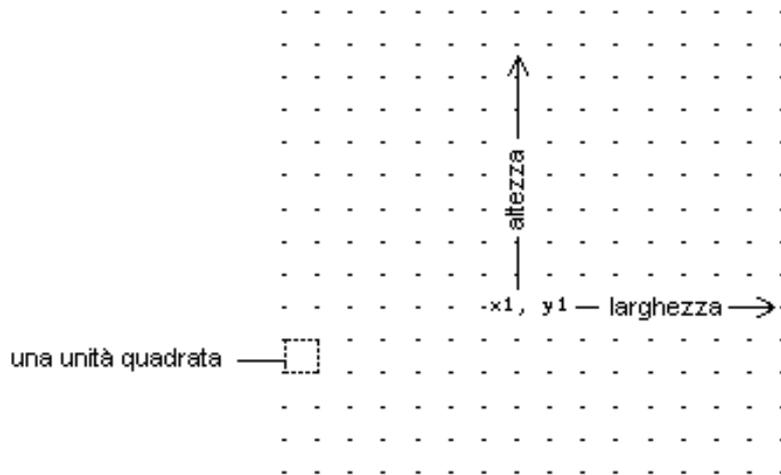
Indica la fine del file per la definizione della forma. Il valore è sempre 0.

Per ottenere il fattore di scala, AutoCAD riduce proporzionalmente la primitiva di un'unità quadrata e poi la moltiplica per l'altezza e la larghezza per ottenere la forma del carattere. Ai codici di carattere (numeri di forma) nel file di definizione delle forme big font può essere associato un valore massimo di 65535. Nella tabella seguente vengono descritti i campi del file big font esteso.

Campi del file big font esteso

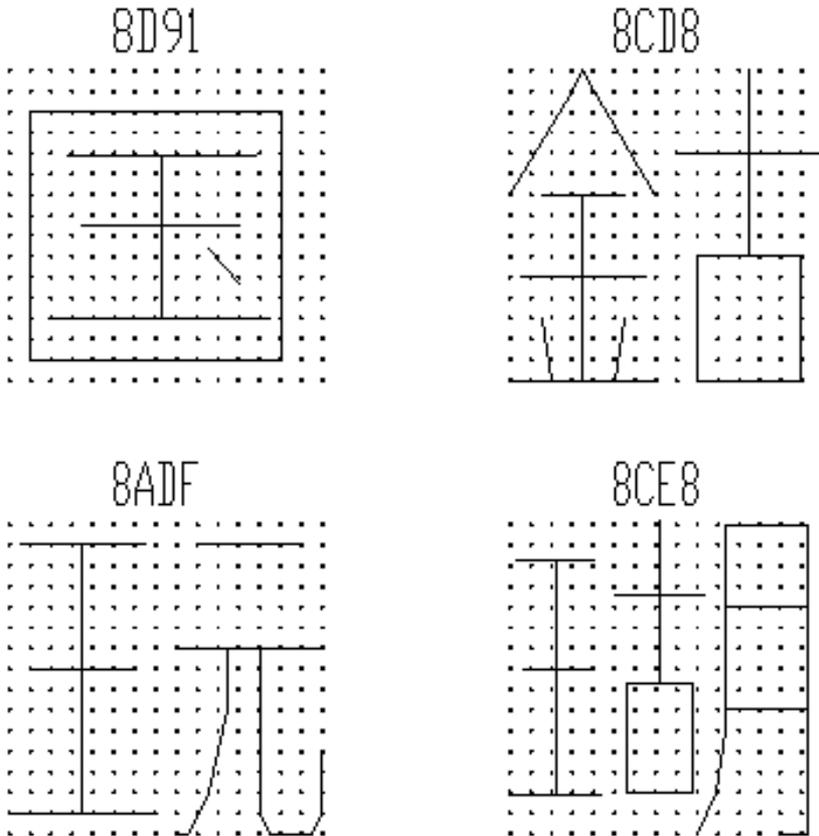
Variable	Valore	Dimensione byte	Descrizione
numero-forma	xxxx	2 byte	Codice di carattere.
codice	7,0	2 byte	Definizione del font esteso.
primitiva#	xxxx	2 byte	Si riferisce al numero di forma secondaria.
punto_base-x		1 byte	Origine X della primitiva.
punto_base-y		1 byte	Origine Y della primitiva.
larghezza		1 byte	Fattore di scala della larghezza della primitiva.
height		1 byte	Fattore di scala dell'altezza della primitiva.
carattere_terminazione	0	1 byte	Definizione della fine della forma.

L'esempio riportato nella figura seguente mostra una matrice di punti 16 x 16 che può essere utilizzata per disegnare un big font esteso, come ad esempio un carattere Kanji. Nell'esempio la distanza tra punti è una unità. Nella figura è indicata una unità quadrata.



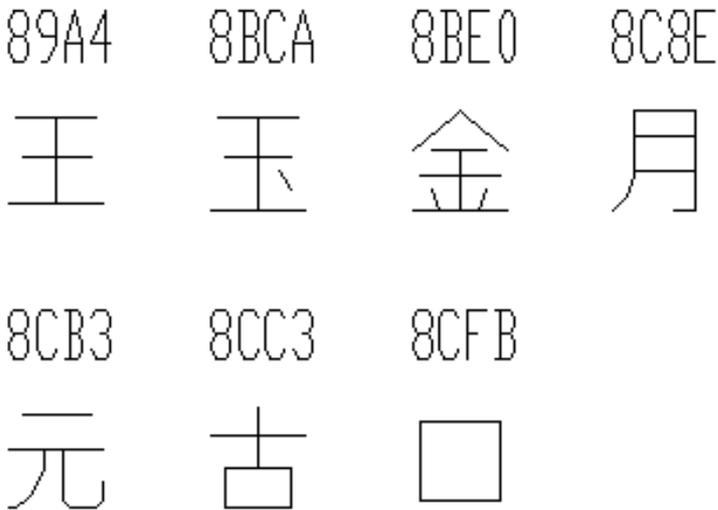
Matrice quadrata per un carattere Kanji

Nella figura di seguito riportata vengono illustrati esempi di caratteri Kanji. Ogni carattere occupa una matrice M x N (le matrici non devono necessariamente essere quadrate) simile a quella mostrata nella figura precedente.



Esempi di caratteri Kanji

Nella figura seguente vengono illustrate primitive Kanji.



Esempi di primitive Kanji

Nota Non tutti i font vengono definiti in una matrice quadrata; alcuni sono definiti in matrici rettangolari.

Di seguito è riportato un esempio di un file di definizione delle forme per un big font esteso.

```
*BIGFONT 50,1,080,09e
*0,5,Extended Font
15,0,2,15,0
*08D91,31,unspecified
```

```

2,0e,8,-7,-15,
7,0,08cfb,0,0,16,16,7,0,08bca,2,3,12,9,
2,8,18,0,2,0e,8,-11,-3,0
*08CD8,31,unspecified
2,0e,8,-7,-15,
7,0,08be0,0,0,8,16,7,0,08cc3,8,0,8,16,
2,8,18,0,2,0e,8,-11,-3,0
*08ADF,31,unspecified
2,0e,8,-7,-15,
7,0,089a4,0,0,8,16,7,0,08cb3,8,0,8,16,
2,8,18,0,2,0e,8,-11,-3,0
*08CE8,39,unspecified
2,0e,8,-7,-15,
7,0,089a4,0,1,5,14,7,0,08cc3,5,2,5,14,7,0,08c8e,9,0,7,
16,2,8,18,0,2,0e,8,-11,-3,0
*089A4,39,primitive
2,0e,8,-7,-15,2,8,1,14,1,0c0,
2,8,-11,-6,1,0a0,2,8,-12,-7,1,
0e0,2,8,-7,13,1,0dc,2,8,11,-1,
2,0e,8,-11,-3,0
*08BCA,41,primitive
2,0e,8,-7,-15,2,8,1,14,1,0c0,
2,8,-11,-6,1,0a0,2,8,-12,-8,1,
0e0,2,0e5,1,0ec,2,063,1,8,
2,-3,2,06f,2,0e,8,-11,-3,0
*08BE0,81,primitive
2,0e,8,-7,-15,2,8,3,9,1,080,
2,8,-10,-4,1,0c0,2,8,-13,-5,1,
0e0,2,8,-7,9,1,09c,2,8,-1,14,
1,8,-6,-5,2,8,8,5,1,8,6,-5,
2,8,-11,-6,1,8,1,-3,2,8,7,3,
1,8,-1,-3,2,8,-3,15,1,01a,2,
012,1,01e,2,8,10,-14,2,0e,8,
-11,-3,0
*08C8E,44,primitive
2,0e,8,-7,-15,2,8,3,15,1,090,0fc,038,
2,8,-6,11,1,090,2,8,-9,-5,1,
090,2,096,1,0ac,8,-1,-3,01a,01a,2,8,
18,0,2,0e,8,-11,-3,0
*08CB3,61,primitive
2,0e,8,-7,-15,2,042,1,02b,02a,018,2,
0d0,1,012,034,2,069,1,01e,040,2,8,
-8,6,1,02b,2,8,4,5,1,08c,2,8,
-3,8,1,03c,2,8,-5,3,1,0e0,2,8,
-12,5,1,0a0,2,8,6,-14,2,0e,8,
-11,-3,0
*08CC3,34,primitive
2,0e,8,-7,-15,2,0c1,1,06c,0a8,064,0a0,2,8,
-5,9,1,09c,2,8,-7,5,1,0e0,2,8,
4,-11,2,0e,8,-11,-3,0
*08CFB,22,primitive
2,0e,8,-7,-15,2,0d2,1,0cc,0c8,0c4,0c0,2,8,
5,-13,2,0e,8,-11,-3,0

```

Alcune tecniche per il disegno consentono di visualizzare molti simboli speciali nelle stringhe di testo. I font di testo standard di AutoCAD possono essere estesi in modo da includervi simboli speciali; ciò presenta tuttavia alcune limitazioni:

- Ogni file di font può contenere al massimo 255 forme.
- La serie di caratteri standard utilizza quasi la metà dei numeri di forme disponibili. Sono disponibili soltanto i codici 1-9, 11-31 e 130-255.
- Se si desidera utilizzare più font di testo, è necessario duplicare le definizioni dei simboli in ogni font.
- Per utilizzare un simbolo speciale, è necessario digitare %%*nnn*, in cui *nnn* è il numero di forma del simbolo.

Il meccanismo del big font evita questi problemi. È possibile selezionare uno o più caratteri usati raramente, come ad esempio la tilde (~) o la barra verticale (|), come codici di escape ed utilizzare il carattere successivo per selezionare il simbolo speciale

appropriato. Ad esempio, è possibile utilizzare il file big font di seguito riportato per disegnare le lettere greche digitando una barra verticale (|, codice ASCII 124) seguita dalla lettera romana equivalente. Poiché il primo byte di ogni carattere è 124, i codici dei caratteri vengono alterati di 124 x 256 oppure di 31744.

```
*BIGFONT 60,1,124,124
*0,4,Greek
sopra, sotto, modalità, 0
*31809,n,uca
. . .   definizione di Alfa maiuscola richiamata da "|A"
*31810,n,ucb
. . .   definizione di Beta maiuscola richiamata da "|B"
*31841,n,lca
. . .   definizione di Alfa minuscola richiamata da "|a"
*31842,n,lcb
. . .   definizione di Beta minuscola richiamata da "|b"
*31868,n,vbar
. . .   definizione di barra verticale richiamata da "||"
. . .
```

Capitolo 3 -- Forme, font e supporto PostScript

File di definizione delle forme

Descrizione dei big font

Uso di un big font

Per utilizzare un big font per disegnare testo, è necessario impostare uno stile mediante il comando STILE e quindi specificare il nome del file big font. Lo stesso stile di testo può utilizzare un normale font di testo ASCII; digitare soltanto i due nomi del file separandoli con una virgola.

Comando: **stile**

Nome dello stile (o ?): *nome*

File di font <corrente>: **txt,greek**

AutoCAD presuppone che il primo nome sia il font normale e che il secondo sia il big font.

Se si digita soltanto un nome, il programma presuppone che sia il font normale ed elimina qualsiasi big font associato.

È possibile modificare un font senza influenzarne un altro utilizzando le virgole iniziali e finali quando si specificano i nomi dei file di font, come riportato nella tabella seguente.

Input per modificare i font

Input	Risultato
<i>normal, big</i>	<i>Vengono specificati sia il font normale che il big font.</i>
<i>normal,</i>	<i>Solo il font normale; il big font rimane invariato.</i>
<i>big</i>	<i>Solo il big font; il font normale rimane invariato.</i>
<i>normal,</i>	<i>Solo il font normale; se necessario, il big font viene eliminato.</i>
RETURN (risposta nulla)	<i>Non viene apportata alcuna modifica.</i>

Quando si utilizza il comando STILE per elencare gli stili o per modificare uno stile esistente, AutoCAD visualizza il file dei font normali, una virgola ed il file big font. Se lo stile prevede un solo file big font, il file verrà visualizzato preceduto da una virgola, come ad esempio in ,greek.

Per ogni carattere di una stringa di testo, AutoCAD ricerca prima il file big font. Se il carattere non viene trovato, il programma ricerca il file dei font normali.

Capitolo 3 -- Forme, font e supporto PostScript

File di definizione delle forme

Descrizione dei font Unicode

I font standard di AutoCAD corrispondono al mappaggio dei caratteri utilizzato dal sistema operativo host. Questo perché i caratteri vengono memorizzati direttamente nel database nel formato in cui vengono ottenuti dalla tastiera. Gli stessi codici di caratteri vengono utilizzati per generare i font. Ciò diventa un problema quando vengono utilizzati i caratteri accentati (8 bit) per i quali esistono molti standard di codifica dei caratteri.

A causa delle limitazioni del mappaggio dei caratteri, è stato necessario fornire un gruppo di font per le varie tabelle codici utilizzate da AutoCAD. Questi font, pur essendo essenzialmente tutti uguali, includono alcuni caratteri in posizioni diverse, in base alla tabella codici per la quale sono definiti. Se la codifica dei font utilizzata non corrisponde a quella del testo contenuto nel disegno, potrebbero essere disegnati caratteri errati.

Con i font Unicode, le stringhe di testo vengono convertite in Unicode *prima* di essere disegnate; in tal modo non è più necessario fornire ulteriori font per altre lingue o piattaforme. Un singolo font Unicode, grazie all'ampio gruppo di caratteri che vi è associato, è in grado di supportare tutte le lingue e le piattaforme. Questa caratteristica è trasparente per l'utente poiché, se necessario, grazie alle diverse tabelle codici, i disegni vengono convertiti nella tabella codici di sistema di AutoCAD al momento del caricamento. I disegni vengono sempre salvati nella tabella codici di sistema di AutoCAD.

Nota Il font Unicode *non* è in grado di fornire l'adeguato supporto per tutte le lingue asiatiche, quindi i big font saranno

Capitolo 3 -- Forme, font e supporto PostScript

File di definizione delle forme

Descrizione dei font Unicode

Formato dei file

Il formato e la sintassi dei file di definizione delle forme Unicode sono praticamente identici ai normali file di definizione delle forme. La principale differenza viene rilevata nella sintassi dell'intestazione del font, come mostrato nel codice riportato di seguito.

```
*UNIFONT, 6, nome-font
sopra, sotto, modalità, codifica, tipo, 0
```

I parametri *nome-font*, *sopra*, *sotto* e *modalità* sono uguali a quelli dei font regolari. I restanti due parametri sono definiti come descritto di seguito.

codifica

È la codifica del font ed assumono uno dei seguenti valori a numero intero.

- 0 Unicode
- 1 Multi-byte 1 compresso
- 2 File di forma.

type

Sono le informazioni sull'incorporazione del font. Specifica se il font è concesso su licenza; in tal caso non deve essere modificato o scambiato (possono essere aggiunti i valori codificati in bit).

- 0 Il font può essere incorporato.
- 1 Il font non può essere incorporato.
- 2 L'incorporazione è di sola lettura.

L'unica differenza tra le definizioni delle forme Unifont e le definizioni delle forme regolari sono i numeri delle forme. Le definizioni delle forme Unifont fornite con AutoCAD utilizzano numeri di forme esadecimali invece che valori decimali. Ciò non è richiesto, ma facilita la creazione di riferimenti incrociati ai numeri delle forme tramite i valori del carattere di controllo\U+.

Capitolo 3 -- Forme, font e supporto PostScript

File di definizione delle forme

Tecnica avanzata per la definizione delle forme

I font forniti con AutoCAD hanno delle limitazioni per quanto riguarda gli esponenti ed i deponenti, anche se risulta abbastanza semplice modificare i file di definizione delle forme per migliorare questa funzione.

Per creare le funzioni relative agli esponenti ed ai deponenti, è necessario effettuare due passi. Per prima cosa si deve spostare sullo schermo verso l'alto e verso il basso la "penna immaginaria" che genera il testo, vettore per vettore e poi ridurre il font relativo alla scala. Per tornare al font normale è necessario eseguire il processo inverso. In questo modo, il font deve riconoscere quattro nuovi tasti, due per gli esponenti e due per i deponenti. Per evitare di alterare le definizioni dei font esistenti, è possibile accedervi utilizzando il tastierino numerico.

L'esempio seguente si riferisce alla creazione di un file di font ROMANS di AutoCAD, anche se un metodo simile può essere applicato a tutti i font di AutoCAD. Con questa procedura vengono aggiunte ad un font quattro nuove definizioni di forma: `super_on`, `super_off`, `sub_on` e `sub_off` che controllano la posizione e la dimensione dei caratteri successivi. Per semplificazione, in questo esempio i caratteri delle parentesi quadre sinistra e destra (`[e]`) ed i caratteri delle parentesi graffe sinistra e destra (`{ e }`) vengono sostituiti con i nuovi caratteri. È possibile sostituire altri caratteri oppure usare un numero di forma compreso nell'intervallo esteso (codici ASCII da 128 a 256). Se viene utilizzato un numero di forma esteso, per posizionare il nuovo carattere sarà necessario adottare il metodo `%nnn` (dove `nnn` rappresenta il valore ASCII del carattere).

- 1 Modificare il file `.shp` (in questo caso `romans.shp`), con un editor di testo ASCII. Si consiglia di creare un nuovo file con il nome `romanss.shp` invece di modificare il file originale.
- 2 Cercare le definizioni delle forme dei caratteri da sostituire. È necessario escludere, facendole diventare delle righe di commento, le definizioni esistenti in modo che vengano eseguite solamente le nuove definizioni. A tal fine, inserire un punto e virgola davanti ad ogni riga di definizione delle forme; di solito la definizione delle forme occupa diverse righe.

I caratteri delle parentesi quadre sinistra e destra hanno i valori ASCII 91 e 93 (valori esadecimali 05B e 05D, se il font è Unicode). I caratteri delle parentesi graffe aperte e chiuse corrispondono ai valori ASCII 123 e 125 (esadecimali 07B e 07D).

- 3 Sommare le prime due cifre riportate sulla seconda riga e dividere il totale per 2, come nell'esempio seguente:

```
*UNIFONT,6,Extended Simplex Roman for UNICODE
21,7,2,0
```

$21 + 7 = 28$, quindi $28 / 2 = 14$. Questo numero verrà usato in seguito.

- 4 Aggiungere le righe riportate di seguito alla fine del file `.shp` file.

```
*228,8,super_on
2,8,(0,14),003,2,1,0
*227,8,super_off
2,004,2,8,(0,-14),1,0
*222,8,sub_on
2,8,(0,-14),003,2,1,0
*221,8,sub_off
2,004,2,8,(0,14),1,0
```

Si considerino le cifre 14 e -14 nelle linee precedenti. Esse rappresentano gli sfalsamenti rispetto all'asse Y per la penna immaginaria. La cifra 14 rappresenta la metà dell'altezza massima di un carattere in questo font ed è il valore corretto sia per gli esponenti che per i deponenti. Tale valore deve essere calcolato per ogni file di font, ma è possibile modificarlo in qualsiasi momento.

A questo punto salvare il file.

- 5 Usare il comando COMPILA per compilare il file `.shp`.

Una volta compilata la forma e definito uno stile appropriato, è possibile accedere ai nuovi comandi relativi alla penna sollevata ed alla penna abbassata digitando i caratteri `[`, `]`, `{` e `}`. La parentesi quadrata aperta (`[`) indica l'inizio della scrittura dell'esponente, mentre la parentesi quadrata chiusa (`]`) consente di tornare alla scrittura normale. Il carattere `{` inizia la scrittura del deponente ed il carattere `}` consente di tornare alla scrittura normale.

Capitolo 3 -- Forme, font e supporto PostScript

Supporto PostScript

Con le funzioni di supporto PostScript di AutoCAD, è possibile usare i font PostScript con i disegni AutoCAD, esportare i disegni AutoCAD come file EPS (Encapsulated PostScript), importare i file EPS nei disegni AutoCAD ed applicare i riempimenti PostScript alle polilinee bidimensionali.

Capitolo 3 -- Forme, font e supporto PostScript

Supporto PostScript

Esportazione di immagini PostScript

Il comando PSOUT consente di esportare la vista corrente di un disegno come un file EPS che può essere inserito in un programma di editoria desktop, di illustrazioni o di presentazione.

Nelle sezioni successive viene descritta la struttura e lo scopo del file di supporto principale per PSOUT, *acad.psf*. In queste sezioni vengono fornite anche informazioni sull'elaborazione avanzata dell'output PostScript.

Capitolo 3 -- Forme, font e supporto PostScript

Supporto PostScript

Esportazione di immagini PostScript

File di supporto PostScript di AutoCAD: *acad.psf*

Il file di supporto PostScript di AutoCAD (*acad.psf*) è il file di supporto principale per i comandi PSOUT e PSMOTIVO di AutoCAD. Questo file ASCII può essere modificato ed è diviso in sezioni che controllano la sostituzione dei font e definiscono i modelli di riempimento PostScript. Include inoltre sezioni *introduttive* contenenti le definizioni e le costanti per la procedura codificata in PostScript. Le sezioni del file *acad.psf* possono apparire in qualsiasi ordine, ma per ragioni di efficienza la sezione **fonts* deve essere la prima del file.

Ogni sezione del file *acad.psf* è introdotta da un'istruzione con un asterisco nella colonna 1 seguito dal nome della sezione. In tutte le sezioni di questo file, le righe che iniziano con un punto e virgola vengono considerate righe di commento.

Il file standard *acad.psf* definisce i font Type 1 forniti con AutoCAD ed anche tutti i font PostScript che appaiono in Adobe Type Manager per Windows, Adobe Plus Pack ed Adobe Font Pack 1. Dal momento che con AutoCAD non viene fornito un programma di scaricamento dei font, sarebbe opportuno modificare il file *acad.psf* in modo da fare riferimento ai font che si trovano nella stampante in uso. Se con la stampante non è stato fornito il programma di scaricamento dei font, è possibile ottenerlo dal forum Adobe su CompuServe. Nelle sezioni successive vengono descritte le sezioni del file *acad.psf*.

***fonts: mappa di sostituzione dei font**

In ogni riga della sezione Fonts viene elencato il nome di un font di AutoCAD (*.shx* o *.pfb*) ed il nome di un font PostScript di sostituzione per gli oggetti di testo che usano quel font quando viene utilizzato il comando PSOUT. I nomi dei font PostScript definiti da Adobe sono sensibili al maiuscolo/minuscolo; lo stesso non è valido per i nomi dei font di AutoCAD. La verifica sui nomi dei font PostScript non viene eseguita. Se un font richiesto non è disponibile al momento della stampa, la maggior parte delle implementazioni PostScript sono in grado di emettere un messaggio di avvertimento e di sostituire quel font con uno di default, ad esempio Courier.

Nell'esempio riportato di seguito, estratto dal file *acad.psf*, vengono mostrati i file di forma e dei font di AutoCAD mappati con i font PostScript. Ad esempio, il file *agd.shx* viene mappato con un font PostScript AvantGarde-Demi.

```
*fonts
;
agd      AvantGarde-Demi
```

agdo	AvantGarde-DemiOblique
agw	AvantGarde-Book
agwo	AvantGarde-BookOblique
bdps	Bodoni-Poster
bkd	Bookman-Demi
bkdi	Bookman-DemiItalic
bkl	Bookman-Light
bkli	Bookman-LightItalic
c	Cottonwood
cibt	City-Blueprint
cob	Courier-Bold
cobo	Courier-BoldOblique
com	Courier
.	.
.	.

Nella prima colonna è riportato il nome con il quale AutoCAD conosce il font; nella seconda è riportato il nome PostScript ufficiale del font. I restanti caratteri di sottolineatura devono essere ignorati; ad esempio, se il file contenente il font AvantGarde-Demi si chiama *agd_____pfb*, la voce della prima colonna è *agd*.

*figureprologue: introduzione alle figure PostScript

Nella sezione **figureprologue* vengono definite le procedure per incorporare le figure PostScript incluse con il comando PSIN nei file EPS (.eps) creati dal comando PSOUT. Per mantenere questo codice nel file *acad.psf*, è possibile modificarlo come necessario in modo che sia adatto per file PostScript insoliti. Questo codice viene emesso nel punto in cui viene incontrata la prima immagine PSIN.

*isofontprologue: ricodifica per il font ISO 8859 Latin/1

Nella sezione **isofontprologue* sono definite le procedure per la ricodifica di un font PostScript standard in modo che sia compatibile con il gruppo di caratteri ISO 8859 Latin/1. Questo codice viene emesso quando in un oggetto di testo viene incontrato per la prima volta un carattere ISO.

*fillprologue: introduzione al riempimento delle polilinee in PostScript

Nella sezione **fillprologue* viene definito il codice trascritto nell'output quando il comando PSOUT incontra la prima richiesta del comando PSMOTIVO. Questa sezione viene utilizzata per riempire le polilinee ed in essa vengono definite le seguenti procedure di riempimento:

Rangefilter

È una procedura di carattere generale per eseguire la verifica dell'intervallo di modelli di riempimento predefiniti e definiti dall'utente.

La procedura Rangefilter può essere utilizzata per ogni modello di riempimento richiamato con i parametri per verificare che tali parametri siano compresi in un intervallo valido per il modello. La sintassi è la seguente:

valore min max Rangefilter risultato

Se il *valore* è compreso nell'intervallo tra il limite minimo (*min*) ed il limite massimo (*max*), allora tale *valore* diventa il *risultato*. In caso contrario, il *risultato* sarà il limite più vicino al *valore*.

Ad esempio, considerare il seguente modello di riempimento per Grayscale:

```
%@Fill
/Grayscale %Grayscale,1, Grayscale=50
{
0 100 Rangefilter 100 div 1 exch sub
setgray fill
} bind def
```

In questo esempio, la procedura Rangefilter viene richiamata immediatamente in corrispondenza dell'argomento della pila passato a Grayscale. Se questo valore non è compreso nell'intervallo, Rangefilter lo imposta sul valore limite più vicino, in questo esempio *min* (0) o *max* (100).

Rangefilter può essere usata anche all'interno di un modello di riempimento per calcolare dinamicamente i limiti per il modello di riempimento.

PreFill

Questa procedura viene richiamata dopo aver stabilito la trasformazione usata per il rendering della polilinea ed esegue il salvataggio della trasformazione nella variabile CMatrix.

DoFill

Questa procedura viene richiamata immediatamente dopo che il percorso da riempire è stato completato, pertanto il percorso della polilinea, su cui viene effettuato il rendering in PostScript, è il percorso corrente. Tale procedura salva la casella di delimitazione nelle variabili standard `Bbxxyy` quindi stabilisce la trasformazione standard usata dalle procedure di riempimento.

EndFill

Questa procedura viene richiamata dopo la procedura DoFill ed è in grado di eseguire qualsiasi azzeramento necessario, anche se in questo caso non ha alcun effetto.

DoOutline

Questa procedura viene richiamata immediatamente dopo EndFill genera il contorno della polilinea. Quando si richiama DoOutline, il percorso corrente corrisponde al percorso del contorno della polilinea e la larghezza di linea PostScript viene impostata alla larghezza della polilinea e scalata in base allo spazio per l'immagine in PostScript.

Per omettere il contorno della polilinea, far precedere il nome del modello di riempimento da un asterisco (*).

***fill: definizione dei propri modelli di riempimento PostScript**

Nella sezione `*fill` vengono definiti i modelli di riempimento PostScript. Tutti i modelli di riempimento vengono descritti tra l'istruzione `*fill` e la successiva riga che presenta un asterisco a colonna 1. Per aggiungere un nuovo modello di riempimento PostScript, modificare questa sezione del file `acad.psf`.

Ogni modello di riempimento viene introdotto da questa riga che ha inizio a colonna 1:

```
%@Fill
```

Subito dopo questa riga, in formato definito in modo estremamente preciso, si trova la prima riga della procedura PostScript che definisce il modello. Nell'esempio riportato di seguito viene definito il modello RGBcolor:

```
%@Fill
/RGBcolor %RGBcolor,3, Red=50, Green=50, Blue=50,
{
/Blue exch 0 100 Rangefilter def
/Green exch 0 100 Rangefilter def
/Red exch 0 100 Rangefilter def
Red 100 div 1 exch sub
Green 100 div 1 exch sub
Blue 100 div 1 exch sub
setrgbcolor fill
} bind def
```

La prima riga della procedura presenta la seguente sintassi:

```
/nome_proc %nome_modello,num_arg, [arg_num=default_num]
```

in cui le variabili sono definite nel seguente modo:

`nome_proc`

È il nome interno di questa procedura di riempimento PostScript. Il nome deve essere univoco; in generale, non può essere la copia di un'altra procedura di riempimento contenuta nel file oppure di un nome PostScript qualsiasi.

`nome_modello`

È il nome del modello digitato per il comando PSMOTIVO.

`num_arg`

È un numero intero che indica il numero di argomenti al modello, da 0 a 25.

`arg_num`

È il nome dell'*ennesimo* argomento che appare nel messaggio di richiesta quando il modello viene applicato con il comando PSMOTIVO.

`default_num`

È il valore di default per l'*ennesimo* argomento quando il modello viene usato per la prima volta. Gli argomenti hanno valori interi o reali ed è necessario specificare un valore di default.

Quando AutoCAD incontra un modello di riempimento PostScript in un disegno esportato con il comando PSOUT, viene individuata la corrispondente definizione nel file *acad.psf*, che viene copiata nel file di output *.eps* (Encapsulated PostScript). Quanto è compreso tra la riga che inizia con "/" (/RGBcolor nell'esempio precedente) fino alla successiva riga "%@" (oppure fino alla fine della sezione o del file) viene trascritto letteralmente nel file di output PostScript.

La funzione del modello di riempimento con il contorno della polilinea viene richiamata come il percorso PostScript corrente. Quando tale funzione viene richiamata, la trasformazione PostScript sarà stata configurata nel seguente stato attendibile:

- La trasformazione del rendering viene impostata in modo tale che le unità di default usate nella funzione di riempimento siano i millesimi di un pollice.
- Le variabili **Bbllx**, **Bbly**, **Bburx** e **Bbury** definiscono le estensioni della casella di delimitazione del percorso in unità di 1/1000 di pollice.
- **gsave** e **grestore** vengono eseguiti di conseguenza all'invocazione della procedura di riempimento per garantire che le relative modifiche allo stato del disegno non vengano applicate anche all'output degli oggetti successivi.
- Gli argomenti passati alla procedura di riempimento appaiono nella pila di operandi nell'ordine in cui essi sono stati dichiarati nel file *acad.psf*: l'ultimo argomento si trova in cima alla pila, mentre gli argomenti dal secondo all'ultimo si trovano in questa posizione nella pila e così via. La procedura di riempimento elimina tutti gli argomenti dalla pila.

Con la prima procedura di riempimento PostScript, il contenuto della sezione **fillprologue* del file *acad.psf* viene trascritto nel file *.eps*. In questa sezione vengono definite le procedure che possono essere usate da tutte le procedure di riempimento.

PSPROLOG: creazione di una sezione introduttiva personalizzata

Con la variabile di sistema PSPROLOG viene indicato al comando PSOUT di fare riferimento ad una sezione introduttiva aggiuntiva, creata dall'utente nel file *acad.psf*. Creando una sezione introduttiva ed inserendovi un codice PostScript, è possibile personalizzare l'aspetto dell'output PostScript in modi diversi. Ciò è utile se si intende eseguire funzioni di output come l'assegnazione di larghezze di linea diverse a colori diversi oppure la creazione di tipi di linea particolari con la funzione PostScript *setdash*. Una singola sezione introduttiva può contenere più funzioni PostScript.

Per creare una sezione introduttiva personalizzata, aggiungere un'intestazione di sezione al file *acad.psf* tramite un editor di testo o un elaboratore di testi. L'intestazione di sezione è una riga di testo preceduta da un asterisco.

```
*linewidths
```

In questo esempio, tutte le righe senza commenti comprese tra **linewidths* e la successiva sezione del file *acad.psf* vengono emesse nel file PostScript.

Nel file *acad.psf* è possibile aggiungere tutte le sezioni introduttive che si desidera (ad esempio, una per ogni dispositivo di stampa in uso); tuttavia, durante l'esecuzione del comando PSOUT viene fatto riferimento soltanto ad una di esse. Usare la variabile di sistema PSPROLOG per determinare a quale sezione fare riferimento.

La variabile di sistema PSPROLOG accetta come valore una stringa, che rappresenta il nome della sezione introduttiva alla quale fare riferimento. Nell'esempio riportato di seguito viene fatto riferimento alla sezione introduttiva **linewidths* creata precedentemente.

Comando: `psprolog`

Nuovo valore per PSPROLOG, o . per nessuno <" ">: `ampiezza righe`

Nel file *acad.psf* è inclusa la sezione introduttiva di esempio **sampleprolog*.

Capitolo 3 -- Forme, font e supporto PostScript

Supporto PostScript

Esportazione di immagini PostScript

Elaborazione PostScript avanzata

Il comando PSOUT codifica la struttura estratta dai disegni AutoCAD nei file PostScript generati dal comando stesso. Applicazioni sofisticate basate su PostScript possono usare queste informazioni per eseguire elaborazioni particolari basate sulle proprietà degli oggetti AutoCAD originali.

Modifiche del layer

Ogni modifica effettuata al layer viene contrassegnata dalla seguente chiamata di funzione:

(nome_layer) ACADLayer

Per poter intraprendere qualsiasi azione basata su di essa, è possibile ridefinire questa chiamata di funzione specificando il nome del layer come argomento.

Modifiche del colore

Le modifiche apportate al colore per la visualizzazione degli oggetti AutoCAD vengono segnalate dalla seguente chiamata di funzione:

num_colore rosso verde blu ACADColor

in cui le variabili sono definite nel seguente modo:

num_colore

È il numero del colore di AutoCAD.

rosso, verde e blu

Sono i componenti dei colori primari (da 0 a 1) per quel colore nella tavolozza standard di 256 colori di AutoCAD.

Per default, questa chiamata di funzione imposta il colore dell'output PostScript con l'operatore setrgbcolor, ma è possibile modificarlo.

Tipi di linea

Le modifiche effettuate al tipo di linea attuale vengono segnalate dalla seguente chiamata di funzione:

(nome_tipol) [serie_punti/lineette] (allineamento) ACADLtype

in cui le variabili sono definite nel seguente modo:

nome_tipol

È il nome del tipo di linea (CONTINUOUS, TRATTEGGIATA e così via).

serie_punti/lineette

Sono le lunghezze delle lineette dei segmenti della penna sollevata e della penna abbassata come specificato nella definizione del font della linea di AutoCAD.

alignment

È una lettera che indica la modalità di allineamento del modello agli oggetti.

È possibile utilizzare questa chiamata di funzione per emulare qualsiasi font di linea PostScript che viene creato dalle caratteristiche del tipo di linea di AutoCAD passate come argomenti. Vedere la sezione **sampleprolog* del file *acad.psf* per le routine di esempio nelle quali sono utilizzate le funzioni ACADLayer, ACADColor ed ACADLtype .

Capitolo 3 -- Forme, font e supporto PostScript

Supporto PostScript

Importazione di immagini PostScript

Nelle sezioni successive vengono descritte le funzioni PostScript che consentono ad AutoCAD di importare immagini PostScript tramite il comando PSIN.

Capitolo 3 -- Forme, font e supporto PostScript

Supporto PostScript

Importazione di immagini PostScript

Interprete PostScript di AutoCAD: *acadps*

L'interprete PostScript di AutoCAD (*acadps.arx*) è un'applicazione ARX che esegue il rendering delle immagini PostScript in oggetti AutoCAD. L'applicazione *acadps* genera oggetti AutoCAD utilizzando la funzione `ads_entmake()` invece di utilizzare il driver di un dispositivo hardware. Quando le immagini PostScript vengono inserite con il comando PSIN, viene richiamata l'applicazione *acadps* per effettuare il rendering di queste immagini rispettando l'indice di qualità impostato. Per impostare l'indice di qualità, utilizzare la variabile di sistema PSQUALITY, come descritto nella tabella delle variabili di sistema in "Variabili di sistema," nella *Guida di riferimento dei comandi di AutoCAD*. Se l'indice di qualità è zero, l'applicazione *acadps* non viene richiamata e per gli oggetti PostScript vengono visualizzati solo i relativi rettangoli di delimitazione e il nome del file.

Quando l'indice di qualità è diverso da zero, il comando PSIN esegue automaticamente il caricamento dell'applicazione *acadps*. Questa applicazione non è residente in AutoCAD; per ridurre il consumo di memoria, essa viene ricaricata ad ogni esecuzione del comando PSIN. Sebbene questo processo sia abbastanza veloce, è possibile evitarlo, se lo si desidera, eseguendo il caricamento dell'applicazione *acadps* con la funzione `arxload`, aggiungendo *acadps* ad *acad.rx* oppure modificando *acad.lsp* in `arxload acadps`.

Per lavorare correttamente, l'applicazione *acadps* deve essere in grado di individuare, nel percorso di ricerca delle librerie di AutoCAD, i propri file di inizializzazione dell'interprete (*acadpsd.ps*, *acadpsf.ps*, *acadpsi.ps* ed *acadpss.ps*), il file della mappa dei font (*fontmap.ps*), le definizioni dei font ed i font.

L'applicazione *acadps* è un'implementazione di Ghostscript Interpreter di Aladdin Enterprises, distribuito secondo le condizioni della licenza Ghostscript General Public License. Una copia di questo accordo di licenza è disponibile nel file *copying.gs*, nella directory *fonts* di AutoCAD. In base alle condizioni di questa licenza, l'utente è libero di distribuire copie dell'applicazione *acadps*. Il codice sorgente dell'applicazione è reperibile direttamente da Autodesk. Questa applicazione viene distribuita gratuitamente. Ghostscript non è un interprete PostScript certificato e viene distribuito da Autodesk senza alcuna garanzia.

Le immagini ed i riempimenti PostScript sono memorizzati in file di disegno come dati estesi e vengono identificati dal nome dell'applicazione AUTOCAD_POSTSCRIPT_FIGURE.

Capitolo 3 -- Forme, font e supporto PostScript

Supporto PostScript

Importazione di immagini PostScript

File della mappa dei font di AutoCAD: *fontmap.ps*

Il file della mappa dei font (*fontmap.ps*) di AutoCAD è il catalogo di tutti i font conosciuti dall'interprete PostScript (*acadps*) di AutoCAD. Durante l'elaborazione del comando PSIN, questo file ASCII esegue il mappaggio dei nomi dei font in linguaggio PostScript sui nomi dei rispettivi file (*.pfb*) di definizione dei font. Per effettuare il caricamento automatico dei font nel momento in cui vengono richiamati, è necessario che siano dichiarati nel file *fontmap.ps*. I font possono essere posizionati in qualsiasi directory del percorso di ricerca ACAD.

Quando un file EPS viene importato con il comando PSIN, l'applicazione *acadps* utilizza *fontmap.ps* per individuare i file *.pfb* che corrispondono ai font richiamati nel file EPS. Se l'applicazione *acadps* trova un file *.pfb* in *fontmap.ps* ma non nel sistema in uso, visualizza un messaggio di avvertimento ed esegue la sostituzione con un font di default. Di conseguenza, non è conveniente dichiarare font che non sono effettivamente installati.

Al momento dell'acquisto, *fontmap.ps* fa riferimento ad un gruppo di font Type 1 forniti con AutoCAD ed anche ai font distribuiti con Adobe Type Manager per Windows, Adobe Plus Pack ed Adobe Font Pack 1. Se sono stati acquistati ulteriori font, è possibile modificare il file della mappa dei font e dichiararli nello stesso modo in cui sono dichiarati quelli esistenti.

Per ulteriori informazioni, fare riferimento al contenuto di *fontmap.ps*, inclusi i commenti che descrivono la sostituzione dei font.

Capitolo 3 -- Forme, font e supporto PostScript

Supporto PostScript

Importazione di immagini PostScript

Errori nelle immagini PostScript importate

Gli errori che si verificano nelle immagini PostScript importate dal comando PSIN vengono gestiti secondo quanto descritto in *PostScript Language Reference Manual*. In seguito alla visualizzazione di messaggi relativi ad errori gravi, l'esecuzione del comando PSIN viene annullata. I messaggi relativi ad errori meno seri, come i riferimenti a font non disponibili, causano invece la sostituzione del font.

Se AutoCAD non può eseguire il rendering per un'immagine PostScript valida, e si desidera comunque includere tale immagine nel proprio disegno, impostare su 0 l'indice di qualità PostScript ed usare il comando PSIN per importare l'immagine. AutoCAD non eseguirà il rendering dell'immagine o la verifica degli errori. Tuttavia, se l'immagine PostScript contiene un errore, qualsiasi file .eps creato dal comando PSOUT genererà un errore al momento della stampa.

Inoltre non si ha alcuna garanzia sul fatto che con l'interprete PostScript di AutoCAD possa essere eseguito correttamente il rendering sui file PostScript che utilizzano un'implementazione Adobe di PostScript. L'interprete è compatibile con Adobe PostScript, ma esistono alcune impercettibili differenze. Se AutoCAD non è in grado di caricare una illustrazione PostScript valida, è possibile impostare PSQUALITY su 0 e caricare comunque l'immagine, che sarà rappresentata in AutoCAD dalla propria casella di delimitazione e dal suo nome di file.

Capitolo 3 -- Forme, font e supporto PostScript

Supporto PostScript

Riferimenti a PostScript

Adobe Type 1 Font Format Version 1.1, Addison-Wesley Publishing Company, Inc., 1990.

Ghostscript General Public License, Aladdin Enterprises, 1989. Una copia di questo accordo di licenza è disponibile nel file *copying.gs*, che si trova nella directory *fonts* di AutoCAD.

PostScript Language Program Design, Adobe Systems Incorporated, Addison-Wesley Publishing Company, Inc., Marzo 1990.

PostScript Language Reference Manual, Adobe Systems Incorporated, Addison-Wesley Publishing Company, Inc., Aprile 1991.

PostScript Language Tutorial and Cookbook, Adobe Systems Incorporated, Addison-Wesley Publishing Company, Inc., Maggio 1989.

Capitolo 4 -- Menu personalizzati

Panoramica

La personalizzazione dei menu risulta particolarmente utile nel caso in cui sia necessario ripetere regolarmente una funzione specifica dell'applicazione. La produttività può essere migliorata aggiungendo una selezione al menu. In questo modo, è possibile eseguire più passaggi di un'operazione usando un'unica selezione di menu e rendendo così automatica un'operazione altrimenti complessa.

I menu sono definiti nei file di menu. È possibile modificare un file di menu esistente oppure crearne uno proprio. Editando il testo in un file di menu, è possibile definire l'aspetto e la posizione delle voci di menu. Successivamente è possibile assegnare *macro di menu* che eseguono specifiche azioni quando viene selezionata una voce di menu.

Le macro di menu possono essere semplici registrazioni di battute che compiono una data operazione oppure possono essere una combinazione complessa di comandi e di codice di programmazione AutoLISP o DIESEL. Le macro di menu più complesse possiedono una capacità decisionale. Una macro di menu è simile ad uno script in quanto invia una serie di comandi. Tuttavia, gli script non hanno capacità decisionale e non possono essere sospesi per interagire nella creazione di menu personalizzati con cui integrare quelli predefiniti di AutoCAD.

Capitolo 4 -- Menu personalizzati

Uso dei file di menu

I file di menu definiscono la funzionalità e l'aspetto delle aree di menu. Le voci di menu in ogni area di menu contengono le stringhe dei comandi di AutoCAD e la sintassi macro che definiscono l'azione conseguente alla selezione della voce di menu. Le seguenti aree di menu vengono definite dai file di menu:

- Menu dei pulsanti del dispositivo di puntamento
- Menu a discesa ed a cursore
- Barre degli strumenti
- Menu a gruppi di immagini
- Menu dello schermo
- Menu di tavoletta di digitalizzazione.
- Stringhe di guida e descrizioni dei comandi
- Tasti di scelta della tastiera.

Queste aree di menu sono descritte più avanti in questo capitolo.

Capitolo 4 -- Menu personalizzati

Uso dei file di menu

Tipi di file di menu

Con il termine *file di menu* si intende il gruppo di file che nel loro insieme definiscono e controllano l'aspetto e la funzionalità delle aree di menu. Nella tabella che segue sono descritti i tipi di file di menu di AutoCAD.

File di menu di AutoCAD

Tipo di file	Descrizione
.mnu	È il file di modello di menu.
.mnc	È il file compilato del menu. Questo file binario contiene le stringhe di comando e la sintassi del menu che definisce la funzionalità e l'aspetto del menu.
MNR	È il file di risorsa del menu. Questo file binario contiene le bitmap usate dal menu.
MNS	È il file sorgente del menu (generato da AutoCAD).
.mnl	È il file LISP di menu. Questi file contengono le espressioni AutoLISP che possono essere usate dal file di menu e vengono caricati in memoria quando un file di menu con lo stesso nome viene a sua volta caricato.

Il tipo dei file *definizione di menu* è *.mnd*. Si tratta di file di menu sorgente contenenti macro. I file di tipo *.mnd* devono essere compilati con il compilatore di menu *mc.exe*.

Capitolo 4 -- Menu personalizzati

Uso dei file di menu

Tipi di file di menu

File DLL delle risorse

Nei file DLL delle risorse è possibile memorizzare le bitmap utilizzate nelle barre degli strumenti ed i menu a discesa ed a cursore. Il nome del file DLL deve pertanto essere uguale a quello del file di menu a cui è associato, alle risorse deve essere assegnato un nome e *non* un numero indicizzato, e il file DLL deve trovarsi nella stessa directory del file di menu che lo utilizza.

Per utilizzare tali risorse nel menu, usare i nomi di risorse appropriati nei parametri *id_small* e *id_big* per i pulsanti delle barre degli strumenti. Per informazioni specifiche, vedere "Barre degli strumenti."

Capitolo 4 -- Menu personalizzati

Uso dei file di menu

Caricamento dei file di menu

Utilizzare il comando MENU per caricare un nuovo menu ed i comandi CARMENU e SCARMENU per caricare e scaricare menu aggiuntivi chiamati menu parziali, nonché per aggiungere e togliere i singoli menu a discesa dalla barra dei menu.

AutoCAD memorizza il nome dell'ultimo menu usato in ogni file di disegno; tale nome è inoltre salvato insieme al disegno, ma viene utilizzato solo per compatibilità con versioni precedenti. All'avvio di AutoCAD viene caricato l'ultimo menu che è stato utilizzato. In AutoCAD Release 14 il menu non viene più caricato tra un disegno e l'altro.

AutoCAD trova e carica il file specificato in base alla sequenza indicata di seguito, usata anche quando AutoCAD carica un nuovo menu con il comando MENU.

- 1 AutoCAD ricerca un file sorgente di menu (*.mns*) con il nome dato seguendo la procedura di ricerca nella libreria. Se AutoCAD trova il file, viene eseguito il passo 2; in caso contrario, viene eseguito il passo 3.
- 2 Dopo aver trovato il file *.mns*, AutoCAD ricerca un file di menu compilato (*.mnc*) con lo stesso nome e nella stessa directory. Se viene trovato un file *.mnc* con la stessa data ed ora o con data ed ora successiva al file *.mns*, AutoCAD carica il file *.mnc*. In caso contrario, compila il file *.mns* in modo da generare e caricare un nuovo file *.mnc* nella stessa directory. Viene quindi eseguito il passo 4.
- 3 Se durante l'esecuzione del passo 1 non viene trovato alcun file *.mns*, AutoCAD ricerca un file di menu compilato *.mnc* con il nome specificato seguendo la procedura di ricerca nella libreria. Se trova il file *.mnc*, lo caricherà ed eseguirà quindi il passo 5; in caso contrario, esegue il passo 6.
- 4 Se AutoCAD non trova né un file *.mns*, né un file *.mnc*, ricerca un file di modello di menu *.mnu* con il nome specificato nel percorso della libreria. Se invece trova il file, compila un file *.mnc* e *.mns*, carica il file *.mnc* ed esegue il passo 5; in caso contrario, esegue il passo 6.
- 5 Una volta trovato, compilato e caricato il file di menu, AutoCAD ricerca un file LISP di menu (*.mnl*), utilizzando la procedura di ricerca nella libreria. Se AutoCAD trova questo file, valuta le espressioni AutoLISP contenute in quel file.
- 6 Se non trova alcun file di menu con il nome specificato, viene visualizzato un messaggio di errore e viene richiesto di specificare un altro nome.

Il file *acad.mnl* contiene il codice AutoLISP usato dal file di menu standard *acad.mnu* e viene caricato ogni volta che si carica il file *acad.mnu*.

Ad ogni compilazione di un file *.mnc*, AutoCAD genera un file di risorse di menu (*.mnr*) che contiene le bitmap usate dal menu. Il file *.mns* è un file ASCII che è inizialmente simile al file *.mnu*, ma senza commenti e formattazione speciale. Il file *.mns* viene modificato da AutoCAD ogni volta che vengono eseguite delle modifiche al contenuto del file di menu tramite l'interfaccia, ad esempio quando viene modificato il contenuto di una barra degli strumenti).

Anche se la posizione iniziale delle barre strumenti è definita nel file *.mnu* o *.mns*, le modifiche apportate allo stato mostra/nascondi e fisso/mobile oppure alla posizione delle barre degli strumenti vengono memorizzate nel registro di sistema. I file *.mns* vengono utilizzati come sorgente per la generazione futura di file *.mnc* e *.mnr*. Se il file *.mnu* viene modificato dopo la creazione del file *.mns*, è necessario usare il comando MENU per caricare in modo esplicito il file *.mnu*, in modo che AutoCAD crei nuovi file di menu in cui siano effettive le modifiche apportate.

Nota Se le barre strumenti sono state modificate tramite l'interfaccia, è necessario tagliare ed incollare le parti modificate del file

.mns nel file .mnu prima di cancellare il file .mns.

Inizialmente il comando MENU richiede il file .mns o .mnc. Per caricare di nuovo un file .mnu modificato, è necessario scegliere Modello di menu dall'elenco Tipo file e il file .mnu desiderato dall'elenco dei file. In tal modo viene impedito che il file .mns venga ricreato inavvertitamente, con una conseguente perdita delle eventuali modifiche alle barre degli strumenti ed ai menu parziali eseguite mediante l'interfaccia. Tuttavia, questa procedura può rivelarsi un po' frustrante durante la rigenerazione e la verifica di un file di menu. La routine AutoLisp riportata di seguito definisce un nuovo comando .MNU, che carica di nuovo il file .mnu corrente senza la visualizzazione della lunga serie di messaggi di richiesta.

```
(defun C:MNU ()
  (command "_menu" (strcat (getvar "nomemenu") ".mnu"))
  (princ)
)
```

Se si aggiunge questo codice al file *acad.lsp*, questo comando verrà automaticamente definito quando si riavvia AutoCAD.

Capitolo 4 -- Menu personalizzati

Uso dei file di menu

Menu di base e parziali

AutoCAD utilizza i concetti dei menu di base e parziali. Il menu di base è l'ultimo menu caricato con il comando MENU. Un menu parziale è qualsiasi menu caricato con il comando CARMENU. Il comando CARMENU carica un menu parziale e consente di aggiungere e disporre in modo diverso i menu a discesa presenti sulla barra dei menu. Quando un menu parziale viene caricato, tutte le aree di menu definite da quel menu vengono caricate in AutoCAD e risultano disponibili all'uso. Le sezioni POP non vengono visualizzate fino a che non vengono esplicitamente inserite in una barra dei menu con AutoLISP o tramite la scheda Barre dei menu della finestra di dialogo Personalizza menu. Quando viene eseguito il caricamento di un menu parziale, AutoCAD utilizza la stessa procedura per la generazione di file .mnc, .mnr e .mns descritta precedentemente. AutoCAD carica inoltre il file .mnl e .dll (file di risorse) associati, se sono disponibili.

Quando si esce da AutoCAD, vengono registrati il nome del menu di base, fino a 24 nome di menu parziali (definiti in base al corrispondente gruppo di menu) e fino a 24 menu a discesa inclusi nella barra dei menu. In tal modo l'interfaccia di AutoCAD risulta costante da una sessione alla successiva.

I menu parziali consentono agli utenti e gli sviluppatori di software di usare effettivamente più menu. AutoCAD fornisce anche un metodo conveniente per il riferimento a specifiche voci di menu. Ad ogni voce di menu viene assegnato un contrassegno di nome, mentre ad ogni file di menu viene assegnato un nome di gruppo di menu. Specificando un nome di gruppo di menu ed un contrassegno di nome, è possibile attivare, disattivare e controllare lo stato di qualsiasi voce di menu.

Capitolo 4 -- Menu personalizzati

Uso dei file di menu

Menu di base e parziali

Verifica dei menu parziali con AutoLISP

Le applicazioni AutoLISP in uso possono essere migliorate grazie ai menu parziali. Per determinare se un particolare file di menu è correntemente caricato, è possibile utilizzare la seguente sintassi con qualsiasi voce di menu POP:

```
(menucmd "Ggruppomenu_contrassegno_nome=?")
```

Se il file di menu è caricato, verrà restituito un valore non nil, in caso contrario verrà restituito il valore nil. Questa sintassi è descritta dettagliatamente nella sezione "Riferimento di menu a discesa ed a cursore."

Quando l'applicazione in uso ha stabilito se il menu è caricato o meno, può eseguire l'azione appropriata.

Caricamento di menu parziali

Gli utenti possono caricare i menu parziali e modificare la barra dei menu tramite la finestra di dialogo Personalizza menu del comando CARMENU. Perché un'applicazione possa caricare un menu parziale, è necessario utilizzare la versione della riga di

comando CARMENU. Nell'esempio riportato di seguito, viene utilizzata la funzione **command** di AutoLISP per eseguire il comando CARMENU e specificare il file di menu *menuprova* come menu parziale da caricare.

```
(command "carmenu" "menuprova")
```

Dopo aver caricato il menu, l'applicazione deve posizionare esplicitamente qualsiasi menu a discesa definito recentemente usando la funzione AutoLISP **menucmd** come suggerito negli esempi riportati di seguito.

```
(menucmd "P6+=sample.pop1")
(menucmd "Gsample.pop1+=sample.pop2")
```

Questa sintassi è illustrata nella sezione "Riferimento di menu a discesa ed a cursore."

Scaricamento di menu parziali

Il comando SCARMENU esegue lo scaricamento del menu parziale dal menu di base, il quale toglie dalla barra dei menu i menu a discesa. Inoltre, è possibile rimuovere singoli menu a discesa senza scaricare l'intero gruppo di menu. La sintassi è simile a quella usata per visualizzare i menu a discesa:

```
(menucmd "Gsample.pop1=-")
(menucmd "Gsample.pop2=-")
```

Con questo codice, i menu a discesa vengono rimossi dal menu corrente; tuttavia, il gruppo di menu è ancora caricato.

Frequenti modifiche al contenuto di una barra dei menu dell'utente degradano la qualità dell'interfaccia utente. Si raccomanda di evitare di cambiare lo stato della barra dei menu visivamente, a meno che l'utente non ne faccia esplicita richiesta. Ad esempio, se un utente desidera scaricare un'applicazione, i menu a cui viene fatto specifico riferimento da quell'applicazione potrebbero essere anch'essi rimossi.

Inoltre, il menu potrebbe essere completamente reinizializzato, rimuovendo tutti i menu parziali correntemente caricati, eseguendo il comando MENU e caricando un nuovo file di menu di base. Questa procedura rimuove tutti i menu parziali nonché le definizioni di contrassegni associate.

Capitolo 4 -- Menu personalizzati

Struttura dei file di menu

I file di menu sono divisi in *sezioni* che fanno riferimento a specifiche aree dell'interfaccia AutoCAD. A seconda della funzionalità associata, un'area di menu può essere definita da una o più sezioni. Ogni sezione contiene *voci di menu* che forniscono istruzioni per l'aspetto e l'azione della selezione del menu. Le voci di menu comprendono i seguenti elementi: un *contrassegno di nome*, un'*etichetta* ed una *macro di menu*. Sebbene le voci di menu siano simili per struttura e funzionamento, ogni sezione utilizza una sintassi speciale per le proprie etichette di voce di menu.

Non è necessario che un file di menu contenga voci per ogni sezione di menu, quanto occorre avere soltanto le entrate che riguardano la propria applicazione. In effetti si consiglia di suddividere sempre i menu in file di menu più piccoli che possono essere caricati e scaricati quando richiesto. A tale scopo, usare il comando MENULOAD. Ciò consente non solo di controllare meglio le risorse del sistema, ma anche di semplificare il lavoro di sviluppo e di gestione in quanto si può lavorare con file più piccoli in base alle diverse esigenze.

Le sezioni dei file di menu sono identificate dalle etichette di sezione che utilizzano il formato *****nome_sezione**. Le etichette di sezione e le aree di menu associate sono elencate nella tabella riportata di seguito.

Etichette di sezione e menu associati

Etichetta di sezione	<i>Area di menu</i>
***MENUGROUP=	<i>Nome di gruppo del file di menu.</i>
***BUTTONS<i>n</i>	<i>Menu pulsanti dispositivo di puntamento.</i>
***AUX<i>n</i>	<i>Menu di dispositivo di puntamento.</i>
***POP<i>n</i>	<i>Area di menu a discesa/cursore.</i>
***TOOLBARS	<i>Definizioni delle barre degli strumenti.</i>
***IMAGE	<i>Area di menu a gruppi di immagini.</i>
***SCREEN	<i>Area di menu di schermo.</i>
***TABLET<i>n</i>	<i>Area di menu di tavoletta.</i>
***HELPSTRINGS	<i>Testo visualizzato nella barra di stato quando una voce di un menu a discesa o cursore viene evidenziata oppure quando il cursore si trova su un pulsante di una barra degli strumenti.</i>
***ACCELERATORS	<i>Definizione di tasti di scelta.</i>

Ogni sezione contiene voci di menu che definiscono l'aspetto e la funzionalità delle selezioni di menu.

Nei file di menu è possibile includere commenti da usare come avvisi di copyright, documentazione o note. Qualsiasi riga che inizia con due barre (//) viene ignorata dal compilatore di menu, come illustrato nell'esempio riportato di seguito.

```
//Questa riga è un commento
```

Capitolo 4 -- Menu personalizzati

Sintassi delle voci di menu

La sintassi generale delle voci di menu risulta consistente in tutte le sezioni di menu che includono voci di menu. Ogni voce di menu può essere composta da un contrassegno di nome, un'etichetta ed una macro di menu (ad eccezione della sezione Image, che non consente contrassegni di nome). Una voce di menu si trova in genere su una riga del file ed utilizza il seguente formato:

```
contrassegno_nome etichetta macro_menu
```

L'esempio di codice di menu POP (a discesa) che segue illustra una semplice voce di menu:

```
##ID_Quit [Esci]^C^C_quit
```

La prima voce ##ID_Quit è il contrassegno di nome, mentre l'etichetta [Esci] visualizza il testo *Esci* nel menu. Quando l'utente seleziona questa voce di menu, viene eseguita la macro di menu ^C^C_quit.

Capitolo 4 -- Menu personalizzati

Sintassi delle voci di menu

Contrassegni di nome

Il contrassegno di nome è una stringa formata da caratteri alfanumerici e dall'etichetta della voce di menu preceduta da un trattino di sottolineatura (_). Questa stringa identifica unicamente un elemento in un file di menu. I contrassegni di nome forniscono la seguente funzionalità:

- Collegamento delle voci dei menu e dei pulsanti delle barre degli strumenti ai messaggi di guida dalla riga di stato ad essi relativi (vedere "Guida specifica di menu").
- Collegamento delle sequenze di tasti ad una macro di menu POP corrispondente (vedere "Tasti di scelta").
- Passaggio da uno stato all'altro delle voci di menu, ad esempio da *attivato* a *disattivato* e da *contrassegnato* a *non contrassegnato*, in macro di menu o AutoLISP (vedere "Controllo della visualizzazione di etichette di voci di menu").

I contrassegni di nomi *non* sono consentiti nelle sezioni Buttons, Aux ed Image, mentre sono consentiti nelle sezioni Screen e Tablet, ma non svolgono alcuna funzione.

Capitolo 4 -- Menu personalizzati

Sintassi delle voci di menu

Etichette

Il formato e l'uso delle etichette delle voci di menu varia a seconda della sezione di menu. L'etichetta è racchiusa tra parentesi quadre ([]) e definisce ciò che deve essere visualizzato o presentato all'utente. Le sezioni di menu prive di interfaccia per la visualizzazione delle informazioni non richiedono etichette (ad esempio, le sezioni Buttons, Aux e Tablet), ma possono essere utilizzate per annotazioni interne. Nella tabella che segue è viene descritto la modalità di utilizzo delle etichette in ciascuna sezione di menu.

Utilizzo delle etichette di sezioni di menu

Sezione di menu	<i>Usa dell'etichetta</i>
BUTTONSn, AUXn e POPn	<i>Definisce il contenuto e la formattazione delle selezioni dei menu a discesa e cursore.</i>
TOOLBARS	<i>Definisce nome, stato (mobile o fissa, nascosta o visibile) e posizione della barra degli strumenti, nonché i vari pulsanti e le corrispondenti proprietà.</i>
IMAGE	<i>Definisce il testo e l'immagine visualizzati nei menu a gruppi di immagini.</i>
SCREEN	<i>Definisce il testo che viene visualizzato nei menu di schermo.</i>
HELPSTRINGS	<i>Definisce i messaggi di guida sulla riga di stato relativi alle voci dei menu POP e delle barre degli strumenti.</i>
ACCELERATORS	<i>Associa l'azione della sequenza di tasti alle macro di menu.</i>

La sintassi e la formattazione specifiche per le etichette di menu sono descritte nelle sezioni successive.

Capitolo 4 -- Menu personalizzati

Sintassi delle voci di menu

Macro di menu

Se si intende includere i parametri di comando in una voce di menu, è necessario conoscere la sequenza in cui quel comando prevede di trovare i propri parametri. Ogni carattere di una voce di menu è significativo, perfino uno spazio. In seguito alla revisione ed all'ulteriore miglioramento di AutoCAD, la sequenza dei messaggi di richiesta per i vari comandi (e talvolta perfino i nomi dei comandi) può variare; per questo motivo, i menu personalizzati possono richiedere un numero minore di modifiche quando si effettua l'aggiornamento di AutoCAD installandone una nuova versione.

La maggior parte degli esempi in questa sezione sono validi specificatamente per le sezioni dei menu di schermo e POP; tuttavia, anche se a volte richiedono piccole modifiche o addirittura nessuna modifica, la maggior parte delle macro di menu funzionano altrettanto bene in tutte le altre sezioni.

Quando l'input di un comando proviene da una voce di menu, le variabili di sistema PICKADD e PICKAUTO vengono impostate rispettivamente su 1 e 0. Ciò consente di mantenere la compatibilità con le release precedenti di AutoCAD e di facilitare la personalizzazione, poiché non è necessario verificare le impostazioni di queste variabili.

Capitolo 4 -- Menu personalizzati

Sintassi delle voci di menu

Macro di menu

Sintassi delle macro di menu

La tabella seguente fornisce un sommario dei caratteri speciali utilizzati nelle macro di menu. L'uso di questi caratteri viene descritto dettagliatamente nelle sezioni successive del presente capitolo.

Caratteri speciali utilizzati nelle macro di menu

Carattere	Descrizione
;	<i>Invia RETURN.</i>
^M	<i>Invia RETURN.</i>
^I	<i>Invia TAB.</i>
SPACEBAR	<i>Inserisce uno spazio; uno spazio tra le sequenze di comandi in una voce di menu equivale a premere la SPACEBAR.</i>
\	<i>Attende l'input dell'utente</i>
_	<i>Converte i comandi e le parole chiave di AutoCAD che seguono.</i>
+	<i>Se è l'ultimo carattere, continua una macro di menu sulla riga seguente.</i>
=*	<i>Visualizza il menu a icone, a discesa o a cursore corrente.</i>
*^C^C	<i>Prefisso per una voce che si ripete.</i>

\$	<i>Codice di carattere speciale che carica una sezione di menu o introduce un'espressione DIESEL macro condizionale (\$M=).</i>
^B	<i>Attiva o disattiva Snap (CTRL+B).</i>
^C	<i>Annulla un comando (ESC).</i>
^D	<i>Attiva o disattiva Coord (CTRL+D).</i>
^E	<i>Imposta il successivo piano isometrico (CTRL+E).</i>
^G	<i>Attiva o disattiva Griglia (CTRL+G).</i>
^H	<i>Invia backspace.</i>
^O	<i>Attiva o disattiva Orto (CTRL+O).</i>
^P	<i>Attiva o disattiva MENECHO.</i>
^Q	<i>Invia alla stampante tutti i messaggi di richiesta, gli elenchi di stato e l'input (CTRL+Q).</i>
^T	<i>Attiva o disattiva la tavoletta (CTRL+T).</i>
^V	<i>Cambia la finestra corrente (CTRL+V).</i>
^Z	<i>Carattere nullo che sopprime l'aggiunta automatica di SPACEBAR alla fine di una voce di menu.</i>

Capitolo 4 -- Menu personalizzati

Sintassi delle voci di menu

Macro di menu

Terminazione delle macro

Quando una voce di menu viene selezionata, AutoCAD inserisce uno spazio alla fine della macro prima di elaborare la sequenza dei comandi. AutoCAD elabora la seguente macro di menu come se fosse stato inserito **linea** SPACEBAR.

```
[Linea]linea
```

A volte, non si desidera che ciò avvenga; ad esempio, il comando TESTO o DIM deve essere completato con RETURN e non con uno spazio. Inoltre, qualche volta per completare un comando è necessario più di un semplice spazio (o RETURN), ma in alcuni editor di testo non è possibile creare una linea con spazi finali. Per risolvere questi problemi si ricorre a due convenzioni speciali:

- Quando un punto e virgola (;) appare in una macro di menu, AutoCAD lo sostituisce con un RETURN.
- Se una linea finisce con un carattere di controllo, una barra rovesciata (\), un segno più (+) o un punto e virgola (;), AutoCAD *non* aggiunge uno spazio. Per un esempio dell'uso di un carattere di controllo come carattere per la terminazione di una voce di menu, vedere "Espressioni macro condizionali."

Si osservi la voce di menu Cancella 1 nell'esempio seguente:

```
[Cancella 1]cancella \;
```

Se questa voce finisce semplicemente con la barra rovesciata (indicante l'input dell'utente), l'operazione di cancellazione non può essere completata, in quanto AutoCAD non aggiunge uno spazio dopo la barra rovesciata. Perciò, questa macro di menu usa un punto e virgola (;) per fare in modo che dopo l'input utente vi sia un RETURN. Di seguito vengono riportati ulteriori esempi.

```
[UCS      ] ucs
[UCS G    ] ucs ;
[Indirizzo] testo \.4 0 DRAFT Inc;;;Via.;;;Città, Stato;
```

Selezionando la prima voce si immette **ucs** e SPACEBAR alla riga di comando, provocando la visualizzazione del seguente messaggio di richiesta:

Origine/Asse-Z/3punti/Entità/Vista/X/Y/Z/Precedente/Ripristina/Salva/Cancella/?/ <Globale>:

Selezionando la seconda voce si immette **ucs**, SPACEBAR e ; (interpretato come RETURN) alla riga di comando, che accetta il valore di default, Globale. L'uguaglianza tra la prima e la seconda voce risulterebbe evidente sullo schermo; naturalmente sarebbe inutile porre entrambi le voci nello stesso menu.

Selezionando la terza voce è possibile visualizzare un messaggio di richiesta per un punto di inizio e poi disegnare l'indirizzo su tre righe. Il primo dei tre punti a virgola (;;;) termina la stringa di testo, il secondo provoca la ripetizione del comando TESTO ed il terzo richiama il posizionamento di default al di sotto della riga precedente.

Capitolo 4 -- Menu personalizzati

Sintassi delle voci di menu

Macro di menu

Attesa dell'input utente

Talvolta risulta utile accettare l'input dalla tastiera o dal dispositivo di puntamento durante l'elaborazione di una macro di menu inserendo una barra rovesciata (\) nel punto in cui si desidera venga inserito l'input.

```
[Cerchio-1]cerchio \1
[Layoff ]layer off \;
```

Cerchio-1 effettua una pausa per richiedere all'utente di digitare il punto centrale e successivamente legge un raggio di lunghezza 1 dal menu. Si noti che non vi è spazio dopo la <:df

Di norma, l'esecuzione della macro di menu riprende dopo l'immissione di una voce. Per questo motivo, non è possibile costruire una macro di menu che accetti un numero variabile di input (come nella selezione di oggetti) e poi continua. Tuttavia, per il comando SELEZ esiste un'eccezione; mediante una barra rovesciata è possibile sospendere l'esecuzione della voce di menu fino a quando la selezione dell'oggetto non è stata completata. Ad esempio, si consideri la seguente voce di menu:

```
[Rosso]selez \modifica il precedente ;proprietà del colore rosso ;
```

Questa voce usa il comando SELEZ per creare un gruppo di selezione composto da uno o più oggetti. Successivamente invia il comando CAMBIA, fa riferimento a questo gruppo di selezione tramite l'opzione Precedente e cambia in rosso il colore di tutti gli oggetti selezionati.

Nota In una macro di menu, dato che la barra rovesciata (\) fa in modo che la macro effettui una pausa in attesa di input da parte dell'utente, *non è possibile* usare tale barra per altri scopi. Quando si specificano i percorsi delle directory utilizzare una barra (/) come carattere di delimitazione del percorso, come in */direct/file*.

Le situazioni che possono provocare ritardi nella ripresa dell'esecuzione di una macro di menu sono le seguenti:

- Se è previsto l'input di un punto, è possibile che le modalità di snap ad oggetto precedano l'immissione del punto vero e proprio.
- Se vengono utilizzati i filtri X/Y/Z, la voce di un menu rimane sospesa fino a quando l'intero punto non è stato accumulato.
- Solo per il comando SELEZ, l'esecuzione della voce di menu non viene ripresa fino a quando la selezione dell'oggetto non è stata completata.
- Se l'utente risponde con un comando trasparente, la macro di menu sospesa rimane tale fino a quando il comando trasparente non è completato e l'input richiesto originariamente non viene ricevuto.
- Se l'utente risponde scegliendo un'altra voce di menu (per fornire le opzioni o per eseguire un comando trasparente), la macro originale viene sospesa e viene elaborata l'ultima voce selezionata fino al suo termine prima che l'esecuzione della macro sospesa venga ripresa.

Capitolo 4 -- Menu personalizzati

Sintassi delle voci di menu

Macro di menu

Supporto per lingue straniere nelle macro di menu

Se si sviluppano menu che è possibile vengano usati con una versione di AutoCAD in un'altra lingua, i comandi e le parole chiave standard di AutoCAD verranno tradotti automaticamente se si fa precedere ogni comando o parola chiave da un trattino di sottolineatura (_). Il file *acad.mnu* usa questa funzione piuttosto di frequente.

L'esempio che segue mostra una parte di un menu POP.

```
[->Arco]
[3punti]^C^C_arc
[Inizio, Centro, Fine]^C^C_arc;\_c
[Inizio, Centro, Angolo]^C^C_arc;\_c;\_a
[Inizio, Centro, Lunghezza]^C^C_arc;\_c;\_1
[Inizio, Fine, Angolo]^C^C_arc;\_e;\_a
```

```
[Inizio, Fine, Raggio]^C^C_arc;\_e;\_r
```

Da notare inoltre che la funzione AutoLISP **getcname** può essere utilizzata per visualizzare il nome inglese o localizzato di un comando di AutoCAD. Per ulteriori informazioni, vedere "getcname."

Capitolo 4 -- Menu personalizzati

Sintassi delle voci di menu

Macro di menu

Annullamento di un comando

Per accertarsi che non vi siano comandi precedenti non completati, utilizzare la stringa **^C^C** in una voce di menu. Ciò equivale a premere ESC per due volte dalla tastiera. Sebbene un solo **^C** consenta di annullare la maggior parte dei comandi, è necessario usare **^C^C** per tornare alla riga di comando di un comando DIM . Quindi, **^C^C** garantisce che AutoCAD torni alla riga di comando nella maggior parte dei casi.

Capitolo 4 -- Menu personalizzati

Sintassi delle voci di menu

Macro di menu

Visualizzazioni e messaggi di richiesta

Di norma, i caratteri letti da una voce di menu compaiono nell'area di comando dello schermo proprio come l'input da tastiera e i messaggi di richiesta vengono visualizzati anche se una voce di menu fornisce le risposte. Per eliminare la visualizzazione di questi messaggi, utilizzare la variabile di sistema **MENUECHO**. Se la visualizzazione dell'input di menu è disattivata, un **^P** nella voce di menu permette di attivarla.

Capitolo 4 -- Menu personalizzati

Sintassi delle voci di menu

Macro di menu

Caratteri di controllo nelle voci di menu

Inserendo un accento circonflesso (^) seguito da un altro carattere, è possibile inserire dei caratteri di controllo ASCII nella parte relativa alle stringhe dei comandi di una voce di menu. Ad esempio, **^C** verrà convertito in un solo carattere, CTRL+C (che quando richiamato da una macro di menu, esegue Annulla e non Copiapp). I caratteri di controllo non alfabetici sono i seguenti:

```
^@      (codice ASCII 0)
^[      (codice ASCII 27)
^\      (codice ASCII 28)
^]      (codice ASCII 29)
^^      (codice ASCII 30)
^_      (codice ASCII 31)
```

Quando viene usato in una macro di menu, l'accento circonflesso (^) viene mappato con il carattere CTRL della tastiera. È possibile unire questo carattere con qualsiasi altro per costruire macro di menu che consentano, ad esempio, di attivare e disattivare la griglia (**^G**) o di annullare un comando (**^C**).

```
[InvertiGriglia]^G
[*Annulla*]^C
```

È possibile che si desideri che una macro di menu digiti uno o più caratteri, ma non li immetta come input finale. Ad esempio, è

possibile creare una serie di voci di menu che agiscano come un tastierino numerico.

```
[1] 1x^H
[2] 2x^H
[3] 3x^H
```

Quando si sceglie una di queste voci, viene inserita la cifra appropriata. Un altro carattere segue la cifra (la lettera x in questo caso), e quel carattere viene rimosso da ^H. (CTRL+H è il codice ASCII per un backspace). Ciascuna di queste voci di menu termina con un carattere di controllo, nel qual caso AutoCAD non aggiunge uno spazio o RETURN a queste voci. Così è possibile scegliere [2], [2], [3], [1], costruendo l'input 2231. Premere RETURN per immettere il numero completato.

Le seguenti voci di menu eseguono la funzione appena descritta, questa volta usando la barra rovesciata.

```
[1] 1\
[2] 2\
[3] 3\
```

Si raccomanda di usare il primo di questi due metodi. Infatti, sebbene il secondo sia più semplice da implementare e nella maggior dei casi produca lo stesso risultato, è possibile che un comando inviato mentre è attiva una pausa di menu non funzioni come previsto.

Per assegnare macro di menu a tasti e sequenze di tasti, vedere "Tasti di scelta."

Capitolo 4 -- Menu personalizzati

Sintassi delle voci di menu

Macro di menu

Lunghezza delle macro di menu

Se una voce del file di menu non rientra in una riga, è possibile continuarla sulla riga successiva. Per far ciò, porre un segno più (+) come ultimo carattere della linea da continuare.

```
[Imposta ]layer gruppo piano-terra;;griglia on; ... ;pieno off;+
limiti 0,0 12,9;stato
```

Questa voce, che potrebbe essere usata per impostare le condizioni iniziali per un nuovo disegno, continua su una seconda riga. Le voci di menu possono estendersi su tante righe quante necessarie.

Nota Le righe di continuazione *non* sono conservate quando viene creato il file .mns.

Si consiglia di inviare macro di menu molto lunghe come comandi definiti da AutoLISP o funzioni definite in un file .mnl. In tal modo la lettura del menu risulta più semplice e viene creato un codice più modulare.

Capitolo 4 -- Menu personalizzati

Sintassi delle voci di menu

Macro di menu

Ripetizione delle macro di menu

Una volta selezionato un comando, è probabile che lo si utilizzi più volte prima di passare ad un altro comando. Nella maggior parte dei casi gli strumenti vengono usati in questo modo: si sceglie uno strumento, si eseguono diverse operazioni con esso, quindi se ne sceglie un altro e così via. Per evitare di selezionare lo strumento prima di utilizzarlo ogni volta, AutoCAD fornisce la possibilità di ripetere un comando, avviandolo mediante una risposta nulla. Questa funzione, tuttavia, non può essere usata per specificare opzioni di comando.

Essa consente di ripetere un comando usato di frequente fino a quando non se ne sceglie un altro. Se una macro di menu comincia con ^C^C immediatamente dopo l'etichetta di voce, la macro viene memorizzata. Quella macro risponderà alle successive richieste della riga di comando fino a quando la sua esecuzione non viene terminata con ESC o mediante la selezione di un'altra voce di menu.

Nota *Non* utilizzare ^C (annulla) all'interno di una voce di menu che comincia con la stringa ^C^C; ciò annulla la ripetizione della

macro di menu.

Di seguito è riportato un esempio dell'approccio ripetitivo o modale della gestione dei comandi.

```
[Sposta ]*^C^CMOVE Singola
[Copia ]*^C^CCOPY Singola
[Cancella ]*^C^CERASE Singola
[Stira ]*^C^CSTRETCH Interseca singola
[Ruota ]*^C^CROTATE Singola
[Scala ]*^C^CSCALE Singola
```

La ripetizione delle macro di menu non funziona per le voci dei menu a gruppi di immagini.

Capitolo 4 -- Menu personalizzati

Sintassi delle voci di menu

Macro di menu

Uso della modalità a selezione singola di oggetti

La selezione singola di oggetti pone la selezione degli oggetti in modalità di selezione singola, disattiva il normale dialogo condotto dalla selezione di oggetti e fa in modo che la selezione restituisca il primo oggetto o i primi oggetti selezionati da un'opzione successiva. In un menu ciò può risultare piuttosto comodo. Ad esempio, si consideri la seguente voce di menu:

```
[Cancella]*^C^CERASE unica
```

Questa voce termina il comando corrente ed attiva il comando CANCELLA con l'opzione di selezione Singola. Dopo aver selezionato questa voce, si può puntare al singolo oggetto da cancellare oppure si può prima puntare ad un'area vuota e poi specificare una finestra. L'oggetto o gli oggetti selezionati in questo modo vengono cancellati e la voce di menu viene ripetuta (a causa dell'asterisco iniziale) in modo tale da poter cancellare qualche altro elemento. La modalità a selezione singola consente una maggiore interazione dinamica con AutoCAD.

Capitolo 4 -- Menu personalizzati

Sintassi delle voci di menu

Macro di menu

Scambio tra menu

AutoCAD fornisce un metodo per la sostituzione del contenuto di una sezione di menu Buttons, Aux, Pop, Screen o Tablet attiva. Il contenuto del nuovo menu può essere quello di un'altra sezione o di un altro sottomenu nel menu di base oppure quello di un menu parziale. Le informazioni sullo scambio tra menu relative ad una sezione di menu vengono fornite insieme alla descrizione di quella sezione.

La seguente sintassi consente di passare da un menu ad un altro da una macro di menu:

```
§sezione= [gruppomenu . ] nomemenu
```

dove

§

Indica ad AutoCAD di caricare una sezione di menu.

sezione

Specifica la sezione di menu. I nomi validi sono:

A1-A4 per le sezioni di menu AUX da 1 a 4
 B1-B4 per le sezioni di menu BUTTONS da 1 a 4
 P0-P16 per le sezioni di menu POP da 0 a 16
 I per le sezioni di menu IMAGE

S per le sezioni di menu SCREEN
 T1-T4 per le sezioni di menu TABLET da 1 a 4

gruppomenu

Specifica il gruppomenu di cui *nomenu* è un membro (non necessario se *nomemenu* è nel menu di base).

nomemenu

Specifica quale sezione o sottomenu da inserire. Etichetta principale o alias per la sezione da caricare.

Le seguenti voci di menu mostrano il riferimento di sottomenu.

```
$S=PARTI
$T1=EDITCMDS
$T2=SCREEN
```

Il meccanismo dei sottomenu può essere attivato durante l'esecuzione di un comando senza che questo venga interrotto. Ad esempio, le seguenti stringhe di comandi sono equivalenti:

```
$S=PARAMETRI_ARCO ARCO
ARCO $S=PARAMETRI_ARCO
```

Ognuno di questi comandi avvia il comando ARCO, passa al sottomenu di schermo PARAMETRI_ARCO ed attende l'immissione di parametri del comando Arco. Il riferimento al sottomenu deve essere seguito da uno spazio in modo da separarlo dai successivi comandi della voce di menu.

Nota È possibile eseguire scambi solo tra menu dello stesso tipo, ad esempio da un menu Aux a un altro menu Aux, da un menu Pop a un altro menu di questo tipo e così via. Lo scambio tra tipi diversi potrebbe produrre risultati imprevedibili e indesiderati. Tuttavia, nell'ambito di un determinato tipo, è possibile spostare qualsiasi menu al posto di un altro. Ciò può produrre risultati inaspettati nel funzionamento dei menu Tablet, in quanto questi, generalmente, non hanno lo stesso numero di macro.

La barra dei menu o il menu cursore attivo può includere una sola occorrenza di ciascun menu Pop.

Capitolo 4 -- Menu personalizzati

Sintassi delle voci di menu

Macro di menu

Espressioni macro condizionali

Il comando \$M= può essere utilizzato all'interno di una macro di menu per introdurre espressioni macro scritte in DIESEL (vedere il capitolo 5, "Linguaggio DIESEL e configurazione della riga di stato"). AutoCAD valuta la parte della voce che segue \$M= e riesamina il risultato come macro di menu. Il formato è:

\$M=espressione

dove \$M= indica ad AutoCAD di valutare la stringa seguente come espressione DIESEL ed *espressione* rappresenta l'espressione DIESEL.

```
[FILLFLIP]FILLMODE $M=$((-1,$(getvar,fillmode))
```

Fillflip attiva e disattiva FILLMODE sottraendo il valore corrente di FILLMODE da 1 e restituendo il valore risultante alla variabile di sistema FILLMODE. Questo metodo può essere usato per attivare e disattivare gli stati delle variabili di sistema i cui valori validi siano 1 o 0.

Se si usa il linguaggio DIESEL a stringhe per eseguire verifiche "if-then", le condizioni possono essere poste nei punti in cui non si desidera che vi sia il normale spazio o punto e virgola di terminazione (risultante in RETURN). Se la voce di menu è seguita da ^Z, AutoCAD non aggiunge RETURN alla fine dell'espressione macro.

Nota Come con altri caratteri di controllo nelle voci di menu, ^Z usato in questo caso è una stringa composta da ^ (un accento circonflesso) e Z, non da CTRL+Z.

Negli esempi seguenti, ^Z viene usato come carattere per la terminazione delle macro di menu.

```
[Spazio Modello]^C^C_mspace $M=$((if,$(=,$(getvar,tilemode),0),$S=mview)^Z
[Spazio Carta]^C^C_pspace $M=$((if,$(=,$(getvar,tilemode),0),$S=mview)^Z
```

Se queste macro di menu non terminassero con ^Z, AutoCAD aggiungerebbe RETURN, rinviando inutilmente l'ultimo comando digitato. Per ulteriori informazioni ed esempi, vedere il capitolo 5, "Linguaggio DIESEL e configurazione della riga di stato".

Capitolo 4 -- Menu personalizzati

Sintassi delle voci di menu

Macro di menu

Uso di AutoLISP nelle macro di menu

È possibile utilizzare variabili ed espressioni AutoLISP, descritte nella parte II "AutoLISP", per creare macro di menu per l'esecuzione di operazioni complesse. In questa sezione vengono riportati alcuni esempi utili.

AutoCAD accetta fino a 255 caratteri di codice AutoLISP alla volta nelle macro di menu. Per usare un numero maggiore di caratteri, interrompere il codice in più moduli separati da punti e virgola (;), in modo tale che AutoCAD possa leggere ed eseguire il codice in blocchi.

Per informazioni circa l'uso di AutoLISP per richiamare una finestra di dialogo da un file di menu, vedere il capitolo 10, "Gestione delle finestre di dialogo".

Per usare AutoLISP in maniera utile ed efficace nelle macro di menu, è possibile porre il codice AutoLISP in un file *.mnl* a parte. Così facendo, il file di menu ed il codice AutoLISP potranno essere scritti e gestiti più facilmente. AutoCAD carica il file *.mnl* quando carica un file di menu con lo stesso nome.

Esecuzione di una macro di menu

Per eseguire da programma una macro di menu Pop come se fosse stata selezionata dall'utente, è possibile utilizzare la seguente sintassi:

```
(menucmd "Gruppomenu ,contrassegno_nome=?")
```

Questa sintassi è valida solo per una macro di menu Pop che fa parte di un menu Pop caricato nella barra dei menu di AutoCAD (deve essere disponibile correntemente per l'utente). Per ulteriori informazioni su questa sintassi, vedere "Riferimento di menu a discesa ed a cursore."

Valori preimpostati

Un'applicazione che utilizza inserimenti di blocchi preimpostati potrebbe fornire voci di menu come queste:

```
[Imposta:    ]
[ LARGFIN ]^C^C(setq LARGFIN (getreal "\nLarghezza della finestra: ))
[ SPESSMUR ]^C^C(setq SPESSMUR (getreal "\nSpessore del muro: ))
[Finestra  ]^C^CINSERT Scala X finestra !LARGFIN Scala Y finestra!SPESSMUR
```

Questo codice dovrebbe inserire il blocco relativo alla finestra, adattando la scala del relativo asse X alla larghezza della finestra corrente e l'asse Y allo spessore corrente del muro. In questo esempio, i valori effettivi provengono dai simboli AutoLISP definiti dall'utente LARGFIN e SPESSMUR. La rotazione deve essere decisa dall'utente in modo tale da consentirgli di far ruotare la finestra sul muro.

Ridimensionamento dei grip

Le seguenti voci di menu consentono di adeguare all'istante la dimensione dei grip:

```
[GRIP-su    ]^P(setvar"gripsize"(1+(getvar"gripsize")))(redraw)(princ)
[GRIP-giù]^P(setvar"gripsize"(1-(getvar"gripsize")))(redraw)(princ)
```

Può risultare utile aggiungere il controllo di validità a queste voci di menu; è possibile fare in modo che i valori inferiori a 0 e maggiori di 255 non siano consentiti per la variabile di sistema GRIPSIZE.

Richiesta di input dell'utente

La seguente voce di menu richiede all'utente di specificare due punti, quindi disegna un poligono rettangolare utilizzando i punti specificati come angoli.

```
[CASELLA](setq a (getpoint "Primo angolo: "));\ +
(setq b (getpoint "Angolo opposto: "));\ +
pline !a (list (car a) (cadr b))!b (list (car b) (cadr a))c;
```

Questa è una funzione avanzata della funzione di personalizzazione dei menu di AutoCAD. Se si desidera creare voci di menu di questo tipo, studiare attentamente l'esempio precedente e le informazioni contenute nella parte II, "AutoLISP". L'esercizio e numerosi tentativi saranno utili per imparare ad usare questa funzione in modo efficiente.

Capitolo 4 -- Menu personalizzati

Gruppi di menu.

L'etichetta *****MENUGROUP=** definisce il contenuto di un file di menu che sarà membro di un particolare gruppo. Lo scopo di questo nome di gruppo è di distinguere le sue voci dalle altre durante l'uso di menu parziali. Questa etichetta *deve* precedere tutte le sezioni di menu che usano il meccanismo del contrassegno di nome, in quanto definisce il nome del gruppo di menu di queste sezioni. Una definizione Menugroup è una stringa composta da un massimo di 32 caratteri alfanumerici (gli spazi ed i segni di punteggiatura non sono consentiti).

Poiché ogni menu ha la propria etichetta di gruppo di menu, più menu parziali possono utilizzare lo stesso contrassegno di nome. Un contrassegno di nome, dunque, deve essere univoco solo all'interno del file di menu in cui è stato definito. Ciò consente, inoltre, di creare applicazioni che caricano file di menu parziali in modo che un menu possa accedere alle voci dell'altro.

I nomi dei gruppi di menu combinati con i nomi o gli alias dei menu POP forniscono la funzionalità seguente:

- Caricamento e scaricamento interattivo di menu parziali (vedere il comando **MENULOAD** nella *Guida di riferimento dei comandi di AutoCAD*).
- Controllo della visualizzazione e del layout dei menu a discesa da macro di menu o da AutoLISP (vedere "Controllo della visualizzazione di etichette di voci di menu").

Nota La sezione Menugroup non contiene voci di menu.

Capitolo 4 -- Menu personalizzati

Menu di pulsanti ed ausiliari

Il formato dei menu di pulsanti (*****BUTTONS n**) e dei menu ausiliari (*****AUX n**) è identico. L'uso degli uni o degli altri dipende dal tipo di dispositivi di puntamento usati. Il mouse di sistema utilizza i menu ausiliari, mentre ogni altro dispositivo di puntamento (ad esempio, un mouse del digitalizzatore o qualsiasi altro dispositivo di input) utilizza i menu di pulsanti. Tutti i riferimenti ai menu dei pulsanti valgono anche per i menu ausiliari; il menu **BUTTONS1** opera esattamente come il menu **AUX1** e così via.

Capitolo 4 -- Menu personalizzati

Menu di pulsanti ed ausiliari

Creazione di menu di pulsanti ed ausiliari

Le sezioni **AUX n** del file di menu definiscono le macro di menu associate ai pulsanti del mouse. Ogni riga di questa sezione rappresenta un pulsante del mouse. Per accedere a ciascun menu dei pulsanti, è possibile usare le sequenze tasto/pulsante indicate nella tabella seguente.

Pulsanti e sezioni di menu associate

Sequenza tasto/pulsante	Sezioni di menu
Pulsante semplice	AUX1 e BUTTONS1
SHIFT + pulsante	AUX2 e BUTTONS2
CTRL + pulsante	AUX3 e BUTTONS3
CTRL + SHIFT + pulsante	AUX4 e BUTTONS4

Sebbene le aree 1-4 siano le sole sezioni attive, è possibile definire ulteriori sezioni e renderle attive. In precedenza, questa operazione poteva essere eseguita con i sottomenu (ad esempio, ****othermenu**). La sintassi ****label** viene ora utilizzata per gli alias delle sezioni nelle sezioni Buttons ed Aux. Per informazioni sullo scambio tra menu, vedere "Scambio dei menu di pulsanti ed ausiliari."

La creazione o la personalizzazione dei menu dei pulsanti può rendere più efficiente e dinamico l'uso dei dispositivi di puntamento.

Selezionando le voci di menu ed aggiungendole al menu dei pulsanti, è possibile personalizzare il dispositivo di puntamento in modo da soddisfare le proprie necessità.

Esaminare il seguente esempio di sezione AUX1 (simile al file *acad.mnu* standard):

```
***AUX1
;
^C^C
^B
^O
^G
^D
^E
^T
```

La prima riga dopo l'etichetta della sezione di menu, ***AUX1, rappresenta il pulsante successivo a quello di selezione sul dispositivo di puntamento. Se il pulsante di selezione è il pulsante numero 1, il punto e virgola (;) assegna RETURN al pulsante numero 2 sul dispositivo di puntamento. La seconda riga dopo l'etichetta della sezione di menu rappresenta il terzo pulsante.

Nota Il pulsante di selezione *non* può essere riassegnato. Esso può essere diverso a seconda del dispositivo di puntamento, in base alle caratteristiche determinate dall'azienda produttrice.

Poiché le etichette nei menu dei pulsanti non vengono visualizzate, è possibile utilizzarle come commenti. L'esempio seguente usa l'area dell'etichetta per annotare il numero del pulsante.

```
***AUX1
[pulsante n.2];
[pulsante n.3]$P0=*
[pulsante n.4]^C^C
```

La macro assegnata al pulsante numero 3 nell'esempio consente di visualizzare un altro menu. Il formato è il seguente:

```
$Pn=*
```

dove \$ è il codice di carattere speciale per il caricamento di un'area di menu, Pn specifica la sezione di menu POP e *= visualizza ciò che è correntemente caricato nell'area di menu specificata.

Perciò, nell'esempio del file *acad.mnu*, scegliendo il pulsante numero 3 è possibile visualizzare il menu assegnato all'area di menu P0 (tale area è il menu a cursore). Di solito, la sezione ***POP0 del file di menu è assegnata all'area di menu P0.

Le altre righe della sezione assegnano ciascuna una sequenza di comando ad ogni pulsante successivo del dispositivo di puntamento, ad esempio, ^C^C (ESC per due volte) al pulsante 4, ^B (attivazione/disattivazione della modalità Snap) al pulsante 5 e così via. Il dispositivo di puntamento in uso può riconoscere tante linee quanti sono i pulsanti assegnabili.

Capitolo 4 -- Menu personalizzati

Menu di pulsanti ed ausiliari

Scambio dei menu di pulsanti ed ausiliari

Le sezioni Buttons ed Aux utilizzano una nuova sintassi per lo scambio dei menu; i sottomenu non sono più disponibili. Al contrario, quelli che in precedenza erano sottomenu, attualmente sono sezioni di menu complete. Ciascuno di questi menu richiede una ***sezione in cui *sezione* è una stringa ininterrotta costituita dalla parola chiave appropriata per indicare di che tipo di menu si tratta (ad esempio ***BUTTONS, ***AUX), seguita dal testo aggiuntivo desiderato.

Nota Il funzionamento dell'utilizzo dei caratteri di testo dopo le parole chiave non è garantito nelle release future di AutoCAD. Per garantire la compatibilità, è necessario utilizzare solo numeri indicizzati che *seguono* le parole chiave. Quindi, anche se etichette come ***AUXTEST e ***BUTTONS1-2 sono valide, è preferibile utilizzare etichette come ***AUX10 o ***BUTTONS15 in quanto la loro compatibilità è garantita a lungo termine.

La sintassi ***etichetta* indica gli alias di una determinata sezione. Le etichette devono trovarsi tra la riga ***sezione e la riga della prima voce di menu per quella sezione. La stringa dell'*etichetta* alias può essere una stringa qualsiasi (non è necessario che contenga una parola chiave). Sebbene per ciascuna sezione sia possibile associare un numero qualsiasi di alias, si consiglia di specificarne il numero minimo possibile in modo da risparmiare risorse di sistema. Le etichette di alias e di sezione possono essere utilizzate per identificare un menu per lo scambio. Ad esempio, nel menu seguente una qualsiasi delle etichette AUX1, alias1 o alias2 può essere utilizzata come indicatore del menu seguente.

```
GRUPPOMENU=test
***AUX1
```

```
**alias1
**alias2
;
^C
```

Quindi, la seguente funzione effettua tutti gli scambi nello stesso menu.

```
(menucmd "B1=test.buttons1")
(menucmd "B1=test.alias1")
(menucmd "B1=test.alias2")
```

Se si utilizza la sintassi ** per assegnare nomi ai menu successivi, quando AutoCAD crea il file .mns ** viene sostituito con ***. Ad esempio, al momento del caricamento il seguente file viene convertito nel file .mns riportato di seguito.

File .mnu di esempio:

```
***AUX1
**alias1
circle
**sub2
linea
**sub3
arc
```

File .mns di esempio:

```
***AUX1
**alias1
circle
***sub2
linea
***sub3
arc
```

L'analizzatore dei menu presuppone che siano stati effettuati errori di battitura e che la sintassi ***sia stata specificata intenzionalmente. Naturalmente, è possibile continuare ad utilizzare sub2 e sub3 come etichette per lo scambio.

Un file di menu può contenere qualsiasi numero di sezioni ***BUTTONS e ***AUX denominati in modo univoco. Tuttavia, si deve tenere presente che ciascun menu occupa una determinata quantità di memoria e di risorse di sistema. Su alcuni sistemi, un numero eccessivo di menu in un singolo file di menu può richiedere più risorse di quelle disponibili. Ciò può dar luogo ad eventi imprevisti ed anomali.

Capitolo 4 -- Menu personalizzati

Menu di pulsanti ed ausiliari

Uso speciale della barra rovesciata

Quando si seleziona una voce di menu con uno dei pulsanti dei menu su un dispositivo di puntamento con più pulsanti, AutoCAD riceve non solo il numero di pulsante, ma anche le coordinate del puntatore a croce sullo schermo nel momento in cui si preme il pulsante. Costruendo attentamente le macro nelle sezioni Buttons e Aux del file di menu, è possibile scegliere di ignorare queste coordinate oppure di usarle con il comando attivato dal pulsante.

Come descritto in precedenza, è possibile includere una barra rovesciata (\) in una voce di menu per effettuare una pausa in attesa di input da parte dell'utente. Per quanto riguarda i menu dei pulsanti, le coordinate del puntatore a croce sullo schermo vengono fornite come input utente quando il pulsante viene premuto. Ciò avviene solo per la prima barra rovesciata nella voce di menu; se la voce non include barre rovesciate, le coordinate del puntatore non vengono utilizzate. Ad esempio, si considerino le seguenti voci di menu:

```
***AUX2
linea
linea \
```

Il primo pulsante di menu invia un comando LINEA ordinario e sollecita il normale messaggio di richiesta Dal punto. Il secondo pulsante di menu invia anch'esso un comando LINEA, ma AutoCAD legge la posizione corrente del puntatore e la utilizza per Dal punto.

Capitolo 4 -- Menu personalizzati

Menu a discesa ed a cursore

I menu a discesa (**POP n) e a cursore (**POP0) vengono visualizzati come menu a cascata. In tal modo, essi permettono di strutturare logicamente i menu senza passare da un'area di menu all'altra. Il menu a cursore fornisce un accesso rapido alle voci di menu utilizzate più spesso come le modalità Snap ad oggetti. Le voci di menu a discesa ed a cursore sono simili alle voci delle altre sezioni di menu; inoltre, le macro di menu definite dall'utente sono simili ai menu di schermo o di tavoletta standard definiti anch'essi dall'utente.

Un menu a discesa può contenere fino ad un massimo di 999 voci di menu. Un menu a cursore invece può contenere un massimo di 499 voci di menu. Entrambi i limiti includono tutti i menu nella gerarchia. Se le voci nel file di menu superano questi limiti, AutoCAD ignora le voci in esubero. Se un menu a discesa o a cursore supera lo spazio disponibile sullo schermo grafico, esso viene troncato in modo da farlo rientrare nello schermo.

I menu a discesa sono sempre *sviluppi* della barra dei menu, ma il menu a cursore viene sempre visualizzato in corrispondenza o vicino al puntatore a croce sullo schermo grafico. La sintassi per entrambe queste sezioni POP n di menu è la stessa, con la sola eccezione che il titolo del menu a cursore non viene incluso nella barra dei menu. Il titolo di menu a cursore non viene visualizzato affatto (anche se è comunque necessario digitare un titolo). Per accedere al menu a cursore il comando da usare è \$P0=*, che può essere inviato da un'altra voce di menu (come una voce di menu BUTTONS n) oppure da un programma AutoLISP o ARX. Mentre il menu a cursore è attivo, la barra dei menu non è disponibile.

Nota Se non è definito alcun menu a discesa (POP1-POP16), il menu a cursore (POP0) non funziona.

4 -- Menu personalizzati

Menu a discesa ed a cursore

Creazione di menu a discesa ed a cursore

Le sezioni di menu POP n controllano i menu a discesa ed il menu a cursore. POP0 controlla il menu a cursore; le sezioni da POP1 a POP16 controllano i menu a discesa attivi per la barra dei menu.

AutoCAD effettua una scansione per ricercare le sezioni di menu POP n durante il caricamento di ciascun file di menu. Per le sezioni di menu che vanno da POP1 a POP16 viene costruita una barra dei menu contenente i corrispondenti titoli. Se non è definita alcuna sezione POP n , AutoCAD inserisce i menu File e Modifica di default.

L'esempio seguente mostra la sintassi usata per creare un menu a discesa o a cursore.

Modifica		
Annulla	Ctrl+Z	***POP2
Ripristina	Ctrl+Y	**MODIFICA
		ID_MnEdit [&Modifica]
		ID_U [&Annulla\tCtrl+Z] _u
		ID_Redo [&Ripristina\tCtrl+Y] ^C^C_redo
		[--]
Taglia	Ctrl+X	ID_Cutclip [Tag&lia\tCtrl+X] '_cutclip
Copia	Ctrl+C	ID_Copyclip [&Copia\tCtrl+C] '_copyclip
Collega		ID_Copylink [C&ollega] ^C^C_copylink
Incolla	Ctrl+V	ID_Pasteclip [&Incolla\tCtrl+V] '_pasteclip
Incolla speciale...		ID_Pastesp [Incolla sp&eciale...] ^C^C_pastespec
Cancella	Canc	ID_Erase [Ca&ncella\tCanc] ^C^C_erase
		[--]
		ID_Links [Collega&menti...] ^C^C_olelinks
Collegamenti...		

A ciascuna sezione di menu può essere associato uno o più alias definiti dalle etichette ****alias** che seguono l'etichetta di sezione di menu. Nell'esempio precedente, ****MODIFICA** è un alias del menu POP2. Per informazioni sullo scambio dei menu, vedere "Scambio dei menu di pulsanti ed ausiliari."

Nota Le sezioni di menu POP n non supportano più la sintassi ****sottomenu**.

Capitolo 4 -- Menu personalizzati

Menu a discesa ed a cursore

Sintassi delle etichette di menu a discesa ed a cursore

La tabella seguente descrive i caratteri che svolgono una funzione particolare quando sono compresi in un'etichetta di menu a discesa o a cursore.

Caratteri speciali per le etichette

Carattere	Descrizione
--	È l'etichetta di voce che si espande fino a diventare una linea di separazione nei menu a discesa o a cursore (quando utilizzata senza nessun altro carattere).
+	Se è l'ultimo carattere, continua la macro sulla riga successiva.
->	È il prefisso di etichetta indicante che la voce del menu a discesa o a cursore possiede un sottomenu.
<-	È il prefisso di etichetta indicante che la voce del menu a discesa o a cursore è l'ultima nel sottomenu.
<-<-...	È il prefisso di etichetta indicante che la voce del menu a discesa o a cursore è l'ultima nel sottomenu e termina il menu di livello superiore (per terminare un menu principale, è necessario inserire <-).
\$(Attiva l'etichetta della voce di menu a discesa o a cursore per valutare una macro di stringhe DIESEL, se \$(sono i primi caratteri.
~	È il prefisso di etichetta che disattiva (visualizza in grigio) una voce di menu.
!.	È il prefisso di etichetta che contrassegna una voce di menu con un segno di spunta.
&	Una E commerciale (&) inserita direttamente prima di un carattere specifica quel carattere come tasto di scelta di menu in un'etichetta di menu a discesa o cursore. Ad esempio, C&o viene visualizzato come Campo.
/c	Specifica il tasto di scelta di menu in un'etichetta di menu a discesa o a cursore. Ad esempio, /aCampo viene visualizzato come Campo.
\t	Specifica che tutto il testo dell'etichetta che si trova a destra di questi caratteri viene posizionato sul lato destro del menu.

Nota I soli caratteri non alfanumerici che possono essere utilizzati come primo carattere in una etichetta di menu sono quelli elencati sopra. I caratteri non alfanumerici non inclusi nell'elenco sono riservati per uso futuro come caratteri di menu speciali.

Capitolo 4 -- Menu personalizzati

Menu a discesa ed a cursore

Titoli delle barre dei menu a discesa

Per quanto riguarda i menu a discesa, la prima etichetta definisce il titolo della barra dei menu, mentre le etichette successive definiscono le voci di menu e sottomenu. L'esempio che segue mostra la parte superiore della sezione del menu a discesa POP2.

```
***POP2
ID_MnEdit      [&Modifica]
ID_U           [&Annulla\tCtrl+Z]_u
##ID_Redo      [&Ripristina\tCtrl+Y]^C^C_redo
```

Sulla prima riga dopo l'etichetta di sezione ***POP2, l'etichetta [&Modifica] causa la visualizzazione di Modifica come titolo della barra del menu con la lettera M sottolineata per indicare che si tratta del tasto di scelta del menu. Il contrassegno di nome associato al titolo del menu (ID_MnEdit) può essere utilizzato per attivare o disattivare l'intero menu. Ai titoli dei menu a discesa non può essere associata una macro di menu.

Nota I menu a cursore (POP0) devono definire un titolo che tuttavia non viene visualizzato.

Capitolo 4 -- Menu personalizzati

Menu a discesa ed a cursore

Sottomenu a discesa

I menu a discesa ed a cursore usano caratteri speciali (come ->, <- e <-<-...) per controllare la gerarchia dei menu a discesa. Tali caratteri indicano i sottomenu e le ultime voci nei sottomenu e possono inoltre terminare tutti i menu principali. Le stringhe con caratteri speciali devono trovarsi all'inizio delle etichette di voce.

Il carattere speciale -> indica che a questa voce è associato un sottomenu, come nell'esempio seguente:

```
[->Filtri coordinate]
```

Se si seleziona il menu Aiuti e si sceglie la voce Filtri coordinate oppure si sposta il cursore all'estremità destra della voce, viene visualizzato il sottomenu Filtri coordinate.

Il carattere speciale <- indica che questa voce è l'ultima di un sottomenu, come nell'esempio seguente:

```
[<- .YZ] .YZ
```

I caratteri speciali <-<-... indicano che questa voce è l'ultima di un sottomenu ed anche di un menu principale, come nell'esempio seguente:

```
[->Testo]
[->Attributi]
[<-<-Estrai...]^C^Cdattext
```

Capitolo 4 -- Menu personalizzati

Menu a discesa ed a cursore

Separazione di etichette di voci di menu

Per creare righe di separazione, utilizzare un'etichetta composta da due trattini.

```
[- -]
```

Poiché la larghezza di ogni menu a discesa o a cursore viene stabilita dall'etichetta più larga, l'esempio precedente finisce per diventare una riga di separazione che si estende per l'intera larghezza del menu. Non è possibile scegliere righe di separazione dal menu e qualsiasi macro di menu ad esse assegnata viene ignorata.

Capitolo 4 -- Menu personalizzati

Menu a discesa ed a cursore

Controllo della visualizzazione di etichette di voci di menu

Un'altra funzione per la personalizzazione di menu a discesa o a cursore consiste nella possibilità di controllare il modo in cui le etichette di voce vengono visualizzate. Le etichette possono essere disattivate (visualizzate in grigio), impedendone l'accesso da parte dell'utente, oppure contrassegnate con un segno di spunta.

Le etichette contengono inoltre espressioni di stringhe DIESEL che consentono di modificare il contenuto delle etichette stesse. Ciò permette di disattivare, contrassegnare o cambiare interattivamente il testo dell'etichetta visualizzata. Per ulteriori informazioni, vedere "Linguaggio DIESEL e configurazione della riga di stato".

Nota Quando si disattivano e contrassegnano le etichette delle voci di menu, assicurarsi di utilizzare una tecnica appropriata per la registrazione delle modifiche che cambiano lo stato dell'etichetta.

4 -- Menu personalizzati

Menu a discesa ed a cursore

Controllo della visualizzazione di etichette di voci di menu

Disattivazione di etichette

Un'etichetta di voce di menu che inizia con una tilde (~) verrà disattivata (visualizzata in grigio). Per convenzione, questo indica che in quel momento la voce non è una selezione valida. Qualsiasi comando associato alla voce non viene inviato e non è possibile accedere ai sottomenu.

Ad esempio, le seguenti etichette di menu sono disattivate.

```
[~Linea]
[~->Plinea]
```

Non è possibile accedere al menu di livello inferiore di un'etichetta disattivata.

Uso di espressioni DIESEL per la disattivazione di etichette

Le etichette di voce di menu possono contenere espressioni di stringhe DIESEL che, in base a determinate condizioni, disattivano o attivano le etichette ogni volta che vengono visualizzate. Ad esempio, l'espressione DIESEL di stringhe all'interno della seguente etichetta di voce di menu disattiva l'etichetta mentre è attivo un comando.

```
[$(if,$(getvar,cmdactive),~)SPOSTA]sposta
```

Uso di AutoLISP per la disattivazione di etichette

La funzione AutoLISP **menucmd** può essere utilizzata per disattivare ed attivare etichette da una macro di menu o da un'applicazione. Per esempi, vedere "Riferimento di menu a discesa ed a cursore."

Capitolo 4 -- Menu personalizzati

Menu a discesa ed a cursore

Controllo della visualizzazione di etichette di voci di menu

Contrassegno di etichette

Inserendo un punto esclamativo e un punto (!.), è possibile contrassegnare un'etichetta di voce di menu con un segno di spunta iniziale (4). Contrassegnando una voce di menu non si riduce la possibilità di sceglierla, anche se una voce contrassegnata può essere disattivata.

Nell'esempio che segue, la voce di menu Linea viene contrassegnata con un segno di spunta.

```
[!.Linea]
```

Uso di espressioni DIESEL per il contrassegno di etichette

Le etichette di voce di menu possono contenere espressioni DIESEL di stringhe che, in base a determinate condizioni, contrassegnano le etichette ogni volta che vengono visualizzate. L'esempio che segue pone un segno di spunta a sinistra delle etichette di menu le cui variabili di sistema relative sono correntemente attivate.

```
[$(if,$(getvar,orthomode),!.)Orto]^O
[$(if,$(getvar,snapmode),!.)Snap]^B
[$(if,$(getvar,gridmode),!.)Griglia]^G
```

Uso di AutoLISP per il contrassegno di etichette

La funzione AutoLISP **menucmd** può essere utilizzata per contrassegnare etichette di una macro di menu o di un'applicazione. Per esempi, vedere "Riferimento di menu a discesa ed a cursore."

Capitolo 4 -- Menu personalizzati

Menu a discesa ed a cursore

Controllo della visualizzazione di etichette di voci di menu

Disattivazione e contrassegno simultanei

È possibile contrassegnare e contemporaneamente disattivare le voci di menu. Il formato è:

```
[~!. testoetichetta]
```

oppure

```
[!.~ testoetichetta]
```

dove ~ è il codice di carattere speciale per disattivare una voce di menu e !. è il codice di carattere speciale per contrassegnare una voce di menu.

Nell'esempio che segue, la voce di menu Linea viene disattivata e contrassegnata con un segno di spunta. Come negli esempi precedenti, è possibile usare un'espressione DIESEL per disattivare e simultaneamente contrassegnare un'etichetta di voce di menu.

```
[~!.Linea]
```

Capitolo 4 -- Menu personalizzati

Menu a discesa ed a cursore

Controllo della visualizzazione di etichette di voci di menu

Riferimento di menu a discesa ed a cursore

Per fare riferimento a un menu a discesa o a cursore, sono disponibili due metodi; con uno viene utilizzato il gruppo di menu e il contrassegno di nome, mentre con l'altro viene utilizzata la posizione assoluta della voce di menu nella gerarchia dei menu. È consigliabile tuttavia adottare il primo metodo che, essendo di tipo dinamico, funziona sempre in modo corretto, indipendentemente dallo stato corrente del menu.

Riferimenti relativi delle voci di menu a discesa ed a cursore

Per fare riferimento a una voce di menu di un cursore a discesa o a cursore in base al corrispondente gruppo di menu e contrassegno di nome, usare la funzione AutoLISP `menucmd`. La sintassi seguente fa riferimento a una voce di menu in base al corrispondente contrassegno di nome.

```
(menucmd "Ggruppomenu . contrassegno_nome=valore")
```

L'esempio seguente disattiva la voce di menu ID_Line inclusa nel gruppo di menu sample. Quello qui riportato funziona a prescindere dalla posizione della voce nel menu.

```
[Disattiva linea] (menucmd "Gsample.ID_Line = ~")
```

Se l'autore di un menu parziale conosce il contenuto del menu di base, la sintassi di una voce di menu potrebbe fare riferimento ad un contrassegno del file di base. Di seguito è riportato un possibile esempio di file di base `acad.mnu` :

```
***MENUGROUP=ACAD
***POP0      (e così via...)
...
***POP6
ID_MnHelp   [?]
ID_Contents [Indice]^C^C_HELP
ID_About    [Informazioni su]^C^C_ABOUT
```

Una voce di un menu parziale può essere modificata in modo da avere un'ulteriore voce di menu che faccia riferimento al contrassegno nel menu di base.

```
***POP2
[Titolo2]
[Disattiva indice guida] (menucmd "Gacad.ID_Contents =~")
```

In questa maniera, più file di menu parziali possono lavorare con specifici file di base. AutoCAD attiverà una rigida definizione della sezione Menugroup in modo tale che due menu non possano definire lo stesso gruppo di menu. Eventuali tentativi di caricamento di un menu con nome di gruppo in conflitto provocheranno l'annullamento della richiesta del comando CARMENU.

Riferimenti assoluti delle voci di menu a discesa ed a cursore

In modo analogo alla procedura descritta nella sezione precedente, è possibile attivare e disattivare una voce di menu utilizzando la sintassi $Pn=xxx$. Il formato è:

```
Pn.i=xxx
```

dove Pn indica la sezione di menu POP n attiva (i valori validi sono compresi nell'intervallo da 0 a 16), i indica il numero della voce di menu e xxx , se specificato, rappresenta una stringa che definisce l'azione.

L'esempio seguente utilizza la funzione AutoLISP `menucmd` per fare riferimento a una voce di menu di un menu a discesa o a cursore. Dal momento che i file di menu di AutoCAD sono dinamici (grazie al caricamento di menu parziali), la sintassi seguente potrebbe non funzionare sempre correttamente.

```
[Disattiva linea vecchia modalità](menucmd "P1.2=~")
```

Questa sintassi si basa sulla posizione della voce di menu e non funziona se l'autore del menu inserisce una nuova voce nella sezione POP1 o se prima di POP1 viene inserito un nuovo menu a discesa con il comando CARMENU.

Se segue il comando \$, la sintassi $Pn=xxx$ può essere utilizzata da una macro di menu. L'esempio seguente disattiva la voce 4 nella sezione POP3.

```
$P3.4=~
```

L'esempio seguente consente di aggiungere un segno di spunta alla voce 1 della sezione POP7.

```
$P7.1=!
```

L'esempio seguente rimuove qualsiasi carattere di disattivazione o di contrassegno dalla voce 1 della sezione POP7.

```
$P7.1=
```

La numerazione delle voci di menu è consecutiva, a prescindere dalla gerarchia del file di menu; la voce 1 è la prima voce dopo il titolo.

```
***POP5
[Aiuti ]                               Titolo
[Visualizza aiuti]'?                   Voce 1
[Annulla ]^C^C^C                       Voce 2
[--]                                    Voce 3
[Annulla ]^C^C_A                       Voce 4
[Ripristina ]^C^C_ripristina           Voce 5
[--]                                    Voce 6
[->Osnap ]                             Voce 7
[Centro ]centro                        Voce 8
```

Per semplificare l'indirizzamento automatico di una voce a prescindere dalla posizione nella gerarchia dei menu, è possibile utilizzare i formati riportati di seguito.

```
$P@.@=xxx
```

Fa riferimento alla voce di menu corrente o a quella scelta più recentemente.

```
$P@.n=xxx
```

Fa riferimento alla voce n nel menu corrente o in quello scelto più recentemente.

Capitolo 4 -- Menu personalizzati

Menu a discesa ed a cursore

Controllo della visualizzazione di etichette di voci di menu

Accesso di AutoLISP allo stato delle etichette

Le funzioni AutoLISP `menucmd` accettano le stringhe di comando `$Pn=xxx`, ma senza \$ iniziale. Con queste funzioni la parte `xxx` della stringa di comando può assumere valori speciali.

`Pn.i=?`

Restituisce in forma di stringa lo stato corrente di disattivazione e contrassegno per la voce specificata (ad esempio, ~ per una voce disattivata, !. per una voce con un segno di spunta e "" per una voce né visualizzata in grigio né contrassegnata).

`Pn.i=#?`

Restituisce lo stesso tipo di stringa descritto per `Pn.i=?`, ma con il prefisso `Pn.i=`. Questo è utile in congiunzione con i formati @, in quanto vengono restituiti il menu effettivo ed il numero di voce.

Ad esempio, se la quinta voce della sezione POP6 è disattivata, il seguente codice `menucmd` restituisce le stringhe riportate di seguito.

```
(menucmd "P6.5=?")      restituisce  "~"
(menucmd "P6.5=#?")    restituisce  "P6.5=~"
```

Per ulteriori informazioni, vedere "Uso di AutoLISP nelle macro di menu" e la parte II di "AutoLISP".

Capitolo 4 -- Menu personalizzati

Menu a discesa ed a cursore

Scambio ed inserimento in menu a discesa

Poiché i menu a discesa di AutoCAD sono menu a tendina, non vi è molta necessità di passare da un menu all'altro. Inoltre, la sostituzione di un menu con un altro può ridurre l'uniformità globale dell'interfaccia utente. L'inserimento e la rimozione di un menu a discesa, tuttavia, risultano operazioni appropriate quando l'utente carica o scarica un'applicazione che richiede un menu aggiuntivo.

Capitolo 4 -- Menu personalizzati

Menu a discesa ed a cursore

Scambio ed inserimento in menu a discesa

Scambio tra menu a discesa

Per sostituire un menu a discesa caricato nella barra dei menu con un altro menu, utilizzare la seguente sintassi:

```
(menucmd "Ggruppomenu1.nomemenu1=gruppomenu2.nomemenu2")
```

Con questa sintassi, è possibile spostare un menu noto indipendentemente dal fatto che sia correntemente situato nella barra dei menu. Questo tipo di scambio può essere eseguito solo con la funzione AutoLISP `menucmd`. Per eseguirlo in una macro di menu, richiamare `menucmd` con i parametri appropriati. Ad esempio codice, il seguente consente di passare dal menu NUOVO1 al menu NUOVO2 (entrambi i menu appartengono allo stesso gruppo di menu MIOGRUP):

```
(menucmd "Gmiogrup.nuovo1=miogrup.nuovo2")
```

Tramite i comandi \$ delle macro di menu è possibile passare tra menu a comparsa diversi in posizione `n` POPspecifiche. Se si adotta questo metodo è tuttavia necessario verificare che il menu a cui si passa sia effettivamente quello desiderato. In seguito alla natura dinamica dei menu di AutoCAD, infatti, un menu inserito nella posizione P6 potrebbe in effetti non trovarsi in quella posizione. Di conseguenza, se si passa da questo menu ad un altro, si rischia di rimuovere un menu diverso da quello desiderato.

In funzione del passaggio tra menu, i nomi delle aree dei menu a discesa attivi sono P1, Pe e così via fino a P16. La seguente macro di menu consente di sostituire il menu che si trova nella posizione P3 con il menu MenuProva nel gruppo di menu MIOMENU.

```
$P3=MioMenu.MenuProva
```

La stessa operazione può essere eseguita utilizzando la funzione `menucmd` nel modo seguente:

```
(menucmd "P3=MioMenu.MenuProva")
```

È possibile utilizzare il comando speciale $\$Pn=*$ da una macro di menu per fare in modo che venga visualizzato il menu assegnato all'area $POPn$.

Nota Lo scambio tra menu a discesa non rispetta le istruzioni Microsoft UI e la compatibilità con le release future di AutoCAD non è garantita.

Capitolo 4 -- Menu personalizzati

Menu a discesa ed a cursore

Scambio ed inserimento in menu a discesa

Inserimento e rimozione di me nu a discesa

Per inserire o rimuovere un menu a discesa, è possibile utilizzare la funzione AutoLISP `menucmd`. La sintassi è simile a quella utilizzata per lo scambio tra menu a discesa con la differenza che la parte sinistra dell'assegnazione corrisponde al menu a discesa davanti al quale si desidera inserire il nuovo menu. La parte destra dell'assegnazione è un più (+), seguito dal nome del gruppo di menu, seguito a sua volta da un punto e dall'alias del menu.

```
(menucmd "Ggruppomenu1 . nomemenu1=+gruppomenu2 . nomemenu2")
```

Per inserire un menu, è inoltre possibile utilizzare la sintassi $Pn=$. La seguente macro di menu consente di inserire un menu dopo il menu P5 (con questo formato è inoltre possibile utilizzare la funzione `menucmd`).

```
$P5=+miomenu.nuovo3
```

Se si utilizza questo metodo di inserimento di menu, è importante ricordare che il menu potrebbe non venire inserito effettivamente nella posizione P6 come previsto. Ciò tuttavia non si verifica mai in due casi: se la barra di menu corrente include solo tre menu, la posizione di un nuovo menu inserito dopo il menu P5 sarà P4; inoltre la numerazione di menu potrebbe non essere sincronizzata quando l'utente inserisce o rimuove un menu con il comando CARMENU o quando un menu viene inserito o rimosso da un'altra applicazione. La sintassi per la rimozione di un menu è la seguente:

```
(menucmd "Ggruppomenu . nomemenu=-")
```

L'esempio seguente consente di rimuovere il menu NUOVO3 incluso nel gruppo MioMenu.

```
(menucmd "Gmiomenu.nuovo3=-")
```

Questo formato è preferibile rispetto al formato $Pn=$, in quanto rimuove solo il menu specificato. L'esempio seguente rimuove qualsiasi menu che si trova nella posizione P4.

```
$P4=-
```

Capitolo 4 -- Menu personalizzati

Barre degli strumenti

La sezione Toolbars specifica il layout di default ed il contenuto delle barre degli strumenti. La sezione `***TOOLBARS` contiene un sottomenu per ogni barra degli strumenti definita dal menu.

Capitolo 4 -- Menu personalizzati

Barre degli strumenti

Creazione di barre degli strumenti

Esistono cinque tipi distinti di voci che possono essere specificati per le barre degli strumenti. Nell'esempio riportato di seguito viene descritta la sintassi di ogni tipo. Tutte le righe eccetto quella di separazione cominciano con un contrassegno standard di nome usato

per associare alla voce le informazioni della guida. Nell'esempio riportato di seguito, la dichiarazione ****TOOLS1** rappresenta un sottomenu che utilizza l'alias **TOOLS1** come etichetta per fare riferimento alla successiva definizione di barra degli strumenti.

```

***TOOLBARS
**TOOLS1
TAG1 [Toolbar ("nomebs", orientamento, visibilit , valx, valy, righe) ]
TAG2 [Button ("nomepl", id_piccola, id_grande) ]macro
TAG3 [Flyout ("nome_icona", id_piccola, id_grande, icona, alias) ]macro
TAG4 [Control (elemento) ]
[--]

```

La quinta riga definisce una linea di separazione (- -).

Capitolo 4 -- Menu personalizzati

Barre degli strumenti

Creazione di barre degli strumenti

Definizioni delle barre degli strumenti.

La prima riga della definizione della barra degli strumenti(TAG1) definisce le caratteristiche per la definizione della barra degli strumenti. Essa usa la parola chiave **Toolbar** seguita da una serie di opzioni contenute tra parentesi. Le opzioni sono:

nomebs

La stringa assegna il nome alla barra degli strumenti. Deve essere composta da caratteri alfanumerici senza segni di punteggiatura, con la sola eccezione del trattino (-) o del trattino di sottolineatura (_). Questo nome, insieme all'alias, permette di fare riferimento alla barra degli strumenti a livello di programmazione.

orientamento

La parola chiave di orientamento. I valori accettabili sono Floating, Top, Bottom, Left e Right, per i quali non viene fatta distinzione tra maiuscolo e minuscolo.

visibilit 

La parola chiave della visibilit . I valori accettati sono Show e Hide, per i quali non viene fatta distinzione tra maiuscolo e minuscolo.

valx

Un valore numerico che specifica la coordinata X in pixel, misurato dal margine sinistro dello schermo al lato destro della barra degli strumenti.

valy

Un valore numerico che specifica la coordinata Y in pixel, misurato dal margine superiore dello schermo al lato superiore della barra degli strumenti.

righe

Un valore numerico che specifica un numero di righe.

Capitolo 4 -- Menu personalizzati

Barre degli strumenti

Creazione di barre degli strumenti

Definizioni dei pulsanti

La seconda riga (TAG2) definisce un pulsante. Essa usa la parola chiave Button seguita da una serie di opzioni contenute tra parentesi. Le opzioni sono:

nomepl

La stringa che assegna il nome al pulsante. Deve essere composta da caratteri alfanumerici senza segni di punteggiatura, con la sola eccezione del trattino (-) o del trattino di sottolineatura (_). La stringa viene visualizzata come una descrizione di comandi quando il cursore è posizionato sopra il pulsante.

id_piccola

La stringa che assegna il nome alla stringa di ID della risorsa relativa alle immagini di piccole dimensioni (bitmap 16 x 15). Deve essere composta da caratteri alfanumerici senza segni di punteggiatura, con la sola eccezione del trattino (-) o del trattino di sottolineatura (_). La stringa può specificare anche una bitmap definita dall'utente; vedere "Bitmap definite dall'utente."

id_grande

La stringa che assegna il nome alla stringa di ID della risorsa relativa alle immagini di grandi dimensioni (bitmap 24 x 22). Deve essere composta da caratteri alfanumerici senza segni di punteggiatura, con la sola eccezione del trattino (-) o del trattino di sottolineatura (_). La stringa può specificare anche una bitmap definita dall'utente; vedere "Bitmap definite dall'utente."

macro

La definizione è seguita da una stringa di comando basata sulla sintassi standard delle voci di menu per le stringhe di comando.

Capitolo 4 -- Menu personalizzati

Barre degli strumenti

Creazione di barre degli strumenti

Definizioni delle icone a comparsa

La terza riga (TAG3) definisce il controllo per le icone a comparsa. Essa usa la parola chiave Toolbar seguita da una serie di opzioni contenute tra parentesi. Le opzioni sono:

nomeicona

La stringa che assegna il nome all'icona a comparsa. Deve essere composta da caratteri alfanumerici senza segni di punteggiatura, con la sola eccezione del trattino (-) o del trattino di sottolineatura (_). La stringa viene visualizzata come una descrizione di comandi quando il cursore è posizionato sopra l'icona a comparsa.

id_piccola

La stringa che assegna il nome alla stringa di ID della risorsa relativa alle immagini di piccole dimensioni (bitmap 16 x 15). Deve essere composta da caratteri alfanumerici senza segni di punteggiatura, con la sola eccezione del trattino (-) o del trattino di sottolineatura (_). La stringa può specificare anche una bitmap definita dall'utente; vedere "Bitmap definite dall'utente."

id_grande

La stringa che assegna il nome alla stringa di ID della risorsa relativa alle immagini di grandi dimensioni (bitmap 24 x 22). Deve essere composta da caratteri alfanumerici senza segni di punteggiatura, con la sola eccezione del trattino (-) o del trattino di sottolineatura (_). La stringa può specificare anche una bitmap definita dall'utente; vedere "Bitmap definite dall'utente."

icona

La parola chiave booleana che controlla se visualizzare sempre la propria icona oppure visualizzare l'ultima icona selezionata (un'altra). I valori accettati sono `OwnIcon` e `OtherIcon`, per i quali non viene fatta distinzione tra maiuscolo e minuscolo.

alias

Il riferimento alla barra degli strumenti per la visualizzazione simile a quella dell'icona a comparsa. L'alias fa riferimento ad un sottomenu della barra degli strumenti definito con la sintassi standard ****alias**.

macro

La definizione è seguita da una stringa di comando basata sulla sintassi standard delle voci di menu per le stringhe di comando.

Capitolo 4 -- Menu personalizzati

-  **Barre degli strumenti**
-  **Creazione di barre degli strumenti**
-  **Definizioni degli elementi di controllo**

La quarta riga (TAG4) definisce un elemento di controllo speciale. Essa utilizza la parola chiave `Control` seguita da un nome, che specifica il tipo di elemento di controllo richiesto contenuto tra parentesi.

elemento

Questo parametro può assumere tre possibili valori, per i quali non viene fatta distinzione tra maiuscolo e minuscolo.

_Layer specifica l'elemento di controllo per il layer. Questo elemento è un elenco a comparsa che controlla i layer correnti presenti nel disegno.

_Linetype specifica l'elemento di controllo per il tipo di linea. Questo elemento è un elenco a comparsa che specifica il tipo di linea corrente.

_Color specifica l'elemento di controllo per il colore. Questo elemento è un elenco a comparsa che specifica il colore corrente.

Capitolo 4 -- Menu personalizzati

-  **Barre degli strumenti**
-  **Creazione di barre degli strumenti**
-  **Bitmap definite dall'utente**

Le bitmap definite dall'utente possono essere usate al posto dei nomi delle risorse relative alle immagini di piccole e grandi dimensioni. Una bitmap definita dall'utente deve avere dimensioni appropriate (16 x 15 pixel per il parametro `id_piccola` e 24 x 22 pixel per il parametro `id_grande`; deve inoltre trovarsi nel percorso `Support`. Specificare una bitmap definita dall'utente indicando il nome del file e l'estensione `.bmp`, come illustrato nell'esempio seguente.

```
TAG34 [Button ("Mio comando", miocmd16.bmp, miocmd24.bmp)]^C^CMIOCMD
```

Capitolo 4 -- Menu personalizzati

Barre degli strumenti

Riferimento di barre degli strumenti

Per controllare Toolbars nei menu parziali, utilizzare la seguente sintassi alla riga di comando Nome barra degli strumenti del comando BAR_STRU:

```
nomegruppo . nome_barrast
```

Il codice AutoLISP seguente visualizza la barra degli strumenti MIABARRA nel gruppo di menu MIOGRUPPO (nel codice si presume che il menu MIOGRUPPO sia già caricato).

```
(command "bar_stru" "miogruppo.miabarra" "mostra")
```

Se *gruppomenu* viene omesso, per default viene utilizzato il menu di base.

Capitolo 4 -- Menu personalizzati

Menu a gruppi di immagini

Un menu a gruppi di immagini viene definito fornendo una sezione ***IMAGE nel file di menu. In questo modo viene sostituita la sezione ***ICON utilizzata in release precedenti (***ICON è ancora valida, ma potrebbe non essere supportata in release future).

AutoCAD visualizza diapositive di immagini in gruppi di 20, insieme ad una casella di riepilogo a scorrimento contenente i nomi di diapositiva associati o testo correlato. I sottomenu a gruppi di immagini non hanno limiti di lunghezza; se un sottomenu a gruppi di immagini contiene più di 20 diapositive, AutoCAD fornisce i pulsanti Seguevole e Precedente che l'utente può premere per scorrere le pagine delle immagini.

Capitolo 4 -- Menu personalizzati

Menu a gruppi di immagini

Voci dei menu a gruppi di immagini

La sezione Image utilizza sottomenu simili a quelli delle sezioni Toolbars e Screen. Come con le sezioni Pop di menu, la prima riga del sottomenu è il relativo titolo. Il titolo viene visualizzato come etichetta della finestra di dialogo che include le immagini. Tra i sottomenu è necessario inserire almeno una riga vuota per eliminare le voci di sottomenu precedenti.

Per la definizione del testo dell'elenco a scorrimento e dell'immagine stessa, nei menu a gruppi di immagini vengono utilizzate etichette di elemento, le quali sono seguite da una macro di menu associata.

Nota I menu a gruppi di immagini *non possono* contenere contrassegni di nome.

Capitolo 4 -- Menu personalizzati

Menu a gruppi di immagini

Voci dei menu a gruppi di immagini

Etichette di menu a gruppi di immagini

Le etichette in un menu a gruppi di immagini generalmente si riferiscono ai nomi dei file delle diapositive anziché alle etichette di

testo visualizzate sullo schermo. Il nome del file di diapositiva, che può contenere una sola diapositiva o parte di una libreria, dovrebbe comparire esattamente non appena lo si digita al comando VSLIDE. Il file di diapositiva contiene l'immagine che deve essere visualizzata in base a quella selezione.

Le etichette di menu a gruppi di immagini vengono visualizzate in una casella di riepilogo a scorrimento che può contenere stringhe lunghe fino ad un massimo di 19 caratteri per etichetta. In genere viene visualizzato il nome del file della diapositiva; tuttavia, sono disponibili anche le seguenti opzioni di etichettatura.

[*nomedia*]

Il nome di diapositiva *nomedia* viene visualizzato nella casella di riepilogo e la diapositiva *nomedia* viene visualizzata come immagine.

[*nomedia, testoetichetta*]

Il testo *testoetichetta* viene visualizzato nella casella di riepilogo e la diapositiva *nomedia* viene visualizzata come immagine.

[*libdia (nomedia)*]

Il nome di diapositiva *nomedia* viene visualizzato nella casella di riepilogo e la diapositiva *nomedia* nella libreria di diapositive *libdia* viene visualizzata come immagine.

[*libdia (nomedia, testoetichetta)*]

Il testo *testoetichetta* viene visualizzato nella casella di riepilogo e la diapositiva *nomedia* nella libreria di diapositive *libdia* viene visualizzata come immagine.

[spazio]

Quando si fornisce il testo spazio come etichetta di icona, nella casella di riepilogo viene visualizzata una riga di separazione e sullo schermo compare un'immagine vuota.

[*testoetichetta*]

Quando il primo carattere di un'etichetta di voce è uno spazio, il testo fornito come *testoetichetta* viene visualizzato nella casella di riepilogo e sullo schermo non compare alcuna immagine. In questo caso, è possibile includere comandi correlati e semplici voci come Esci senza che sia necessario creare diapositive che contengano tali parole.

Capitolo 4 -- Menu personalizzati

Menu a gruppi di immagini

Voci dei menu a gruppi di immagini

Macro dei menu a gruppi di immagini

Le macro dei menu a gruppi di immagini possono svolgere la stessa funzione di altre macro di menu; tuttavia non è possibile utilizzare la funzione di ripetizione delle macro di menu. Queste macro di menu possono contenere comandi di menu, compresi i comandi \$I=. Perciò è possibile costruire menu gerarchici a gruppi di immagini nei quali una selezione visualizza un altro menu a gruppi di immagini, e così via. Poiché l'attivazione di questi menu è sequenziale anziché nidificata, non vi sono limiti alla complessità delle strutture che si possono creare.

Capitolo 4 -- Menu personalizzati

Menu a gruppi di immagini

Visualizzazione dei menu a gruppi di immagini

Il comando macro \$I= visualizza il menu a gruppi di immagini. Per poter visualizzare un menu a gruppi di immagini, è prima necessario caricarlo. La sintassi che segue carica un menu di questo tipo.

\$I=[*gruppomenu .*] *nomemenu*

Il comando macro \$I=* visualizza il menu a gruppi di immagini caricato. Ad esempio, le seguenti macro consentono di caricare e visualizzare il menu POLI_IMMAGINE nel menu di base.

```
$I=poli_immagine $I=*
```

L'esempio che segue carica e visualizzare il menu a gruppi di immagini MIOBLOCCO da un gruppo di menu parzialmente caricato MIOGRUPPO.

```
$I=miogruppo.mioblocco $I=*
```

Per caricare e visualizzare menu a gruppi di immagini, è anche possibile utilizzare la funzione **menucmd**. Il seguente codice produce gli stessi risultati del codice precedente.

```
(menucmd "I=miogruppo.mioblocco")
(menucmd "I=*")
```

Capitolo 4 -- Menu personalizzati

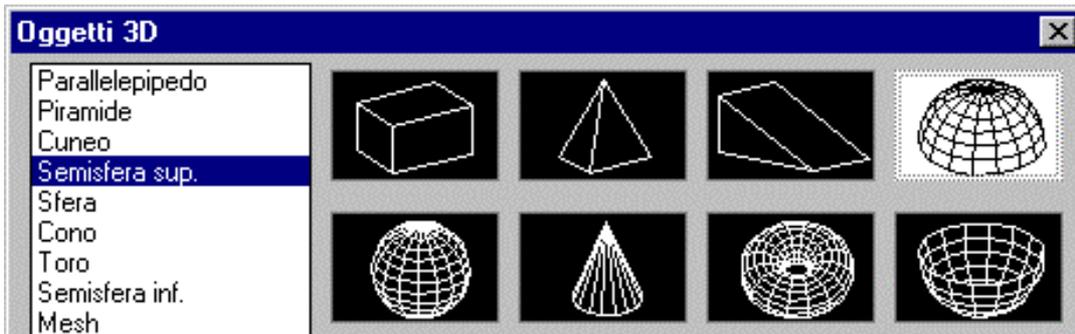
Menu a gruppi di immagini

Menu a gruppi di immagini di esempio

Di seguito è riportato un sottomenu a gruppi di immagini di esempio chiamato ****image_3DObjects**.

```
**3DOBJECTS
[Oggetti 3D]
[acad(box3d)]^c^cai_box
[acad(Piramide)]^c^cai_pyramid
[acad(Cuneo)]^c^cai_wedge
[acad(Tronco)]^c^cai_dome
[acad(Sfera)]^c^cai_sphere
[acad(Cono)]^c^cai_cone
[acad(Toro)]^c^cai_torus
[acad(Faccia)]^c^cai_dish
[acad(Mesh)]^c^cai_mesh
```

La figura seguente mostra una parte del menu a gruppi di immagini risultante.



Menu ****image_3DObjects** di esempio

Di seguito è riportato un esempio di un menu a gruppi di immagini usato per inserire varie parti elettroniche. L'etichetta di testo è una voce che passa ad un altro sottomenu a gruppi di immagini contenente vari elementi di fissaggio.

```
***IMAGE
**IPARTS
[Parti elettroniche]
[cap]^Cinsert cap
[res]^Cinsert res
[neon]^Cinsert neon
[triodo]^Cinsert triodo
[tetrodo]^Cinsert tetrodo
[ Fissaggio]$I=ifast $I=*
**IFAST
[Fissaggio]
[nut632]...
```

Per attivare questo menu a gruppi di immagini, è possibile scegliere da un qualsiasi menu una voce di menu come la seguente.

```
[Parti elettroniche]$I=iparts $I=*
```

Nelle variazioni riportate di seguito, le immagini vengono richiamate da una libreria di diapositive chiamata named *elib*; nella casella di riepilogo compare solo il nome della diapositiva

```
***ICON
**IPARTS
[Parti elettroniche]
[elib(cap)]^Cinsert cap
[elib(res)]^Cinsert res
[elib(neon)]^Cinsert neon
[elib(triodo)]^Cinsert triodo
[elib(tetrodo)]^Cinsert tetrodo
```

Quando vengono visualizzate successive diapositive di icone della stessa libreria, il file della libreria rimane aperto. Per questa ragione, il tempo necessario per visualizzare un menu a icone viene sensibilmente ridotto.

Capitolo 4 -- Menu personalizzati

Menu a gruppi di immagini

Preparazione di diapositive per menu a gruppi di immagini

Qualsiasi diapositiva generata da AutoCAD può essere utilizzata come immagine. Tuttavia, l'uso ottimale dei menu a gruppi di immagini prevede che si ponga molta attenzione nel preparare le diapositive che serviranno come immagini. Seguire queste linee generali:

- Puntare alla semplicità. Quando un menu a gruppi di immagini viene visualizzato, l'utente deve attendere che tutte le immagini compaiano sullo schermo prima di poter effettuare una selezione. Se si utilizza il menu a gruppi di immagini per mostrare all'utente numerosi simboli complessi, fare in modo che le immagini siano versioni semplificate dei simboli anziché la loro copia completa. Un'immagine dovrebbe essere la più semplice possibile e, allo stesso tempo, dovrebbe poter essere riconosciuta immediatamente.
- Riempire la casella. Lo spazio sullo schermo è limitato e le immagini compaiono su piccole parti dello schermo intero. Durante la creazione di una diapositiva per un'immagine, accertarsi di riempire lo schermo con l'immagine prima di digitare il comando GENDIA. Se l'immagine è molto larga e corta, oppure lunga e sottile, il menu a gruppi di immagini avrà un aspetto migliore se l'immagine viene centrata sullo schermo con il comando PAN prima di creare la diapositiva.

Le immagini vengono visualizzate con un rapporto di formato di 1,5:1 (1,5 unità di larghezza per 1 unità di altezza). Se l'area grafica ha un rapporto di formato differente, può essere difficile produrre diapositive di immagini centrate nel menu a gruppi di immagini. Se si lavora all'interno di una finestra mobile con rapporto di formato 1,5:1, è possibile posizionare l'immagine per essere sicuri che apparirà esattamente nel modo in cui viene visualizzata nel menu a gruppi di immagini.

I comandi seguenti impostano una finestra correttamente proporzionata. In questo esempio, viene usato un disegno senza alcuna finestra e la variabile di sistema TILEMODE è impostata su 1.

Comando: **tilemode**

Nuovo valore per TILEMODE <1>: **0**

Digitazione Spazio carta. Usare FINMUL per inserire finestre Spazio modello.

Comando: **finmul**

ON/OFF/Hideplot/Fit/2/3/4/Restore/<Primo punto>: **0,0**

Altro angolo: 3,2 **3,2**

Comando: **zoom**

Tutto/Centrato/Dinamico/Estensione/Precedente/Scala(X/XC)/Finestra/<Tempo reale>: **e**

Comando: **mspace**

Questa finestra è ora proporzionata per un gruppo di immagini.

- Le aree solide come le polilinee vuote, i tracciamenti ed i solidi pieni sono visibili nei gruppi di immagine solo se la diapositiva è stata generata dopo aver eseguito il comando OMBRA. Altrimenti vengono visualizzati come contorni.
- Tenere sempre a mente lo scopo principale di queste immagini. Non eccedere nel loro uso mentre si codificano in simboli dei concetti astratti. L'utilità primaria delle immagini consiste innanzitutto nel fatto che l'utente può selezionare un simbolo grafico.

Capitolo 4 -- Menu personalizzati

Menu di schermo

La sezione relativa al menu di schermo controlla l'area del menu di schermo. Per default, il menu di schermo viene disattivato. Per attivarlo, scegliere Visualizza il menu di schermo di AutoCAD nella finestra di disegno nella scheda Visualizzazione della finestra di dialogo Preferenze.

Capitolo 4 -- Menu personalizzati

Menu di schermo

Creazione di menu di schermo

L'etichetta di sezione ***SCREEN rappresenta l'inizio dei menu di schermo di AutoCAD. L'etichetta della sezione del sottomenu riportata di seguito è identificata dalla stringa **S. Quando diverse voci separate fanno riferimento a questo sottomenu, conviene utilizzare un nome semplice e conciso.

<i>Menu di schermo</i>	<i>Sezione del file di menu di schermo</i>
	***SCREEN
	**S
AutoCAD	[AutoCAD]^C^C^P(ai_rootmenus) ^P
****	[* * * *]\$\$=ACAD.OSNAP
FILE	[FILE]\$\$=ACAD.01_FILE
MODIFICA	[MODIFICA]\$\$=ACAD.02_EDIT
VISUAL 1	[VISUAL1]\$\$=ACAD.03_VIEW1
VISUAL 2	[VISUAL2]\$\$=ACAD.04_VIEW2
INSERISC	[INSERISCI]\$\$=ACAD.05_INSERT
FORMATO	[FORMATO]\$\$=ACAD.06_FORMAT
STRUMEN1	[STRUMEN1]\$\$=ACAD.07_TOOLS1
STRUMEN2	[STRUMEN2]\$\$=ACAD.08_TOOLS2
DISEGNA1	[DISEGNA1]\$\$=ACAD.09_DRAW1
DISEGNA2	[DISEGNA2]\$\$=ACAD.10_DRAW2
QUOTE	[QUOTE]\$\$=ACAD.11_DIMENSION
EDITA 1	[EDITA 1]\$\$=ACAD.12_MODIFY1
EDITA 2	[EDITA 2]\$\$=ACAD.13_MODIFY2
	[?]\$\$=ACAD.14_HELP
	[AIUTI]\$\$=ACAD.ASSIST
	[ULTIMO]\$\$=ACAD.

AIUTI
PREC

Capitolo 4 -- Menu personalizzati

Menu di schermo

Sottomenu di schermo

Le etichette di sottomenu dei menu di schermo hanno il seguente formato:

```
**nomemenu [numinizio]
```

Il *nomemenu* è una stringa composta da un massimo di 33 caratteri contenente lettere, cifre e i caratteri di dollaro (\$), trattino (-) e trattino di sottolineatura (_). L'etichetta di sottomenu deve trovarsi da sola su una riga del file di menu e *non* deve contenere spazi. Un *numinizio* intero opzionale, che specifichi la riga iniziale del sottomenu, può seguire il *nomemenu*.

Mentre un sottomenu può contenere un numero qualsiasi di voci, ogni area di menu possiede un numero limitato di voci accessibili a causa della dimensione dello schermo. Ad esempio, se un sottomenu di menu di schermo ha 21 voci ma lo schermo può visualizzare solo 20 voci per volta, non è possibile accedere all'ultima voce del sottomenu.

Quando un sottomenu viene attivato, le relative voci sostituiscono di solito quelle del menu precedente partendo dall'inizio (casella di menu 1) e continuando attraverso tutte le voci del sottomenu. In tal modo, un sottomenu può sostituire solo una parte del menu precedente. Per specificare il punto da cui deve iniziare la sostituzione con una voce di menu diversa da 1, è possibile aggiungere un numero di voce dopo l'etichetta di sezione o di sottomenu, come nell'esempio seguente:

```
**SAMPLE 3
```

Quando il sottomenu SAMPLE viene attivato, le prime due caselle di menu non vengono cambiate e la sostituzione di sottomenu comincia a partire dalla casella 3. Se si specifica un numero di voce negativo, la sostituzione inizia con il numero di voci appropriato calcolato a partire dalla *fine* del menu (parte inferiore dello schermo).

Per ripristinare le voci di menu precedenti, una voce di menu deve inviare il seguente codice senza etichetta di sottomenu.

```
$$=
```

AutoCAD memorizza gli ultimi otto sottomenu. Se si supera questo numero, i primi menu vengono esclusi.

Il seguente esempio di sezione di menu Screen mostra l'uso dei sottomenu.

```
***SCREEN
[menuFACILE]
                                     Riga vuota
[DISEGNA... ]$$=Draw_Root
[MODIFICA... ]$$=Edit_Root
                                     Riga vuota
[Ciao      ]end
                                     Tre righe vuote portano a dieci le righe di
                                     questa pagina di menu e ricoprono le voci
                                     visualizzate dai sottomenu.
                                     Dato che sotto questa linea non vi è
                                     alcun sottomenu, questo compare in tutti i menu.
                                     Richiama
                                     il menu principale.
                                     Il 2 dopo il nome del sottomenu avvia
                                     questo menu sulla linea dopo [menuFACILE].

**Draw_Root 2
[Linea     ]linea
[Cerchio   ]cerchio
[Arco      ]arco
                                     Almeno una riga vuota.

**Edit_Root 2
[Cancella  ]$$=Sel_obj cancella
[Copia     ]$$=Sel_obj copia
[Sposta    ]$$=Obj_sel sposta
                                     Notare l'uso di un alias di menu.
                                     Almeno due righe vuote coprono
                                     le voci del menu Sel_obj.
                                     Per richiamare questo menu, si può usare sia Obj_sel
                                     che Sel_obj.

**Obj_sel 2
**Sel_obj 2
[Ultimo    ]last
```

```
[Precedente ]precedente
[Finestra ]finestra
[Intersezione]intersezione
```

```
[ -PREC- ]$$=
***BUTTONS1
;
redraw
```

*\$\$= richiama il menu precedente.
Menu pulsanti dispositivo di puntamento.
Assegna RETURN al pulsante 2
Assegna il comandoRIDIS
al pulsante 3.*

L'esempio precedente contiene tre sottomenu: Draw_Root, Edit_Root e Sel_obj.

Draw_Root e Edit_Root vengono richiamati dal menu principale di schermo quando si seleziona la voce di menu DISEGNA o MODIFICA. Il sottomenu Draw_Root fornisce tre voci di selezione corrispondenti a comandi AutoCAD. Il sottomenu Edit_Root contiene inoltre tre voci di selezione, ciascuna delle quali richiama il sottomenu Sel_obj prima di eseguire il comando appropriato.

In tutti i casi una voce di selezione -PRINCIPALE- richiama il menu di schermo principale. Un menu di schermo ricopre (cancella) solo tante righe del menu di schermo precedente quante ne può contenere. Se uno schermo di menu contiene più voci di quante siano le caselle sullo schermo, oppure se un menu dei pulsanti contiene più voci di quanti siano i pulsanti disponibili, le voci in eccesso vengono ignorate.

Nota Per rendere più lunghi i sottomenu nei file di menu in modo tale che possano coprire i menu precedenti è possibile utilizzare righe vuote. Inoltre, è possibile includere righe vuote per rendere il file più facile da leggere.

Selezionando una voce chiamata ZOOM dal menu di schermo principale, è possibile attivare un sottomenu contenente le opzioni per il comando ZOOM. Per informazioni su un metodo alternativo per richiamare un sottomenu di comandi, vedere "Variabile di sistema MENUCTL."

L'esempio seguente fa riferimento al sottomenu **01_FILE nel gruppo di menu ACAD.

```
[FILE ]$$=ACAD.01_FILE
```

La maggior parte dei menu di schermo nel file *acad.mnu* vengono caricati alla casella di menu 3, consentendo alle etichette di menu[AutoCAD] e [***] di restare sullo schermo.

L'esempio seguente mostra come un sottomenu **01_FILE venga visualizzato sullo schermo. Notare che la prima riga (per Nuovo) viene visualizzata in corrispondenza della casella di menu 3.

Menu di schermo Sezione del file di menu di schermo

AutoCAD	**01_FILE 3
****	[Nuovo]^C^C_new
Nuovo	[Apri]^C^C_open
Apri	[Salva]^C^C_qsave
Salvavel	[Salva con nome]^C^C_saveas
Salvacon	[Esporta]^C^C_export
Esporta	[Configura]^C^C_config
CFG	[Stampa...]^C^C_plot
Plot	[Controlla]^C^C_audit
Verif	[Ripristina]^C^C_recover
Recupera	[Elimina]^C^C_purge
Elimina	[Esci]^C^C_quit
Fine	
AIUTI	
PREC	

Le voci di menu Aiuti e Ultimo vengono visualizzate nella parte inferiore dell'area del menu di schermo, in quanto fanno parte della sezione di sottomenu **S, che non viene sovrascritta dal sottomenu **01_FILE.

Capitolo 4 -- Menu personalizzati

Menu di schermo

Etichette di voci

Se una voce di un menu di schermo non contiene una etichetta di voci, soli i primi otto caratteri di una macro di menu appaiono sul menu di schermo. Nell'esempio seguente il comando verrebbe visualizzato come SNAP 0.0.

SNAP 0.001

Se la voce contiene una etichetta, i primi otto caratteri dell'etichetta vengono visualizzati nella casella del menu di schermo appropriata; tutti gli altri caratteri possono fungere da caratteri.

Nota Il numero massimo di voci di menu visualizzabili sullo schermo dipende dal sistema in uso. Per richiamare il numero di caselle di menu di schermo, utilizzare la variabile di sistema SCREENBOXES.

Capitolo 4 -- Menu personalizzati

Menu di schermo

Variabile di sistema MENUCTL

La variabile di sistema MENUCTL controlla il passaggio automatico da un sottomenu di schermo all'altro quando viene inviato un

comando corrispondente. Quando la variabile MENUCTL è impostata su 1 (On) e viene richiamato un comando AutoCAD da una voce di menu, AutoCAD invia \$\$=*nomecmd* (dove *nomecmd* indica il nome del comando) per richiamare un sottomenu di schermo con il nome uguale a quello del comando. Il menu standard *acad.mnu* sfrutta questa funzione impostando 1 per la variabile MENUCTL dal file *acad.mnl*. Impostando 0 (Off) per questa variabile si modifica il funzionamento del menu standard, ma ciò può risultare utile per i menu personalizzati meno recenti.

Capitolo 4 -- Menu personalizzati

Menu di tavoletta

AutoCAD consente di configurare fino ad un massimo di quattro aree di tavoletta di digitalizzazione come aree di menu per input di comandi. Le sezioni del file di menu con etichette da TABLET1 a TABLET4 definiscono le macro di menu associate con le selezioni di tavoletta in queste aree.

Capitolo 4 -- Menu personalizzati

Menu di tavoletta

Creazione di menu di tavoletta

Le voci di menu nelle sezioni TABLET*n* usano la stessa sintassi di quelle delle altre sezioni. Le etichette di voce vengono trattate come quelle delle sezioni BUTTONS*n*; possono essere usate come commenti e non vengono visualizzate.

Le aree dei menu di tavoletta definite con il comando TAVOLET Cfg vengono divise in caselle di selezione di menu di uguali dimensioni; queste sono determinate dal numero di colonne e di righe specificate in ogni area. Queste caselle di selezione corrispondono direttamente alle righe che seguono le etichette di sezione TABLET*n* seguendo l'ordine da sinistra -a-destra e dall'alto -verso-il basso (a prescindere se contengono o meno del testo).

Per esempio, se si configura un'area di menu per cinque colonne e quattro righe, la voce di menu sulla riga immediatamente dopo l'etichetta di sezione corrisponde alla casella di selezione all'estrema sinistra nella riga superiore. Allo stesso modo, la voce di menu sull'ottava riga dopo l'etichetta di sezione corrisponde alla terza casella a sinistra nella seconda riga. AutoCAD può riconoscere fino ad un massimo di 32.766 voci di menu in ogni sezione di tavoletta, una quantità ben più che sufficiente per qualsiasi menu di tavoletta.

Nella sezione ***TABLET1 del file *acad.mnu* è possibile aggiungere le proprie macro di menu. Le voci di menu in quest'area corrispondono alle 225 caselle presenti nella parte superiore del modello di tavoletta (righe da A ad 1 e colonne da 1 a 25). Personalizzare soltanto le righe contenenti le etichette di menu.

Individuare nel file di menu la riga contenente la sezione ***TABLET1 e notare se le successive 225 righe contengono etichette di menu.

```
***TABLET1
[A-1]
[A-2]
[A-3]
.
.
.
[I-25]
```

Queste etichette corrispondono al sistema a griglie del modello. La macro di menu può essere aggiunta dopo l'etichetta di menu [*riga-colonna*] corrispondente, utilizzando il formato descritto precedentemente in questo capitolo. Si consiglia di *non* modificare le righe che seguono la casella [I-25].

Capitolo 4 -- Menu personalizzati

Guida specifica di menu

I messaggi di Guida della riga di stato rappresentano un elemento importante del sistema di Guida nativo. Questi sono i messaggi semplici, descrittivi che compaiono nella riga di stato quando viene scelta una voce di menu. La sezione di menu Helpstrings fornisce il supporto per questa forma di aiuto in linea.

L'esempio che segue mostra un semplice file di menu che utilizza la sezione Helpstrings.

```
***MENUGROUP=esempio
***POP1
ID_Title  [/TTitolo]
ID_Cancel [Annulla comando]^C^C
ID_Line   [/LLinea]^C^C_line
          [Disattiva linea](menucmd "Gsample.ID_Line = ~")
          [Controlla linea](menucmd "Gsample.ID_Line = !")
***POP2
[/2Title2]
[Altro menu a discesa](menucmd "Gsample.ID_Line = ~")
***HELPSTRINGS
ID_Title  [Questo è il menu Titolo]
ID_Cancel [Questa voce annulla il comando precedente]
ID_Line   [Questo comando disegna una linea semplice]
```

La sintassi per la sezione Helpstrings prevede il contrassegno di nome seguito da un'etichetta. Quando una voce di menu viene evidenziata, il contrassegno di nome per quella voce viene richiesto per un'entrata corrispondente nella sezione ***HELPSTRINGS. Se esiste questa corrispondenza, la stringa contenuta all'interno dell'etichetta viene visualizzata nella riga di stato.

Capitolo 4 -- Menu personalizzati

Tasti di scelta

AutoCAD supporta tasti di scelta-definiti dall'utente. Di seguito è riportato un breve esempio di una sezione Accelerators.

```
***ACCELERATORS
ID_Line   [SHIFT+CONTROL+"L"]
[CONTROL+"Q"]^C^C_quit
[CONTROL+SHIFT+"Z"]^C^Czoom extents
```

La sezione Accelerators contiene voci in uno dei seguenti formati Il primo è un contrassegno di nome (come ID_Line) seguito da un'etichetta contenente i tasti di modifica. Questi sono seguiti da un singolo carattere oppure da una stringa speciale i tasti virtuali (come "F12") racchiusa tra virgolette. Questo tipo di voce assegna una sequenza di tasti ad una voce di menu. È possibile concatenare più di un tasto di modifica con un altro usando il simbolo più (+), come nel primo esempio. Quando una sequenza speciale di tasti viene riconosciuta, la voce di menu associata al contrassegno di nome viene eseguita come se l'utente avesse scelto tale voce.

Il secondo metodo di definizione di un tasto di scelta usa un'etichetta contenente un tasto di modifica ed una stringa di tasti, seguiti da una sequenza di comandi. Questo metodo assegna una sequenza di tasti ad una stringa di comando e non possiede una voce di menu corrispondente.

La seguente tabella contiene l'elenco dei tasti di modifica validi.

Tasti di modifica validi

Stringa	Descrizione
CONTROL	Il tasto CTRL
SHIFT	Il tasto MAIUSC, presente sia a destra che a sinistra.

La tabella seguente elenca i tasti virtuali speciali. Questi tasti devono essere racchiusi tra virgolette.

Tasti virtuali speciali

Stringa	Descrizione	Eccezioni
"F1"	TastoF1	Sebbene sia possibile assegnare il tasto F1 ad una macro di menu, questa operazione è sconsigliata in quanto F1 è in genere

		<i>associato alla Guida. L'uso di un tasto di modifica con F1 è consentito.</i>
"F2"	TastoF2	<i>Senza alcun tasto di modifica, alterna lo stato della finestra di testo.</i>
"F3"	TastoF3	<i>Senza alcun tasto di modifica, esegue DDOSNAP.</i>
"F4"	TastoF4	<i>Senza alcun tasto di modifica, attiva e disattiva TABMODE.</i>
"F5"	TastoF5	<i>Senza alcun tasto di modifica, attiva e disattiva PIANOASS.</i>
"F6"	TastoF6	<i>Senza alcun tasto di modifica, attiva e disattiva COORDS.</i>
"F7"	TastoF7	<i>Senza alcun tasto di modifica, attiva e disattiva GRIDMODE.</i>
"F8"	TastoF8	<i>Senza alcun tasto di modifica, attiva e disattiva ORTHOMODE.</i>
"F9"	TastoF9	<i>Senza alcun tasto di modifica, attiva e disattiva SNAPMODE.</i>
"F11"	TastoF11	Nessuno
"F12"	TastoF12	Nessuno
"INSERT"	TastoINS	Nessuno
"DELETE"	TastoCANC	Nessuno
"ESCAPE"	TastoESC	<i>Sebbene sia possibile assegnare il tasto ESC ad una macro di menu, questa operazione è sconsigliata in quanto tale tasto è in genere associato alla funzione Annulla.</i>

Le sequenze CTRL+ESC e CTRL+MAIUSC+ESC non possono essere assegnate ad una macro di menu perché sono controllate da Windows.

		<i>L'uso del tasto di modifica MAIUSC è consentito.</i>
"UP"	TastoFRECCIA SU	<i>Deve essere utilizzato con il tasto di modifica CTRL.</i>
"DOWN"	TastoFRECCIA GIÙ	<i>Deve essere utilizzato con il tasto di modifica CTRL.</i>
"LEFT"	TastoFRECCIA SINISTRA	<i>Deve essere utilizzato con il tasto di modifica CTRL.</i>
"RIGHT"	TastoFRECCIA DESTRA	<i>Deve essere utilizzato con il tasto di modifica CTRL.</i>
"NUMPAD0"	Tasto0	Nessuno
"NUMPAD1"	Tasto1	Nessuno
"NUMPAD2"	Tasto2	Nessuno
"NUMPAD3"	Tasto3	Nessuno
"NUMPAD4"	Tasto4	Nessuno
"NUMPAD5"	Tasto5	Nessuno
"NUMPAD6"	Tasto6	Nessuno
"NUMPAD7"	Tasto7	Nessuno
"NUMPAD8"	Tasto8	Nessuno
"NUMPAD9"	Tasto9	Nessuno

Nota il tasto F10 viene utilizzato dal sistema operativo Windows come alternativa al tasto ALT e di conseguenza non è configurabile da parte dell'utente.

Gli utenti che eseguono spesso l'immissione di coordinate troveranno i seguenti miglioramenti di menu particolarmente utili.

```
[ "NUMPAD5 " ] @x^h
[ "NUMPAD6 " ] <0
[ "NUMPAD9 " ] <45
[ "NUMPAD8 " ] <90
[ "NUMPAD7 " ] <135
[ "NUMPAD4 " ] <180
[ "NUMPAD1 " ] <-135
[ "NUMPAD2 " ] <-90
[ "NUMPAD3 " ] <-45
```

Se il codice seguente viene aggiunto nella sezione Accelerators, il tastierino numerico viene modificato nel modo seguente: il tasto RETURN immette il simbolo @ e gli altri tasti numerici immettono il simbolo minore di (<), seguito dal valore angolare rappresentato dalla posizione corrispondente nel tastierino numerico. Ad esempio, se si desidera disegnare un quadrato con lato pari a 3 unità, è necessario digitare quanto segue:

Comando: **linea**

Dal punto: *specificare il punto iniziale*

Al punto: *(premere 5) 3 (premere 6)*

Al punto: *(premere 5) 3 (premere 2)*

Al punto: *(premere 5) 3 (premere 4)*

Al punto: **c**

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Panoramica

È possibile utilizzare il linguaggio DIESEL (Direct Interpretively Evaluated String Expression Language) per modificare la riga di stato di AutoCAD mediante la variabile di sistema MODEMACRO. Il linguaggio DIESEL può essere utilizzato anche nelle voci di menu come linguaggio per le macro al posto di AutoLISP. Le espressioni DIESEL utilizzano le stringhe come input e generano risultati sotto forma di stringhe.

Poiché le espressioni DIESEL gestiscono esclusivamente stringhe, le variabili di sistema USERS1-5 sono utili per passare informazioni da una routine AutoLISP ad un'espressione DIESEL. Le espressioni DIESEL possono essere valutate dalle routine AutoLISP tramite la funzione **menucmd** di AutoLISP.

Argomenti di questo capitolo

{button ,JI(','Status_Line_Configuration4545MODEMACRO_al_u0105')} Configurazione della riga di stato -- MODEMACRO

{ewc ,JI(','DIESEL_Expressions_in_Menus_al_u0105')} Espressioni DIESEL nei menu

{button ,JI(','DIESEL_Expressions_in_AutoLISP_al_u0105')} Espressioni DIESEL in AutoLISP

{ewc ,JI(','Catalog_of_DIESEL_String_Functions_al_u0105')} Catalogo delle funzioni delle stringhe DIESEL

{ewc ,JI(','Error_Messages_al_u0105')} Messaggi di errore

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Configurazione della riga di stato -- MODEMACRO

La riga di stato costituisce una risorsa molto importante, in quanto fornisce all'utente informazioni importanti senza interromperne il lavoro. Il valore calcolato della variabile di sistema MODEMACRO viene visualizzato in un riquadro della barra di stato nella parte inferiore della finestra di AutoCAD. Il numero di caratteri che è possibile visualizzare nella riga di stato è limitato soltanto dalle dimensioni della finestra di AutoCAD e da quelle del monitor in uso. Con l'aumentare delle dimensioni del riquadro MODEMACRO, i riquadri di default vengono spostati a destra. Questi riquadri possono inoltre essere rimossi completamente dallo schermo.

La maggior parte dei dati a disposizione di AutoCAD può essere visualizzata sulla riga di stato mediante la variabile MODEMACRO. Le funzioni riguardanti i calcoli, le scelte e le modifiche di MODEMACRO risulteranno molto utili nel momento in cui si desidera ottenere una riga di stato personalizzata in base alle proprie esigenze.

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Configurazione della riga di stato -- MODEMACRO

Variabile MODEMACRO

La variabile di sistema MODEMACRO controlla la riga di stato definita dall'utente. Quando si avvia AutoCAD, questa variabile è impostata come stringa nulla e non viene salvata nel disegno, nel file di configurazione o in altre parti. Se si desidera impostare MODEMACRO su un valore specifico ogni volta che si apre un disegno, è possibile caricare una definizione per MODEMACRO mediante la funzione **S::STARTUP** definita dal file *acad.lsp*.

MODEMACRO è una variabile utente di tipo stringa. Per questa variabile è possibile impostare qualsiasi valore di stringa e la sua lunghezza è limitata dalle restrizioni imposte da AutoLISP per le variabili di stringa e dalla dimensione del buffer per le comunicazioni da AutoLISP ad AutoCAD. Una lunghezza massima di stringa pari a 460 caratteri dovrebbe essere adeguata in tutti i sistemi. È possibile impostare MODEMACRO con il comando MODIVAR oppure digitando **modemacro** alla riga di comando. Se l'impostazione di MODEMACRO viene modificata, è possibile provare vari formati per la riga di stato; tuttavia, il numero massimo di caratteri che è possibile digitare in questo modo è 255.

Se MODEMACRO è una stringa nulla, impostata digitando un punto (.) oppure passando alla variabile stessa una stringa vuota ("") mediante la funzione AutoLISP **setvar**, AutoCAD visualizza la riga di stato standard.

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Configurazione della riga di stato -- MODEMACRO

Definizioni di MODEMACRO

Se il valore di MODEMACRO non è una stringa nulla, tale valore determina ciò che viene visualizzato nella riga di stato delle modalità. La MODEMACRO più semplice (e la meno utile) consiste in una stringa di testo costante. Ad esempio, per visualizzare sulla riga di stato il nome della propria azienda, si può digitare quanto riportato di seguito.

Comando: **modemacro**

Nuovo valore per MODEMACRO, o . per nessuno <"">: **Finanziaria Rossi**

Questa MODEMACRO visualizza sempre lo stesso testo; la riga di stato non riflette le modifiche nello stato interno di AutoCAD, in quanto quest'ultimo viene modificato solo quando si modifica MODEMACRO.

Per fare in modo che la riga di stato rifletta lo stato corrente di AutoCAD, includere nella riga di stato "espressioni macro" (usando il linguaggio DIESEL). Tali espressioni macro sono scritte nella forma riportata di seguito:

```
$(funzione, arg1, arg2, ...)
```

dove **funzione** è il nome della funzione DIESEL (simile al nome di una funzione AutoLISP) e *arg1*, *arg2* e così via sono gli argomenti per la funzione, interpretati secondo la definizione della funzione. A differenza di AutoLISP, le espressioni macro del linguaggio DIESEL presentano un solo tipo di dati: stringhe. Le macro che operano su numeri esprimono i numeri sotto forma di stringhe e li riconvertono come richiesto.

Definire una riga di stato più interessante, ad esempio una riga di stato che riporti il nome dello stile di testo corrente. Utilizzando il comando MODEMACRO, la riga di stato potrebbe essere definita nel seguente modo.

Comando: **modemacro**

Nuovo valore per MODEMACRO, o . per nessuno <"">: **Style: \$(getvar, textstyle)**

Nota Questo esempio e quelli che seguono mostrano la stringa di MODEMACRO che continua sulle righe successive. Tale stringa viene digitata come unica stringa lunga nell'area della riga del messaggio di richiesta.

Con la funzione **\$(getvar, nomevar)** è possibile richiamare qualsiasi variabile di sistema. La sua impostazione corrente sostituisce l'espressione macro nella riga di stato. Quindi, quando si scambiano i layer, il valore di MODEMACRO viene ricalcolato. Se è stato modificato, il nuovo layer viene visualizzato sullo schermo.

Le espressioni possono essere nidificate ed essere complesse. Si supponga di voler visualizzare il valore e l'angolo di snap corrente (in gradi) sulla riga di stato. L'esempio che segue usa espressioni nidificate per convertire l'angolo di snap da radianti in gradi, troncando il valore sull'intero.

Comando: **modemacro**

Nuovo valore per MODEMACRO, o . per nessuno <"">: **Snap: \$(getvar, snapunit)
\$(fix,\$(*,\$(getvar,snapang),\$(/,180,3.14159)))**

I valori possono anche essere visualizzati nelle modalità UNITA lineari ed angolari correnti nel modo riportato di seguito.

Comando: **modemacro**

Nuovo valore per MODEMACRO, o . per nessuno <"">: **Snap: \$(rtos,\$(index,0,
\$(getvar,snapunit))),\$(rtos,\$(index,1,\$(getvar,snapunit)) \$(angtos,\$(getvar,snapang))**

DIESEL copia il suo input direttamente sull'output fino a che non trova il carattere del dollaro (\$) o una stringa tra virgolette. È possibile utilizzare le stringhe tra virgolette per sopprimere la valutazione delle sequenze di caratteri che altrimenti verrebbero interpretate come funzioni DIESEL. Nelle stringhe tra virgolette è possibile includere altre virgolette utilizzando virgolette consecutive. Nell'esempio riportato di seguito, il layer corrente è impostato su LAYOUT, ed a MODEMACRO è stata assegnata una stringa.

Comando: **modemacro**

Nuovo valore per MODEMACRO, o . per nessuno <"">: **"\$(getvar,clayer)= ""\$(getvar,clayer)""**

La riga di stato visualizza quindi quanto riportato di seguito.

```
$(getvar,clayer)="LAYOUT"
```

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Configurazione della riga di stato -- MODEMACRO

Definizioni di MODEMACRO con AutoLISP

Per impostare le definizioni MODEMACRO è possibile utilizzare AutoLISP. Gli esempi riportati di seguito possono essere salvati come file di testo in formato ASCII e caricati con il comando AutoLISP **load**.

Il comando AutoLISP riportato di seguito definisce una MODEMACRO che simula la riga di stato corrente incorporata. Poiché AutoLISP non può continuare le stringhe da una riga all'altra, usare la funzione AutoLISP **strcat** per assemblare la stringa MODEMACRO completa costituita da stringhe di componenti più brevi.

```
(defun C:ACADMODE ( )
  (setvar "modemacro"
    (strcat
      "Layer $(substr,$(getvar,clayer),1,8) "
      "$(if,$(getvar,orthomode), Orto) "
      "$(if,$(getvar,snapmode), Snap) "
      "$(if,$(getvar,tabmode), Tavolet) "
      "$(if,$(=,$(getvar,tilemode),0), "
      "    "$(if,$(=,$(getvar,cvport),1), P) "
      ") "
    )
  )
)
```

Questa routine AutoLISP può essere salvata in un file chiamato *acadmode.lsp*. Quando viene caricata ed eseguita, produce una riga di stato che si comporta esattamente nello stesso modo della riga di stato standard. Tuttavia, questa non è l'applicazione più utile di questa funzione; infatti viene fornita solo come esempio. Poiché non si vedrà alcuna differenza nella riga di stato dopo che la routine è stata caricata, per sapere che la MODEMACRO realmente viene utilizzata è possibile cambiare *Layer* in *L*.

Il file *acad.lsp* di esempio riportato di seguito utilizza la funzione **S::STARTUP** per assegnare alla variabile MODEMACRO una stringa definita dal file AutoLISP *mode1.lsp*.

```
;;; File ACAD.LSP di esempio che usa S::STARTUP per
;;; caricare il file MODEL.LSP che definisce una stringa;
(defun S::STARTUP ( )
  (load "model")
  (princ)
)
;;; È possibile definire o caricare
;;; altri file AutoLISP.
```

Quando il file AutoLISP (*mode1.lsp*) riportato di seguito viene caricato, utilizza la variabile di sistema MODEMACRO per definire una riga di stato che visualizza la stringa **L:** seguita dai primi otto caratteri del nome del layer, quindi dal nome del disegno e da una parte del percorso, e che termina con la prima lettera del nome di ogni modalità correntemente attiva. La posizione del nome del disegno rimane costante, indipendentemente dalla lunghezza del nome del layer.

```
;;; MODEL.LSP
;;;
(setvar "modemacro"
  (strcat
    "L:$(substr,$(getvar,clayer),1,8) "
    "$(substr,          ,1,$(-,8,$(strlen,$(getvar,clayer)))) "
  )
)
;;;      ^^^^^^^^ Notare gli 8 spazi
"<.."
  "$(if,$(eq,$(getvar,dwgname),ANONIMO),ANONIMO,"
  "$(substr,$(getvar,dwgname),"
  "    "$(if,$(>,$(strlen,$(getvar,dwgprefix)),7),"
  "      "$(-,$(strlen,$(getvar,dwgprefix)),7),1"
  "    ),"
  "    "$(strlen,$(getvar,dwgname)) "
  ") "
  ") "
">"
  "$(if,$(getvar,orthomode), O, )"
  "$(if,$(getvar,snapmode), S, )"
)
```

```

    "$ (if, $(getvar, tabmode), T, ) "
    "$ (if, $(and, "
      "$ (=, $(getvar, tilemode), 0), $(=, $(getvar, cvport), 1)), P) "
  )
)

```

Come mostrato nel codice precedente, un rientro appropriato del testo rispetto al margine migliora la leggibilità dei file AutoLISP e delle stringhe DIESEL.

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Espressioni DIESEL nei menu

È possibile implementare le espressioni di stringhe DIESEL nei file di menu ed utilizzarle come metodo aggiuntivo per la creazione delle macro. Tali espressioni possono restituire valori di stringa in risposta ai comandi standard di AutoCAD, alle routine AutoLISP ed ADS e ad altre macro di menu. Esse possono anche restituire valori di stringa al menu stesso, modificando in questo modo l'aspetto o il contenuto dell'etichetta di un menu.

Se non si conoscono bene le procedure per la personalizzazione dei menu, prima di continuare questa sezione leggere capitolo 4, "Menu personalizzati".

Un'espressione DIESEL utilizzata in una voce di menu deve seguire il formato \$section=sottomenu dove il nome della sezione è M ed il sottomenu è l'espressione DIESEL desiderata. Generalmente, una macro di menu può essere implementata più facilmente con AutoLISP.

Gli esempi riportati di seguito mostrano due voci di menu che producono lo stesso risultato, una utilizza l'espressione DIESEL mentre l'altra utilizza l'espressione AutoLISP.

La voce di menu riportata di seguito utilizza l'espressione DIESEL.

```
[Ps/Ms]^C^C^P$M=$(if, $(=, $(getvar, cvport), 1), spaziom, spazioc)
```

La voce di menu riportata di seguito utilizza l'espressione AutoLISP.

```
[-Ps/Ms-]^C^C^P (if (= (getvar "cvport") 1) (command "spaziom")+
(command "spazioc")) (princ)
```

Entrambe le voci di menu consentono di passare dallo spazio carta allo spazio modello e viceversa (se TILEMODE è impostata su 0), ma l'espressione DIESEL è più corta e viene valutata in modo trasparente, senza dover eseguire la chiamata alla funzione AutoLISP **princ**. Se in entrambi i casi viene omesso il carattere speciale ^P (che attiva e disattiva MENU ECHO), l'espressione DIESEL visualizza solamente il comando inviato, mentre l'espressione AutoLISP visualizza l'intera riga di codice.

Poiché il valore restituito da un'espressione DIESEL è una stringa di testo, può essere utilizzato in risposta ad una chiamata alla funzione AutoLISP **getxxx**. Ciò consente alle voci di menu di valutare le condizioni correnti del disegno e di restituire un valore ad una routine AutoLISP.

L'esempio successivo presuppone quattro condizioni.

- La routine AutoLISP è caricata in memoria.
- La parte di menu interessata è inclusa nel menu corrente.
- I simboli da inserire sono sia alti che larghi una unità.
- La variabile DIMSCALE è impostata sul fattore di scala del disegno, cioè per un disegno che deve essere stampato su plotter con scala 1" = 10' il fattore di scala è 120, mentre per un disegno con scala 1/4" = 1' il fattore di scala è 48.

Se si carica e si esegue la routine AutoLISP di esempio, AutoCAD inserisce il simbolo secondo la dimensione e la posizione specificate. Quando si esegue la stampa su un plotter avente la stessa scala specificata da DIM-SCALE, le dimensioni dei simboli corrisponderanno a quelle specificate.

Di seguito viene riportata una routine AutoLISP di esempio.

```

(defun C:SYMIN ( )
  (setq sym
    (getstring "\nDigitare il nome del simbolo: "))
    ; Richiede il nome di un simbolo
  (menucmd "s=symsize")
    ; Passa dal menu di schermo
    ; al sottomenu symsize
  (setq siz

```

```

    (getreal "\nSelezionare la dimensione del simbolo: ")
    ; Richiede la dimensione del simbolo
    p1 (getpoint "\nPunto di inserimento: ") ; Richiede il punto di inserimento:
    (command "inser" sym p1 siz siz 0) ; Invia il comando INSER
    ; usando il simbolo, il
    ; punto di inserimento e la dimensione
desiderati
    (menucmd "s=") ; Passa al menu di schermo
    ; precedente
    (princ) ; Esce senza effettuare modifiche
)

```

Nota Per verificare la validità dell'input da parte dell'utente, una routine AutoLISP di uso frequente dovrebbe includere il controllo degli errori.

Le espressioni DIESEL presenti nel file di menu riportato di seguito moltiplicano il valore corrente di DIMSCALE per il valore specificato e restituiscono un fattore di scala appropriato. Questa operazione non può essere eseguita con un codice AutoLISP equivalente; un valore restituito da un'espressione AutoLISP generalmente non può essere utilizzato come risposta ad una chiamata alla funzione **getxxx**, come la funzione **getreal** dell'esempio precedente.

Di seguito è riportata una parte del file di menu.

```

**symsize 3
[ DIMENSIONI ]
[ 3/8" ]$M= $(*, $(getvar,dimscale),0.375)
[ 1/2" ]$M=$(*,$(getvar,dimscale),0.5)
[ 5/8" ]$M=$(*,$(getvar,dimscale),0.625)

```

Le espressioni DIESEL possono anche restituire valori di stringa alle etichette delle voci di menu a discesa, in modo da poter disattivare o modificare il modo in cui i menu vengono visualizzati. Per utilizzare un'espressione DIESEL nell'etichetta di un menu a discesa, è necessario essere sicuri che il primo carattere sia il carattere \$.

Nell'esempio riportato di seguito, il layer corrente è impostato su BASE e quanto segue viene utilizzato come parte dell'etichetta nella sezione *****POPn** di un file di menu.

```

[$ (eval,"Layer corrente: " $(getvar,clayer))]

```

Il risultato è che il menu a discesa appropriato viene visualizzato ed aggiornato ogni volta che il layer corrente viene modificato.

Layer corrente: BASE

Questo metodo può essere utilizzato anche per modificare interattivamente il testo visualizzato in un menu a discesa. Utilizzare una routine AutoLISP che assegni alle variabili di sistema USERS1-5 il testo selezionato, che può essere richiamato da una macro DIESEL in un'etichetta di menu.

Nota La larghezza dei menu a discesa e dei menu a cursore viene determinata quando il file di menu viene caricato. Le etichette di menu generate o modificate dalle espressioni DIESEL dopo che un menu è stato caricato, vengono troncate per poter rientrare nella larghezza del menu esistente.

Se si presume che un'etichetta di menu generata da un'espressione DIESEL sarà troppo larga, è possibile utilizzare l'esempio riportato di seguito per assicurarsi che la larghezza del menu sia sufficiente per le etichette. Questo esempio visualizza i primi 10 caratteri del valore corrente della variabile di sistema USERS3.

```

[$ (eval,"Valore corrente: " $(getvar,users3))+
  $(if, $(eq,$(getvar,users3),""), 10 spazi )]^C^Cusers3

```

Non è possibile utilizzare spazi finali in un'etichetta per aumentare la larghezza del menu, perché tali spazi vengono ignorati quando il menu viene caricato. Ogni spazio utilizzato per aumentare la larghezza di un'etichetta di menu deve essere specificato nell'ambito di un'espressione DIESEL.

Nell'esempio riportato di seguito viene utilizzata la stessa espressione DIESEL dell'etichetta ed una parte della voce di menu. Questo esempio presenta un modo pratico per inserire nel disegno il giorno e l'ora correnti.

```

[$ (edtime,$(getvar,date),DDD", "D MON YYYY)]^C^Ctext +
  \ $M=$(edtime,$(getvar,date),DDD", "D MON YYYY);

```

Una macro DIESEL può essere utilizzata anche per disattivare o contrassegnare le etichette dei menu a discesa. L'etichetta riportata di seguito visualizza ERASE disattivato (non selezionabile) mentre un comando è attivo. Il testo viene visualizzato in modo normale quando non ci sono comandi attivi.

```

[ $(if,$(getvar,cmdactive),~)CANCELLA]cancella

```

È possibile utilizzare un metodo simile per collocare un contrassegno accanto ad una voce di un menu a discesa o per modificare interattivamente il carattere utilizzato per il contrassegno stesso.

Nota L'utilizzo di DIESEL nelle etichette di menu è specifico per le piattaforme utilizzate, in quanto alcune piattaforme non aggiornano le etichette di menu dopo il caricamento iniziale del file di menu.

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Espressioni DIESEL in AutoLISP

È possibile utilizzare le espressioni DIESEL nelle routine AutoLISP chiamando la funzione AutoLISP **menu-cmd**. Il formato è simile a quello utilizzato per le espressioni DIESEL nelle voci di menu.

La parte di codice riportata di seguito imposta la variabile `c_time` secondo l'ora corrente.

```
(setq c_time (menucmd "M=$(edtime,$(getvar,date),HH:MM a/p)"))
```

È possibile utilizzare AutoLISP per fare esercitazioni con DIESEL. La routine di esempio riportata di seguito definisce un nuovo comando che è possibile utilizzare per digitare espressioni DIESEL alla riga di comando.

```
;;; DIESEL.LSP
;;; Consente di digitare espressioni DIESEL alla riga di comando
(defun C:DIESEL ( / dsl )
  (while (/= dsl "M=")
    (setq dsl (strcat "M=" (getstring T "\nDIESEL: ")))
    (princ (menucmd dsl))
  )
  (princ)
)
```

Dopo che questa routine è stata definita, digitando **diesel** alla riga di comando viene visualizzato un messaggio di richiesta DIESEL. A questo punto è possibile digitare qualsiasi espressione DIESEL. Se l'espressione digitata è valida, restituisce il risultato; in caso contrario, restituisce il messaggio di errore DIESEL appropriato. Questa routine continua a visualizzare il messaggio di richiesta DIESEL fino a quando non si fornisce una risposta nulla (premere RETURN).

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

Le funzioni DIESEL effettuano il richiamo dello stato, il calcolo e la visualizzazione. In questa sezione vengono descritte le funzioni disponibili.

Nota Per tutte le funzioni è possibile utilizzare al massimo 10 parametri, incluso il nome della funzione stessa; se questo limite viene superato, viene visualizzato un messaggio di errore DIESEL.

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

+ (addizione)

Restituisce la somma dei numeri *val1*, *val2*, ..., *val9*.

```
$(+, val1 [, val2, ..., val9] )
```

Se lo spessore corrente è impostato su 5, la stringa DIESEL riportata di seguito restituisce 15.

```
$(+, $(getvar,spessore),10)
```

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

- (sottrazione)

Restituisce il risultato della sottrazione da *val1* dei numeri da *val2* a *val9*.

§ (-, *val1* [, *val2*, ..., *val9*])

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

* (moltiplicazione)

Restituisce il risultato della moltiplicazione dei numeri *val1*, *val2*, ..., *val9*.

§ (*, *val1* [, *val2*, ..., *val9*])

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

/ (divisione)

Restituisce il risultato della divisione del numero *val1* per *val2*, ..., *val9*.

§ (/ , *val1* [, *val2*, ..., *val9*])

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

= (uguale a)

Se i numeri *val1* e *val2* sono uguali, la stringa restituisce 1, altrimenti restituisce 0.

§ (=, *val1*, *val2*)

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

< (minore di)

Se il numero *val1* è minore di *val2*, la stringa restituisce 1, altrimenti restituisce 0.

§ (< , *val1*, *val2*)

L'espressione riportata di seguito richiama il valore corrente di HPANG; se il valore è minore di quello memorizzato nella variabile di sistema USERR1, restituisce 1. Se in USERR1 è memorizzato il valore 10.0 e l'impostazione corrente di HPANG è 15.5, la stringa

riportata di seguito restituisce 0.

```
$( <, $(getvar,hpang),$(getvar,userri1) )
```

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

> (maggiore di)

Se il numero *val1* è maggiore di *val2*, la stringa restituisce 1, altrimenti restituisce 0.

```
$( >, val1, val2 )
```

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

!= (diverso da)

Se i numeri *val1* e *val2* sono diversi, la stringa restituisce 1, altrimenti restituisce 0.

```
$( !=, val1, val2 )
```

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

<= (minore di o uguale a)

Se il numero *val1* è minore o uguale a *val2*, la stringa restituisce 1, altrimenti restituisce 0.

```
$( <=, val1, val2 )
```

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

>= (maggiore di o uguale a)

Se il numero *val1* è maggiore o uguale a *val2*, la stringa restituisce 1, altrimenti restituisce 0.

```
$( >=, val1, val2 )
```

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

and

Restituisce l'AND logico a livello bit degli interi da *val1* a *val9*.

```
$(and, val1 [, val2,..., val9] )
```

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

angtos

Restituisce il valore angolare nel formato e con la precisione specificati.

```
$(angtos, valore [, modalità, precisione] )
```

Modifica il *valore* dato come angolo nel formato specificato da *modalità* e *precisione* secondo quanto definito per la funzione AutoLISP analoga. Nella tabella riportata di seguito vengono mostrati i valori per *modalità*. Se *modalità* e *precisione* vengono omessi, vengono utilizzati i valori correnti scelti dal comando UNITA.

Valori di unità angolari

Valore per modalità	Formato stringa
0	Gradi
1	Gradi/minuti/secondi
2	Gradi centesimali
3	Radiani
4	Unità topografiche

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

edtime

Restituisce l'ora e la data formattate in base ad una determinata figura.

```
$(edtime, ora, figura)
```

Modifica la data del calendario Giuliano di AutoCAD fornita da *ora* (ottenuto, ad esempio, da **\$(getvar,date)** secondo la *figura* fornita. La *figura* è costituita da frasi di formato sostituite da rappresentazioni specifiche della data e dell'ora. I caratteri non interpretabili come frasi di formato vengono copiati così come sono nel risultato di **\$(edtime)**. Le frasi di formato vengono definite come mostrato nella tabella riportata di seguito, in cui come esempio vengono utilizzate come data ed ora Giovedì 5 Settembre 1998 4:53:17.506.

Frasi di formato *edtime*

Formato	Output	Formato	Output
D	5	H	4
DD	05	HH	04
DDD	Gio	MM	53
DDDD	Giovedì	SS	17
M	9	MSEC	506
MO	09	AM/PM	AM
MON	Set	am/pm	am
MONTH	Settembre	A/P	A
YY	98	a/p	a
YYYY	1998		

Nota Digitare l'intera frase AM/PM come mostrato nella tabella precedente; se AM viene usato da solo, la A verrà letta letteralmente e la M restituirà il mese corrente.

Se nella figura appare AM/PM, le frasi H e HH daranno l'ora nel formato a 12 ore (12:00-12:59 1:00-11:59) anziché nel formato a 24 ore (00:00-23:59).

L'esempio riportato di seguito utilizza la data e l'ora della tabella precedente. Notare che la virgola deve essere specificata tra virgolette perché viene letta come separatore di argomenti.

```
$(edtime, $(getvar,date),DDD", " DD MON YYYY - H:MMam/pm)
```

Restituisce quanto riportato di seguito.

```
Gio, 5 Set 1998 - 4:53am
```

Se *ora* è uguale a 0, vengono utilizzate l'ora e la data in cui è stata eseguita la macro più esterna. Ciò evita chiamate multiple lunghe e che richiedono molto tempo a `$(getvar, date)` e assicura che tutte le stringhe composte da macro `$(edtime)` multiple utilizzino la stessa ora.

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

eq

Se le stringhe *val1* e *val2* sono identiche, la stringa restituisce 1, altrimenti restituisce 0.

```
$(eq, val1, val2)
```

L'espressione riportata di seguito prende il nome del layer corrente; se il nome corrisponde al valore della stringa memorizzato nella variabile di sistema `USERS1`, restituisce 1. Il presupposto è che la stringa "PARTE12" sia memorizzata nella variabile `USERS1` e che il layer corrente sia lo stesso.

```
$(eq, $(getvar,users1),$(getvar,clayer)) restituisce 1
```

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

eval

Passa la stringa *str* al programma di valutazione DIESEL e restituisce il risultato della valutazione.

```
$(eval, str)
```

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

fix

Tronca il numero reale *value* ad un intero eliminando eventuali parti frazionarie.

```
$(fix, valore)
```

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

getenv

Restituisce il valore della variabile di ambiente *varname*.

```
$(getenv, nomevar)
```

Se non è definita alcuna variabile con quel nome, viene restituita la stringa nulla.

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

getvar

Restituisce il valore della variabile di sistema con il *varname* specificato.

```
$(getvar, nomevar)
```

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

if

Valuta le espressioni in base a determinate condizioni.

```
$(if, espressione, esegui_se_vero[, esegui_se_falso] )
```

Se *espressione* è diversa da zero, valuta e restituisce *esegui_se_vero*. Altrimenti, valuta e restituisce *esegui_se_falso*. Notare che la parte non scelta da *espressione* non viene valutata.

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

index

Restituisce la porzione specificata di una stringa delimitata da virgole.

```
$(index, quale, stringa)
```

Si presume che l'argomento *string* contenga uno o più valori delimitati dal carattere di separazione degli argomenti delle macro, cioè la virgola. L'argomento *quale* permette di selezionare quale di questi valori deve essere estratto, con 0 definito per il primo elemento. Questa è la funzione utilizzata più di frequente per estrarre la coordinata X, Y o Z dalle coordinate del punto restituite da \$(getvar).

Questa funzione può essere usata dalle applicazioni per richiamare i valori memorizzati come stringhe delimitate da virgole dalle variabili di sistema da USERS1-5.

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

linelen

Restituisce la lunghezza, in caratteri, della riga di stato più lunga che può essere visualizzata per l'utente.

```
$(linelen)
```

È possibile utilizzare questa funzione per cambiare il formato della riga di stato, a seconda delle caratteristiche del video in uso. Questa funzione è utile solo per la configurazione della riga di stato MODEMACRO.

Nota Attualmente spazio disponibile per MODEMACRO sulla riga di stato è fisso e pari a caratteri. La funzione **\$(linelen)** restituisce sempre 64. Nelle future release di AutoCAD potrebbe non essere più disponibile.

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

nth

Valuta e restituisce l'argomento selezionato da *quale*.

```
$(nth, quale, arg0 [, arg1,..., arg7] )
```

Se *quale* è 0, *nth* restituisce *arg0*, e così via. Notare la differenza tra **\$(nth)** e **\$(index)**; **\$(nth)** restituisce alla funzione una delle serie di argomenti, mentre **\$(index)** estrae un valore da una stringa delimitata da virgole trasmessa ad un singolo argomento. Gli argomenti non selezionati da *quale* non vengono valutati.

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

or

Restituisce l'OR logico a livello bit degli interi compresi tra *val1* e *val9*.

```
$(or, val1 [, val2,..., val9] )
```

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

rtos

Restituisce il valore reale nel formato e con la precisione specificati.

```
$(rtos, valore [, modalità, precisione] )
```

Modifica il *valore* specificato come numero reale nel formato indicato dalla *modalità* e dalla *precisione* definite dalla funzione AutoLISP analoga. Se *modalità* e *precisione* vengono omesse, vengono utilizzati i valori correnti selezionati con il comando UNITA.

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

strlen

Restituisce la lunghezza in caratteri di *stringa*.

```
$(strlen, stringa)
```

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

substr

Restituisce la sottostringa di *stringa*, partendo dal carattere *inizio* ed estendendosi per il numero di caratteri indicato dall'argomento *lunghezza*.

```
$ (substr, stringa, inizio[, lunghezza] )
```

I caratteri nella stringa vengono numerati a partire da 1. Se l'argomento *lunghezza* viene omissso, viene restituita tutta la lunghezza rimanente della stringa.

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

upper

Restituisce la *stringa* convertita in lettere maiuscole secondo le regole locali correnti.

```
$ (upper, stringa)
```

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Catalogo delle funzioni delle stringhe DIESEL

xor

Restituisce l'operatore XOR logico a livello bit degli interi compresi tra *val1* e *val9*.

```
$ (xor, val1 [, val2,..., val9] )
```

Capitolo 5 -- Linguaggio DIESEL e configurazione della riga di stato

Messaggi di errore

Di solito, se si compie un errore in un'espressione DIESEL, l'errore risulta subito chiaro. A seconda della natura dell'errore, DIESEL inserisce un'indicazione dell'errore nel flusso di output.

Messaggi di errore DIESEL

Messaggio di errore	Descrizione
\$?	Errore di sintassi (di solito manca una parentesi destra o una stringa).
\$(funz,??)	Argomenti non corretti nella funz.
\$(funz)??	La funzione funz è sconosciuta.
\$(++)	La stringa di output è troppo lunga; la valutazione viene troncata.

Capitolo 6 -- Interfacce di programmazione

Panoramica

Oltre alle interfacce della riga di comando e dei menu, AutoCAD fornisce funzioni per l'esecuzione di comandi script ed interfacce di programmazione definibili dall'utente che possono essere utilizzate per operare su disegni e database. Le interfacce definibili dall'utente descritte in questo capitolo sono Automazione ActiveX (conosciuta in precedenza come Automazione OLE) ed AutoLISP. L'interfaccia ADS è obsoleta, ma viene comunque descritta in questo capitolo per coloro che utilizzano applicazioni precedenti alla Release 14. Il tipo di interfaccia che verrà utilizzato è determinato dalle richieste dell'applicazione e dall'esperienza di programmazione dell'utente. AutoCAD utilizza anche l'interfaccia dell'applicazione ARX per aumentare il gruppo di comandi standard di AutoCAD.

In questo capitolo vengono descritti i concetti e gli utilizzi delle varie interfacce di programmazione e viene spiegato come caricare ed utilizzare le applicazioni create da altri sviluppatori.

Per informazioni dettagliate sulla programmazione in AutoLISP, vedere la parte II "AutoLISP". Il linguaggio di programmazione ADS è descritto in *ARX Developer's Guide*. I programmi AutoLISP possono utilizzare finestre di dialogo con le loro applicazioni. Le finestre di dialogo programmabili sono descritte nella parte III "Finestre di dialogo programmabili".

Argomenti di questo capitolo

{button ,JI(','Command_Scripts_al_u0106')} Comandi di tipo script

{button ,JI(','ActiveX_Automation_al_u0106')} Automazione ActiveX

{ewc ,JI(','AutoLISP_al_u0106')} AutoLISP

{ewc ,JI(','ARX4545AutoCAD_Runtime_Extension_al_u0106')} ARX (AutoCAD Runtime Extension)

{ewc ,JI(','ADS_AL_U0106')} ADS

Capitolo 6 -- Interfacce di programmazione

Comandi di tipo script

AutoCAD dispone di una funzione *script* che consente di leggere comandi da un file di testo. Questa funzione è utile per l'esecuzione di una sequenza di comandi che possono essere richiamati quando AutoCAD viene avviato (utilizzando un formato speciale del comando **acad**), oppure è possibile eseguire uno script dall'interno di AutoCAD utilizzando il comando SCRIPT. La funzione script consente di creare facilmente una successione di schermate per la dimostrazione di prodotti ed a scopo commerciale.

I file di script vengono creati esternamente ad AutoCAD, utilizzando un editor di testo (come Blocco note di Windows) oppure un elaboratore di testi (come Word) che consenta di salvare il file in formato ASCII. L'estensione del file deve essere *.scr*.

I file di script possono contenere commenti. Ogni riga che inizia con un punto e virgola (;) viene considerata una riga di commento ed AutoCAD la ignora durante l'elaborazione del file di script.

Quando l'input di un comando proviene da uno script, le variabili di sistema PICKADD e PICKAUTO vengono impostate rispettivamente su 1 e 0. Ciò consente di mantenere la compatibilità con le release precedenti di AutoCAD e di facilitare la personalizzazione, poiché non è necessario verificare le impostazioni di queste variabili.

La funzione Annulla di AutoCAD considera uno script come un *gruppo*, reversibile tramite un singolo comando A. Tuttavia, l'esecuzione del file di script potrebbe essere rallentata poiché ogni comando presente nello script provoca un'immissione nel registro degli annullamenti. Se lo si desidera, è possibile usare ANNULLA Controllo Nessuno per disattivare la funzione Annulla prima di eseguire lo script oppure all'inizio dello script stesso. Quando l'esecuzione del file di script è terminata, attivare di nuovo la funzione Annulla (ANNULLA Controllo Tutto).

Tutti i riferimenti a nomi di file lunghi che contengono spazi devono essere racchiusi tra virgolette. Ad esempio, per aprire il disegno *miacasa.dwg* da uno script, utilizzare la sintassi seguente:

```
open "miacasa"
```

Lo script corrente viene terminato quando viene richiamato un altro comando SCRIPT.

Capitolo 6 -- Interfacce di programmazione

Comandi di tipo script

Come richiamare uno script durante il caricamento di AutoCAD

Per richiamare uno script all'avvio di AutoCAD, utilizzare il seguente formato di comando:

acad [disegno_esistente] [/t template] [/v view] /b script-file

Il file di script deve essere l'ultimo parametro contenuto nella riga di chiamata del programma **acad**; si presume che l'estensione del file sia `.scr`. Se il file di script non viene trovato, AutoCAD visualizza un messaggio indicante che è impossibile aprire il file. .

Considerare quanto riportato di seguito. Ogni volta che si inizia un nuovo disegno, viene attivata la griglia, la scala del tipo di linea globale viene impostata su 3.0 ed il layer 0 viene impostato come layer corrente con il colore rosso. Ciò può essere effettuato usando un disegno prototipo, ma per seguire lo scopo di questo capitolo, si consiglia di eseguire tutte queste operazioni con lo script riportato di seguito, memorizzato nel file `setup.scr`.

<code>griglia on</code>	<i>Attiva la griglia</i>
<code>scalat1 3.0</code>	<i>Imposta la scala per i tipi di linea</i>
<code>gruppo layer 0 colore rosso 0</code>	<i>Seleziona il layer corrente e ne imposta il colore</i>
	<i>Riga vuota per terminare il comando LAYER</i>

Per creare un disegno usando il file `MioModello.dwt` come modello, lanciare AutoCAD nel modo seguente:

acad /t MioModello /b setup

Questo comando crea un nuovo disegno e procede con l'immissione delle sequenze di comandi di setup dal file `setup.scr`. Al termine dell'esecuzione del file di script, viene visualizzata la riga di comando.

Nota In tal caso, non è più possibile iniziare un nuovo disegno con uno specifico nome. Il nome di file viene applicato al disegno quando quest'ultimo viene salvato.

È necessario conoscere la sequenza di messaggi di richiesta di AutoCAD per fornire la sequenza appropriata di risposte nel file di script. Tenere presente che i messaggi di richiesta ed i nomi dei comandi di AutoCAD nelle release successive possono cambiare, quindi potrebbe essere necessario rivedere i propri script quando viene eseguito l'aggiornamento ad una nuova versione di AutoCAD. In modo analogo, evitare di utilizzare le abbreviazioni; l'aggiunta successiva di altri comandi potrebbe dar luogo ad ambiguità. Inoltre, tenere presente che ogni spazio vuoto in un file di script è significativo; AutoCAD accetta lo spazio o RETURN come terminatore di comando o di campo dati.

Al messaggio di richiesta di sistema viene specificato lo script per la riconfigurazione, utilizzando la seguente sintassi.

acad -r nome-disegno file-script

Per eseguire uno script per un disegno che utilizza il prototipo di default (un nuovo disegno senza nome), utilizzare la sintassi seguente:

acad /b make_dwg

Capitolo 6 -- Interfacce di programmazione

Comandi di tipo script

Creazione di dimostrazioni con diapositive

Gli script sono utili per la creazione di dimostrazioni con diapositive. Generalmente la velocità alla quale è possibile visualizzare le diapositive è limitata dal numero di accessi al disco necessari per leggere il file che le contiene. Tuttavia, è possibile precaricare dal disco in memoria la diapositiva successiva mentre il pubblico osserva quella corrente e quindi visualizzare velocemente la nuova diapositiva presente in memoria.

Per eseguire il precaricamento di una diapositiva, specificare un asterisco prima del nome del file nel comando VISDIA. Il successivo comando VISDIA rileva che una diapositiva è stata precaricata e la visualizza senza richiedere il nome del file. Ad esempio si consideri lo script riportato di seguito:

<code>VISDIA DIAPO1</code>	<i>Inizia la dimostrazione delle diapositive, carica DIAPO1</i>
<code>VISDIA *DIAPO2</code>	<i>Precarica la diapositiva DIAPO2</i>

PAUSA 2000	<i>Consente al pubblico di osservare DIAPO1</i>
VISDIA	<i>Visualizza DIAPO2</i>
VISDIA *DIAPO3	<i>Precarica DIAPO3</i>
PAUSA 2000	<i>Consente al pubblico di osservare DIAPO2</i>
VISDIA	<i>Visualizza DIAPO3</i>
PAUSA 3000	<i>Consente al pubblico di osservare DIAPO3</i>
RSCRIPT	<i>Ricomincia con la prima istruzione</i>

Il tempo di accesso al disco necessario per caricare la diapositiva successiva si sovrappone a quello di visione della diapositiva corrente. Possono essere utilizzati anche ulteriori pause.

Capitolo 6 -- Interfacce di programmazione

Automazione ActiveX

Automazione ActiveX (conosciuta in precedenza come Automazione OLE) è una nuova interfaccia di programmazione per AutoCAD, che consente di sviluppare script, macro ed applicazioni di terzi utilizzando gli ambienti di programmazione di Automazione come Visual Basic 4.0. Mediante questa interfaccia, AutoCAD espone oggetti programmabili che possono essere manipolati dai controllori di Automazione (ad esempio Visual Basic ed Excel).

Con Automazione è possibile creare e manipolare oggetti di AutoCAD da qualsiasi applicazione che funga da controllore di Automazione. In questo modo, Automazione la programmazione delle macro tra applicazioni, una funzione che non esiste in AutoLISP. Con Automazione è possibile combinare le funzioni di molte applicazioni in un'unica applicazione.

Gli oggetti esposti vengono chiamati oggetti di Automazione, che a loro volta espongono metodi e proprietà. I metodi sono funzioni che eseguono un'azione su un oggetto, mentre le proprietà sono funzioni che impostano o forniscono informazioni sullo stato di un oggetto.

Per creare applicazioni che fanno uso degli oggetti di Automazione esposti da AutoCAD, vedere *Automazione Active X* nella Guida in linea.

Capitolo 6 -- Interfacce di programmazione

Automazione ActiveX

Uso di applicazioni di Automazione

In teoria, qualsiasi tipo di applicazione può accedere in AutoCAD agli oggetti di Automazione esposti. Queste applicazioni possono essere file eseguibili indipendenti, file DLL e macro in applicazioni come Word o Excel. Le più comuni sono probabilmente i file eseguibili indipendenti. Se si utilizzano applicazioni di sviluppatori di applicazioni, seguire le istruzioni su installazione ed uso relative. Questa sezione descrive alcuni modi per eseguire un file eseguibile indipendente da AutoCAD.

Capitolo 6 -- Interfacce di programmazione

Automazione ActiveX

Uso di applicazioni di Automazione

Lancio di un'applicazione dalla riga di comando

Per definire un nuovo comando di AutoCAD che esegua un comando esterno per il lancio dell'applicazione, è possibile utilizzare il file *acad.pgp*. L'esempio seguente definisce il comando RUNAPP1, che lancia l'applicazione *app1.exe* nella directory *C:\vbapps* (aggiungere questo codice alla sezione dei comandi esterni del file *acad.pgp*).

```
RUNAPP1, start c:\vbapps\app1, 0
```

Se l'applicazione richiede parametri dalla riga di comando, utilizzare quanto segue:

```
RUNAPP2, start c:\vbapps\app2, 0, *Parameters: ,
```

L'esempio precedente definisce il comando RUNAPP2; questo comando richiede la specifica di alcuni parametri, i quali vengono successivamente passati all'applicazione in uso. Per ulteriori informazioni sul file *acad.pgp*, vedere "File dei parametri di programma: acad.pgp."

Per lanciare un'applicazione è anche possibile utilizzare la funzione **startapp** di AutoLISP. In questo modo, una routine AutoLISP può eseguire un'applicazione che fa uso di Automazione. Naturalmente, dopo il lancio dell'applicazione esterna, AutoLISP non ha controllo sulle azioni dell'applicazione stessa, ma è comunque possibile utilizzare AutoLISP per individuare ed eseguire applicazioni differenti in base a certi parametri.

Capitolo 6 -- Interfacce di programmazione

Automazione ActiveX

Uso di applicazioni di Automazione

Lancio di un'applicazione da un menu

Dopo la definizione di un nuovo comando per il lancio di una applicazione (come descritto nella sezione precedente), è possibile utilizzare quel comando in una macro di menu, che può essere richiamata da una voce di menu in qualsiasi area di menu. Se vengono utilizzate solo una o due applicazioni, è possibile aggiungerle ad uno dei menu a discesa standard; se invece si utilizza un gruppo di applicazioni, è possibile aggiungere un menu a discesa o una barra degli strumenti dedicata specificatamente a tali applicazioni. Il capitolo 4 "Menu personalizzati" descrive tutte le opzioni disponibili per la personalizzazione dei menu.

L'esempio che segue è un file di menu completo che definisce un nuovo menu a discesa denominato MieApp. Questo file definisce il gruppo di menu MIEAPP e due gruppi di voci di menu. Le prime tre voci di menu utilizzano la funzione AutoLISP **startapp** per lanciare l'applicazione associata, mentre le ultime due presuppongono che i comandi APP4 e APP5 siano stati definiti nel file *acad.pgp*. Questo file include inoltre una sezione Helpstring, che fornisce la guida sulla riga di stato quando la voce di menu relativa è evidenziata. La sezione Accelerators definisce i tasti di scelta che eseguono le macro di menu. Queste sezioni utilizzano contrassegni di nome (ad esempio ID_App1) per collegare le loro azioni associate.

```
// MIEAPP Menu
//
***MENUGROUP=MIEAPP
// Questa sezione definisce il nuovo menu a discesa
***POP1
ID_MenuMieApps [Mie&App]
ID_MiaApp1 [App1 ] ^C^C^P(startapp "app1") (princ) ^P
ID_MiaApp2 [App2 ] ^C^C^P(startapp "app2") (princ) ^P
ID_MiaApp3 [App3 ] ^C^C^P(startapp "app3") (princ) ^P
[--]
ID_MiaApp4 [App4 ] ^C^CAPP4
ID_MiaApp5 [App5 ] ^C^CAPP5
// Questa sezione definisce i messaggi della riga di stato che
// vengono visualizzati quando una voce di menu associata viene evidenziata
***HELPSTRINGS
ID_MiaApp1 [Questa è APP1]
ID_MiaApp2 [Questa è APP2]
ID_MiaApp3 [Questa è APP3]
ID_MiaApp4 [Questa è APP4]
ID_MiaApp5 [Questa è APP5]
// Questa sezione definisce i tasti di scelta che eseguono
// le macro di menu nei menu a discesa relativi
***ACCELERATORS
ID_MiaApp1 [CTRL+MAIUSC+"1"]
ID_MiaApp2 [CTRL+MAIUSC+"2"]
ID_MiaApp3 [CTRL+MAIUSC+"3"]
ID_MiaApp4 [CTRL+MAIUSC+"4"]
ID_MiaApp5 [CTRL+MAIUSC+"5"]
```

Dopo aver salvato le sezioni nel file *mieapp.mnu* utilizzare CARMENU per aggiungere questo menu alla barra dei menu. Quando si

carica il suddetto file, è necessario specificare l'estensione di file *.mnu*.

Capitolo 6 -- Interfacce di programmazione

AutoLISP

AutoLISP si basa sul linguaggio di programmazione LISP che è molto facile da apprendere nonostante le sue caratteristiche avanzate. AutoCAD è dotato di un interprete LISP incorporato che viene utilizzato per digitare il codice AutoLISP alla riga di comando o per caricare il codice AutoLISP da file esterni. Le applicazioni o le routine AutoLISP possono interagire in molti modi con AutoCAD. Queste routine possono richiedere l'input dell'utente, accedere direttamente ai comandi incorporati di AutoCAD e modificare o creare oggetti nel database di disegno. Attraverso la creazione di routine AutoLISP è possibile aggiungere ad AutoCAD comandi specifici di funzionamento. Molti dei comandi standard di AutoCAD sono in effetti applicazioni AutoLISP.

Poiché AutoCAD legge direttamente il codice AutoLISP, non è richiesta alcuna compilazione. Digitando il codice alla riga di comando è possibile vedere immediatamente i risultati. Tale caratteristica rende il linguaggio AutoLISP uno strumento di facile utilizzo per l'esecuzione di prove, indipendentemente dalla grado di esperienza di programmazione. Dopo aver acquisito una certa esperienza con AutoLISP, ci si renderà conto di utilizzarlo come un'estensione ai comandi base di AutoCAD.

Anche se non si è interessati alla scrittura di applicazioni AutoLISP, il pacchetto di AutoCAD include molte routine utili nelle directory *sample* e *bonus*. Le routine sono inoltre disponibili come prodotti shareware e tramite altre società. Sapere come caricare ed utilizzare queste routine può aumentare la produttività.

Nota Quando l'input di comandi proviene dalla funzione AutoLISP **command**, le variabili di sistema PICKADD e PICKAUTO vengono impostate per default rispettivamente su 1 e 0. Ciò consente di mantenere la compatibilità con le release precedenti di AutoCAD e di facilitare la personalizzazione, poiché non è necessario verificare le impostazioni di queste variabili.

Capitolo 6 -- Interfacce di programmazione

AutoLISP

Uso di applicazioni AutoLISP

Le applicazioni AutoLISP sono memorizzate in file di testo ASCII con estensione *.isp*. Questi file hanno generalmente una parte di intestazione che descrive le routine, il loro utilizzo e tutte le istruzioni specifiche. Questa intestazione può contenere anche commenti relativi al nome dell'autore ed informazioni sull'uso delle routine. I commenti sono preceduti da un punto e virgola (;). Questi file possono essere visualizzati e modificati con un editor di testo o un elaboratore di testi in grado di produrre file di testo ASCII.

Prima di poter utilizzare un'applicazione AutoLISP è necessario caricarla. Per eseguire il caricamento di un'applicazione, è possibile usare il comando APPLOAD (anch'esso un'applicazione AutoLISP) o la funzione AutoLISP **load** (vedere "APPLOAD" nella *Guida di riferimento dei comandi di AutoCAD*). Caricando un'applicazione AutoLISP, viene eseguito il caricamento del codice AutoLISP dal file *.isp* alla memoria di sistema.

Per caricare un'applicazione con la funzione **load** è necessario digitare il relativo codice AutoLISP alla riga di comando. Se la funzione **load** ha esito positivo, visualizzerà alla riga di comando il valore dell'ultima espressione contenuta nel file che, generalmente, è il nome dell'ultima funzione definita nel file oppure sono le istruzioni sull'uso della funzione definita recentemente. Se la funzione **load** ha esito negativo, restituirà un messaggio di errore AutoLISP. L'esito negativo della funzione **load** può essere causato dalla presenza nel file di codifica errata oppure dalla errata digitazione alla riga di comando del nome di file. La sintassi per la funzione **load** è:

```
(load nomefile [se_fallisce])
```

Questa sintassi mostra che la funzione **load** ha due argomenti: *nomefile*, obbligatorio e *se_fallisce*, opzionale. Quando un file AutoLISP viene caricato alla riga di comando, generalmente viene fornito solo l'argomento *nomefile*. Con il seguente esempio viene caricato il file AutoLISP *miofile.isp*.

Comando: (load "miofile")

L'estensione *.isp* non è necessaria. Questo formato è valido per qualsiasi file con estensione *.isp* contenuto nella libreria corrente (vedere "Percorso di ricerca per le librerie").

Per caricare un file AutoLISP che non si trova nel percorso della libreria, è necessario che nell'argomento *nomefile* siano specificati il percorso completo ed il nome del file. Quando viene specificato il percorso di una directory, è necessario usare una barra (/) o due

barre inverse (\) come separatore, poiché una singola barra inversa ha un significato particolare in AutoLISP.

Comando: `(load "d:/file/piulisp/nuovfile")`

Capitolo 6 -- Interfacce di programmazione

AutoLISP

Caricamento ed esecuzione automatici

Probabilmente, dopo aver creato una libreria di routine AutoLISP utili, sarà opportuno caricarle ad ogni esecuzione di AutoCAD. Inoltre, potrebbe essere necessario eseguire alcuni comandi o funzioni in un determinato momento durante una sessione di disegno.

AutoCAD carica automaticamente il contenuto di due file definibili dall'utente: *acad.lsp* e *.mnl*; il secondo è associato al menu corrente. Se uno di questi file definisce una funzione del tipo speciale **S::STARTUP**, questa routine viene eseguita immediatamente dopo la completa inizializzazione del disegno. La funzione **S::STARTUP** è descritta in "Funzione S::STARTUP: esecuzione automatica."

Capitolo 6 -- Interfacce di programmazione

AutoLISP

Caricamento ed esecuzione automatici

Caricamento automatico del file *acad.lsp*

Con il file *acad.lsp* è possibile caricare una libreria di routine AutoLISP tutte le volte che AutoCAD viene avviato. Ogni volta che viene iniziato un disegno, AutoCAD cerca un file *acad.lsp* nel percorso della libreria. Se il file viene trovato, viene caricato in memoria.

Se la reinizializzazione di AutoLISP è attivata, il file *acad.lsp* viene caricato ad ogni avvio di nuovi disegni o ad ogni apertura di disegni esistenti. Se invece è disattivata, il file viene caricato solo all'avvio di AutoCAD. La reinizializzazione di AutoLISP è controllata dall'opzione Ricarica AutoLISP tra i disegni nella scheda Compatibilità della finestra di dialogo Preferenze e dalla variabile di sistema LISPINIT.

Nota Non modificare il file *acadr14.lsp*. Il file *acadr14.lsp* contiene le funzioni definite da AutoLISP che sono richieste da AutoCAD. Questo file viene caricato in memoria immediatamente prima del caricamento del file *acad.lsp*.

Nel file *acad.lsp* può essere inserito il codice AutoLISP relativo ad una o più routine oppure solo una serie di chiamate alla funzione **load**. Si consiglia di utilizzare il secondo metodo, poiché è più semplice eseguire successive modifiche. Se il codice riportato di seguito viene salvato come un file *acad.lsp*, i file *miaapp1.lsp*, *build.lsp* e *counter.lsp* vengono caricati ogni volta che AutoCAD viene avviato.

```
(load "miaapp1")
(load "build")
(load "counter")
```

AutoCAD ricerca il file *acad.lsp* nell'ordine definito dal percorso della libreria; tuttavia, è possibile avere file *acad.lsp* diversi in ogni directory di disegno. Pertanto, per determinati tipi di disegni o lavori è possibile caricare routine AutoLISP specifiche.

Se si verifica un errore AutoLISP durante il caricamento del file *acad.lsp*, le righe successive all'istruzione che ha provocato l'errore vengono ignorate e non vengono caricate. Generalmente i file specificati in *acad.lsp* che non esistono o che non si trovano nel percorso della libreria di AutoCAD provocano errori. Pertanto, potrebbe essere necessario usare l'argomento *se_fallisce* con la funzione **load**. L'esempio precedente di file *acad.lsp* può essere scritto di nuovo per specificare l'argomento *se_fallisce*, come mostrato nell'esempio riportato di seguito.

```
(princ (load "miaapp1" ""\nMIAAPPL.LSP non è stato caricato.))
(princ (load "build" ""\nBUILD.LSP non è stato caricato.))
(princ (load "counter" ""\nCOUNTER.LSP non è stato caricato.))
(princ)
```

Se una chiamata alla funzione **load** ha esito positivo, restituisce il valore dell'ultima espressione contenuta nel file che generalmente riporta il nome dell'ultima funzione definita oppure un messaggio relativo all'uso della funzione. Se la funzione ha esito negativo,

restituisce il valore dell'argomento *se_fallisce*. Nell'esempio precedente, il valore restituito dalla funzione **load** viene passato alla funzione **princ**, provocando la visualizzazione del valore sulla riga di comando. Ad esempio, se si verifica un errore mentre AutoCAD carica il file *miaapp1.lsp*, la funzione **princ** visualizza il seguente messaggio ed AutoCAD continua a caricare i rimanenti due file.

MIAAPP1.LSP non è stato caricato.

Nota Se la funzione **command** viene utilizzata in un file *acad.lsp* o *.mnl*, deve essere richiamata solo dall'interno di una istruzione **defun**. Utilizzare la funzione **S::STARTUP** per definire i comandi che devono essere eseguiti immediatamente all'inizio di una sessione di disegno.

Capitolo 6 -- Interfacce di programmazione

AutoLISP

Caricamento ed esecuzione automatici

Caricamento automatico del file *.mnl*

L'altro tipo di file che AutoCAD carica automaticamente è associato al file di menu corrente ed ha l'estensione *.mnl*. Quando AutoCAD carica un file di menu, ricerca un file *.mnl* con il nome di file corrispondente. Se il file viene trovato, viene caricato in memoria.

Questa funzione assicura che AutoCAD esegua il caricamento delle funzioni AutoLISP che sono necessarie per il corretto funzionamento di un menu. Ad esempio, per il menu standard di AutoCAD, *acad.mnu*, è necessario che il file *acad.mnl* venga caricato correttamente. Questo file definisce numerose funzioni AutoLISP utilizzate dal menu. Il file *.mnl* viene caricato dopo il file *acad.lsp*.

Nota Se un file di menu viene caricato con la funzione AutoLISP **command** con una sintassi simile a (command "menu" "nuovmenu"), il file *.mnl* associato non viene caricato fino a quando la routine AutoLISP non è stata eseguita completamente.

Ad esempio, se viene creato un nuovo menu chiamato *nuovmenu.mnu* ed è necessario caricare tre file AutoLISP (*nuovo1.lsp*, *nuovo2.lsp* e *nuovo3.lsp*) affinché il menu funzioni correttamente, è opportuno creare un file di testo ASCII chiamato *nuovmenu.mnl* nel modo seguente:

```
(load "nuovo1")
(load "nuovo2")
(load "nuovo3")
(princ "\nUtilità nuovmenu... caricate.")
(princ)
```

In questo esempio, le chiamate alla funzione **princ** possono essere utilizzate per visualizzare messaggi di stato. La prima chiamata alla funzione **princ** visualizza sulla riga di comando quanto riportato di seguito.

Utilità nuovmenu... caricate.

La seconda chiamata alla funzione **princ** esce senza salvare dalla funzione AutoLISP. Senza questa seconda chiamata alla funzione **princ**, il messaggio verrebbe visualizzato due volte. Come detto precedentemente, con chiamate alla funzione **load** è possibile includere, per ulteriore precauzione, l'argomento *se_fallisce*.

Capitolo 6 -- Interfacce di programmazione

AutoLISP

Caricamento ed esecuzione automatici

Caricamento automatico dei comandi

Quando un comando viene caricato automaticamente con le funzioni descritte precedentemente, la definizione del comando occupa la memoria del sistema indipendentemente dal fatto che il comando venga effettivamente utilizzato. La funzione AutoLISP **autoload** rende disponibile un comando senza dover effettuare il caricamento nella memoria dell'intera routine. Aggiungendo al file *acad.lsp* il codice riportato di seguito, viene effettuato il caricamento automatico dei comandi CMD1, CMD2 E CMD3 dal file *cmds.lsp* E DEL COMANDO NEWCMD dal file *newcmd.lsp*.

```
(autoload "CMDS" ' ("CMD1" "CMD2" "CMD3"))
```

```
(autoload "NEWCMD" ' ("NEWCMD"))
```

La prima volta che alla riga di comando viene digitato un comando caricato automaticamente, AutoLISP carica l'intera definizione del comando dal file ad esso associato. Per ulteriori informazioni sulla funzione **autoload**, vedere "**autoload**" nel capitolo 13. In AutoLISP sono inoltre disponibili funzioni ad autocaricamento per applicazioni ARX (vedere "**autoload**" e "**autoarxload**" nel capitolo 13).

Capitolo 6 -- Interfacce di programmazione

ARX (AutoCAD Runtime Extension)

ARX (AutoCAD Runtime Extension) è un ambiente di programmazione in linguaggio compilato per lo sviluppo di applicazioni AutoCAD. L'ambiente ARX supporta attualmente la libreria ADS con ADSRX ed in futuro supporterà molte altre librerie.

Per migliorare le prestazioni, le applicazioni ARX vengono attivate nella stessa elaborazione e nello stesso spazio di memoria di AutoCAD. Le interfacce API possono essere esportate in modo più efficiente dall'ambiente ARX rispetto all'ambiente di programma ADS.

Capitolo 6 -- Interfacce di programmazione

ARX (AutoCAD Runtime Extension)

Uso di applicazioni ARX

Per caricare una applicazione ARX, usare l'opzione Load del comando ARX. DOPO IL CARICAMENTO, TUTTI I COMANDI DEFINITI DA TALE APPLICAZIONE RISULTANO DISPONIBILI NELLA RIGA DI COMANDO.

Le applicazioni ARX occupano memoria di sistema. Quando non è più necessario usare un'applicazione e si desidera rimuoverla dalla memoria, è possibile usare l'opzione Unload del comando ARX.

Una applicazione ARX può essere caricata anche con la funzione AutoLISP **arxload**. La sintassi per la funzione **arxload** è quasi identica a quella della funzione **load** utilizzata con i file AutoLISP ed a quella della funzione **xload** utilizzata con le applicazioni ADS. Se la funzione **arxload** esegue correttamente il caricamento del programma ARX, restituisce il nome del programma. La sintassi per la funzione **arxload** è:

```
(arxload nomefile [se_fallisce])
```

I due argomenti per la funzione **arxload** sono *nomefile* e *se_fallisce*. Come per la funzione **load**, l'argomento *nomefile* è obbligatorio e deve essere il percorso completo del file del programma ARX che si intende caricare. L'argomento *se_fallisce* è opzionale e generalmente non viene usato quando si esegue il caricamento di programmi ARX dalla riga di comando. Con il seguente esempio viene caricata l'applicazione ARX *miaapp.arx*.

```
(arxload "miaapp")
```

Come per i file AutoLISP, AutoCAD ricerca il file specificato nel percorso della libreria. Per caricare un file che non si trova nel percorso della libreria, è necessario fornire il percorso completo del file. Assicurarsi di usare una singola barra (/) o due barre rovesciate (\) come separatore di directory quando si specifica il nome completo del percorso.

Se si tenta di caricare una applicazione che è già stata caricata, verrà visualizzato un messaggio di errore. Prima di usare **arxload**, è necessario usare la funzione **arx** per controllare quali sono le applicazioni caricate.

Per scaricare una applicazione con AutoLISP, utilizzare la funzione **arxunload**. Con il seguente esempio viene scaricata l'applicazione *miaapp.lsp*.

```
(arxunload "miaapp")
```

Utilizzando la funzione **arxunload**, dalla memoria non viene rimossa solamente l'applicazione, ma anche le definizioni dei comandi ad essa associati.

Capitolo 6 -- Interfacce di programmazione

ARX (AutoCAD Runtime Extension)

Caricamento automatico delle applicazioni ARX

Il file *acad.rx* contiene un elenco dei file di programma ARX che vengono caricati automaticamente quando si avvia AutoCAD. È possibile modificare questo file con un editor di testo o con un programma di elaborazione testi che crea dei file in formato testo ASCII. Questo file può essere personalizzato nel modo desiderato, effettuando aggiunte e cancellazioni e rendendo disponibili per l'uso i programmi ARX appropriati. AutoCAD legge questo file in modo simile al file *acad.rx*.

Le funzioni di caricamento automatico possono essere utilizzate con i comandi di AutoCAD definiti in ARX. Vedere "Caricamento automatico dei comandi" e "**autoarxload**" nel capitolo 13.

Le applicazioni ARX possono essere caricate da un file *.mnl* utilizzando la funzione **arxload**. Ciò assicura che un programma ARX, richiesto per il corretto funzionamento di un menu, verrà caricato quando il file di menu sarà caricato.

Capitolo 6 -- Interfacce di programmazione

ARX (AutoCAD Runtime Extension)

ADSRX

ADSRX è un sottogruppo di funzioni ARX, equivalenti alle funzioni disponibili in precedenza con ADS. ADSRX è disponibile con le librerie ed i file di intestazione forniti con AutoCAD e consente di trasferire le applicazioni ADS nelle applicazioni ARX.

Poiché lo scopo principale di ADSARX è quello di facilitare l'importazione di applicazioni ADS in ARX, ADSRX non verrà incluso nelle release future.

Capitolo 6 -- Interfacce di programmazione

ADS

ADS è un ambiente di programmazione C per lo sviluppo di applicazioni di AutoCAD divenuto oramai obsoleto. Le informazioni contenute in questa sezione vengono fornite per coloro che ancora fanno uso delle applicazioni ADS. Molte di queste applicazioni verranno ancora utilizzate con AutoCAD in futuro, tuttavia, poiché l'interfaccia API non è più supportata, è consigliabile sostituire le applicazioni ADS con applicazioni ARX.

Capitolo 6 -- Interfacce di programmazione

ADS

Uso di applicazioni ADS

Il caricamento di un'applicazione ADS è simile al caricamento di un file AutoLISP. Per caricare un'applicazione ADS dalla riga di comando di AutoCAD è necessario utilizzare la funzione AutoLISP **xload**. La sintassi della funzione **xload** è pressoché identica a quella della funzione **load** usata con i file AutoLISP. Se la funzione **xload** esegue correttamente il caricamento del programma ADS, restituisce il nome del programma. La sintassi per la funzione **xload** è:

```
(xload nomefile [se_fallisce])
```

I due argomenti per la funzione **xload** sono *nomefile* e *se_fallisce*. Come per la funzione **load**, l'argomento *nomefile* è obbligatorio e deve essere il percorso completo del file del programma ADS che si intende caricare. L'argomento *se_fallisce* è opzionale e generalmente non viene usato quando si esegue il caricamento di programmi ADS dalla riga di comando. L'esempio di riga di comando riportato di seguito esegue il caricamento dell'applicazione ADS *miaapp.exe*.

comando: (**xload "miaapp"**)

Come per i file AutoLISP, AutoCAD ricerca il file specificato nel percorso della libreria. Per caricare un file che non si trova nel percorso della libreria, è necessario fornire il percorso completo del file. Ricordarsi di usare una singola barra (/) o due barre rovesciate (\) come separatore di directory quando si specifica il nome completo del percorso.

Se si tenta di caricare un programma che è già stato caricato, verrà visualizzato il messaggio riportato di seguito, dove xxx è il nome dell'applicazione.

Applicazione "xxx" già caricata.

Le applicazioni ADS occupano memoria di sistema. Quando non è più necessario usare un'applicazione e si desidera rimuoverla dalla memoria, è possibile usare la funzione AutoLISP **xunload**. Con il seguente esempio viene scaricata l'applicazione *miaapp*.

Comando: (**xunload "miaapp"**)

Utilizzando la funzione **xunload**, dalla memoria non viene rimossa solamente l'applicazione, ma anche le definizioni dei comandi ad essa associati.

Capitolo 6 -- Interfacce di programmazione



ADS

Caricamento automatico delle applicazioni ADS

Il file *acad.ads* contiene un elenco dei file di programma ADS che vengono caricati automaticamente quando si avvia AutoCAD. È possibile modificare questo file con un editor di testo o con un programma di elaborazione testi che crea dei file in formato testo ASCII. È possibile personalizzare questo file nel modo desiderato, aggiungendo altre informazioni oppure cancellando quelle esistenti e rendendo disponibili per l'uso i programmi ADS appropriati.

Poiché AutoCAD ricerca il file *acad.ads* nell'ordine specificato dal percorso della libreria, è possibile avere un file *acad.ads* diverso in ogni directory di disegno. Ciò rende disponibili specifici programmi ADS per determinati tipi di disegni. Ad esempio, è possibile tenere disegni di modellazione tridimensionale in una directory chiamata *acadlvr/3d_dsgn*. Se questa directory è impostata come directory corrente, è possibile copiare il file *acad.ads* in quella directory e modificarlo come riportato di seguito.

```
miaappl  
diversaapp
```

Se il nuovo file *acad.ads* viene inserito nella directory *acadlvr/3d_dsgn* ed AutoCAD viene avviato con questa directory come quella corrente, questi nuovi programmi ADS verranno caricati e saranno disponibili dalla riga di comando di AutoCAD. Poiché il file *acad.ads* originale si trova ancora nella directory con i file di programma di AutoCAD, il file *acad.ads* di default verrà caricato se AutoCAD viene avviato da una directory che non contiene un file *acad.ads*.

I programmi ARX possono essere caricati da un file *.mnl* utilizzando la funzione **xload**. Ciò assicura che un programma ADS, richiesto per il corretto funzionamento di un menu, verrà caricato quando il file di menu sarà caricato.

È possibile utilizzare le funzioni di caricamento automatico utilizzate dalla maggior parte dei comandi di AutoCAD definiti in ADS. Vedere "Caricamento automatico dei comandi" e "**autoload**" nel capitolo 13.

Parte II AutoLISP

Panoramica

AutoLISP, una implementazione del linguaggio di programmazione LISP, fa parte del pacchetto di AutoCAD. Con AutoLISP è possibile scrivere programmi e funzioni macro in un linguaggio ad alto livello, adatto ad applicazioni grafiche. AutoLISP risulta flessibile e semplice da usare.

AutoLISP è la guida completa al linguaggio di programmazione AutoLISP. Come sottoinsieme di Common LISP, AutoLISP ne segue la sintassi e le convenzioni, ma contiene funzioni aggiuntive specifiche di AutoCAD. Tra i numerosi testi che trattano il linguaggio di programmazione LISP, si consiglia la lettura di LISP di Winston e Horn (seconda edizione) e Looking at LISP di Tony Hasemer, entrambi pubblicati da Addison-Wesley.

Numerosi programmi AutoLISP, inclusi molti esempi presenti in questo manuale, sono contenuti nella directory *sample* di AutoCAD. Alcuni di questi sono disponibili come prodotti shareware sviluppati da altre società. Poiché il codice AutoLISP può essere immesso alla riga di comando, l'apprendimento di questo linguaggio risulterà facile. Una volta imparato AutoLISP, sarà possibile utilizzarlo per

umentare il numero di comandi di AutoCAD.

Questa parte del manuale è dedicata all'uso delle funzioni AutoLISP ed alla scrittura di applicazioni AutoLISP. Per una panoramica generale su AutoLISP e per informazioni sul caricamento e sull'uso di applicazioni AutoLISP esistenti, vedere il capitolo 6, "Interfacce di programmazione."

Parte II AutoLISP

Panoramica

AutoLISP, una implementazione del linguaggio di programmazione LISP, fa parte del pacchetto di AutoCAD. Con AutoLISP è possibile scrivere programmi e funzioni macro in un linguaggio ad alto livello, adatto ad applicazioni grafiche. AutoLISP risulta flessibile e semplice da usare.

AutoLISP è la guida completa al linguaggio di programmazione AutoLISP. Come sottoinsieme di Common LISP, AutoLISP ne segue la sintassi e le convenzioni, ma contiene funzioni aggiuntive specifiche di AutoCAD. Tra i numerosi testi che trattano il linguaggio di programmazione LISP, si consiglia la lettura di LISP di Winston e Horn (seconda edizione) e Looking at LISP di Tony Hasemer, entrambi pubblicati da Addison-Wesley.

Numerosi programmi AutoLISP, inclusi molti esempi presenti in questo manuale, sono contenuti nella directory *sample* di AutoCAD. Alcuni di questi sono disponibili come prodotti shareware sviluppati da altre società. Poiché il codice AutoLISP può essere immesso alla riga di comando, l'apprendimento di questo linguaggio risulterà facile. Una volta imparato AutoLISP, sarà possibile utilizzarlo per aumentare il numero di comandi di AutoCAD.

Questa parte del manuale è dedicata all'uso delle funzioni AutoLISP ed alla scrittura di applicazioni AutoLISP. Per una panoramica generale su AutoLISP e per informazioni sul caricamento e sull'uso di applicazioni AutoLISP esistenti, vedere il capitolo 6, "Interfacce di programmazione."

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Panoramica

Questo capitolo introduce i concetti di base del linguaggio di programmazione AutoLISP. Poiché il codice AutoLISP non è compilato, è possibile immettere il codice sulla riga di comando e vedere subito il risultato. Gli esempi in questo capitolo sono formattati in modo da poter essere facilmente digitati sulla riga di comando così, da facilitare le operazioni per gli utenti non esperti.

Argomenti di questo capitolo

{button ,JI(','AutoLISP_Expressions_al_u0301')} Espressioni AutoLISP

{ewc ,JI(','AutoLISP_Program_Files_al_u0301')} File di programma AutoLISP

{ewc ,JI(','AutoLISP_Variables_al_u0301')} Variabili AutoLISP

{ewc ,JI(','Number_Handling_al_u0301')} Calcoli numerici

{button ,JI(','String_Handling_al_u0301')} Gestione delle stringhe

{button ,JI(','Equality_and_Conditional_al_u0301')} Uguaglianza e condizionale

{ewc ,JI(','List_Handling_al_u0301')} Gestione degli elenchi

{button ,JI(','Symbol_and_Function_Handling_al_u0301')} Gestione di simboli e funzioni

{ewc ,JI(','Error_Handling_al_u0301')} Gestione degli errori

{ewc ,JI(','Application_Handling_al_u0301')} Gestione delle applicazioni

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Espressioni AutoLISP

Quando si digita del testo alla riga di comando, AutoCAD lo interpreta confrontando i caratteri con un elenco interno di nomi di comandi validi. Se una parola digitata corrisponde ad una delle parole nell'elenco, AutoCAD valuta la definizione del comando che esegue l'operazione richiesta. Se AutoCAD riceve del testo in codice AutoLISP, lo invia all'interprete AutoLISP, alla cui base opera un programma di valutazione. Tale programma legge una riga di codice, la valuta e restituisce un risultato. Il codice deve essere strutturato come un'espressione AutoLISP e può essere letto da un file o derivare da un'immissione da parte dell'utente alla riga di comando.

Tutte le espressioni AutoLISP hanno il seguente formato:

```
(funzione argomenti)
```

Ciascuna espressione inizia con una parentesi aperta ed è costituita da un nome di funzione e da un elenco opzionale di argomenti, ciascuno dei quali può essere esso stesso un'espressione. L'espressione termina con una parentesi chiusa. Ogni espressione restituisce un valore che può essere utilizzato dall'espressione che la include. Se quest'ultima non esiste, AutoLISP restituisce il valore alla riga di comando di AutoCAD. Ad esempio, il seguente codice contiene tre funzioni.

```
(fun1 (fun2 argomenti) (fun3 argomenti))
```

La prima funzione, **fun1**, ha due argomenti, mentre le altre due funzioni, **fun2** e **fun3**, hanno un argomento ciascuna. Le funzioni **fun2** e **fun3** sono incluse nella funzione **fun1**, quindi i loro valori di ritorno vengono trasferiti come argomenti alla funzione **fun1**. Questa li valuta e restituisce il valore alla riga di comando.

Se si digita un'espressione AutoLISP sulla riga di comando di AutoCAD, AutoLISP valuta l'espressione e visualizza il risultato, quindi riappare la riga di comando. Il seguente esempio mostra l'uso della funzione * (**moltiplicazione**), che accetta come argomenti uno o più numeri reali.

Comando: (* 2 27)

54

Dato che questo esempio di codice non presenta espressioni che iniziano con una parentesi aperta oppure che includono degli appunti, il valore ritorna alla riga di comando.

Le espressioni nidificate all'interno di altre espressioni restituiscono i loro risultati alle espressioni che le includono. Nel seguente esempio il risultato della funzione + (addizione) viene utilizzato come uno degli argomenti della funzione * (moltiplicazione).

Comando: (* 2 (+ 5 10))

30

Se si digita un numero incorretto di parentesi chiuse, AutoLISP visualizza il seguente messaggio di richiesta:

n>

In questo esempio, *n* è un numero che indica quanti livelli di parentesi rimangono aperti. Se compare tale messaggio, occorre digitare un numero *n* di parentesi chiuse nell'espressione da valutare.

Comando: (* 2 (+ 5 10

2>))

30

Un errore comune è quello di omettere le virgolette (") alla fine di una stringa di testo. In tal caso, le parentesi chiuse vengono interpretate come parte della stringa e non modificano il valore di *n*. Per correggere tale errore occorre annullare la funzione premendo ESC e digitarla di nuovo correttamente.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Espressioni AutoLISP

Tipi di dati in AutoLISP

Il programma di valutazione di AutoLISP analizza le espressioni secondo l'ordine ed il tipo di dati contenuti all'interno delle parentesi. Prima di poter utilizzare completamente AutoLISP, occorre comprendere le differenze tra i tipi di dati ed il modo in cui vengono

utilizzati.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Espressioni AutoLISP

Tipi di dati in AutoLISP

Subroutine e Subroutine esterne

La maggior parte delle funzioni AutoLISP descritte nel capitolo 13, "AutoLISP: catalogo delle funzioni", sono subroutine incorporate chiamate *subr*. Le funzioni descritte nel capitolo 14, "Accesso ai comandi definiti esternamente e alle variabili di sistema", sono definite da applicazioni ADSRX ed ARX esterne chiamate *subroutine esterne*. I nomi delle subroutine e delle subroutine esterne possono essere digitati indifferentemente in caratteri maiuscoli o minuscoli.

La funzione **princ** è una subroutine. La funzione **acad_strlsort** è una subroutine esterna.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Espressioni AutoLISP

Tipi di dati in AutoLISP

Numeri interi

Per numeri interi si intendono numeri che non contengono una virgola decimale. I numeri interi di AutoLISP sono numeri con segno di 32 bit e con valori che variano da +2,147,483,648 a -2,147,483,647. Sebbene AutoLISP internamente utilizzi valori a 32 bit, da AutoLISP ad AutoCAD possono essere trasferiti solo valori a 16 -bit. Di conseguenza, non è possibile passare ad AutoCAD un valore maggiore di +32767 o minore di -32768. Quando si utilizza esplicitamente un numero intero in una espressione AutoLISP, tale valore è noto come una *costante*. Numeri come 2, -56 e 1,200,196 sono numeri interi validi in AutoLISP.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Espressioni AutoLISP

Tipi di dati in AutoLISP

Numeri reali

Per numero reale si intende un numero contenente un punto decimale. I numeri tra -1 e 1 devono comprendere uno zero iniziale. I numeri reali sono memorizzati nel formato a virgola mobile a doppia precisione, che fornisce una precisione di almeno 14 cifre significative, anche se nell'area della riga di comando di AutoCAD ne vengono visualizzate soltanto 6. I numeri reali possono anche essere espressi in notazione scientifica, cioè con una e o E opzionale seguita dall'esponente del numero (ad esempio, 0.0000041 equivale a 4.1e-6). Quando si utilizza esplicitamente un numero reale in una espressione AutoLISP, tale valore viene detto costante. Numeri come 3.1, 0.23, -56.123 e 21.000.000.0 sono numeri reali validi in AutoLISP.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Espressioni AutoLISP

Tipi di dati in AutoLISP

Stringhe

Per stringa si intende un gruppo di caratteri racchiuso tra virgolette. All'interno delle stringhe racchiuse tra virgolette, la barra rovesciata (\) consente l'inserimento di caratteri di controllo o codici di escape. Ogni singola stringa può contenere fino a 132 caratteri. Quando si utilizza esplicitamente una stringa tra virgolette in un'espressione AutoLISP, tale valore viene detto *stringa letterale* o *stringa costante*.

Esempi di stringhe valide sono "stringa 1" e "\nDigitare il primo punto:".

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Espressioni AutoLISP

Tipi di dati in AutoLISP

Elenchi

Un elenco di AutoLISP è un gruppo di valori correlati separati da spazi e racchiusi in parentesi. Gli elenchi costituiscono un metodo efficace per la memorizzazione di valori correlati. AutoCAD esprime i punti tridimensionali sotto forma di un elenco di tre numeri reali.

Esempi di elenchi sono (1.0 1.0 0.0), ("questo" "quello" "l'altro") ed (1 "UNO").

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Espressioni AutoLISP

Tipi di dati in AutoLISP

Gruppi di selezione

Per gruppi di selezione si intendono gruppi di uno o più oggetti (entità). Con le routine di AutoCAD, è possibile aggiungere o eliminare oggetti dai gruppi di selezione in modo interattivo.

Nel seguente esempio la funzione **ssget** è utilizzata per restituire un gruppo di selezione contenente tutti gli oggetti di un disegno.

Comando: (**ssget "X"**)

<Gruppo di selezione: 1>

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Espressioni AutoLISP

Tipi di dati in AutoLISP

Nomi di entità

Per nome di entità si intende un'etichetta numerica assegnata agli oggetti di un disegno. Si tratta in realtà di un puntatore mantenuto da AutoCAD in un file, grazie al quale AutoLISP può individuare il record dell'oggetto di database ed i suoi vettori (se sono visualizzati). Le funzioni AutoLISP possono fare riferimento a questa etichetta per operare la selezione degli oggetti ed analizzarli in vario modo. Al suo interno, AutoCAD fa riferimento agli oggetti come ad entità.

Nel seguente esempio la funzione `entlast` viene utilizzata per ottenere il nome dell'ultimo oggetto immesso nel disegno.

Comando: `(entlast)`
<Nome di entità: 60000016>

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

-  **Espressioni AutoLISP**
-  **Tipi di dati in AutoLISP**
-  **Descrittori di file**

I descrittori di file sono etichette alfanumeriche assegnate ai file aperti in AutoLISP. Quando una funzione AutoLISP deve accedere ad un file (a scopo di lettura o scrittura), è necessario fare riferimento alla relativa etichetta.

Nel seguente esempio il file `miainfo.dat` viene aperto per la lettura.

Comando: `(apri "miainfo.dat" "r")`
<File: #34614>

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

-  **Espressioni AutoLISP**
-  **Tipi di dati in AutoLISP**
-  **Simboli e variabili**

AutoLISP utilizza i simboli per la memorizzazione dei dati. I nomi dei simboli possono essere digitati indifferentemente in maiuscolo o minuscolo e possono essere costituiti da qualsiasi sequenza di caratteri alfanumerici e di notazione, con le seguenti eccezioni: () . ' " ; . Un nome di simbolo non può essere costituito solo da caratteri numerici.

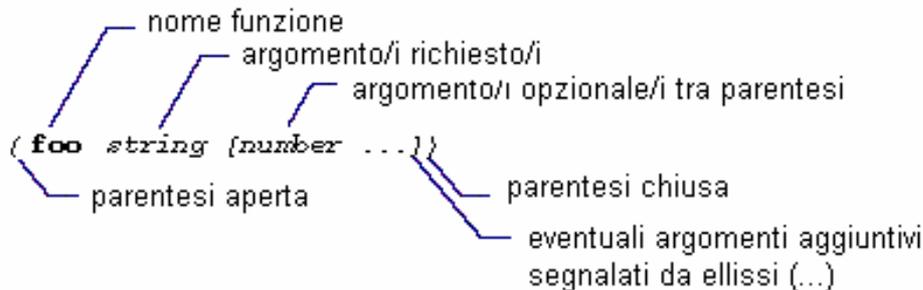
Da un punto di vista strettamente tecnico, le applicazioni AutoLISP sono costituite da simboli o da valori costanti, come stringhe, numeri reali e numeri interi. Per esigenze di chiarezza, questo manuale utilizza il termine *simbolo* per indicare il nome di un simbolo con il quale vengono memorizzati dati statici, come le funzioni incorporate o definite dall'utente. Il termine *variabile* è utilizzato per indicare il nome di un simbolo con il quale vengono memorizzati i dati di un programma. Nel seguente esempio la funzione `setq` viene utilizzata per assegnare il valore di stringa "questa è una stringa" alla variabile `str1` :

Comando: `(setq str1 "questa è una stringa")`
"questa è una stringa"

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

-  **Espressioni AutoLISP**
-  **Sintassi delle funzioni AutoLISP**

La presente pubblicazione utilizza le seguenti convenzioni per descrivere la sintassi delle funzioni AutoLISP.



Nell'esempio precedente, la funzione `foo` ha un argomento richiesto, `stringa`, e un argomento opzionale, `numero`. Possono essere forniti argomenti `numero` aggiuntivi. Spesso il nome dell'argomento indica il tipo di dati previsti. Gli esempi della tabella seguente mostrano chiamate valide e non valide per la funzione `foo`.

Richiami validi e non validi per la funzione foo

Richiami validi	Richiami non validi
(foo "catch")	(foo 44 13)
(foo "catch" 22)	(foo "foe" 44 13)
(foo "catch" 22 31)	(foo)

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

File di programma AutoLISP

Sebbene il codice AutoLISP possa essere digitato alla riga di comando, le operazioni di debug e di verifica sono molto più semplici se si carica il codice AutoLISP da un file piuttosto che digitarlo nuovamente ogni volta che si apportano perfezionamenti. Il codice AutoLISP è archiviato in file di testo ASCII con l'estensione `.lsp` o `.mnl`. Tuttavia, è possibile caricare il codice AutoLISP da qualsiasi file di testo ASCII se si fornisce alla funzione `load` il nome completo del file. La sintassi delle espressioni AutoLISP nei file corrisponde essenzialmente a quella delle espressioni digitate alla riga di comando.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

File di programma AutoLISP

Commenti

I commenti sono generalmente inclusi nei file di programma AutoLISP. Essi sono utili sia al programmatore che ai futuri utenti che desiderino eventualmente modificare il programma secondo le esigenze personali. Utilizzare i commenti per i seguenti usi:

- Attribuire il titolo, il nome dell'autore e la data di creazione.
- Fornire istruzioni sull'uso di una routine.
- Creare note esplicative all'interno di una routine.
- Creare annotazioni personali durante l'esecuzione del debug.
- Consentire l'inserimento di caratteri che migliorino l'aspetto estetico.

I commenti iniziano con un punto e virgola (;) e proseguono fino alla fine della riga.

```
; Tutta questa riga è un commento
(setq area (* pi r r)) ; Calcolare l'area del cerchio.
```

Tutto il testo racchiuso tra `;` ... `;` viene ignorato; ciò consente di inserire i commenti all'interno di una riga di codice o di prolungarli per più righe. Questo tipo di commenti è detto *interno alla riga*.

```
(tmode;|alcune note in questo punto|;(getvar "tilemode"))
```

L'esempio seguente mostra un commento che occupa più righe:

```
(setvar "orthomode" 1) ;|il commento inizia in questo punto
e continua in questa riga,
e termina qui|; (princ "\nModalità ORTHO impostata su On.")
```

Può essere molto utile fare un ampio uso dei commenti quando si scrivono i programmi AutoLISP. Alcuni dei file *.isp* forniti con AutoCAD (ma non tutti) presentano buoni esempi di stile di commento. Il file *ai_utils.isp* contiene numerosi commenti e alcune routine AutoLISP utili e interessanti. Nello sviluppare uno stile di commento, si tengano presenti i seguenti punti:

- La parte descrittiva dell'intestazione può fornire una descrizione abbastanza dettagliata sull'uso del file rendendo così superfluo il ricorso ad altro materiale di documentazione.
- Il numero dei punti e virgola che precedono un commento e la posizione del commento all'interno del file possono contribuire ad indicare il contenuto o l'importanza del commento.
- Molti monitor possono visualizzare solo 80 colonne di testo; un eventuale commento che inizi alla 81ª colonna potrebbe non venire visualizzato.

Le regole del linguaggio sono flessibili ed è molto più importante la presenza dei commenti, piuttosto che la loro conformità a particolari regole di layout.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

File di programma AutoLISP

Indentazione e allineamento

L'ampio uso di parentesi in AutoLISP può rendere il programma difficile da leggere. La tecnica tradizionale per ovviare a questa confusione è l'indentazione. Grazie ad essa il programma viene disposto graficamente in modo che le righe di codice nidificate più profondamente inizino più a destra. La profondità delle parentesi può essere così elevata che i rientri tipici standard sono di soli due spazi verso destra per ciascun livello, a differenza degli standard tipici del linguaggio C, che sono di quattro spazi per livello.

In AutoLISP, più spazi tra i nomi di variabili, tra le costanti e tra i nomi di funzioni equivalgono ad uno spazio singolo. Anche la fine di una riga viene considerata come uno spazio singolo.

Le due espressioni seguenti producono lo stesso risultato.

```
(setq test1 123 test2 456)
(setq
  test1 123
  test2 456
)
```

Nota Sebbene in AutoLISP le tabulazioni siano di solito interpretate come spazi, è consigliabile non utilizzarle. Alcune piattaforme possono interpretarle in modo diverso, portando a risultati imprevedibili.

Le regole relative all'indentazione sono più difficili da spiegare che da illustrare tramite esempi. Il miglior modo per sviluppare un buon layout di codice consiste nel leggere e imitare il codice di altri autori, che si ritiene sia di facile lettura. Inoltre, numerosi sistemi per la modifica di codice LISP comprendono dei programmi che leggono il codice LISP disposto in modo arbitrario e lo riformattano, inserendovi le indentazioni adeguate.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Variabili AutoLISP

Le variabili AutoLISP assumono il tipo di dati proprio del valore ad esse assegnato. Le variabili mantengono i loro valori originali fino a che non ne vengono assegnati loro di nuovi. Per assegnare dei valori alle variabili viene utilizzata la funzione AutoLISP **setq**.

```
(setq nome_variabile1 valore1 [nome_variabile2 valore2 ...])
```

La funzione **setq** assegna alla variabile di cui è stato indicato il nome il valore specificato. Essa restituisce il valore come risultato della funzione. Se si utilizza la funzione **setq** alla riga di comando, imposta la variabile e visualizza il valore.

Comando: **(setq val 3 abc 3.875)**

3.875

Comando: **(setq layr "MURI-ESTERNI")**

"MURI-ESTERNI"

Comando:

Ciascuna variabile occupa un certo spazio in memoria, quindi è buona norma di programmazione riutilizzare i nomi di variabili o impostare il valore nil per quelle variabili i cui valori non sono più necessari. Impostando il valore nil per una variabile, si rilascia la memoria occupata dal valore della variabile stessa. Se non si ha più bisogno della variabile val, è possibile rilasciarla dalla memoria mediante la seguente espressione:

Comando: **(setq val nil)**

nil

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Variabili AutoLISP

Uso delle variabili alla riga di comando

Dopo aver assegnato un valore ad una variabile, è possibile utilizzarla in risposta ai messaggi di richiesta di AutoCAD che vengono visualizzati sulla riga di comando. Ciò costituisce un metodo per la memorizzazione ed il riutilizzo di nomi complessi e di numeri. Se si desidera utilizzare il valore di una variabile come risposta ad una richiesta di AutoCAD, digitare il nome della variabile preceduto da un punto esclamativo (!). Ad esempio, per utilizzare il valore della variabile abc in risposta alla richiesta di un valore in numeri reali, digitare **!abc**.

Distanza di colonne: **!abc**

Se il proprio disegno ha un layer contrassegnato dal nome MURI-ESTERNI e si vuole utilizzare il comando LAYER per renderlo corrente, è possibile digitare il nome della variabile invece del nome completo del layer.

Comando: **layer**

?/Def/coRrente/Nuovo/ON/OFF/Colore/Tipolinea/conGela/SCongela/BLocca/SBlocca: **r**

Nuovo layer corrente <0>: **!layr**

?/Def/coRrente/Nuovo/ON/OFF/Colore/Tipolinea/conGela/SCongela/BLocca/SBlocca:

Comando:

Inoltre, è possibile rispondere ad un messaggio di richiesta di AutoCAD con un'espressione AutoLISP. Se si sa già che si utilizzerà lo stesso valore per una serie di richieste, è possibile digitare la seguente espressione:

Angolo: **(setq ang 47.338)**

Questa espressione effettua due operazioni: assegna il valore 47.338 alla variabile ang e restituisce tale valore alla richiesta Angolo. Quando si desidererà utilizzare nuovamente quel valore, sarà sufficiente digitare **!ang**.

Dato che le stringhe di testo ordinario possono iniziare con il carattere ! o (, per poter utilizzare una variabile o un'espressione per fornire la stringa di testo ai comandi TESTO e DEFATT, è necessario impostare il valore 1 per la variabile di sistema TEX-TEVAL. Non è possibile utilizzare le variabili o le espressioni per fornire le stringhe di testo al comando TESTODIN o nelle finestre di dialogo. Non è possibile inoltre usare una variabile per inviare un comando di AutoCAD. Ad esempio, se si imposta per la stringa "linea" la variabile x e poi si digita **!x** in risposta alla richiesta comando di AutoCAD, viene visualizzato il valore "linea"; il comando LINEA non viene eseguito. La funzione **command** esegue i comandi di AutoCAD dall'interno delle espressioni AutoLISP. Per informazioni, vedere "**command**" nel capitolo 13.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Variabili AutoLISP

Variabili predefinite

AutoCAD fornisce tre variabili predefinite che possono essere utilizzate nelle applicazioni AutoLISP.

PAUSE

È definita come una stringa composta da un'unica barra rovesciata (\). Tale variabile è utilizzata con la funzione **command** per permettere l'inserimento di input da parte dell'utente.

PI

È definita come la costante π (p greco). Ha un valore approssimativo di 3.1415926.

T

È definita come la costante T. È utilizzata come un valore diverso da zero.

Nota È possibile modificare il valore di queste variabili con la funzione **setq**. Dato tuttavia che in altre applicazioni l'uniformità dei valori può essere un fattore fondamentale, si raccomanda di non modificare le variabili.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Calcoli numerici

AutoLISP dispone di funzioni per l'esecuzione di operazioni con i numeri interi e con i numeri reali. Le funzioni per i calcoli numerici sono le seguenti:

+ (addizione)	abs	gcd	minusp
- (sottrazione)	atan	log	rem
* (moltiplicazione)	cos	logand	sin
/ (divisione)	exp	logior	sqrt
~ (NOT a livello bit)	expt	lsh	zerop
1+ (incremento)	fix	max	
1- (decremento)	float	min	

Le funzioni per la gestione delle funzioni sono descritte nel capitolo 13, "AutoLISP: catalogo delle funzioni".

Oltre all'esecuzione di calcoli matematici complessi nelle applicazioni, le funzioni aritmetiche possono facilitare l'uso ordinario di AutoCAD. Se si disegna un dettaglio di un elemento di connessione in acciaio, che utilizza un bullone di 2.5", con un diametro di 0.5" e con 13 fili per pollice, quanti fili ci saranno in tutto? Alla riga di comando utilizzare la funzione * (moltiplicazione).

Comando: (*** 2.5 13**)

32.5

Le funzioni aritmetiche che hanno come argomento *numero*, invece di , , ad esempio, *num* o *angolo*, restituiscono valori diversi se si indicano numeri interi o numeri reali come argomenti. Se tutti gli argomenti sono numeri interi, il valore restituito è anch'esso un numero intero. Tuttavia, se uno o tutti gli argomenti sono numeri reali, il valore restituito è un numero reale. Per fare in modo che la propria applicazione trasferisca numeri reali, occorre assicurarsi che almeno uno degli argomenti sia un numero reale.

Comando: (**/ 12 5**)

2

Comando: (**/ 12.0 5**)

2.4

Sebbene AutoLISP utilizzi al proprio interno valori a 32 bit, da AutoLISP ad AutoCAD possono essere passati solo valori a 16 bit. Di conseguenza non è possibile passare ad AutoCAD valori maggiori di +32767 o minori di -32768. Se è necessario utilizzare un valore che supera tali limiti, è possibile utilizzare la funzione **float** per convertire il valore in numero reale, dato che tali numeri sono trasferiti come valori a 32 bit.

Comando: (**setq 32bit (float (* 3 30000))**)

90000.0

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione delle stringhe

AutoLISP dispone di funzioni per la gestione dei valori di stringa. Di seguito sono riportate le funzioni di gestione delle stringhe.

strcase	strlen	wcmatch
strcat	substr	

Le seguenti funzioni convertono i valori di stringa in valori numerici e i valori numerici in valori di stringa. Tali funzioni sono illustrate nella sezione "Conversioni."

angtof	atof	distof
angtos	atoi	itoa
ascii	chr	rtos

Le seguenti funzioni visualizzano i valori di stringa alla riga di comando. Tali funzioni sono illustrate nella sezione "Controllo della visualizzazione."

prin1	print
princ	prompt

La funzione **strcase** restituisce la conversione di tutti i caratteri alfanumerici di una stringa in maiuscolo o minuscolo. Essa accetta due argomenti: una stringa e un argomento opzionale che specifica se i caratteri debbano essere restituiti in maiuscolo o in minuscolo. Se questo argomento opzionale viene omissso, il suo valore viene considerato uguale a nil e la funzione **strcase** restituisce i caratteri convertiti in maiuscolo.

Comando: **(strcase "Questa è una VERIFICA.")**

"QUESTA È UNA VERIFICA."

Se viene specificato un secondo argomento con un valore diverso da nil, i caratteri vengono restituiti in minuscolo. AutoLISP fornisce la variabile predefinita T da utilizzare in situazioni come questa, in cui un valore diverso da nil viene utilizzato come un interruttore di tipo vero/falso. Tuttavia, non è necessario utilizzare la variabile T in questa occasione; è possibile utilizzare qualsiasi valore diverso da -nil, se esso si adatta meglio alla propria applicazione.

Comando: **(strcase "Questa è una VERIFICA.") T)**

"questa è una verifica"

La funzione **strcat** combina più stringhe in un'unica stringa. Ciò è utile per posizionare una stringa variabile all'interno di una stringa costante. Il seguente esempio di codice imposta per una variabile il valore di una stringa e utilizza quindi la funzione **strcat** per inserire tale stringa all'interno di un'altra stringa.

Comando: **(setq str "PRIMA") (setq bigstr (strcat "Questa è una" str "verifica.))**

"Questa è una PRIMA verifica."

Il valore di una singola stringa è limitato a 132 caratteri. È possibile creare stringhe di lunghezza illimitata utilizzando la funzione **strcat** per combinare più stringhe insieme.

Se per la variabile bigstr è impostato il valore della stringa precedente, è possibile utilizzare la funzione **strlen** per ottenere il numero di caratteri, compresi gli spazi, che compongono tale stringa.

Comando: **(strlen bigstr)**

19

La funzione **substr** restituisce la stringa secondaria di una stringa. Essa ha due argomenti richiesti e un argomento opzionale. Il primo argomento è dato dalla stringa, il secondo argomento è un numero intero che specifica il primo carattere della stringa che si vuole inserire nella stringa secondaria. Se viene fornito il terzo argomento, esso specifica il numero dei caratteri da includere nella stringa secondaria. Se il terzo argomento non viene fornito, **substr** restituisce il carattere iniziale e tutti i caratteri che lo seguono.

È possibile utilizzare la funzione **substr** per privare il nome di un file dei tre caratteri di estensione. Per prima cosa, occorre impostare in una variabile il nome di un file. Secondo il tipo di applicazione, ciò verrà effettuato richiedendo una variabile di sistema o l'immissione di input da parte dell'utente.

Comando: **(setq nomfile "filegran.txt")**

"filegran.txt"

Occorre quindi ottenere una stringa che contenga tutti i caratteri ad eccezione degli ultimi quattro (il punto e i tre caratteri di estensione). Utilizzare la funzione **strlen** per ottenere la lunghezza della stringa e sottrarre quattro da tale valore. Quindi utilizzare la funzione **substr** per specificare il primo carattere della stringa secondaria e la sua lunghezza.

Comando: **(setq newlen (- (strlen nomfile) 4))**

7

Comando: **(substr nomfile 1 newlen)**

"filegran"

Se la propria applicazione non necessita del valore di newlen, è possibile combinare queste due righe di codice in una.

Comando: **(substr nomfile 1 (- (strlen nomfile) 4))**

"filegran"

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione delle stringhe

Caratteri di controllo nelle stringhe

All'interno delle stringhe racchiuse tra virgolette, la barra rovesciata (<:df

Caratteri di controllo

Codice	<i>Descrizione</i>
\\	<i>Carattere \</i>
\"	<i>Carattere "</i>
\e	<i>Carattere Escape</i>
\n	<i>Carattere nuova riga</i>
\r	<i>Carattere di ritorno a capo</i>
\t	<i>Carattere di tabulazione</i>
\nnn	<i>Carattere con codice ottale nnn</i>
\U+xxxx	<i>Sequenza di caratteri Unicode</i>
\M+nxxxx	<i>Sequenza di caratteri a byte multiplo</i>

La funzione **prompt** espande i caratteri di controllo e li visualizza alla riga di comando. La funzione **prompt** accetta come suo unico argomento una stringa, la stampa alla riga di comando e restituisce il valore nil. Tutti i caratteri di controllo presenti nella stringa sono espansi. Per informazioni, vedere "**prompt**" nel capitolo 13.

Quando è necessario utilizzare la barra rovesciata (<:df

Comando: **(prompt "Il \"nomefile\" è: D:\ACAD\TEST.TXT. ")**

Il "nomefile" è: D:\ACAD\TEST.TXT. nil

Il carattere di controllo (\e) può essere utilizzato per inviare sequenze al sistema operativo al fine di modificare i grafici di visualizzazione, controllare i movimenti del cursore e ridefinire i tasti. Vedere la documentazione relativa al sistema operativo per informazioni sulle sequenze escape riconosciute. Un uso diffuso delle sequenze escape riguarda la ridefinizione dei tasti della tastiera sui sistemi DOS. Il seguente codice utilizza la funzione **prompt** per assegnare la stringa **qsave** RETURN al tasto F12.

Comando: **(prompt "\e[0;134;qsave';13p")**

nil

La stringa dell'esempio precedente è composta essenzialmente da quattro parti. La parte **\e[** segnala a DOS che ciò che segue è una sequenza escape, la parte **0;134;** è il codice che specifica il tasto F12. Quando il tasto F12 viene premuto, la parte 'qsave'; è la **stringa che viene trasferita alla riga di comando e 13p** specifica che dopo la stringa viene passato RETURN. È possibile utilizzare qualsiasi stringa di comando valida al posto di **qsave**, purché la stringa trasferita alla riga di comando sia racchiusa tra apici e sia seguita da un punto e virgola.

Per inserire una interruzione di riga in un punto specifico di una stringa, utilizzare il carattere nuova riga (\n).

Comando: **(prompt "Questo è un esempio del \ncarattere nuova riga. ")**

Questo è un esempio del

carattere nuova riga. nil

Il carattere di ritorno (\r) torna all'inizio della riga corrente. Ciò è utile per visualizzare l'informazione incrementale (come ad esempio il numero degli oggetti analizzati).

Quando AutoLISP valuta un'espressione, restituisce il valore dell'ultima espressione. Dato che la funzione **prompt** restituisce il valore nil, negli esempi precedenti tale valore segue le stringhe espansi. Anche la funzione **princ** stampa una stringa alla riga di

comando; tuttavia, invece del valore nil, essa restituisce l'espressione (in questo caso una stringa). La funzione **princ** differisce dalla funzione **prompt** anche per il fatto che non richiede alcun argomento (i suoi argomenti sono opzionali). Di conseguenza, se un'espressione in AutoLISP termina con una chiamata alla funzione **princ** senza la specificazione di alcun argomento, il nil finale viene omesso in quanto non include alcun valore da restituire. Tale concetto è noto come *uscita tranquilla*.

Il carattere di tabulazione (\t) può essere utilizzato nelle stringhe per effettuare indentazioni o allineamenti con altre stringhe di testo per le quali è stata utilizzata la tabulazione. Si noti nel seguente esempio, l'uso della funzione **princ** per eliminare il valore nil finale.

Comando: (prompt "\nNome\tUfficio\n- - - -\t- - - -")

1> \nCarla\t101\nGianni\t102\nSamuele\t103\n") (princ)

Nome	Ufficio
-----	-----
Carla	101
Gianni	102
Samuele	103

Per inserire caratteri diversi da quelli della tastiera standard, utilizzare i caratteri di controllo \nnn, in cui nnn è il codice ottale del carattere che si vuole visualizzare. Vedere l'Appendice A, "Codici ASCII", per un elenco dei caratteri ASCII standard il cui codice decimale va da 0 a 127, con il corrispondente codice decimale, ottale ed esadecimale. I codici associati con i caratteri estesi (con codici decimali da 128 a 256) dipendono dal gruppo di caratteri installato. I codici estesi variano per le diverse piattaforme e possono variare anche per le applicazioni e le periferiche di una stessa piattaforma. La sezione "Conversione di codici ASCII" fornisce un'applicazione di esempio che visualizza tutti i caratteri del sistema nella finestra di testo di AutoCAD e li memorizza su un file.

Se la propria finestra di disegno utilizza il font Courier, il seguente codice visualizza il simbolo " ± ":

Comando: (prompt "Il valore è \261 7001. ") (princ)

Il valore è ± 7001.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione delle stringhe

Corrispondenza con caratteri jolly

La funzione **wcmatch** consente alle applicazioni di confrontare una stringa con un modello di caratteri jolly. Questa funzione può essere utilizzata insieme alla funzione **sset** per creare un gruppo di selezione e insieme alla funzione **entget** per richiamare i dati di entità estesi in base al nome dell'applicazione.

La funzione **wcmatch** confronta un'unica stringa con un modello. Essa restituisce T se la stringa corrisponde al modello e nil se invece la stringa non corrisponde al modello. I modelli di caratteri jolly sono simili alle normali espressioni utilizzate da molti sistemi e applicazioni. Nel modello, i caratteri alfabetici e numerici sono interpretati in modo letterale; le parentesi possono essere utilizzate per specificare caratteri opzionali o una serie di caratteri o di cifre; il punto interrogativo (?) corrisponde ad un singolo carattere; l'asterisco (*) corrisponde ad una serie di caratteri; altri caratteri speciali hanno significati particolari all'interno del modello. Se si utilizza il carattere * all'inizio o alla fine del modello di ricerca, è possibile individuare la parte desiderata in qualsiasi punto della stringa. Per ulteriori informazioni, vedere "**wcmatch**" nel capitolo 13.

Negli esempi seguenti, una variabile di stringa contrassegnata dal nome corrispondenza è stata dichiarata e inizializzata:

Comando: (setq corrispondenza "questa è una stringa - testo1 testo2 fine")

"questa è una stringa - testo1 testo2 fine"

Il codice seguente controlla che la variabile corrispondenza inizi con i cinque caratteri "questa":

Comando: (wcmatch corrispondenza "questa*")

T

L'esempio seguente illustra l'uso delle parentesi quadrate nel modello. In questo caso, **wcmatch** restituisce T se la variabile corrispondenza contiene "verifica4", "verifica5", "verifica6" o "verifica9" (notare l'uso del carattere *):

Comando: (wcmatch corrispondenza "*verifica[4-69]*")

nil

Tuttavia,

Comando: (wcmatch corrispondenza "*verifica[4-61]*")

T

perché la stringa contiene "verifica1".

La stringa modello può specificare più modelli, separati da virgole. Il codice seguente restituisce T se la variabile corrispondenza è uguale a "ABC", se *inizia* con "XYZ" oppure *termina* con "fine".

Comando: (**wcmatch corrispondenza "ABC,XYZ",*fine"**)

T

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Uguaglianza e condizionale

AutoLISP comprende funzioni che eseguono verifiche di eguaglianza, così come il raggruppamento condizionale e l'iterazione. Di seguito sono riportate le funzioni di uguaglianza e le funzioni condizionali.

= (uguale a)	and	oppure
/= (diverso da)	Boole	repeat
< (minore di)	cond	while
<= (minore di o uguale a)	eq	
> (maggiore di)	equal	
>= (maggiore di o uguale a)	if	

Tutte le funzioni condizionali e di uguaglianza sono descritte nel capitolo 13, "AutoLISP: catalogo delle funzioni".

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione degli elenchi

AutoLISP dispone di funzioni per la gestione degli elenchi, elencate qui di seguito:

append	foreach	listp	reverse
assoc	last	mapcar	subst
car e cdr	length	member	
cons	list	nth	

Questa sezione fornisce esempi delle funzioni **append**, **assoc**, **car**, **cons**, **list**, **nth** e **subst**. Ogni funzione per la gestione degli elenchi è descritta nel capitolo 13, "AutoLISP: catalogo delle funzioni".

Gli elenchi costituiscono un metodo efficace per la memorizzazione di valori correlati. Numerose funzioni AutoLISP forniscono gli elementi di base per la programmazione di applicazioni di grafica bidimensionale e tridimensionale. Tali funzioni restituiscono valori punto sotto forma di elenco.

La funzione **list** fornisce un metodo semplice per raggruppare gli elementi collegati, che non necessitano di tipi di dati simili. Il seguente codice raggruppa tre elementi correlati come elenco.

Comando: (**setq lst1 (list 1.0 "Uno" 1)**)

(1.0 "Uno" 1)

È possibile richiamare un elemento specifico da un elenco con la funzione **nth**. La funzione **nth** accetta due argomenti. Il primo è un numero intero che specifica quale elemento restituire. Uno 0 specifica il primo elemento in un elenco, 1 specifica il secondo elemento e così via. Il secondo argomento è rappresentato dall'elenco stesso. Il codice seguente restituisce il secondo elemento nell'elenco lst1.

Comando: (**nth 1 lst1**)

"Uno"

Le funzioni **car** e **cdr** forniscono un altro modo per estrarre elementi da un elenco. Per degli esempi sull'uso di **car** e **cdr**, vedere "Elenchi punto."

Esistono tre funzioni per modificare un elenco esistente. La funzione **append** restituisce un elenco con nuovi elementi aggiunti alla fine dell'elenco, mentre la funzione **cons** aggiunge i nuovi elementi all'inizio dell'elenco. La funzione **subst** restituisce un elenco con un nuovo elemento sostituito per ogni vecchio elemento. Queste funzioni non modificano l'elenco originale, bensì restituiscono un

elenco modificato. Per modificare un elenco originale è necessario sostituire espressamente il vecchio elenco con il nuovo.

La funzione **append** esegue qualsiasi numero di elenchi come un unico elenco. Di conseguenza tutti gli argomenti di questa funzione devono essere elenchi. Il seguente codice aggiunge un altro "Uno" all'elenco lst1. Notare l'uso della funzione **quote** (o ') **che consente di trasformare facilmente la stringa** "Uno" in un elenco.

Comando: **(setq lst2 (append lst1 `("Uno")))**
 (1.0 "Uno" 1 "Uno")

La funzione **cons** combina un singolo elemento con un elenco. È possibile aggiungere un'altra stringa "Uno" all'inizio del nuovo elenco, lst2, con la funzione **cons**.

Comando: **(setq lst3 (cons "Uno" lst2))**
 ("Uno" 1.0 "Uno" 1 "Uno")

È possibile sostituire tutte le occorrenze di un elemento in un elenco con un nuovo elemento con la funzione **subst**. Il seguente codice sostituisce tutte le stringhe "Uno" con la stringa "uno".

Comando: **(setq lst4 (subst "uno" "Uno" lst3))**
 ("uno" 1.0 "uno" 1 "uno")

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione degli elenchi

Elenchi punto

AutoLISP segue le seguenti convenzioni per la gestione di coordinate grafiche. I punti sono espressi come *elenchi* di due o tre numeri racchiusi tra parentesi tonde.

punti bidimensionali

espressi come elenchi di due numeri reali (X e Y rispettivamente), come
 (3.4 7.52)

punti tridimensionali

espressi come elenchi di tre numeri reali (XY e Z rispettivamente), come
 (3.4 7.52 1.0)

Le variabili punto contengono le componenti X, Y e (in opzione) Z. Il primo elemento dell'elenco è la componente X del punto, il secondo è la componente Y e il terzo, se presente, è la componente Z. È possibile utilizzare la funzione incorporata **list** per creare tali elenchi.

Comando: **(list 3.875 1.23)**
 (3.875 1.23)

Comando: **(list 88.0 14.77 3.14)**
 (88.0 14.77 3.14)

Per assegnare coordinate particolari ad una variabile punto, è possibile utilizzare una delle seguenti espressioni:

Comando: **(setq pt1 (list 3.875 1.23))**
 1> **pt2 (list 88.0 14.77 3.14)**
 1> **abc 3.45**
 1> **pt3 (list abc 1.23)**
 (3.45 1.23)

L'ultima espressione utilizza il valore della variabile abc come componente X del punto.

Se tutti i numeri di un elenco sono valori costanti, è possibile utilizzare la funzione **quote** invece della funzione **list** per definire esplicitamente l'elenco. La funzione **quote** restituisce un'espressione non valutata.

Comando: **(setq pt1 (quote (4.5 7.5)))**

(4.7 7.5)

L'apice (') può essere usato come abbreviazione della funzione quote. Il seguente codice produce gli stessi risultati del codice precedente.

Comando: (setq pt1 '(4.5 7.5))

(4.7 7.5)

È possibile fare riferimento alle singole componenti X, Y e Z di un punto, utilizzando tre funzioni incorporate aggiuntive, **car**, **cadr** e **caddr**. I seguenti esempi mostrano come estrarre le coordinate X, Y e Z da un elenco relativo ad un punto tridimensionale. Per la variabile pt è impostato il punto (1.5 3.2 2.0).

Comando: (setq pt '(1.5 3.2 2.0))

(1.5 3.2 2.0)

La funzione **car** restituisce il primo elemento di un elenco. In questo caso imposta per la variabile x_val il valore X del punto pt.

Comando: (setq x_val (car pt))

1.5

La funzione **cadr** restituisce il secondo elemento di un elenco. In questo caso imposta per la variabile y_val il valore Y del punto pt.

Comando: (setq y_val (cadr pt))

3.2

La funzione **caddr** restituisce il terzo elemento di un elenco. In questo caso imposta per la variabile z_val il valore Z del punto pt.

Comando: (setq z_val (caddr pt))

2.0

È possibile utilizzare il seguente codice per definire gli angoli inferiore sinistro e superiore (pt1 e pt2) di un rettangolo.

Comando: (setq pt1 '(1.0 2.0) pt2 ' (3.0 4.0))

(3.0 4.0)

È possibile utilizzare le funzioni **car** e **cadr** per impostare per la variabile pt3 il valore dell'angolo superiore sinistro del rettangolo, estraendo la componente X di pt1 e la componente Y di pt2.

Comando: (setq pt3 (list (car pt1) (cadr pt2)))

(1.0 4.0)

L'espressione precedente imposta per pt3 il punto (1.0,4.0).

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione degli elenchi

Coppie puntate

Un altro modo in cui AutoCAD usa gli elenchi per organizzare i dati è con un tipo speciale di elenco detto *coppie puntate*. Una coppia puntata è un elenco che deve comunque contenere due elementi. Quando viene visualizzata una coppia puntata, AutoLISP stampa un punto tra gli elementi dell'elenco. La maggior parte delle funzioni di gestione di elenchi non accettano coppie puntate come argomenti, quindi è necessario assicurarsi di passare il corretto tipo di elenco alla funzione.

Oltre ad aggiungere un elemento all'inizio di un elenco, la funzione **cons** può creare una coppia puntata. Se il secondo argomento alla funzione **cons** è un atomo (una costante o un valore tra apici), viene creata una coppia puntata.

Comando: (setq d1 (cons 1 "Uno"))

(1 . "Uno")

Le funzioni **car**, **cdr** e **assoc** sono utili per la gestione di coppie puntate. Il codice seguente crea un *elenco di associazione*. Un elenco di associazione è un elenco di elenchi ed è il metodo usato da AutoCAD per gestire i dati di definizione di entità (vedere il capitolo 9, "Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli"). Il seguente codice crea un elenco di associazione di coppie puntate.

Comando: (setq d2 (list d1 (cons 2 "Due")(cons 3 "Tre")))

((1 . "Uno") (2 . "Due") (3 . "Tre"))

La funzione **assoc** restituisce un elenco specifico dall'interno di un elenco di associazione, indipendentemente dalla posizione specifica dell'elenco di associazione. La funzione **assoc** ricerca un elemento chiave specifico negli elenchi.

Comando: (**assoc 2 d2**)

(2 . "Due")

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione di simboli e funzioni

AutoLISP dispone di funzioni per la gestione di simboli e delle variabili. Di seguito sono riportate le funzioni di gestione dei simboli.

atom	not	quote	type
atoms-family	null	set	
boundp	numberp	setq	

Ogni funzione per la gestione dei simboli è descritta nel capitolo 13, "AutoLISP: catalogo delle funzioni".

AutoLISP fornisce funzioni per la gestione di uno o più gruppi di funzioni. Di seguito sono riportate le funzioni per la gestione delle funzioni.

apply	eval	progn	untrace
defun	lambda	trace	

Questa sezione fornisce esempi sulla funzione **defun**. Ogni funzione per la gestione delle funzioni è descritta nel capitolo 13, "AutoLISP: catalogo delle funzioni".

Con AutoLISP è possibile definire le proprie funzioni. Una volta definite, è possibile utilizzarle alla riga di comando o all'interno di altre espressioni AutoLISP nello stesso modo in cui si usano le funzioni standard. È inoltre possibile creare i propri comandi AutoCAD, in quanto essi sono semplicemente tipi speciali di funzioni.

La funzione **defun** (vedere "**defun**" in capitolo 13, "AutoLISP: catalogo delle funzioni") combina un gruppo di espressioni in una funzione o comando. Tale funzione richiede almeno tre argomenti, il primo dei quali è il nome della funzione (il nome del simbolo), da definire. Il secondo argomento è l'elenco di argomenti e simboli locali usati dalla funzione. L'elenco degli argomenti può essere nil o un elenco vuoto (). Gli elenchi argomento sono discussi dettagliatamente in "Funzioni con argomenti." Se vengono forniti simboli locali, essi sono separati dagli argomenti da una barra (/). I simboli locali sono discussi in "Simboli locali nelle funzioni." Dopo questi argomenti sono le espressioni che costruiscono la funzione; deve esserci almeno un'espressione in una definizione di funzione.

```
(defun nome_simbolo ( args / simbolo locale )
  espressioni
)
```

Il codice seguente definisce una funzione che non accetta argomenti e visualizza un messaggio sulla riga di comando. L'elenco argomento viene passato come elenco vuoto (()).

Comando: (**defun DONE () (prompt "Inciaio! ") ")**
DONE

Una volta definita la funzione **DONE**, è possibile usarla come qualsiasi altra funzione. Poiché essa stampa il messaggio ciao! su una nuova riga di comando, è possibile usare la funzione **DONE** nel modo seguente:

Comando: (**prompt "Il valore è 127. ")(DONE)(princ)**
 Il valore è 127
 ciao!

Le funzioni che non accettano argomenti potrebbero sembrare inutili. Tuttavia, esse possono essere usate per sapere lo stato di certe variabili di sistema o condizioni e restituire un valore che indica quei valori.

Nota Per evitare conflitti con funzioni non correlate, considerare il prefisso del nome della funzione **S::** "reserved". Usare questo prefisso solo per la funzione speciale **S::STARTUP**. Vedere "Funzione S::STARTUP: esecuzione automatica."

AutoCAD può automaticamente caricare le funzioni ogni volta che viene avviata una nuova sessione. (vedere "Caricamento ed esecuzione automatici").

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione di simboli e funzioni

C:FunzioniXXX

Se una funzione viene definita con un nome avente il formato **C:XXX**, può essere inviata alla riga di comando allo stesso modo di un comando incorporato di AutoCAD. Questa funzione può essere usata per aggiungere nuovi comandi in AutoCAD o per ridefinire comandi esistenti.

Per poter utilizzare le funzioni come comandi di AutoCAD, accertarsi che siano conformi alle seguenti regole:

- Il nome della funzione deve utilizzare il formato **C:XXX** in caratteri maiuscoli o minuscoli. La parte del nome costituita da **C:** deve sempre essere presente; la parte **XXX** può essere un nome di comando di propria scelta. Le funzioni **C:XXX** possono essere utilizzate per sostituire i comandi incorporati di AutoCAD (vedere "Ridefinizione dei comandi di AutoCAD").

Nota In questo caso, **C:** non si riferisce ad un'unità disco. È un prefisso speciale che denota una funzione della riga di comando.

- La funzione deve essere definita senza argomenti, ma sono consentiti simboli locali.

Una funzione definita in questo modo può essere inviata in modo trasparente da qualsiasi messaggio di richiesta di qualsiasi comando incorporato di AutoCAD, sempre che tale funzione non richiami la funzione **command**. Quando si invia un comando definito **C:XXX** in modo trasparente, è necessario che la parte **XXX** da una virgoletta semplice (**'**).

È possibile inviare in modo trasparente un comando incorporato mentre è attivo un comando **C:XXX**, digitando prima del nome una virgoletta semplice (**'**), come si farebbe con tutti i comandi inviati in modo trasparente. Tuttavia, non è possibile inviare un comando **C:XXX** in modo trasparente mentre è attivato un altro comando **C:XXX**. Quando viene chiamata una funzione definita come un comando dal codice di un'altra funzione AutoLISP, occorre utilizzare il nome completo, ad esempio (**C:CIAO**).

Se la definizione della funzione viene seguita da una chiamata alla funzione **setfunhelp** è possibile associare un file di Guida ed un argomento a un comando definito dall'utente. Quando un utente richiama la Guida in modo trasparente (**'help**) ad un messaggio di richiesta viene visualizzato l'argomento specificato con **setfun-help**.

Generalmente non è possibile utilizzare un'istruzione AutoLISP per rispondere ai messaggi di richiesta di un comando AutoLISP implementato. Tuttavia, se la routine AutoLISP utilizza la funzione **initget**, con determinate funzioni è possibile immettere un input arbitrario da tastiera. Ciò consente ad un comando AutoLISP implementato di accettare come risposta un'istruzione AutoLISP. Inoltre, i valori restituiti da un'espressione DIESEL possono eseguire alcune elaborazioni del disegno corrente e restituire i relativi valori ad AutoLISP. Per informazioni sul linguaggio DIESEL, vedere il capitolo 5, "Linguaggio DIESEL e configurazione della riga di stato"..

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione di simboli e funzioni

C:FunzioniXXX

Aggiunta di comandi

Usando il formato **C:XXX**, è possibile definire un comando che visualizzi un messaggio semplice.

Comando: **(defun C:CIAO () (prompt "Ciao a tutti. \n") (princ))**

C:CIAO

CIAO è ora definito come comando.

Comando: **ciao**

Ciao a tutti.

Questo nuovo comando può essere eseguito in modo trasparente perché non richiama la funzione **command**.

Comando: **linea**

Dal punto: **'ciao**

Ciao a tutti.

Dal punto:

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione di simboli e funzioni

C:FunzioniXXX

Ridefinizione dei comandi di AutoCAD

Tramite AutoLISP, i comandi esterni e gli alias è possibile definire comandi di AutoCAD personalizzati. È possibile utilizzare il comando NUOVDEF per ridefinire un comando incorporato di AutoCAD con un comando definito dall'utente avente lo stesso nome. Per ripristinare la definizione incorporata di un comando, utilizzare il comando RIDEF. Il comando NUOVDEF è valido solo per la sessione di modifica corrente.

È sempre possibile attivare un comando la cui definizione è stata modificata, specificando il suo vero nome, cioè il nome del comando preceduto da un punto. Ad esempio, se viene modificata la definizione di ESCI, è sempre possibile accedere al comando digitando **.quit**.

Considerare il seguente esempio: l'utente desidera che AutoCAD gli ricordi di utilizzare il comando PLINEA ogni volta che utilizza il comando LINEA. È possibile definire la funzione AutoLISP **C:LINEA** per sostituire il comando LINEA normale nel seguente modo:

```
Comando: (defun C:LINEA ()
1> (princ "non dovresti usare PLINEA?\n")
1> (command ".LINEA") (princ) )
C:LINEA
```

In questo esempio, la funzione **C:LINEA** è impostata in modo da inviare il relativo messaggio e quindi eseguire il normale comando LINEA, utilizzando il suo vero nome LINEA. Prima che AutoCAD utilizzi la nuova definizione del comando LINEA, è necessario modificare la definizione del comando incorporato LINEA. Vedere "Procedura per la ricerca dei comandi" per informazioni sull'ordine in cui AutoCAD riconosce i nomi dei comandi.

```
Comando: nuovodef
Nome comando: linea
```

A questo punto, se si digita **linea** alla riga di comando, AutoCAD utilizza la funzione AutoLISP **C:LINEA**:

```
Comando: linea
Non dovresti usare PLINEA?
.LINEA Dal punto:
Dal punto:
```

L'esempio precedente presume che la variabile di sistema CMDECHO sia impostata a 1 (On). Se CMDECHO è impostata a 0 (Off) AutoCAD non visualizza i messaggi di richiesta risultanti da una chiamata alla funzione **command**. Il codice seguente usa la variabile di sistema CMDECHO per controllare la visualizzazione dei messaggi di comando.

```
Comando: (defun C:LINEA ()
1> (setvar "cmdecho" 0)
1> (princ "non dovresti usare PLINEA?\n")
1> (command ".LINEA") (setvar "cmdecho" 1) (princ) )
C:LINEA
```

Questa funzione potrebbe essere impiegata, ad esempio, in un sistema di gestione di disegno. È possibile ridefinire i comandi NUOVO, APRI, ESCI e FINE per scrivere informazioni su un file di log prima di concludere la sessione di modifica.

Nota È consigliabile proteggere i menu, gli script e i programmi AutoLISP utilizzando le forme precedute dal punto per tutti i comandi. Ciò assicura che le applicazioni utilizzino le definizioni di comandi assunte anziché un comando ridefinito.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione di simboli e funzioni

Simboli locali nelle funzioni

AutoLISP dispone di un metodo per definire un elenco di simboli (variabili) disponibili solo per le funzioni definite dall'utente. Essi vengono detti *simboli locali*. L'uso di tali simboli assicura che le variabili nelle proprie funzioni non siano influenzate dalle altre applicazioni e non rimangano disponibili dopo che la funzione di richiamo ha eseguito l'operazione.

Molte funzioni definite dall'utente vengono utilizzate come funzioni di utilità all'interno di applicazioni più estese. Le funzioni definite dall'utente spesso contengono una serie di variabili i cui valori e usi sono specifici di quella funzione.

L'esempio seguente mostra l'uso di simboli locali in una funzione definita dall'utente (assicurarsi che ci sia almeno uno spazio tra la barra e i simboli locali).

```
Comando: (defun LOCAL (/ aaa bbb)
1> (setq aaa "A" bbb "B")
1> (princ (strcat "\naaa ha il valore " aaa ))
1> (princ (strcat "\nbbb ha il valore " bbb))
1> (princ )
LOCAL
```

Prima di verificare la nuova funzione, assegnare alle variabili aaa e bbb valori diversi da quelli della funzione **LOCAL**.

```
Comando: (setq aaa 1 bbb 2)
2
```

È possibile verificare che per le variabili aaa e bbb siano effettivamente impostati i valori desiderati utilizzando il punto esclamativo (!).

```
Comando: !aaa
1
Comando: !bbb
2
```

A questo punto, verificare la funzione **LOCAL**.

```
Comando: (local)
aaa ha il valore A
bbb ha il valore B
```

Notare che la funzione ha utilizzato per le variabili aaa e bbb i valori locali della funzione stessa. Anche il punto esclamativo (!) può essere utilizzato per verificare se per le variabili aaa e bbb siano ancora impostati i valori non locali.

```
Comando: !aaa
1
Comando: !bbb
2
```

Oltre a garantire che le variabili siano locali per una funzione specifica, tale tecnica assicura che la memoria utilizzata per quelle variabili sia disponibile per altre funzioni.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione di simboli e funzioni

Funzioni con argomenti

Con AutoLISP è possibile definire funzioni che accettano argomenti. A differenza di molte funzioni standard AutoLISP, le funzioni definite dall'utente non possono avere argomenti opzionali. Di conseguenza, quando viene utilizzata una funzione definita dall'utente che accetta argomenti, è necessario fornire i valori per tutti gli argomenti.

I simboli da utilizzare come argomenti vengono inizializzati nell'elenco argomenti prima dei simboli locali. Gli argomenti sono considerati come tipi di particolari di simboli locali: anche le variabili argomento infatti non sono disponibili al di fuori della funzione. Non è possibile definire una funzione con più argomenti con lo stesso nome.

Il seguente codice definisce una funzione che accetta due argomenti di stringa, li combina con un'altra stringa e restituisce la stringa risultante.

```
Comando: (defun ARGS ( arg1 arg2 / ccc )
1> (setq ccc "Stringa costante")
1> (strcat ccc " , " arg1 " , " arg2 )
ARGTEST
```

Questo tipo di funzione può essere utilizzata più volte nell'ambito di un'applicazione per combinare due stringhe variabili con una stringa costante, in un ordine specifico. Dato che essa restituisce un valore, è possibile salvare tale valore in una variabile per i successivi usi all'interno dell'applicazione.

```
Comando: (setq newstr (ARGTEST "Stringa 1" "Stringa 2")
"Stringa costante, Stringa 1, Stringa 2")
```

Per la variabile newstr è ora impostato il valore delle tre stringhe combinate.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione degli errori

AutoLISP dispone di funzioni per la gestione degli errori. Di seguito sono riportate le funzioni di gestione degli errori.

alert *error* exit quit

Ogni funzione per la gestione degli errori è descritta nel capitolo 13, "AutoLISP: catalogo delle funzioni".

AutoLISP dispone di un metodo per la gestione degli errori dell'utente o degli errori di programma. La funzione ***error*** assicura che AutoCAD torni ad uno stato particolare nel caso in cui si verifichi un errore. Mediante questa funzione definibile dall'utente è possibile individuare la condizione di errore e restituire un messaggio adeguato all'utente.

Se AutoLISP riscontra un errore nel corso dell'elaborazione, visualizza un messaggio nel seguente formato:

Errore: testo

In tale messaggio, *testo* descrive l'errore. Se la funzione ***error*** è definita, ovvero non restituisce il valore nil, invece di visualizzare il messaggio di errore. AutoLISP la esegue, passando come suo unico argomento *testo*. Se la funzione ***error*** non è definita o è destinata a restituire come valore nil, l'elaborazione di AutoLISP si arresta e viene visualizzato il codice in cui è contenuta la funzione che genera l'errore e le funzioni che la richiamano, fino a 100 livelli. È consigliabile che tali errori siano mantenuti durante le operazioni di debugging del programma.

Il codice relativo all'ultimo errore viene memorizzato nella variabile di sistema ERRNO di AutoCAD, da cui è possibile recuperarlo utilizzando la funzione **getvar**. Vedere anche i messaggi di errore e i codici di errore nel capitolo 16, "Codici e messaggi di errore AutoLISP". Vedere ****error**** nel capitolo 13, "AutoLISP: catalogo delle funzioni".

Prima di definire una funzione ***error*** personalizzata, salvare il contenuto corrente di ***error*** in modo che il gestore di errore precedente possa essere ripristinato quando si esce. Se si verifica una condizione di errore, AutoCAD chiama la funzione ***error*** definita correntemente e le passa un argomento costituito da una stringa di testo che descrive la natura dell'errore. La funzione ***error*** personalizzata dovrebbe essere definita in modo da uscire senza produrre alcun effetto nel caso in cui venga premuto ESC (annulla) o venga chiamata una funzione **exit**. La procedura standard per ottenere ciò consiste nell'includere le seguenti istruzioni nella routine personalizzata di gestione degli errori.

```
(if
  (or
    (/= msg "Funzione cancellata")
    (= msg "esci./.continua")
  )
  (princ)
  (princ (strcat "\nErrore: " msg))
)
```

Questo codice esamina il messaggio di errore che gli viene passato e assicura che l'utente sia informato della natura dell'errore. Se l'utente annulla la routine mentre essa è in esecuzione, questa non restituirà alcun messaggio. Allo stesso modo, se si verifica una

condizione di errore nella programmazione del codice e viene chiamata la funzione **exit**, non viene restituito alcun messaggio. Si presuppone che la natura dell'errore sia già stata spiegata utilizzando le istruzioni di visualizzazione. Occorre includere una chiamata finale alla funzione **princ**, se non si desidera ottenere un valore di ritorno alla fine di una routine di gestione degli errori.

Se si effettuano chiamate di tipo UNDO per annullare una routine in una volta sola, è necessario fornire chiamate UNDO di equilibratura che verrebbero effettuate se l'uscita della routine avvenisse in modo normale. Si potrebbe anche desiderare che AutoCAD annulli tutto. I programmi possono utilizzare le variabili di sistema UNDOCTL e UNDOMARKS per determinare come e se AutoCAD debba effettuare chiamate di tipo UNDO.

L'inconveniente principale delle routine di gestione dell'errore è legato al fatto che si tratta di normali funzioni AutoLISP, che possono essere annullate dall'utente. È consigliabile quindi che siano il più possibile brevi e veloci. In tal modo sarà più probabile che vengano eseguite per intero una volta chiamate.

Molti comandi ARX dispongono di proprie funzioni o variabili di sistema che forniscono informazioni sugli errori, come descritto nel capitolo 14, "Accesso ai comandi definiti esternamente e alle variabili di sistema".

È inoltre possibile avvisare l'utente sulle condizioni di errore visualizzando una finestra di avviso, contenente un messaggio fornito dal programma. Per visualizzare tale finestra, richiamare la funzione **alert**.

La seguente chiamata alla funzione **alert** visualizza una finestra di avviso.

```
(alert "File non trovato")
```

Per ulteriori informazioni, vedere "**alert**" nel capitolo 13.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione delle applicazioni

AutoLISP dispone di funzioni per la gestione delle applicazioni e delle funzioni definite esternamente. Di seguito sono riportate le funzioni di gestione dell'applicazione.

ads	arxunload	autoload	xload
arx	autoarxload	load	xunload
arxload	autoload	startapp	

Questa sezione fornisce esempi delle funzioni **load**, **arx** e **arxload**. Vedere "Caricamento automatico dei comandi" per esempi della funzione **autoload**. Tutte le funzioni per la gestione delle applicazioni sono descritte nel capitolo 13, "AutoLISP: catalogo delle funzioni".

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione delle applicazioni

Caricamento di applicazioni AutoLISP

È possibile salvare le definizioni delle funzioni in file con estensione *.lsp* e caricarli utilizzando la funzione AutoLISP **load** oppure includerle in un file *acad.lsp* o *.mnl* che viene caricato all'avviamento di AutoCAD. Il caricamento di un file *.lsp* comporta l'elaborazione delle sue espressioni. Generalmente, nei file *.lsp* viene utilizzata la funzione **defun** per memorizzare gruppi di funzioni nella memoria del computer, per successive esecuzioni.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione delle applicazioni

Caricamento di applicazioni AutoLISP

Funzione S::STARTUP: esecuzione automatica

Se la funzione **S::STARTUP** definita dall'utente è contenuta nel file *acad.lsp* o in un file *.mnl*, essa viene richiamata quando si immette un nuovo disegno o se ne apre uno già esistente. In questo modo, è possibile includere una funzione **defun** di **S::STARTUP** nel proprio file *acad.lsp* per eseguire tutte le operazioni di configurazione.

Se si desidera ignorare i comandi standard di AutoCAD ESCI e FINE e sostituirli con versioni personalizzate, utilizzare un file *acad.lsp* che contenga quanto segue:

```
(defun C:ESCI ( )
  ... propria definizione...
)
(defun C:FINE ( )
  ... propria definizione...
)
(defun S::STARTUP ( )
  (command "nuovdef" "esci")
  (command "nuovdef" "fine")
)
```

Prima che il disegno venga inizializzato, mediante la funzione **defun**, vengono impostate nuove definizioni per i comandi ESCI e FINE. Una volta che l'inizializzazione del disegno è stata completata, viene richiamata la funzione **S::STARTUP** e ai comandi ESCI e FINE vengono assegnate nuove definizioni in sostituzione di quelle standard.

Dato che la funzione **S::STARTUP** può essere definita in molti ambiti diversi (in un file *acad.lsp* file, in un file *.mnl* o in qualsiasi altro file AutoLISP caricato da uno di questi ultimi), è possibile sovrascrivere una funzione **S::STARTUP** precedentemente definita. L'esempio seguente mostra un metodo per garantire che la funzione di avviamento personalizzata operi con altre funzioni.

```
(defun MYSTARTUP ( )
  ... funzione di avviamento-personalizzata ...
)
(setq S::STARTUP (append S::STARTUP MYSTARTUP))
```

Il codice precedente unisce la funzione di avvio e quella di una funzione **S::STARTUP** esistente, e la ridefinisce in modo da includere il codice di avvio. Tale codice funziona indipendentemente dall'esistenza di una funzione **S::STARTUP**.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione delle applicazioni

Caricamento di applicazioni AutoLISP

Tecniche di caricamento

I programmi AutoLISP vengono generalmente forniti in file *.lsp* da caricare mediante la funzione **load**. Oltre a creare funzioni che l'utente può richiamare, la funzione **load** consente al programmatore di fornire istruzioni all'utente sulle modalità di utilizzo del programma. Dato che AutoLISP elabora le espressioni nel file, se viene richiamata la funzione **princ**, **print**, **prin1** o simili, è possibile visualizzare tutte le informazioni ad esse relative alla riga di comando. È anche possibile eliminare ciò che AutoLISP normalmente visualizzerebbe in modo che l'ultima riga del file sia (**princ**).

Il caricamento di un file fornisce altre opportunità per richiamare le funzioni AutoLISP. Qualsiasi codice in un file *.lsp* che non fa parte di un'istruzione **defun** verrà eseguito al momento del caricamento del file. Oltre alla visualizzazione di informazioni descritte precedentemente, tale funzione può essere utilizzata per impostare determinati parametri o per eseguire qualsiasi altro tipo di procedura di inizializzazione.

Capitolo 7 -- Nozioni fondamentali sull'uso di AutoLISP

Gestione delle applicazioni

Caricamento di applicazioni ADS e ARX

Prima di richiamare un comando o una funzione ADS o ARX definiti da un'applicazione AutoLISP, è necessario verificare che il comando o la funzione siano disponibili. Le funzioni **ads** e **arx** restituiscono un elenco delle applicazioni ADS e ARX attualmente caricate.

Il seguente esempio di codice definisce la funzione **INLIST** che accetta due argomenti. Il primo argomento, *lst*, è un elenco di stringhe mentre il secondo, *str*, è una stringa. La funzione **INLIST** restituisce T se *str* è la sottostringa di stringhe in *lst*. È possibile includere questa definizione di funzione nell'applicazione per facilitare la verifica di un'applicazione ADS o ARX eventualmente caricata.

```
(defun INLIST ( lst str / ct tmpstr ret )
  (setq ct 0 str (strcase str))
  (repeat (length lst)
    (setq tmpstr (strcase (nth ct lst)))
    (if (wcmatch tmpstr (strcat "*" str "*"))
      (setq ret T)
    )
    (setq ct (1+ ct))
  )
  ret
)
```

Se la funzione **INLIST** viene definita, il seguente codice controlla se l'applicazione Render ARX è caricata. Se Render non è caricato, il codice carica il file *render.arx*. Il codice di esempio fornisce inoltre un argomento *se_fallisce* (simile a quello usato dalla funzione **load** appena descritta) per notificare l'utente se non viene trovato il file *render.arx*. Questo codice usa l'elenco di stringhe generato dalla funzione **ads** (percorso e nomi file) come argomento *lst*. La stringa passata come argomento *str* riconosce le minuscole e maiuscole, e deve corrispondere a quella usata dalla piattaforma. È possibile controllare la corrispondenza di maiuscole e minuscole della piattaforma immettendo la funzione **arx** o **ads** alla riga di comando.

```
(if (not (INLIST (arx) "RENDER") )
  (setq err (arxload "render" nil))
)
(if (not err) (princ "\nErrore: RENDER.ARX non caricato."))
```

Capitolo 8 -- Funzioni di utilità generali

Panoramica

In AutoLISP sono disponibili varie funzioni per esaminare il contenuto del disegno attualmente caricato. Questo capitolo contiene informazioni generali relative a tali funzioni ed al loro uso insieme ad altre funzioni.

Argomenti di questo capitolo

```
{button ,JI(','Query_and_Command_Functions_al_u0302')} Funzioni di richiesta e di comando
{button ,JI(','Display_Control_al_u0302')} Controllo della visualizzazione
{button ,JI(','Getting_User_Input_al_u0302')} Input dell'utente
{ewc ,JI(','Geometric_Uilities_al_u0302')} Utilità geometriche
{button ,JI(','Conversions_al_u0302')} Conversioni
{button ,JI(','File_Handling_al_u0302')} Gestione di file
{button ,JI(','Device_Access_and_Control_al_u0302')} Controllo ed accesso a dispositivi
{button ,JI(','ASE_AutoLISP_Interface_al_u0302')} Interfaccia ASE AutoLISP
```

Capitolo 8 -- Funzioni di utilità generali

Funzioni di richiesta e di comando

Le funzioni di richiesta e di comando descritte in questa sezione consentono di accedere direttamente ai comandi di AutoCAD ed alle utilità per il disegno. L'azione eseguita da tali funzioni dipende dallo stato corrente delle variabili d'ambiente e di sistema di AutoCAD e dal disegno attualmente caricato. Le funzioni di richiesta e di comando sono le seguenti:

acad_colordlg	getcfg	getvar	setvar
command	getenv	setcfg	ver

In questa sezione sono riportati alcuni esempi relativi alle funzioni **command**, **getcfg**, **getvar**, **setcfg** e **setvar**. Le funzioni **help**, **getenv** e **ver** sono descritte nel capitolo 13, "AutoLISP: catalogo delle funzioni". La funzione **help** consente un accesso migliore alla guida di AutoCAD.

Capitolo 8 -- Funzioni di utilità generali

Funzioni di richiesta e di comando

Sottomissione di comandi

La funzione **command** invia un comando di AutoCAD direttamente alla relativa riga di comando. Tale funzione dispone di un elenco di argomenti di lunghezza variabile. Questi argomenti devono corrispondere ai tipi e valori previsti dalla sequenza di richieste del comando, quali stringhe, valori reali, numeri interi, punti, nomi di entità o nomi di gruppi di selezione. Dati quali angoli, distanze e punti possono essere passati come stringhe o valori (numeri interi, valori reali o elenchi di punti). Una stringa vuota ("") equivale a digitare uno spazio o premere RETURN sulla tastiera.

Esistono alcune restrizioni relative ai comandi che possono essere richiamati con la funzione **command**. Per informazioni, vedere "**command**" nel capitolo 13.

La parte di codice di seguito riportata mostra esempi di chiamate alla funzione **command**.

```
(command "cerchio" "0,0" "3,3")
(command "altezza" 1)
(setq p1 '(1.0 1.0 3.0))
(setq rad 4.5)
(command "cerchio" p1 rad)
```

Se, quando vengono richiamate queste funzioni, AutoCAD si trova in corrispondenza della riga di comando, effettua le azioni riportate di seguito.

- 1 La prima chiamata alla funzione **command** passa i punti al comando CERCHIO come stringhe. Viene disegnato un cerchio con il centro in corrispondenza di (0.0,0.0) e passante per (3.0,3.0).
- 2 La seconda chiamata passa un numero intero al comando ALTEZZA. L'altezza corrente viene modificata in (1.0).
- 3 L'ultima chiamata utilizza un punto tridimensionale ed un valore reale (a virgola mobile), che vengono memorizzati come variabili e passati per riferimento al comando CERCHIO. Viene disegnato un altro cerchio [estruso] con centro in corrispondenza di (1.0,1.0,3.0) e con un raggio pari a 4.5.

Capitolo 8 -- Funzioni di utilità generali

Funzioni di richiesta e di comando

Sottomissione di comandi

Supporto per altre lingue

Se si sviluppano programmi AutoLISP che potrebbero essere usati con una versione di AutoCAD in un'altra lingua, i comandi e le parole chiave standard di AutoCAD verranno tradotti automaticamente se si fa precedere ogni comando o parola chiave da un trattino di sottolineatura (_).

```
(command "_line" pt1 pt2 pt3 "_c")
```

Se si utilizza il prefisso costituito da un punto, per evitare di utilizzare comandi ridefiniti, è possibile collocare il punto ed il carattere di sottolineatura in qualsiasi ordine; sia "_line" che ".line" sono validi.

Capitolo 8 -- Funzioni di utilità generali

Funzioni di richiesta e di comando

Sottomissione di comandi

Pausa per input dell'utente

Se è in esecuzione un comando di AutoCAD e come argomento di **command** viene rilevato il simbolo predefinito PAUSE, l'esecuzione del comando viene sospesa per consentire l'input diretto dell'utente (in genere, selezione di punti o trascinamento). Questa operazione è simile all'uso della barra rovesciata per la sospensione nei menu.

Se si invia un comando trasparente mentre una funzione **command** è sospesa, tale funzione rimane sospesa. In questo modo, gli utenti durante la sospensione possono utilizzare i comandi 'ZOOM e 'PAN. La funzione rimane sospesa fino a quando AutoCAD non riceve un input valido e nessun comando trasparente è in esecuzione. Ad esempio, il codice di seguito indicato avvia il comando CERCHIO, imposta il centro in corrispondenza di (5,5) e si interrompe per consentire all'utente di trascinare il raggio del cerchio sullo schermo. Quando l'utente specifica il punto desiderato (o digita il raggio desiderato), la funzione riprende disegnando una linea da (5,5) a (7,5).

```
(command "cerchio" "5,5" pause "linea" "5,5" "7,5" "")
```

L'input di menu non viene sospeso da una pausa AutoLISP. Se una voce di menu è attiva quando la funzione **command** si interrompe per l'input, tale richiesta di input può essere eseguita dal menu. Se si desidera che anche la voce di menu venga sospesa, specificare una barra rovesciata (\) in tale voce. Quando viene individuato un input valido, sia la funzione **command** che la voce di menu vengono riprese.

Capitolo 8 -- Funzioni di utilità generali

Funzioni di richiesta e di comando

Sottomissione di comandi

Trasmissione di punti di selezione a comandi di AutoCAD

Alcuni comandi di AutoCAD (quali TAGLIA, ESTENDI e RACCORDO) richiedono all'utente di specificare un *punto di selezione* oltre all'oggetto stesso. Per passare queste coppie di dati punto ed oggetto mediante la funzione **command** senza utilizzare PAUSE, è necessario prima memorizzarle come variabili. I punti possono essere passati come stringhe all'interno della funzione **command** o possono essere definiti al di fuori di essa e passati come variabili, come indicato nel seguente esempio. Questa parte di codice mostra un metodo per passare un nome di entità ed un punto di selezione alla funzione **command**.

```
(command "cerchio" "5,5" "2")      Disegna un cerchio
(command "linea" "3,5" "7,5")     Disegna una linea
(setq el (entlast))              Richiama l'ultimo nome entità
(setq pt '(5 7))                 Imposta il punto pt
```

```
(command "trim" el "" pt "")
```

Esegue l'operazione di taglio

Se, quando vengono richiamate queste funzioni, AutoCAD si trova in corrispondenza della riga di comando, effettua le azioni riportate di seguito.

- 1 Disegna un cerchio con il centro in corrispondenza di (5,5) ed un raggio pari a 2.
- 2 Disegna una linea da (3,5) a (7,5).
- 3 Crea una variabile el che è il nome dell'ultimo oggetto aggiunto al database. Per ulteriori informazioni sugli oggetti e sulle funzioni per la gestione di tali oggetti, vedere il *capitolo 9*, "Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli".
- 4 Crea una variabile pt che è un punto sul cerchio. Questo punto seleziona la parte di cerchio che si desidera tagliare.
- 5 Esegue il comando TAGLIA selezionando l'oggetto el ed il punto specificato da pt.

Capitolo 8 -- Funzioni di utilità generali

Funzioni di richiesta e di comando

Variabili di sistema e d'ambiente

Le funzioni **getvar** e **setvar** consentono alle applicazioni AutoLISP di controllare e modificare il valore delle variabili di sistema di AutoCAD. Queste funzioni utilizzano una stringa per specificare il nome della variabile. La funzione **setvar** specifica un tipo di valore previsto dalla variabile di sistema. Le variabili di sistema di AutoCAD sono di vari tipi: numeri interi, valori reali, stringhe, punti bidimensionali e punti tridimensionali. I valori forniti come argomenti per la funzione **setvar** devono essere del tipo previsto. Specificando un tipo non valido, viene generato un errore AutoLISP. Per un elenco delle variabili di sistema e dei relativi tipi, vedere "Variabili di sistema" nella *Guida di riferimento dei comandi di AutoCAD*.

La parte di codice di seguito riportata assicura che i successivi comandi RACCORDO utilizzino un raggio pari almeno ad 1:

```
(if (< (getvar "filletrad") 1)
  (setvar "filletrad" 1)
)
```

La funzione **getenv** consente alle routine AutoLISP di accedere alle variabili d'ambiente del sistema operativo attualmente definite.

Capitolo 8 -- Funzioni di utilità generali

Funzioni di richiesta e di comando

Controllo della configurazione

AutoCAD utilizza il file *acad14.cfg* per memorizzare le informazioni relative alla configurazione. La sezione App-Data di questo file è destinata ad utenti e sviluppatori per la memorizzazione di informazioni di configurazione relative alle loro applicazioni. Le funzioni **getcfg** e **setcfg** consentono alle applicazioni AutoLISP di controllare e modificare il valore dei parametri presenti nella sezione AppData.

Capitolo 8 -- Funzioni di utilità generali

Controllo della visualizzazione

AutoLISP prevede anche delle funzioni per il controllo della visualizzazione AutoCAD, compresa la visualizzazione di finestre di disegno e di testo. Alcune funzioni di questo tipo richiedono l'input da parte dell'utente o dipendono da esso. Le funzioni per il controllo della visualizzazione sono le seguenti:

graphscr	menucmd	print	textpage
grdraw	gruppomenu	prompt	textscr
grtext	prin1	redraw	vports

grvecs princ terpri

Le funzioni **terpri** e **vports** sono descritte nel capitolo 13.

Capitolo 8 -- Funzioni di utilità generali

Controllo della visualizzazione

Visualizzazione di messaggi sulla riga di comando

Le funzioni di output di base sono **prompt**, **princ**, **prin1** e **print**. La funzione **prompt** visualizza un messaggio (una stringa) sulla riga di comando di AutoCAD e restituisce nil. Le funzioni **princ**, **prin1** e **print** visualizzano un'espressione (non necessariamente una stringa) in corrispondenza della riga del messaggio di richiesta, restituiscono l'espressione ed opzionalmente inviano l'output in un file. Le differenze sono le seguenti: **princ** visualizza le stringhe non incluse tra virgolette; **prin1** e **print** visualizzano le stringhe racchiuse tra virgolette; **print** posiziona una riga vuota prima dell'espressione ed uno spazio dopo di essa.

Le funzioni per la gestione dei file di seguito riportate possono inoltre visualizzare l'output nell'area della riga di comando. Per ulteriori informazioni su queste funzioni, vedere "Gestione di file."

read-char read-line write-char write-line

La lunghezza di una stringa visualizzata dalla funzione **prompt** non deve superare la lunghezza della riga del messaggio di richiesta della finestra di disegno, che in genere è costituita da un massimo di 80 caratteri.

L'esempio seguente mostra le differenze tra le quattro funzioni di output di base ed il modo in cui gestiscono la stessa stringa di testo. AutoLISP fornisce numerosi caratteri di controllo da utilizzare nelle stringhe; per un elenco completo vedere "Caratteri di controllo nelle stringhe."

```
(setq str "La \"tolleranza\" possibile è ± 1/4")
(prompt str) stampa La "tolleranza" possibile è ± 1/4"
e restituisce nil
(princ str) stampa La "tolleranza" possibile è ± 1/4"
e restituisce "La "tolleranza" possibile è ± 1/4""
(prin1 str) stampa La "tolleranza" possibile è ± 1/4"
e restituisce "La "tolleranza" possibile è ± 1/4""
(print str) stampa <riga vuota>
"La "tolleranza" possibile è ± 1/4"" <spazio>
e restituisce "La "tolleranza" possibile è ± 1/4""
```

Capitolo 8 -- Funzioni di utilità generali

Controllo della visualizzazione

Controllo dei menu

La funzione **menucmd** controlla la visualizzazione dei menu delle finestre di disegno. Inoltre visualizza, modifica o richiede uno dei sottomenu del menu corrente ed accetta un argomento di stringa che specifica il sottomenu e l'azione da eseguire su quel sottomenu.

La funzione **menucmd** utilizza un argomento di stringa costituito da due campi separati da un segno di uguale nel modo seguente:

```
"area_menu=azione"
```

Con questa sintassi è possibile caricare un sottomenu in una specifica area di menu oppure eseguire un'azione su una voce di menu o su un'area di menu attualmente caricata. Il campo *area_menu* indica quale parte del menu deve ricevere l'azione. In questo campo può essere specificata un'area di menu, come P0 (per il menu a cursore) o S (per il menu di schermo) oppure una specifica voce di menu. Nel campo *azione* viene indicata l'azione da eseguire sull'area di menu o la voce di menu oppure un sottomenu da caricare nell'area di memoria. Le aree di menu che possono ricevere un'azione sono le stesse utilizzate nei riferimenti del file di menu.

In ogni area di menu è presente il sottomenu attualmente caricato. Per default, il primo sottomenu successivo all'etichetta di una sezione di menu viene caricato in quell'area di menu.

Se *area_menu* indica un menu a discesa o un menu a gruppo di immagini, *azione* può essere un asterisco (*). Ciò provoca la visualizzazione del menu; quando sono richiamati, i menu a discesa ed i menu di gruppi di immagini non vengono visualizzati automaticamente. Su Windows, solo il menu P0 (cursore) ed i menu di gruppo di immagine vengono visualizzati con l'asterisco. Per ulteriori informazioni relative ai file di menu, vedere il capitolo 4, "Menu personalizzati".

Nota *Non* includere nell'argomento della stringa il segno del dollaro che introduce le istruzioni simili in un file di menu. Inoltre, *non* includere nel campo *azione* dell'argomento di stringa gli asterischi che precedono le etichette dei sottomenu nel file di menu.

La chiamata della funzione **menucmd** di seguito indicata consente la visualizzazione del sottomenu di schermo **OSNAP definito nel file di menu corrente, presupponendo che il menu di schermo sia attivo.

```
(menucmd "S=OSNAP")
```

In Windows, è possibile fare riferimento al gruppo di menu. Ciò può essere utile nel caso in cui siano stati caricati più menu che contengono lo stesso sottomenu. Con il codice riportato di seguito viene visualizzato il sottomenu di schermo **OSNAP nel gruppo di menu ACAD.

```
(menucmd "S=ACAD.OSNAP")
```

La funzione **menucmd** può eseguire il caricamento dei sottomenu nelle aree di menu BUTTONS ed AUX. Probabilmente, i pulsanti del digitalizzatore avranno funzioni diverse a seconda che la modalità Tavola sia attiva o disattiva. Nella sezione ***BUTTONS1 è possibile definire due sottomenu, **DIGBUTTONS- e **TABBUTTONS-, e passare dall'uno all'altro con il codice riportato di seguito.

```
(menucmd "B1=DIG-BUTTONS") Attiva il sottomenu DIG-BUTTONS
```

```
(menucmd "B1= TAB-BUTTONS") Attiva il sottomenu TAB-BUTTONS
```

Con il codice riportato di seguito, il menu ***POP0 viene caricato e visualizzato nell'area di menu P0 (cursore).

```
(menucmd "P0=POP0") Carica il menu ***POP0 nell'area di menu P0
```

```
(menucmd "P0=*") Visualizza questo menu
```

Se si è certi che in una particolare area di menu è stato caricato il menu corretto, non è necessario caricarlo specificatamente ogni volta che si desidera visualizzarlo.

La seguente chiamata visualizza il menu a discesa attualmente caricato nella posizione P1, del primo menu a discesa.

```
(menucmd "P1=*")
```

Se si utilizza "P1=" senza caricare prima il menu, è possibile che si verifichino risultati non previsti. Sebbene sia possibile caricare virtualmente qualsiasi menu in una posizione di menu a cursore o menu a discesa, si consiglia di utilizzare solo menu specificatamente destinati a tale area di menu. Ad esempio, se si dispone di un sottomenu chiamato **MORESTUFF, è possibile caricarlo nella posizione P1 utilizzando il seguente codice:

```
(menucmd "P1=MORESTUFF") Carica il menu **MORESTUFF nella  
posizione P1
```

```
(menucmd "P1=*") Visualizza questo menu
```

Tale menu rimane in questa posizione fino a quando non viene sostituito dal caricamento di un altro menu, come nel seguente esempio:

```
(menucmd "P1=POP1")
```

Se il proprio menu utilizza le funzioni per disattivare (visualizzazione in grigio) e contrassegnare, è possibile ripristinare e modificare lo stato di un'etichetta di menu con la funzione **menucmd**. La chiamata seguente ripristina lo stato corrente della quarta etichetta nel menu a discesa P2.

```
(menucmd "P2.4=#?") Se disattivato restituisce "P2.4=~"
```

Queste chiamate di funzione attivano e disattivano la stessa etichetta:

```
(menucmd "P2.4=") Attiva l'etichetta
```

```
(menucmd "P2.4=~") Disattiva l'etichetta
```

Inoltre, è possibile posizionare e rimuovere i contrassegni che si trovano sul lato sinistro delle etichette di menu.

Il metodo di gestione delle voci di menu descritto precedentemente funziona relativamente bene con un solo menu statico. Tuttavia, diventa inaffidabile quando le posizioni delle voci di menu vengono modificate durante il caricamento di più file di menu parziali. È possibile utilizzare le funzioni del gruppo di menu e contrassegno del nome per tenere traccia delle voci di menu. Invece di specificare una voce di menu in base alla sua posizione nel file di menu, è possibile specificare il gruppo di menu ed il contrassegno di nome associati a quella voce di menu.

Quando il gruppo di menu viene utilizzato per attivare, disattivare e contrassegnare etichette di menu, è necessario fare precedere il nome del gruppo da una G, come mostrato nell'esempio riportato di seguito.

```
(menucmd "Gacad.ID_Nuovo=~") Disattiva l'etichetta
(menucmd "Gacad.ID_Nuovo=") Attiva l'etichetta
```

Una funzione AutoLISP oltre ad attivare e disattivare le etichette di menu, può anche modificare il testo visualizzato nell'etichetta posizionando un'espressione di stringa DIESEL nell'etichetta stessa. Poiché DIESEL accetta come input solo stringhe, è possibile passare le informazioni all'espressione DIESEL mediante una variabile di sistema USERS1-5 impostata su un valore restituito dalla funzione.

La funzione **menucmd** può essere utilizzata anche per calcolare le espressioni di stringhe DIESEL all'interno di una funzione AutoLISP. La routine di seguito riportata restituisce l'ora corrente:

```
(defun C:CTIME ( / ctim)
  (setq ctim
    (menucmd "M=$(edtime, $(getvar, date), H:MMam/pm) "))
  (princ (strcat "\nL'ora corrente è " ctim))
  (princ)
)
```

Per informazioni sull'uso delle espressioni DIESEL con AutoLISP ed un elenco delle funzioni DIESEL, vedere il capitolo 5, "Linguaggio DIESEL e configurazione della riga di stato"..

Capitolo 8 -- Funzioni di utilità generali

Controllo della visualizzazione

Controllo delle finestre di disegno e di testo

È possibile controllare la visualizzazione di finestre di disegno e di testo da un'applicazione AutoLISP. Nelle installazioni di AutoCAD a singolo schermo una chiamata alla funzione **graphscr** visualizza la finestra di disegno, mentre una chiamata alla funzione **textscr** visualizza la finestra di testo. L'uso di queste funzioni equivale ad attivare e disattivare il tasto funzione Grafico/Testo. La funzione **textpage** è simile alla funzione **textscr**, con la sola differenza che cancella le informazioni presenti nella finestra di testo prima di visualizzarla, in modo analogo ai comandi STATO e LISTA di AutoCAD.

La funzione **redraw** è simile al comando RIDIS di AutoCAD, ma consente un maggiore controllo sulle informazioni visualizzate. Tale funzione non solo ridisegna l'intera area di disegno ma può anche specificare di ridisegnare un singolo oggetto o di annullare il disegno, cancellando tutti i dati presenti sullo schermo. Se si tratta di un oggetto complesso come una polilinea o un blocco, la funzione **redraw** può disegnare o annullare il disegno dell'intero oggetto o della relativa intestazione. Tale funzione può anche evidenziare o annullare l'evidenziazione degli oggetti specificati.

Capitolo 8 -- Funzioni di utilità generali

Controllo della visualizzazione

Controllo della grafica a basso livello

In AutoLISP sono disponibili delle funzioni che controllano la grafica a basso livello e che consentono l'accesso diretto allo schermo di disegno ed ai dispositivi di input di AutoCAD.

La funzione **grtext** visualizza il testo direttamente nelle aree di stato o di menu con o senza evidenziazione. La funzione **grdraw** disegna un vettore nella finestra corrente, controllando il colore e l'evidenziazione. La funzione **grvecs** disegna più vettori.

Nota Poiché queste funzioni dipendono dal codice in AutoCAD, l'azione che esse eseguono può essere diversa a seconda della release. Non esiste alcuna garanzia che le applicazioni che richiamano queste funzioni siano compatibili con release successive. Inoltre, tali funzioni dipendono dalla configurazione hardware corrente. In particolare, le applicazioni che richiamano la funzione **grtext** non funzionano in modo uguale su tutte le configurazioni, a meno che lo sviluppatore non le utilizzi esattamente come descritto (vedere il capitolo 4, "Menu personalizzati") ed eviti funzioni che dipendono in modo specifico dall'hardware. Infine, queste funzioni, essendo a basso livello, non visualizzano quasi nessun messaggio di errore e possono alterare in modo impreveduto la visualizzazione dello schermo di disegno (per la risoluzione di questo problema, vedere l'esempio riportato di seguito).

La sequenza di seguito riportata ripristina la visualizzazione di default della finestra di disegno da errori causati da chiamate non corrette alla funzione **grtext**, **grdraw** o **grvecs**:

```
(grtext) Ripristina il testo standard
```

(redraw)

Capitolo 8 -- Funzioni di utilità generali

Input dell'utente

Diverse funzioni attivano un'applicazione AutoLISP per richiedere all'utente l'immissione di dati. Di seguito sono riportate le funzioni di input utente.

entsel	getfiled	getpoint	nentsel
getangle	getint	getreal	nentselp
getcorner	getkeyword	getstring	
getdist	getorient	initget	

Capitolo 8 -- Funzioni di utilità generali

Input dell'utente

Funzioni getxxx

Tutte le funzioni **getxxx** che richiedono l'input dell'utente vengono sospese per consentire l'immissione di dati del tipo indicato e restituiscono il valore inserito. L'applicazione può specificare un messaggio di richiesta opzionale da visualizzare prima che la funzione venga sospesa. La seguente tabella contiene un elenco delle funzioni **getxxx** e del tipo di input utente richiesto.

Input dell'utente consentito per le funzioni getxxx

Nome funzione	<i>Tipo di input dell'utente</i>
getint	<i>Un valore intero dalla riga del messaggio di richiesta.</i>
getreal	<i>Un valore intero o reale dalla riga del messaggio di richiesta.</i>
getstring	<i>Una stringa dalla riga del messaggio di richiesta.</i>
getpoint	<i>Un valore punto dalla riga del messaggio di richiesta o selezionato dallo schermo.</i>
getcorner	<i>Un valore punto (l'angolo opposto di una casella) dalla riga del messaggio di richiesta o selezionato dallo schermo.</i>
getdist	<i>Un valore (di distanza) intero o reale indicato nella riga del messaggio di richiesta o determinato selezionando punti sullo schermo.</i>
getangle	<i>Un valore di angolo (nel formato corrente per l'angolo) dalla riga del messaggio di richiesta o determinato selezionando punti sullo schermo.</i>
getorient	<i>Un valore di angolo (nel formato corrente per l'angolo) dalla riga del messaggio di richiesta o determinato selezionando punti sullo schermo.</i>
getkeyword	<i>Una parola chiave predefinita o la relativa abbreviazione dalla riga del messaggio di richiesta.</i>

Nota Sebbene le funzioni **getvar**, **getcfg** e **getenv** inizino con le lettere *g*, *e* e *t*, non sono funzioni che richiedono l'input dell'utente. Tali funzioni sono illustrate nella sezione "Funzioni di richiesta e di comando."

Le funzioni **getint**, **getreal** e **getstring** vengono sospese per consentire all'utente l'input dalla riga del messaggio di richiesta di AutoCAD. Tali funzioni restituiscono un valore solo dello stesso tipo di quello richiesto.

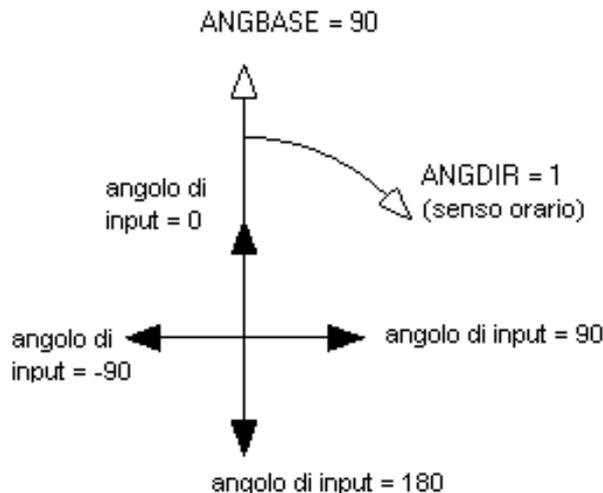
Le funzioni **getpoint**, **getcorner** e **getdist** vengono sospese per consentire all'utente l'input dalla riga del messaggio di richiesta o dai punti selezionati sullo schermo di disegno. Le funzioni **getpoint** e **getcorner** restituiscono valori punto tridimensionale, mentre la funzione **getdist** restituisce un valore reale.

Entrambe le funzioni **getangle** e **getorient** vengono sospese per consentire l'input di un valore di angolo dalla riga del messaggio di richiesta o come definito in base ai punti selezionati sullo schermo di disegno. Per la funzione **getorient**, l'angolo zero è sempre sulla destra: "Est" oppure "ore 3". Per la funzione **getangle**, l'angolo zero è il valore di ANGBASE, che può essere impostato su qualsiasi angolo. Entrambe le funzioni **getangle** e **getorient** restituiscono un valore di angolo (reale) in radianti misurato in senso antiorario da una base (angolo zero), per **getangle** uguale a ANGBASE, e per **getorient** sulla destra.

Ad esempio, ANGBASE è impostato su 90 gradi (nord), e ANGDIR è impostato su 1 (senso orario per angoli crescenti). La tabella seguente mostra i valori restituiti (in radianti) dalle funzioni **getangle** e **getorient** per valori di input rappresentativi (in gradi).

Possibili valori restituiti da *getangle* e *getorient*

Input (in gradi)	<i>getangle</i>	<i>getorient</i>
0	0.0	1.5708
-90	1.5708	3.14159
180	3.14159	4.71239
90	4.71239	0.0



La funzione **getangle** rispetta le impostazioni di **ANGDIR** ed **ANGBASE** quando accetta l'input. È possibile utilizzare tale funzione per ottenere una determinata rotazione per l'inserimento di un blocco, poiché l'input di 0 gradi restituisce sempre 0 radianti. La funzione **getorient** rispetta solo l'impostazione di **ANGDIR**. Utilizzare tale funzione per ottenere angoli come quello della linea di base per un oggetto di testo. Ad esempio, con le precedenti impostazioni di **ANGBASE** e **ANGDIR**, per una riga di testo creata ad un angolo di 0 gradi, **getorient** restituisce un valore di angolo di 90 gradi.

Le funzioni che richiedono l'input dell'utente utilizzano anche il controllo degli errori di AutoCAD. Errori banali vengono intercettati da AutoCAD e non vengono restituiti dalla funzione di input utente. Una precedente chiamata alla funzione **initget** fornisce ulteriori funzioni di filtro, riducendo la necessità del controllo degli errori.

La funzione **getkeyword** viene sospesa per l'input di una parola chiave o della relativa abbreviazione. Le parole chiave devono essere definite con la funzione **initget** prima della chiamata alla funzione **getkeyword**. Tutte le funzioni che richiedono l'input dell'utente (tranne **getstring**) possono accettare i valori di parole chiave oltre ai valori normalmente restituiti, a condizione che per definire le parole chiave sia stata richiamata la funzione **initget**.

Tutte le funzioni di input utente consentono un argomento *messaggio_di_richiesta* opzionale. Si consiglia di utilizzare questo argomento e non una precedente chiamata alla funzione **prompt** o **princ**. Se nella chiamata alla funzione di input utente viene fornito un argomento *messaggio_di_richiesta*, tale messaggio di richiesta viene ripetuto se l'input dell'utente non è valido. Se non viene fornito un argomento *messaggio_di_richiesta* e l'utente inserisce informazioni non corrette, in corrispondenza della riga del messaggio di richiesta appare il seguente messaggio:

Riprovare:

Tale messaggio può creare confusione, poiché è possibile che il messaggio di richiesta originale abbia causato uno scorrimento all'esterno dell'area della riga di comando.

Nota In genere l'utente di AutoCAD *non può* rispondere ad una funzione che richiede l'input inserendo un'espressione AutoLISP. Se la routine AutoLISP in uso utilizza la funzione **initget**, per determinate funzioni è consentito un input da tastiera arbitrario che può permettere di rispondere ad un comando AutoLISP con un'istruzione AutoLISP. Ciò è descritto in "Input arbitrario da tastiera."

Capitolo 8 -- Funzioni di utilità generali

Input dell'utente

Controllo delle condizioni delle funzioni di input utente

La funzione **initget** consente di controllare la successiva chiamata di funzioni che richiedono l'input dell'utente. La funzione **initget**

stabilisce diverse opzioni che possono essere utilizzate dalla successiva funzione **entsel**, **nentsel**, **nentselp** o **getxxx** (e non dalle funzioni **getstring**, **getvar** e **getenv**). Questa funzione accetta i due argomenti *bit* e *stringa*, entrambi opzionali. L'argomento *bit* specifica uno o più bit di controllo che attivano o disattivano determinati valori di input per la successiva chiamata alla funzione di input utente. L'argomento *stringa* può specificare le parole chiave che verranno riconosciute dalla successiva chiamata alla funzione di input utente.

I bit di controllo e le parole chiave determinate dalla funzione **initget** vengono applicati solo alla successiva chiamata della funzione di input utente. In seguito, vengono eliminati. L'applicazione non deve richiamare la funzione **initget** una seconda volta per annullare delle condizioni speciali.

Capitolo 8 -- Funzioni di utilità generali

Input dell'utente

Controllo delle condizioni delle funzioni di input utente

Opzioni di input per funzioni di input utente

Il valore dell'argomento *bit* limita i tipi di input dell'utente per la successiva chiamata della funzione che richiede questo input. In tal modo, viene ridotto il controllo degli errori. Di seguito, vengono indicati alcuni valori di bit disponibili. 1 impedisce l'input nullo, 2 impedisce l'input 0 (zero) e 4 impedisce l'input negativo. Se questi valori vengono utilizzati con una successiva chiamata alla funzione **getint**, l'utente deve necessariamente inserire un valore intero maggiore di 0.

Per impostare più condizioni contemporaneamente, aggiungere i valori in qualsiasi combinazione per creare un valore *bit* compreso tra 0 e 255. Se per *bit* non viene impostato alcun valore o il valore 0, alla successiva chiamata della funzione di input utente non viene applicata alcuna condizione di controllo. Per un elenco completo delle impostazioni bit **initget**, vedere "**initget**" nel capitolo 13.

```
(initget (+ 1 2 4))
(getint "\nQuanti anni hai? ")
```

Questa sequenza richiede l'età dell'utente. AutoCAD visualizza un messaggio di errore e mostra di nuovo il messaggio di richiesta se l'utente tenta di inserire un valore negativo o zero, se preme solo il tasto RETURN o se inserisce una stringa (la funzione **getint** rifiuta tentativi di inserimento di un valore diverso da un numero intero).

Capitolo 8 -- Funzioni di utilità generali

Input dell'utente

Controllo delle condizioni delle funzioni di input utente

Opzioni per parole chiave

L'argomento opzionale *stringa* specifica un elenco di parole chiave riconosciute dalla successiva chiamata della funzione di input utente.

La funzione **initget** consente di riconoscere abbreviazioni di parole chiave oltre alle parole chiave specificate per esteso. Tale funzione restituisce una parola chiave predefinita se l'input dell'utente corrisponde esattamente alla parola chiave (indipendentemente dal maiuscolo e minuscolo) o all'abbreviazione corrispondente. Esistono due metodi per l'abbreviazione delle parole chiave, entrambi descritti in "Specifiche delle parole chiave."

La seguente funzione definita dall'utente mostra una chiamata alla funzione **getreal**, preceduta da una chiamata alla funzione **initget**, che specifica due parole chiave. L'applicazione controlla queste parole chiave ed imposta di conseguenza il valore di input.

```
(defun C:GETNUM (/ num)
  (initget 1 "Pi Two-pi")
  (setq num (getreal "Pi/Two-pi/<numero>: "))
  (cond
    ((eq num "Pi") pi)
    ((eq num "Two-pi") (* 2.0 pi))
    (T num)
```

)
)

Questa chiamata alla funzione **initget** inibisce un input nullo (*bit* = 1) e determina un elenco di due parole chiave, "Pi" e "Two-pi". La funzione **getreal** viene quindi utilizzata per ottenere un numero reale, inviando il seguente messaggio di richiesta:

Pi/Two-pi/<numero>:

Il risultato viene inserito nel simbolo locale num. Se l'utente digita un numero, tale numero viene restituito da **C:GETNUM**. Tuttavia, se l'utente digita la parola chiave Pi (o semplicemente P), **getreal** restituisce la parola chiave Pi. La funzione **cond** rileva questa parola chiave e restituisce in questo caso il valore di p. La parola chiave Two-pi viene gestita in modo simile.

Nota La funzione **initget** può essere utilizzata anche per attivare **entsel**, **nentsel** e **nentselp** per accettare l'input di parole chiave. Per ulteriori informazioni su queste funzioni, vedere "Gestione degli oggetti" e "**entsel**" "**nentsel**" e "**nentselp**" nel capitolo 13,.

Capitolo 8 -- Funzioni di utilità generali

Input dell'utente

Controllo delle condizioni delle funzioni di input utente

Input arbitrario da tastiera

La funzione **initget** consente, inoltre, un input da tastiera arbitrario per la maggior parte delle funzioni **getxxx**. Questo input viene passato nuovamente all'applicazione come stringa. Per consentire all'utente di richiamare una funzione AutoLISP quando viene visualizzato un messaggio che richiede una funzione **getxxx**, è possibile scrivere un'applicazione usando questa funzione.

Le seguenti funzioni costituiscono un metodo per consentire la risposta AutoLISP ad una chiamata di una funzione **getxxx**:

```
(defun C:ARBENTRY ( / pt1)
  (initget 128) ;Imposta bit immissione arbitrario.
  (setq pt1 (getpoint "\nPunto: ")); Richiama il valore utente.
  (if (= 'STR (type pt1)) ; Se è una stringa, la converte
    (setq pt1 (eval (read pt1))) ; in un simbolo e tenta la valutazione
    ; come funzione; in caso contrario
    pt1 ; restituisce semplicemente il valore.
  )
)
(defun REF ( )
  (setvar "LASTPOINT" (getpoint "\nPunto di riferimento: "))
  (getpoint "\nPunto successivo: " (getvar "LASTPOINT"))
)
```

La sequenza di comando di seguito riportata può essere accettata se entrambe le funzioni **C:ARBENTRY** e **REF** sono state caricate.

Comando: **arbentry**

Punto: (**ref**)

Punto di riferimento: *Selezionare un punto*

Punto successivo: **@1,1,0**

Capitolo 8 -- Funzioni di utilità generali

Input dell'utente

Controllo delle condizioni delle funzioni di input utente

Convalida dell'input

Si consiglia di proteggere il codice da errori causati inavvertitamente dall'utente. Le funzioni AutoLISP **getxxx** che richiedono l'input dell'utente eseguono questa azione di controllo. Tuttavia, è necessario verificare sempre che gli altri requisiti di programma, non controllati dalle funzioni **getxxx**, siano soddisfatti. In caso contrario, l'integrità dei programmi potrebbe essere danneggiata.

Capitolo 8 -- Funzioni di utilità generali

Utilità geometriche

Esiste un gruppo di funzioni che consente alle applicazioni di ottenere informazioni di geometria pura e dati geometrici dal disegno. Le funzioni di utilità geometriche sono le seguenti:

angle	inters	polar
distance	osnap	textbox

La funzione **angle** individua l'angolo in radianti tra una linea e l'asse X del Sistema di Coordinate Utente (UCS) corrente, la funzione **distance** individua la distanza tra due punti e la funzione **polar** individua un punto mediante le coordinate polari (in base ad un punto iniziale). La funzione **inters** individua l'intersezione tra due linee. Le funzioni **osnap** e **textbox** vengono descritte separatamente.

La parte di codice di seguito riportata mostra le chiamate alle funzioni di utilità geometriche:

```
(setq pt1 '(3.0 6.0 0.0))
(setq pt2 '(5.0 2.0 0.0))
(setq base '(1.0 7.0 0.0))
(setq rads (angle pt1 pt2))      Angolo nel piano XY del Sistema di Coordinate Utente (UCS)
                               corrente (il valore viene restituito in radianti)

(setq len (distance pt1 pt2))   Distanza in spazio tridimensionale
(setq endpt (polar base rads len))
```

La chiamata alla funzione **polar** imposta endpt su un punto che si trova ad una distanza da (1,7) uguale alla distanza di pt1 da pt2, e ad un angolo dall'asse X uguale all'angolo tra pt1 e pt2.

Capitolo 8 -- Funzioni di utilità generali

Utilità geometriche

Snap ad oggetto

La funzione **osnap** è in grado di individuare un punto utilizzando una delle modalità Snap ad oggetti di AutoCAD. Le modalità Snap vengono specificate in un argomento di stringa.

La seguente chiamata alla funzione **osnap** ricerca il punto medio di un oggetto accanto a pt1:

```
(setq pt2 (osnap pt1 "midp"))
```

La seguente chiamata ricerca il punto medio, il punto finale o il centro di un oggetto il più possibile vicino a pt1:

```
(setq pt2 (osnap pt1 "midp,endp,center"))
```

In entrambi gli esempi, pt2 viene impostato sul punto di snap se è stato individuato un punto che soddisfa i requisiti osnap. Se più di un punto di snap soddisfa i requisiti, il punto viene selezionato in base all'impostazione della variabile di sistema SORTENTS. In caso contrario, pt2 viene impostato su nil.

Nota La variabile di sistema APERTURE determina la prossimità consentita di un punto selezionato ad un oggetto quando si utilizza la funzione Snap ad oggetti.

Capitolo 8 -- Funzioni di utilità generali

Utilità geometriche

Estensioni di testo

La funzione **textbox** restituisce le coordinate diagonali di una casella che racchiude un oggetto di testo. Tale funzione utilizza come singolo argomento un elenco di definizioni di entità del tipo restituito da **entget** (un elenco di associazione di valori e codici di gruppo). Questo elenco può contenere una descrizione completa dell'elenco di associazione dell'oggetto di testo o solo un elenco che descrive la stringa di testo.

I punti restituiti da **textbox** descrivono la casella di delimitazione (una casella immaginaria che racchiude l'oggetto di testo) dell'oggetto di testo considerando come relativo punto di inserimento il punto (0,0,0) e come relativo angolo di rotazione 0. Il primo elenco restituito è il punto (0.0 0.0 0.0) a meno che l'oggetto di testo non sia obliquo o verticale oppure contenga lettere con tratti discendenti (come g e p). Il valore del primo elenco di punti specifica la distanza di sfalsamento dal punto di inserimento del testo all'angolo inferiore sinistro del rettangolo più piccolo che racchiude il testo. Il secondo elenco di punti specifica l'angolo superiore destro di questa casella. Gli elenchi di punti restituiti descrivono sempre gli angoli inferiore sinistro e superiore destro di questa casella di delimitazione, indipendentemente dall'orientamento del testo da misurare.

L'esempio seguente mostra minimo consentito l'elenco di definizioni di entità che la funzione **textbox** accetta. Poiché non vengono fornite ulteriori informazioni, **textbox** utilizza per l'altezza e lo stile del testo i valori di default correnti.

Comando: **(textbox '(1 . "Ciao a tutti"))**
 ((0.0 0.0 0.0) (2.80952 1.0 0.0))

I valori attuali restituiti dalla funzione **textbox** saranno diversi a seconda dello stile corrente del testo.

L'esempio seguente descrive un metodo per fornire un elenco di definizioni di entità alla funzione **textbox**.

Comando: **testodin**
 Giustificato/Stile/<Punto iniziale>: **1,1**
 Altezza <1.0000>: RETURN
 Angolo di rotazione <0>: RETURN
 Testo: **verifica**
 Testo: RETURN
 Comando: **(setq e (entget (entlast)))**
 ((-1 . <Nome di entità: 6000001c>) (0 . "TESTO") (8 . "0")
 (10 1.0 1.0 0.0) (40 . 1.0) (1 . "verifica") (50 . 0.0)
 (41 . 1.0)(51 . 0.0) (7 . "STANDARD") (71 . 0) (72 . 0)
 (11 0.0 0.0 0.0) (210 0.0 0.0 1.0) (73 . 0))
 Comando: **(textbox e)**
 ((0.0 0.0 0.0) (0.8 0.2 0.0))

La figura di seguito riportata mostra i risultati dell'applicazione della funzione **textbox** ad un oggetto di testo con altezza 1.0 La figura mostra anche la linea di base ed il punto di inserimento del testo.

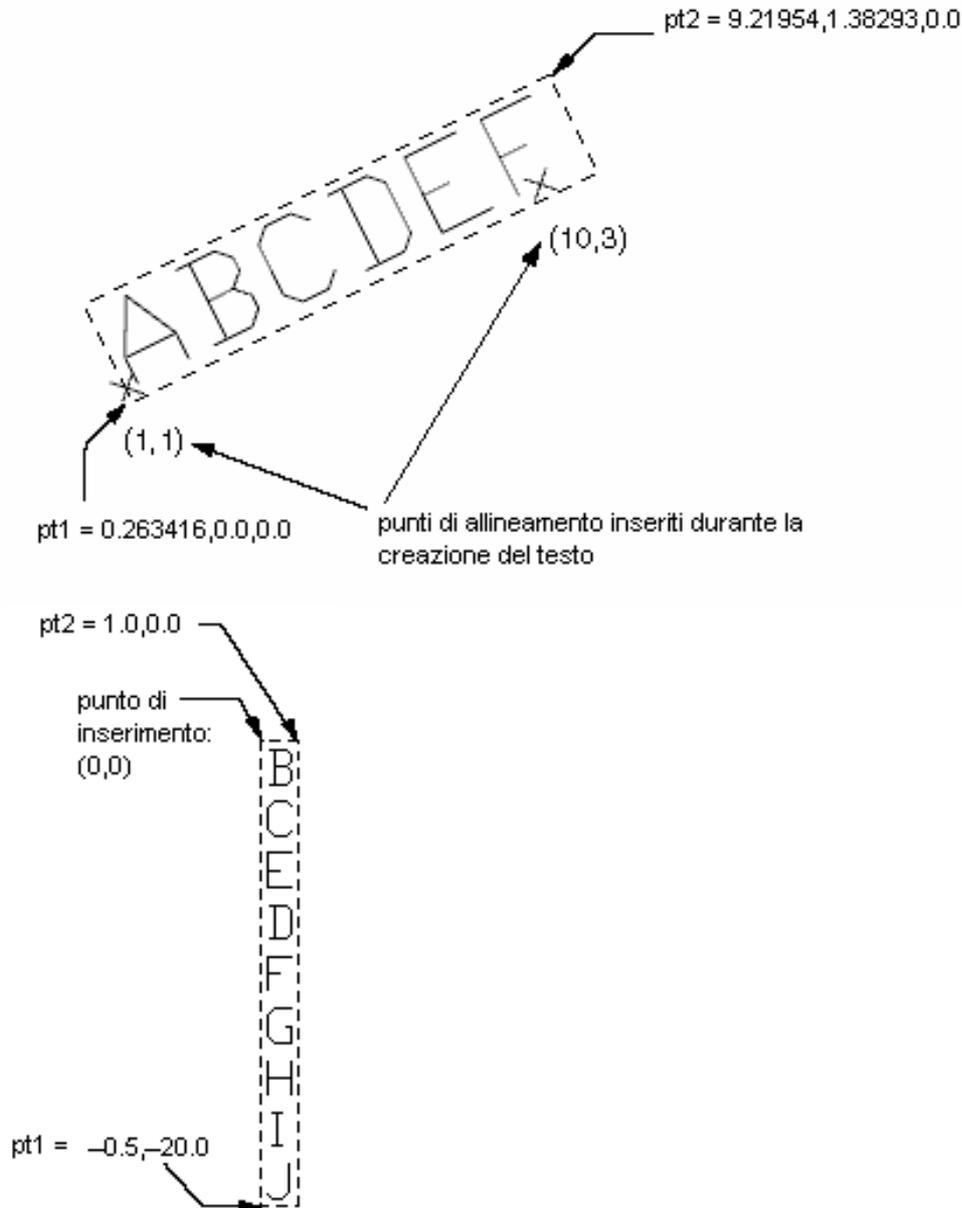


Punti restituiti dalla funzione **textbox**

Se il testo è verticale o ruotato, pt1 è ancora l'angolo inferiore sinistro e pt2 è l'angolo superiore destro; il punto inferiore sinistro, se

necessario, può avere sfalsamenti negativi.

La figura seguente mostra i valori punto (pt1 e pt2) che la funzione **textbox** restituisce per esempi di testo verticale. In entrambi gli esempi, l'altezza delle lettere è 1.0. Per il testo allineato, l'altezza viene regolata in modo da adattarsi ai punti di allineamento.



Testo verticale ed allineato

Quando si utilizzano stili di testo verticali, i punti vengono restituiti da sinistra a destra e dal basso verso l'alto come per gli stili orizzontali; in tal modo, il primo elenco di punti conterrà sfalsamenti negativi dal punto di inserimento del testo.

Indipendentemente dallo stile o dall'orientamento del testo, i punti restituiti dalla funzione **textbox** sono tali che il punto di inserimento del testo (codice di gruppo 10) viene convertito direttamente nel punto di origine del Sistema di Coordinate Oggetto (OCS). È possibile fare riferimento a questo punto quando le coordinate restituite dalla funzione **textbox** vengono convertite in punti che definiscono le reali estensioni del testo. Le due routine di esempio riportate di seguito utilizzano la funzione **textbox** per posizionare una casella intorno al testo selezionato indipendentemente dall'orientamento.

La prima routine utilizza la funzione **textbox** per disegnare una casella intorno all'oggetto di testo selezionato.

```
(defun C:TBOX ( / textent tb ll ur ul lr)
  (setq textent (car (entsel "\nSelezionare il testo: ")))
```

```

(command "ucs" "Oggetto" textent)
(setq tb (textbox (list (cons -1 textent)))
  ll (car tb)
  ur (cadr tb)
  ul (list (car ll) (cadr ur))
  lr (list (car ur) (cadr ll))
)
(command "pline" ll lr ur ul "Chiudi")
(command "ucs" "p")
(princ)
)

```

La seconda routine, riportata di seguito, effettua la stessa operazione della prima eseguendo calcoli geometrici mediante le funzioni AutoLISP **sin** e **cos**. Il risultato è corretto solo se il Sistema di Coordinate Utente (UCS) corrente è parallelo al piano dell'oggetto di testo.

```

(defun C:TBOX2 ( / textent ang sinrot cosrot
  t1 t2 p0 p1 p2 p3 p4)
  (setq textent (entget (car (entsel "\nSelezionare il testo: "))))
  (setq p0 (cdr (assoc 10 textent))
    ang (cdr (assoc 50 textent))
    sinrot (sin ang)
    cosrot (cos ang)
    t1 (car (textbox textent))
    t2 (cadr (textbox textent))
    p1 (list
      (+ (car p0)
        (- (* (car t1) cosrot) (* (cadr t1) sinrot))
      )
      (+ (cadr p0)
        (+ (* (car t1) sinrot) (* (cadr t1) cosrot))
      )
    )
    p2 (list
      (+ (car p0)
        (- (* (car t2) cosrot) (* (cadr t2) sinrot))
      )
      (+ (cadr p0)
        (+ (* (car t2) sinrot) (* (cadr t2) cosrot))
      )
    )
    p3 (list
      (+ (car p0)
        (- (* (car t2) cosrot) (* (cadr t2) sinrot))
      )
      (+ (cadr p0)
        (+ (* (car t2) sinrot) (* (cadr t2) cosrot))
      )
    )
    p4 (list
      (+ (car p0)
        (- (* (car t1) cosrot) (* (cadr t2) sinrot))
      )
      (+ (cadr p0)
        (+ (* (car t1) sinrot) (* (cadr t2) cosrot))
      )
    )
  )
  (command "pline" p1 p2 p3 p4 "c")
  (princ)
)

```

Capitolo 8 -- Funzioni di utilità generali

Conversioni

Le funzioni descritte in questa sezione sono utilità per la conversione di unità e tipi di dati. Di seguito sono riportate le funzioni di conversione.

angtof	atof	cvunit	rtos
angtos	atoi	distof	trans
ascii	chr	itoa	

Le funzioni di conversione sono descritte anche nel capitolo 13.

Capitolo 8 -- Funzioni di utilità generali

Conversioni

Conversioni di stringhe

Le funzioni **rtos** (da reale a stringa) e **angtos** (da angolo a stringa) convertono valori numerici in AutoCAD in valori di stringa che possono essere utilizzati in output o come dati di testo. La funzione **rtos** converte un valore reale e la funzione **angtos** converte un angolo. Il formato della stringa risultante viene controllato dal valore delle variabili di sistema di AutoCAD: le unità e la precisione vengono specificate da LUNITS e LUPREC per valori reali (lineari) e da AUNITS ed AUPREC per valori angolari. Per entrambe le funzioni, la variabile di quotatura DIMZIN controlla il modo in cui zeri iniziali e finali vengono scritti nella stringa risultante.

Le parti di codice di seguito riportate mostrano chiamate alla funzione **rtos** ed i valori restituiti (presupponendo che la variabile DIMZIN sia uguale a 0). La precisione (terzo argomento per **rtos**) è impostata su 4 posizioni nella prima chiamata e su 2 posizioni nelle altre chiamate.

```
(setq x 17.5)
(setq str "\nValore formattato come ")
(setq fmtval (rtos x 1 4))      Modalità 1 = scientifica
(princ (strcat str fmtval))   visualizza Valore formattato come 1.7500E+01
(setq fmtval (rtos x 2 2))      Modalità 2 = decimale
(princ (strcat str fmtval))   visualizza Valore formattato come 17.50
(setq fmtval (rtos x 3 2))      Modalità 3 = ingegneristica
(princ (strcat str fmtval))   visualizza Valore formattato come 1'-5.50"
(setq fmtval (rtos x 4 2))      Modalità 4 = architettonica
(princ (strcat str fmtval))   visualizza Valore formattato come 1'-5 1/2"
(setq fmtval (rtos x 5 2))      Modalità 5 = frazionarie
(princ (strcat str fmtval))   visualizza Valore formattato come 17 1/2
```

Quando la variabile di sistema UNITMODE è impostata su 1, specificando che le unità vengono visualizzate così come digitate, la stringa restituita da **rtos** è diversa per le unità ingegneristiche (*modalità 3*), architettoniche (*modalità 4*) e frazionarie (*modalità 5*). Ad esempio, le prime due righe del precedente output di esempio sono uguali, ma le ultime tre righe appaiono nel modo seguente:

```
Valore formattato come 1'5.50"
Valore formattato come 1"5-1/2"
Valore formattato come 17-1/2
```

Poiché la funzione **angtos** considera la variabile di sistema ANGBASE, il codice di seguito indicato restituisce sempre "0".

```
(angtos (getvar "angbase"))
```

Non esiste alcuna funzione AutoLISP che restituisce una versione di stringa (nella precisione/modalità corrente) della quantità di rotazione di ANGBASE dallo zero reale (Est) o di un angolo arbitrario in radianti.

Per individuare la quantità di rotazione di ANGBASE dallo zero AutoCAD (Est) o la dimensione di un angolo arbitrario, è possibile effettuare una delle seguenti operazioni:

- Aggiungere l'angolo desiderato al valore di ANGBASE corrente e controllare se il valore assoluto del risultato è maggiore di 2π ($2 * \pi$). In tal caso, sottrarre 2π ; se il risultato è negativo, aggiungere 2π . Quindi, utilizzare la funzione **angtos** sul risultato.

- Memorizzare il valore di ANGBASE in una variabile temporanea, impostare ANGBASE su 0 e valutare la funzione **angtos**. Quindi, impostare per ANGBASE il valore originale.

Sottraendo il risultato di (atof (angtos 0)) da 360 gradi (2π radianti o 400 gradi) si ottiene comunque la rotazione di ANGBASE da 0.

La funzione **distof** (distanza dalla virgola mobile) completa la funzione **rtos**, pertanto, le seguenti chiamate, che utilizzano stringhe generate negli esempi precedenti, restituiscono tutte lo stesso valore: 17.5. Notare l'uso della barra rovesciata (\) con le modalità 3 e 4.

```
(distof "1.7500E+01" 1)   Modalità 1 = scientifica
(distof "17.50" 2)       Modalità 2 = decimale
(distof "1'-5.50\" 3)     Modalità 3 = ingegneristica
(distof "1'-5 1/2\" 4)     Modalità 4 = architettonica
(distof "17 1/2" 5)      Modalità 5 = frazionaria
```

Le parti di codice di seguito riportate mostrano chiamate alla funzione **angtos** simili ed i valori restituiti (presupponendo sempre che la variabile DIMZIN sia uguale a 0). La precisione (il terzo argomento per **angtos**) è impostata su 0 posizioni nella prima chiamata, su 4 posizioni nelle successive tre chiamate e su 2 posizioni nell'ultima.

```
(setq ang 3.14159 str2 "\nAngolo formattato come ")
(setq fmtval (angtos ang 0 0))   Modalità 0 = gradi
(princ (strcat str2 fmtval))    visualizza Angolo formattato come 180
(setq fmtval (angtos ang 1 4))   Modalità 1 = gradi/min/sec
(princ (strcat str2 fmtval))    visualizza Angolo formattato come 180d0'0"
(setq fmtval (angtos ang 2 4))   Modalità 2 = gradi
(princ (strcat str2 fmtval))    visualizza Angolo formattato come 200.0000g
(setq fmtval (angtos ang 3 4))   Modalità 3 = radianti
(princ (strcat str2 fmtval))    visualizza Angolo formattato come 3.1416r
(setq fmtval (angtos ang 4 2))   Modalità 4 = topografica
(princ (strcat str2 fmtval))    visualizza Angolo formattato come 0
```

Quando la variabile di sistema UNITMODE restituisce una stringa in unità topografiche (*modalità 4*), tale variabile influisce anche sulle stringhe restituite dalla funzione **angtos**. Se UNITMODE è uguale a 0, la stringa restituita può includere spazi (ad esempio, "N45d E"); se UNITMODE è uguale a 1, la stringa non contiene spazi (ad esempio, "N45dE").

La funzione **angtof** completa la funzione **angtos**; pertanto, tutte le seguenti chiamate restituiscono lo stesso valore: 3.14159.

```
(angtof "180" 0)           Modalità 0 = gradi
(angtof "180d0'0\" 1)      Modalità 1 = gradi/min/sec
(angtof "200.0000g" 2)    Modalità 2 = gradi
(angtof "3.14159r" 3)     Modalità 3 = radianti
(angtof "0" 4)            Modalità 4 = topografica
```

Se si dispone di una stringa che specifica una distanza in piedi e pollici o un angolo in gradi, minuti e secondi, è necessario far precedere il simbolo delle virgolette da una barra rovesciata (\) in modo che tale simbolo non venga interpretato come la fine della stringa. Gli esempi precedenti delle funzioni **angtof** e **distof** dimostrano tale operazione.

Capitolo 8 -- Funzioni di utilità generali

Conversioni

Conversione angolare

Se l'applicazione necessita la conversione di valori angolari da radianti in gradi, è possibile usare la funzione **angtos**, che restituisce una stringa, e quindi convertire tale stringa in un valore mobile con la funzione **atof**.

```
(setq pt1 '(1 1) pt2 '(1 2))
(setq rad (angle pt1 pt2))
(setq deg (atof (angtos rad 0 2)))   restituisce 90.0
```

Comunque, un metodo più efficiente potrebbe essere quello di includere nell'applicazione una funzione **DTR** (in gradi o in radianti). Il seguente codice mostra questo metodo:

```
; Convertire angolo in radianti in gradi
(defun DTR (d)
```

```
(* pi (/ d 180.0))
)
```

Dopo aver definito la funzione, è possibile usare la funzione **DTR** nell'applicazione:

```
(setq deg (DTR rad))           restituisce      90.0
```

È possibile che si debba inoltre convertire i gradi in radianti. Il seguente codice mostra questo metodo:

```
; Convertire angolo in gradi in radianti
(defun DTR (d)
  (* pi (/ d 180.0))
)
```

Capitolo 8 -- Funzioni di utilità generali

Conversioni

Conversione di codici ASCII

AutoLISP prevede le funzioni **ascii** e **chr** per la gestione di codici decimali ASCII. La funzione **ascii** restituisce il valore decimale ASCII associato ad una stringa, la funzione **chr** restituisce il carattere associato ad un valore decimale ASCII.

Per visualizzare i caratteri di sistema con i relativi codici in forma ottale, decimale ed esadecimale, salvare il codice AutoLISP di seguito riportato in un file chiamato *ascii.sp*. Quindi, caricare il file e digitare il nuovo comando ASCII in corrispondenza della riga di comando. Questo comando visualizza i codici ASCII sullo schermo e li inserisce in un file chiamato *ascii.txt*. La funzione **C:ASCII** usa inoltre la funzione **BASE**. Questa funzione di conversione può risultare utile in altre applicazioni.

```
; BASE converte un numero intero decimale in una stringa su altra base
(defun BASE ( bas int / ret yyy zot )
  (defun zot ( i1 i2 / xxx )
    (if (> (setq xxx (rem i2 i1)) 9)
      (chr (+ 55 xxx))
      (itoa xxx)
    )
  )
  (setq ret (zot bas int) yyy (/ int bas))
  (while (>= yyy bas)
    (setq ret (strcat (zot bas yyy) ret))
    (setq yyy (/ yyy bas))
  )
  (strcat (zot bas yyy) ret)
)
(defun C:ASCII ( / chk out ct code dec oct hex )
  (initget "Si")
  (setq chk (getstring "\nScrittura sul file ASCII.TXT, continuare? <Y>: "))
  (if (or (= chk "Si") (= chk ""))
    (progn
      (setq out (open "ascii.txt" "w") chk 1 code 0 ct 0)
      (princ "\n \n CHAR   DEC   OCT   HEX \n")
      (princ "\n \n CHAR   DEC   OCT   HEX \n" out)
      (while chk
        (setq dec (strcat " " (itoa code))
              oct (base 8 code) hex (base 16 code))
        (setq dec (substr dec (- (strlen dec) 2) 3))
        (if (< (strlen oct) 3) (setq oct (strcat "0" oct)))
        (princ (strcat "\n " (chr code) "      " dec " "
                      oct " " hex ) )
        (princ (strcat "\n " (chr code) "      " dec " "
                      oct " " hex ) out)
        (cond
          ((= code 255) (setq chk nil))
          ((= ct 20)
            (setq xxx (getstring
```

```

        "\n \nPremere 'E' per uscire o qualsiasi tasto per continuare: ")
    (if (= (strcase xxx) "X")
        (setq chk nil)
        (progn
            (setq ct 0)
            (princ "\n \n CHAR    DEC    OCT    HEX \n")
        )
    )
    )
    )
    (setq ct (1+ ct) code (1+ code))
    )
    (close out)
    (setq out nil)
    )
    )
    (princ)
    )

```

Capitolo 8 -- Funzioni di utilità generali

Conversioni

Conversione di unità

Il file *acad.unt* definisce diverse conversioni tra unità reali, ad esempio, da miglia in chilometri, da Fahrenheit in Celsius e così via. La funzione **cvunit** prende un valore espresso in un sistema di unità e restituisce il valore equivalente in un altro sistema. I due sistemi di unità vengono specificati da stringhe contenenti espressioni di unità definite nel file *acad.unt*.

La funzione **cvunit** *non* converte dimensioni incompatibili. Ad esempio, non converte pollici in grammi.

La prima volta che la funzione **cvunit** esegue una conversione in o da un'unità durante una sessione di editazione del disegno, è necessario considerare la stringa che specifica l'unità nel file *acad.unt*. Se l'applicazione in uso dispone di molti valori per eseguire conversioni da un sistema di unità ad un altro, si consiglia di convertire il valore 1.0 mediante una singola chiamata alla funzione **cvunit** ed utilizzare poi il valore restituito come fattore di scala nelle successive conversioni. Tale operazione può essere effettuata per tutte le unità definite nel file *acad.unt* eccetto le scale della temperatura, che comprendono uno sfalsamento oltre ad un fattore di scala.

Capitolo 8 -- Funzioni di utilità generali

Conversioni

Conversione di unità

Conversione da pollici in metri

Se le unità di disegno correnti sono unità ingegneristiche o architettoniche (piedi e pollici), la routine che segue converte in metri una distanza in pollici specificata dall'utente:

```

(defun C:I2M ( / eng_len metric_len eng metric)
  (princ "\nConversione da pollici in metri. ")
  (setq eng_len
    (getdist "\nDigitare una distanza in pollici: "))
  (setq metric_len (cvunit eng_len "pollici" "metri"))
  (setq eng (rtos eng_len 2 4)
    metric (rtos metric_len 2 4))
  (princ
    (strcat "\n\t" eng " pollici= " metric " metri. "))
  (princ)

```

)

Capitolo 8 -- Funzioni di utilità generali

Conversioni

Conversione di unità

File di definizione delle unità

Con il file di definizione delle unità *acad.unt* di AutoCAD è possibile definire i fattori per convertire dati espressi in un gruppo di unità in un altro gruppo di unità. Le definizioni presenti nel file *acad.unt* sono in formato ASCII e vengono utilizzate dalla funzione per la conversione di unità **cvunit**.

È possibile rendere disponibili nuove unità aggiungendo le relative definizioni al file *acad.unt* mediante un editor di testo. Una definizione è costituita da due righe: il *nome unità* e la *definizione unità*. La prima riga deve contenere un asterisco (*) nella prima colonna, seguito dal nome dell'unità. Per un nome unità possono esistere diverse abbreviazioni o nomi alternativi separati da virgole. Se un nome unità è definito sia al singolare che al plurale, è possibile specificare entrambe le definizioni utilizzando il seguente formato:

```
*[ [comune] [ ( [singolare.] plurale) ] ]...
```

È possibile specificare più espressioni (singolare e plurale). Tali espressioni non devono trovarsi necessariamente alla fine della parola e non è obbligatoria la forma plurale.

```
*pollice(i)
*millenni(o.i)
*pied(e.i) o (piede.piedi)
```

La riga successiva a **nome unità* definisce l'unità come *fondamentale* o *derivata*.

Unità fondamentali

Un'unità fondamentale è un'espressione in costanti. Se la riga che segue il **nome unità* inizia con un carattere diverso dal segno di uguale (=), definisce unità fondamentali. Tale riga è composta da cinque numeri interi e due numeri reali nel seguente formato:

c, e, h, k, m, r1, r2

I cinque numeri interi corrispondono agli esponenti di queste cinque costanti:

- c** velocità della luce nel vuoto
- e** carica dell'elettrone
- h** costante di Planck
- k** costante di Boltzman
- m** massa a riposo dell'elettrone

Come gruppo, questi esponenti definiscono le dimensioni dell'unità: lunghezza, massa, tempo, volume e così via.

Il primo numero reale (r1) è un moltiplicatore ed il secondo (r2) è uno sfalsamento additivo (utilizzato solo per conversioni di temperature). La definizione di unità fondamentale consente di esprimere l'unità in diversi modi; l'uso di maiuscole e minuscole non è rilevante. Di seguito viene riportato un esempio di unità fondamentale.

```
*metro(i),metro(i),m
-1,0,1,0,-1,4.1214856408e11,0
```

In questo esempio, le costanti che costituiscono un metro sono:

$$\left(\frac{1}{c} \times h \times \frac{1}{m}\right) \times (4.1214856 \times 10^{11})$$

Unità derivate

Un'unità derivata viene definita utilizzando termini di altre unità. Se la riga che segue il **nome unità* inizia con un segno di uguale (=), definisce unità derivate. Operatori validi in queste definizioni sono * (moltiplicazione), / (divisione), + (addizione), - (sottrazione) e ^ (elevazione a potenza). È possibile specificare un'unità predefinita assegnandole un nome e, se definite, è possibile utilizzare anche abbreviazioni. Gli elementi di una formula vengono moltiplicati insieme a meno che non siano stati specificati altri operatori aritmetici.

Ad esempio, il database di unità definisce i nomi di sottomultipli e multipli adimensionati in modo che sia possibile specificare un'unità come "micro-pollici" digitando **micro inch**. Di seguito sono riportati degli esempi di definizioni di unità derivate.

```
; Unità di area
*circoscrizione(i)
=93239571.456 meter^2
```

La definizione di una circoscrizione viene specificata come 93,239,571.456 metri quadrati.

```
; Unità elettromagnetiche
*volt(s),v
=watt/ampere
```

In questo esempio, un volt viene definito come un watt diviso per un ampere. Nel file *acad.unt*, sia i watt che gli ampere vengono definiti in termini di unità fondamentali.

Commenti dell'utente

Per includere commenti, iniziare la riga con un punto e virgola. Il commento continua fino alla fine della riga.

```
; Tutta questa riga è un commento
```

Per ulteriori informazioni ed esempi, vedere il file *acad.unt*.

Capitolo 8 -- Funzioni di utilità generali

Conversioni

Trasformazioni del sistema di coordinate

La funzione **trans** converte un punto o uno spostamento da un sistema di coordinate in un altro. Tale funzione prende un argomento punto *pt*, il quale può essere interpretato come un punto tridimensionale o un vettore di spostamento tridimensionale, contraddistinto da un argomento di spostamento chiamato *spost*. L'argomento *spost* deve essere un valore diverso da zero, se *pt* deve essere considerato come un vettore di spostamento, altrimenti *pt* viene considerato come un punto. Un argomento *da* specifica il sistema di coordinate nel quale viene espresso *pt* e l'argomento *a* specifica il sistema di coordinate desiderato. Di seguito viene riportata la sintassi per la funzione **trans**.

```
(trans pt da a [spost])
```

I sistemi di coordinate AutoCAD riportati di seguito possono essere specificati dagli argomenti *da* ed *a*.

WCS

World Coordinate System, Sistema di Coordinate Globali: il sistema di coordinate di "riferimento". Tutti gli altri sistemi di coordinate vengono definiti in base al WCS che non viene mai modificato. I valori misurati in base a tale sistema rimangono sempre uguali nonostante le modifiche apportate agli altri sistemi di coordinate.

UCS

User Coordinate System, Sistema di Coordinate Utente: il sistema di coordinate "operativo". L'utente specifica un UCS per rendere più semplice l'esecuzione delle operazioni di disegno. Tutti i punti passati ai comandi di AutoCAD, inclusi quelli restituiti da funzioni esterne e routine AutoLISP, sono punti nell'UCS corrente, a meno che l'utente non specifichi prima di essi un asterisco (*) in corrispondenza della riga di comando. Se si desidera che l'applicazione invii le coordinate dei sistemi WCS, OCS o DCS ai comandi di AutoCAD, è necessario prima convertirle in UCS richiamando la funzione **trans**.

OCS

Object Coordinate System, Sistema di Coordinate Oggetto: i valori punto restituiti da **entget** sono espressi in questo sistema di coordinate in relazione all'oggetto stesso. Questi punti, in genere, vengono convertiti nel sistema WCS oppure nel sistema UCS o DCS corrente in base all'uso previsto per l'oggetto. I punti, viceversa, devono essere convertiti in un OCS prima di essere scritti nel database mediante la funzione **entmod** o **entmake**. **Questo sistema è anche detto Sistema di Coordinate di Entità (Entity Coordinate System).**

DCS

Display Coordinate System, Sistema di Coordinate di Visualizzazione: il sistema di coordinate nel quale vengono convertiti gli oggetti prima della relativa visualizzazione. L'origine del DCS è il punto memorizzato nella variabile di sistema di AutoCAD TARGET ed il relativo asse Z è la direzione di visualizzazione. In altre parole, una finestra è sempre una vista

piana del DCS relativo. Queste coordinate possono essere utilizzate per determinare il punto in cui un oggetto verrà visualizzato per l'utente AutoCAD.

Quando i codici a numero intero *da* ed *a* sono 2 e 3, in un qualsiasi ordine, 2 indica il DCS per la finestra dello spazio del modello corrente e 3 indica il DCS per lo spazio carta (DCSSC). Quando il codice 2 viene utilizzato con un codice a numero intero diverso da 3 (o un altro metodo di specifica del sistema di coordinate), si presume che tale codice indichi il DCS dello spazio corrente, sia se si tratta di spazio carta che di spazio modello. Si presume, inoltre, che l'altro argomento indichi un sistema di coordinate nello spazio corrente.

PSDCS

DCS dello Spazio Carta: questo sistema di coordinate può essere convertito *solo* nel o dal DCS della finestra dello spazio modello attualmente attivo. Questa è essenzialmente una trasformazione bidimensionale, nella quale le coordinate X ed Y sono sempre scalate e sfalsate se l'argomento *spost* è 0. La coordinata Z *viene* scalata ma mai convertita; pertanto, può essere utilizzata per individuare il fattore di scala tra i due sistemi di coordinate. Il DCSSC (codice a numero intero 2) può essere convertito solo nella finestra di spazio del modello corrente: se l'argomento *da* è uguale a 3, l'argomento *a* deve essere uguale a 2 e viceversa.

Entrambi gli argomenti *da* ed *a* possono specificare un sistema di coordinate in uno dei seguenti modi:

- Come un codice a numero intero che specifica il WCS, l'UCS o il DCS corrente (sia della finestra corrente che dello spazio carta).
- Come un nome di entità restituito da una delle funzioni per gruppi di selezione o per nomi di entità descritte nel capitolo 9, "Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli". Questo nome specifica l'OCS dell'oggetto denominato. Per oggetti piani, l'OCS può essere diverso dal WCS, come descritto nel *Manuale dell'utente di AutoCAD*. Se OCS non presenta differenze, la conversione tra OCS e WCS è un'operazione di identità.
- Come un vettore d'estrusione tridimensionale. I vettori di estrusione sono sempre rappresentati in coordinate Globali; un vettore di estrusione di (0,0,1) specifica il WCS stesso.

La tabella seguente riporta un elenco dei codici a numero intero validi che possono essere utilizzati come argomenti *a* e *da*.

Codici del sistema di coordinate

Codice	Sistema di coordinate
0	Globale (WCS)
1	Utente (UCS corrente)
2	Visualizzazione; DCS della finestra corrente quando utilizzato con il codice 0 o 1; DCS della finestra dello spazio del modello corrente quando utilizzato con il codice 3
3	DCS dello Spazio Carta, DCSSC (utilizzato solo con il codice 2)

L'esempio seguente converte un punto dal WCS nell'UCS corrente.

```
(setq pt '(1.0 2.0 3.0))
(setq cs_from 0)           WCS
(setq cs_to 1)            UCS
(trans pt cs_from cs_to 0)  spost = 0 indica che pt è un punto
```

Se l'UCS corrente viene ruotato di 90 gradi in senso antiorario sull'asse Globale Z, la chiamata alla funzione **trans** restituisce un punto (2.0,-1.0,3.0). Tuttavia, se si scambiano i valori *a* e *da*, il risultato cambia come mostrato nel seguente codice.

```
(trans pt cs_to cs_from 0) ; il risultato è (-2.0,1.0,3.0)
```

Capitolo 8 -- Funzioni di utilità generali

Conversioni

Trasformazioni del sistema di coordinate

Trasformazioni di punti

Se si stanno effettuando trasformazioni di punti con la funzione **trans** ed è necessario rendere più veloce l'esecuzione di tale parte di un programma, è possibile costruire la propria matrice di trasformazione in AutoLISP utilizzando **trans** una volta per trasformare ogni "vettore di base" (0 0 0), (1 0 0), (0 1 0) e (0 0 1). La scrittura di funzioni di moltiplicazione di matrici in AutoLISP può essere difficile; potrebbe pertanto non essere conveniente, a meno che nel programma in uso non vengano eseguite molte trasformazioni.

Capitolo 8 -- Funzioni di utilità generali

Gestione di file

AutoLISP fornisce funzioni per la gestione di file e di I/O di dati. Tali funzioni sono le seguenti:

close	help	read-line	write-line
findfile	open	setfunhelp	
getfiled	read-char	write-char	

In questa sezione vengono forniti esempi delle funzioni **findfile**, **getfiled** ed **help**. Le altre funzioni per la gestione dei file sono descritte nel capitolo 13.

Capitolo 8 -- Funzioni di utilità generali

Gestione di file

Ricerca di file

Un'applicazione può utilizzare la funzione **findfile** per ricercare un determinato nome file. L'applicazione può specificare la directory in cui eseguire la ricerca o può utilizzare il percorso di libreria AutoCAD corrente.

Nella seguente parte di codice, la funzione **findfile** ricerca il nome file richiesto in base al percorso di libreria AutoCAD:

```
(setq refname "refc.dwg")
(setq fil (findfile refname))
(if fil
  (setq refname fil)
  (princ (strcat "\nImpossibile trovare il file " nomerif ". " ))
)
```

Se la chiamata alla funzione **findfile** viene eseguita regolarmente, la variabile **refname** viene impostata su una stringa contenente il percorso completo, ad esempio:

```
"/home/work/ref/refc.dwg"
```

Nota Quando si specifica un nome di percorso DOS, affinché tale nome venga riconosciuto da AutoLISP, è necessario specificare prima di una barra rovesciata (<:df

La funzione **getfiled** visualizza una finestra di dialogo contenente un elenco di file disponibili, con il tipo di estensione specificata, nella directory indicata. In tal modo, le routine AutoLISP possono accedere alla finestra di dialogo di AutoCAD per la ricerca dei file.

Una chiamata alla funzione **getfiled** utilizza quattro argomenti che determinano l'aspetto e la funzionalità della finestra di dialogo. L'applicazione deve specificare i seguenti valori di stringa, ognuno dei quali può essere nil: un titolo, posizionato nella parte superiore della finestra; un nome di default, visualizzato nella casella di modifica nella parte inferiore della finestra ed un tipo di estensione, che determina i file iniziali forniti per la selezione nella casella di riepilogo. L'argomento finale è un valore intero che specifica come la finestra di dialogo interagisce con i file selezionati.

Questa semplice routine utilizza la funzione **getfiled** per consentire all'utente di visualizzare la propria struttura di directory e selezionare un file:

```
(defun C:DDIR ( )
  (setq dfil (getfiled "Elenco directory" "" "" 2))
  (princ (strcat "\nVariabile 'dfil' impostata sul file " dfil ))
  (princ)
)
```

Questo è un comando particolarmente utile. La variabile **dfil** viene impostata sul file selezionato, che può essere utilizzato da altre funzioni AutoLISP o come risposta ad un messaggio di richiesta della riga di comando per un nome file. Per utilizzare questa variabile in risposta ad un messaggio di richiesta della riga di comando, digitare **!dfil**.

Nota Non è possibile utilizzare **!dfil** in una finestra di dialogo. Questa variabile è valida solo se specificata dalla riga di comando.

Per ulteriori informazioni, vedere "**getfiled**" nel capitolo 13.

Capitolo 8 -- Funzioni di utilità generali

Gestione di file

Accesso ai file della Guida

La funzione **help** consente l'accesso ai file della guida di AutoCAD (.ahp) e di Windows (.hlp). In base all'estensione del file della guida, la funzione **help** richiama il programma di visualizzazione della guida di AutoCAD o di Windows con il file specificato. È possibile utilizzare questa funzione per aggiungere una funzione di guida alle proprie applicazioni. La parte di codice di seguito riportata richiama il file della guida di AutoCAD di default e fornisce informazioni sul comando LINEA.

```
(help "" "linea")
```

È possibile creare un file della guida che fornisca informazioni relative alle specifiche applicazioni o procedure utilizzate. Il comando seguente definito dall'utente visualizza il file della guida *piùaiuto.ahp*.

```
(defun C:MIOAIUTO ( )
  (setq hlpfil "piùaiuto.ahp")
  (princ)
)
```

Vedere "Personalizzazione della documentazione in linea" per informazioni relative alla creazione e modifica dei file della guida.

La funzione **setfunhelp** fornisce la guida per i comandi definiti dall'utente. Dopo aver definito il nuovo comando, aggiungendo una chiamata a **setfunhelp** vengono associati al comando un file ed un argomento specifici della guida. Il seguente esempio assegna all'argomento MIOCOM nel file *piùaiuto.ahp* il comando definito dall'utente MIOCOM.

```
(defun C:MIOCOM ( )
  .
  . Definizione del comando
  .
)
(setfunhelp c:miocom "piùaiuto.ahp" "miocom")
```

Capitolo 8 -- Funzioni di utilità generali

Controllo ed accesso a dispositivi

In AutoLISP sono disponibili funzioni che consentono di accedere ai dati da diversi dispositivi di input. Le funzioni per il controllo e l'accesso ai dispositivi sono le seguenti:

grrread tablet

Le funzioni per la gestione dei file di seguito riportate possono, inoltre, leggere l'input dal buffer della tastiera. Per ulteriori informazioni su queste funzioni, vedere "Gestione di file."

read-char read-line

Capitolo 8 -- Funzioni di utilità generali

Controllo ed accesso a dispositivi

Accesso all'input dell'utente

La funzione **gread** restituisce l'input dell'utente "grezzo" immesso mediante la tastiera o il dispositivo di puntamento (mouse o digitalizzatore); se la chiamata a **gread** attiva la tracciatura, la funzione restituisce una coordinata digitalizzata che può essere utilizzata, ad esempio, per il trascinamento.

Nota Non esiste alcuna garanzia che le applicazioni che richiamano la funzione **gread** siano compatibili con release successive. Poiché la compatibilità dipende dalla configurazione hardware corrente, le applicazioni che richiamano **gread** non funzionano in modo uguale su tutte le configurazioni.

Capitolo 8 -- Funzioni di utilità generali

Controllo ed accesso a dispositivi

Graduazione delle tavolette

Gli utenti di AutoCAD possono calibrare una tavoletta di digitalizzazione utilizzando il comando TAVOLET. Vedere "TABLET" nel *Guida di riferimento dei comandi di AutoCAD*. La funzione **tablet** attiva le applicazioni per la gestione della graduazione impostando le graduazioni direttamente e salvando tali impostazioni per uso futuro.

Il primo argomento per la funzione **tablet** è un *codice* a numero intero. Se il *codice* è uguale a 0, la funzione restituisce la graduazione corrente. Se il *codice* è uguale a 1, la graduazione viene impostata in base ai restanti argomenti. Le graduazioni vengono espresse come quattro punti tridimensionali (oltre al *codice*). I primi tre punti (*riga1*, *riga2* e *riga3*) sono tre righe della matrice di trasformazione della tavoletta. Il quarto punto, la *direzione*, è un vettore perpendicolare al piano sul quale si presume si trovi la superficie della tavoletta espresso nel Sistema di Coordinate Globali (WCS). Quando la graduazione viene impostata con il comando TAVOLET, si presuppone che la superficie della tavoletta si trovi sul piano XY dell'UCS corrente.

Nota La variabile di sistema TABMODE controlla se la modalità tavoletta è attivata (1) o disattivata (0). È possibile controllare tale modalità utilizzando la funzione **setvar**.

La seguente routine di esempio richiama la graduazione corrente della tavoletta e la memorizza nella variabile tcal.

```
(defun C:TABGET ( )
  (setq tcal (tablet 0))
  (if tcal
    (princ
      (strcat "\nConfigurazione salvata, "
        "usare TABSET per la graduazione.")
    )
    (princ "\nGraduazione non ottenibile ")
  )
  (princ)
)
```

Se la routine precedente è terminata con esito positivo, il simbolo tcal contiene ora l'elenco restituito dalla funzione relativa alla tavoletta. Di seguito viene riportato un esempio dell'elenco restituito:

```
(1 (0.00561717 -0.000978942 -7.5171)
  (0.000978942 0.00561717 -9.17308)
  (0.0 0.0 1.0)
  (0.0 0.0 1.0)
)
```

Per ripristinare i valori richiamati dalla routine precedente per la graduazione, è possibile utilizzare la routine **C:TABSET**.

```
(defun C:TABSET ( )
  (if (not (apply 'tablet tcal))
    (princ "\nImpossibile reimpostare la graduazione. ")
    (progn
      (princ "\nGraduazione Tavolet reimpostata. ")
      (setvar "tabmode" 1)
    )
  )
)
```

```

    (if (= (getvar "tabmode") 0)
      (princ "\nImpossibile attivare la modalità Tavolet ")
    )
  )
)
(princ)
)

```

La matrice di trasformazione passata come *riga1*, *riga2* e *riga3* è una matrice di trasformazione 3×3 destinata a trasformare un punto bidimensionale. Tale punto viene espresso come un vettore di colonna in *coordinate omogenee* (aggiungendo 1.0 come terzo elemento), in modo che la trasformazione sia la seguente:

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1.0 \end{bmatrix} = \begin{bmatrix} M_{00} & M_{01} & M_{02} & M_{03} \\ M_{10} & M_{11} & M_{12} & M_{13} \\ M_{20} & M_{21} & M_{22} & M_{23} \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1.0 \end{bmatrix}$$

Il calcolo di un punto è simile al caso tridimensionale. AutoCAD trasforma il punto utilizzando le seguenti formule:

$$\begin{aligned} X' &= M_{00}X + M_{01}Y + M_{02} \\ Y' &= M_{10}X + M_{11}Y + M_{12} \\ D' &= M_{20}X + M_{21}Y + 1.0 \end{aligned}$$

Per trasformare nuovamente il vettore risultante in un punto bidimensionale, i primi due componenti vengono divisi per il terzo (il fattore di scala D'), ottenendo il punto (X'/D', Y'/D').

Per trasformazioni di proiezione, particolarmente frequenti, **tablet** esegue l'intero calcolo. Tuttavia, per trasformazioni affini ed ortogonali, M20 e M21 sono entrambi 0, in modo che D' sia 1.0. Il calcolo di D' e la divisione vengono omissi; il punto bidimensionale risultante è semplicemente (X', Y').

Come si può dedurre dal paragrafo precedente, una trasformazione affine è un caso speciale ed uniforme di una trasformazione di proiezione. Una trasformazione ortogonale è un caso speciale di una trasformazione affine: non solo M20 e M21 sono pari a zero, ma M00 = M11 e M10 = -M01.

Nota Quando si imposta una graduazione, se la *direzione* non è normalizzata, l'elenco restituito *non* è uguale all'elenco fornito; AutoCAD normalizza il vettore di direzione prima di restituirlo. Inoltre, assicura che il terzo elemento nella terza colonna (*riga3[Z]*) sia uguale a 1. Questa situazione non si verifica se la graduazione viene impostata utilizzando i valori richiamati da AutoCAD mediante **tablet**, ma può presentarsi se il programma in uso calcola automaticamente la trasformazione.

Capitolo 8 -- Funzioni di utilità generali

Interfaccia ASE AutoLISP

Oltre a fornire l'accesso diretto al gruppo di comandi ASE dalle applicazioni AutoLISP (vedere il capitolo 14, "Accesso ai comandi definiti esternamente e alle variabili di sistema"), AutoLISP consente un accesso interno alla maggior parte dei comandi ASE. Inoltre, fornisce alle applicazioni AutoLISP l'interfaccia al modulo eseguibile ASE. L'interfaccia di programma ASE AutoLISP fornisce alle applicazioni AutoLISP diverse funzionalità. L'interfaccia AutoLISP per ASE è descritta nella *ASE Developer's Guide*.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Panoramica

La maggior parte delle funzioni AutoLISP che gestiscono i gruppi di selezione e gli oggetti identificano un gruppo o un oggetto mediante il suo *nome entità*. Prima di poter gestire un gruppo di selezione o un oggetto, un'applicazione AutoLISP deve ottenere il

nome corrente richiamando una delle funzioni che restituiscono il nome di un gruppo di selezione o di un'entità.

Per i gruppi di selezione, che sono validi solo nella sessione corrente, la possibilità di cambiare i nomi non causa alcun problema, mentre diventa problematica per gli oggetti, in quanto questi ultimi vengono salvati nel database dei disegni. Un'applicazione che deve far riferimento alle stesse entità nello stesso disegno (o negli stessi disegni) in momenti differenti può utilizzare i *gestori* di oggetti.

AutoLISP utilizza le tabelle di simboli per mantenere elenchi di dati grafici e non grafici riguardanti un disegno, come layer, tipi di linea e definizioni di blocchi. Ad ogni voce delle tabelle di simboli è associato un nome entità e un gestore; le voci possono inoltre essere gestite in un modo simile a quello in cui vengono gestite le altre entità AutoCAD.

Argomenti di questo capitolo

{button ,JI(','Selection_Set_Handling_al_u0303')} Gestione dei gruppi di selezione

{ewc ,JI(','Object_Handling_al_u0303')} Gestione degli oggetti

{button ,JI(','Extended_Data4545Xdata_al_u0303')} Dati estesi

{ewc ,JI(','Xrecord_Objects_al_u0303')} Oggetti xrecord

{button ,JI(','Symbol_Table_and_Dictionary_Access_al_u0303')} Accesso alle tabelle di simboli e ai dizionari

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione dei gruppi di selezione

AutoLISP fornisce le funzioni per gestire i gruppi di selezione. Di seguito sono riportate le funzioni per tale gestione.

ssadd	ssget	ssmemb
ssdel	sslenght	ssname

La funzione **ssget** fornisce i mezzi più generali per la creazione di un gruppo di selezione. La creazione di un gruppo di selezione può essere eseguita in uno dei modi riportati di seguito.

- Specificando esplicitamente gli oggetti da selezionare utilizzando le opzioni Ultimo, Precedente, Finestra, Implicito, Poligono finestra, Interseca, Poligono interseca o Intercetta.
- Specificando un singolo punto.
- Selezionando l'intero database.
- Richiedendo all'utente di selezionare gli oggetti.

Con una qualsiasi di queste opzioni, è possibile utilizzare il *filtraggio* per specificare un elenco di attributi e condizioni che gli oggetti selezionati devono rispettare.

Nota I nomi dei gruppi di selezione e delle entità sono temporanei, vale a dire che sono validi solo per la sessione di disegno corrente.

Il primo argomento di **ssget** è una stringa che descrive quale opzione di selezione utilizzare. I due argomenti successivi, *pt1* e *pt2*, specificano i valori dei punti per le opzioni attinenti (se non sono validi per l'opzione devono essere omissi). Un elenco di punti, *elenco-pt*, deve essere fornito come argomento per i metodi di selezione che consentono la selezione per mezzo di poligoni (cioè Intercetta, Poligono interseca e Poligono finestra). L'ultimo argomento, *elenco-filtri*, è opzionale. Se viene fornito un *elenco-filtri*, esso specifica l'elenco dei valori dei campi dell'entità usati nel filtraggio. I filtri di selezione vengono descritti dettagliatamente in "Elenchi dei filtri dei gruppi di selezione." Nella tabella che segue vengono riassunti i valori di modalità disponibili e gli argomenti usati con ognuno di essi. Un *elenco-filtri* può essere utilizzato come argomento aggiuntivo per tutte le modalità di selezione riportate nell'elenco.

Opzioni di selezione per ssget

Modalità	Metodo di selezione	Prototipi
nessuna	Selezione utente o punto singolo (se è specificato <i>pt1</i>).	(ssget) o (ssget <i>pt1</i>)
"U"	Ultimo oggetto creato visibile sullo schermo.	(ssget "U")

Nota Se viene specificata la *modalità* "X", ma non viene fornito alcun *elenco-filtri*, **ssget** seleziona *tutte* le entità presenti nel database, comprese quelle sui layer che sono disattivate, congelate o al di fuori della parte visibile dello schermo.

Il codice riportato di seguito mostra le chiamate alla funzione **ssget**.

<code>(setq pt1 '(0.0 0.0 0.0)</code>	<i>Imposta pt1, pt2, pt3 e pt4 come valori di punto</i>
<code>pt2 '(5.0 5.0 0.0)</code>	
<code>pt3 '(4.0 1.0 0.0)</code>	
<code>pt4 '(2.0 6.0 0.0)</code>	
<code>)</code>	
<code>(setq ssl (ssget))</code>	<i>Chiede all'utente di selezionare un oggetto generale e pone tali voci in un gruppo di selezione</i>
<code>(setq ssl (ssget "P"))</code>	<i>Crea un gruppo di selezione con gli oggetti oggetti selezionati.</i>
<code>(setq ssl (ssget "U"))</code>	<i>Crea un gruppo di selezione dell'ultimo oggetto aggiunto al database che è visibile sullo schermo</i>
<code>(setq ssl (ssget pt2))</code>	<i>Crea un gruppo di selezione di un oggetto che passa per il punto (5,5)</i>
<code>(setq ssl (ssget "F" pt1 pt2))</code>	<i>Crea un gruppo di selezione degli oggetti all'interno della finestra da (0,0) a (5,5)</i>
<code>(setq ssl (ssget "IN" (list pt2 pt3 pt4)))</code>	<i>Crea un gruppo di selezione degli oggetti che intersecano la delimitazione e definiti da punti (5,5), (4,1) e (2,6)</i>
<code>(setq ssl (ssget "FP" (list pt1 pt2 pt3)))</code>	<i>Crea un gruppo di selezione degli oggetti all'interno del poligono definito dai punti (0,0), (5,5) e (4,1)</i>
<code>(setq ssl (ssget "X"))</code>	<i>Crea un gruppo di selezione di tutti gli oggetti all'interno del database</i>

Dopo che un'applicazione ha terminato l'uso di un gruppo di selezione, è importante togliere tale gruppo dalla memoria. A tal fine, impostarlo su nil.

```
(setq ssl nil)
```

Nota Si consiglia di *non* cercare di gestire contemporaneamente un numero elevato di gruppi di selezione. Per un'applicazione AutoLISP non possono essere aperti più di 128 gruppi di selezione nello stesso momento. Nel sistema in uso questo limite potrebbe essere ancora più basso. Una volta raggiunto il limite, AutoCAD rifiuta la creazione di ulteriori gruppi di selezione. Tenere aperto contemporaneamente un numero minimo di gruppi e, appena possibile, impostare su nil i gruppi di selezione non necessari. Se è stato raggiunto il numero massimo di gruppi, è necessario chiamare **gc** (vedere "Spazio per i nodi" per informazioni sul riordino) prima di poter utilizzare un'altra funzione **ssget**.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione dei gruppi di selezione

Elenchi dei filtri dei gruppi di selezione

Un elenco di filtri di entità è un elenco di associazioni che utilizza i codici di gruppo DXF nello stesso formato dell'elenco restituito da **entget**. Per un elenco dei codici di gruppo, vedere l'appendice C "Codici di gruppo DXF". La funzione **ssget** riconosce tutti i codici di gruppo, ad eccezione dei nomi di entità (gruppo -1), dei gestori (gruppo 5) e dei codici di dati estesi (gruppi maggiori di 1000). Se in un *elenco-filtri* viene utilizzato un codice di gruppo non valido, **ssget** lo ignora. Per ricercare gli oggetti con dati estesi, usare il codice -3 come descritto in "Filtraggio di dati estesi."

Quando come ultimo argomento per **ssget** viene fornito un *elenco-filtri*, la funzione analizza gli oggetti selezionati e crea un gruppo di selezione contenente i nomi di tutte le entità principali corrispondenti ai criteri specificati. Ad esempio, è possibile ottenere un gruppo di selezione che includa tutti gli oggetti di un dato tipo, su un dato layer o di un determinato colore.

L'*elenco-filtri* specifica quale o quali proprietà delle entità devono essere controllate e quali valori costituiscono una corrispondenza.

Gli esempi riportati di seguito mostrano i metodi riguardanti l'uso di un *elenco-filtri* con varie opzioni di selezione della *modalità*.

<code>(setq ssl (ssget</code>	<i>Chiede all'utente la selezione generale degli oggetti</i>
<code>'((0 . "TEXT"))</code>	<i>ma aggiunge gli oggetti di testo solo al gruppo di selezione</i>
<code>)</code>	
<code>(setq ssl (ssget "P"</code>	<i>Crea un gruppo di selezione degli ultimi oggetti</i>
<code>'((0 . "LINE"))</code>	<i>selezionati che sono anche oggetti linea</i>
<code>)</code>	
<code>(setq ssl (ssget "W" pt1 pt2</code>	<i>Crea un gruppo di selezione di tutti gli oggetti nella</i>

```
'((8 . "PIANO9"))          finestra che sono anche sul layer PIANO9
)
(setq ssl (ssget "X"        Crea un gruppo di selezione di tutti gli oggetti nel
'((0 . "CIRCLE")))        database che sono oggetti cerchio
)
```

Se si conoscono sia il codice che il valore desiderato, l'elenco può essere racchiuso tra virgolette, come mostrato precedentemente. Se viene specificato da una variabile, l'elenco deve essere costruito (mediante le funzioni **list** e **cons**).

```
(setq lay_name "PIANO3")
(setq ssl
  (ssget "X"              Crea un gruppo di selezione di tutti gli oggetti
    (list (cons 8 lay_name)) del database che sono sul layer PIANO3
  )
)
```

Se l'*elenco-filtri* specifica più proprietà, un'entità viene inclusa nel gruppo di selezione solo se corrisponde a *tutte* le condizioni specificate, come nell'esempio che viene riportato di seguito.

```
(ssget "X" (list (cons 0 "CIRCLE") (cons 8 lay_name) (cons 62 1)))
```

Questo codice seleziona solo gli oggetti cerchio sul layer PIANO3 che sono di colore rosso. Questo tipo di verifica esegue un'operazione AND booleana. Verifiche aggiuntive riguardanti le proprietà degli oggetti vengono descritte in "Raggruppamento logico delle verifiche dei filtri."

La funzione **ssget** filtra un disegno analizzando le entità selezionate e comparando i campi di ogni entità principale con l'elenco di filtraggio specificato. Se le proprietà di un'entità corrispondono a *tutti* i campi specificati nell'elenco di filtraggio, l'entità viene inclusa nel gruppo di selezione restituito, altrimenti, non viene inclusa. Se nessuna delle entità selezionate corrisponde ai criteri di filtraggio selezionati, la funzione **ssget** restituisce nil.

Nota Il significato di certi codici di gruppo può essere differente da un'entità all'altra e non tutti i codici di gruppo risultano presenti in tutte le entità. Se un particolare codice di gruppo viene specificato in un filtro, le entità che non contengono tale codice vengono escluse dal gruppo di selezione restituito da **ssget**.

Quando la funzione **ssget** filtra un disegno, il gruppo di selezione richiamato potrebbe includere le entità sia dello spazio carta che dello spazio modello. Tuttavia, quando il gruppo di selezione viene passato ad un comando AutoCAD, vengono utilizzate solo le entità dello spazio correntemente attivo. Lo spazio a cui un'entità appartiene viene specificato dal valore del suo gruppo 67, come descritto nell'appendice C "Codici di gruppo DXF"..

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione dei gruppi di selezione

Elenchi dei filtri dei gruppi di selezione

Modelli di caratteri jolly negli elenchi dei filtri

I nomi dei simboli specificati negli elenchi di filtraggio possono includere modelli di caratteri jolly. I modelli riconosciuti da **ssget** sono gli stessi riconosciuti dalla funzione **wcmatch** e sono descritti nella sezione "Corrispondenza con caratteri jolly" e nella descrizione di "**wcmatch**" nel capitolo 13.

Nota Quando si esegue il filtraggio di blocchi anonimi, è necessario introdurre il carattere * con un apice inverso (`) in quanto il carattere * viene letto da **ssget** come carattere jolly. Ad esempio, è possibile richiamare un blocco anonimo denominato *U2 con il codice riportato di seguito.

```
(ssget "X" '((2 . "`*U2")))
```

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione dei gruppi di selezione

Elenchi dei filtri dei gruppi di selezione

Filtraggio di dati estesi

Usando l'*elenco-filtri* di **ssget**, è possibile selezionare per una particolare applicazione tutte le entità contenenti dati estesi (vedere "Dati estesi"). A tal fine, utilizzare il codice di gruppo -3 come mostrato nell'esempio riportato di seguito.

```
(ssget "X" '((0 . "CIRCLE") (-3 ("NOMEAPP"))))
```

Mediante questo codice vengono selezionati tutti i cerchi che includono i dati estesi per l'applicazione "NOMEAPP". Se nell'elenco del gruppo -3 vengono inclusi più nomi di applicazioni, verrà eseguita implicitamente un'operazione AND e l'entità dovrà contenere i dati estesi per tutte le applicazioni specificate. Quindi, l'istruzione riportata di seguito causerebbe la selezione di tutti i cerchi con dati estesi sia per l'applicazione "APP1" che per l'applicazione "APP2".

```
(ssget "X" '((0 . "CIRCLE") (-3 ("APP1") ("APP2"))))
```

Poiché è consentita la corrispondenza con i caratteri jolly, entrambe le istruzioni riportate di seguito determinano la selezione di tutti i cerchi con dati estesi per una di tali applicazioni o per entrambe.

```
(ssget "X" '((0 . "CIRCLE") (-3 ("APP[12]"))))
(ssget "X" '((0 . "CIRCLE") (-3 ("APP1,APP2"))))
```

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione dei gruppi di selezione

Elenchi dei filtri dei gruppi di selezione

Verifiche relazionali

A meno che non sia stato specificato diversamente, per ogni voce presente nell'*elenco-filtri* è implicita un'equivalenza. Per i gruppi numerici (valori interi, reali, punti e vettori), è possibile specificare altre relazioni includendo un codice di gruppo speciale -4 che specifica un operatore relazionale. Il valore del gruppo -4 è una stringa che indica l'operatore di verifica da applicare al gruppo successivo presente nell'*elenco-filtri*. Per maggiori informazioni, vedere "Verifiche relazionali."

Con il codice riportato di seguito vengono selezionati tutti i cerchi con raggio (codice di gruppo 40) maggiore o uguale a 2.0.

```
(ssget "X" '((0 . "CIRCLE") (-4 . ">=") (40 . 2.0)))
```

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione dei gruppi di selezione

Elenchi dei filtri dei gruppi di selezione

Raggruppamento logico delle verifiche dei filtri

È possibile eseguire la verifica di gruppi anche creando espressioni booleane nidificate che utilizzino gli operatori di raggruppamento logici mostrati in "Raggruppamento logico delle verifiche dei filtri." Tali operatori vengono specificati dai gruppi -4, come gli operatori relazionali. Essi vengono accoppiati e devono essere correttamente bilanciati nell'elenco dei filtri, altrimenti la chiamata **ssget** non avrà esito positivo. Di seguito, è riportato un esempio degli operatori di raggruppamento in un elenco di filtri.

```
(ssget "X"
' (
  (-4 . "<"OR")
  (-4 . "<AND")
  (0 . "CIRCLE")
  (40 . 1.0)
```

```

(-4 . "AND>")
(-4 . "<AND")
(0 . "LINE")
(8 . "ABC")
(-4 . "AND>")
(-4 . "OR>")
)
)

```

Con questo codice vengono selezionati tutti i cerchi con raggio 1.0 e tutte le linee sul layer "ABC". Per gli operatori di raggruppamento, è possibile utilizzare indifferentemente sia lettere maiuscole che minuscole.

Gli operatori di raggruppamento non possono essere utilizzati all'interno del gruppo -3. I nomi di applicazioni multiple specificati nel gruppo -3 utilizzano un operatore AND implicito. Se si desidera verificare i dati estesi utilizzando altri operatori di raggruppamento, specificare gruppi -3 separati e raggrupparli nel modo desiderato. Per selezionare per l'applicazione "APP1" o "APP2", ma non per entrambe, tutti i cerchi aventi dati estesi, digitare quanto riportato di seguito.

```

(ssget "X"
'((0 . "CIRCLE")
(-4 . "<XOR")
(-3 ("APP1"))
(-3 ("APP2"))
(-4 . "XOR>")
)
)

```

È possibile semplificare la codifica degli operatori di raggruppamento utilizzati in modo frequente impostandoli uguali ad un simbolo. L'esempio precedente potrebbe essere riscritto nel modo riportato di seguito. Notare che in questo esempio è necessario racchiudere esplicitamente tra virgolette ogni elenco.

```

(setq <xor '(-4 . "<XOR")
xor> '(-4 . "XOR>") )
(ssget "X"
(list
' (0 . "CIRCLE")
<xor
' (-3 ("APP1"))
' (-3 ("APP2"))
xor>
)
)

```

Come si può vedere, questo metodo può non essere valido per piccole parti di codice, ma può essere utile in applicazioni di dimensioni maggiori.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione dei gruppi di selezione

Elenchi dei filtri dei gruppi di selezione

Gestione dei gruppi di selezione

Dopo che un gruppo di selezione è stato creato, è possibile aggiungervi entità o eliminare da esso entità già esistenti mediante le funzioni **ssadd** e **ssdel**. Per creare un nuovo gruppo di selezione, è possibile utilizzare la funzione **ssadd**, come mostrato nell'esempio che segue. La parte di codice riportata di seguito crea un gruppo di selezione che include la prima e l'ultima entità del disegno corrente. Le funzioni **entnext** e **entlast** vengono descritte più avanti in questo capitolo.

```

(setq fname (entnext))           Prende la prima entità del disegno.
(setq lname (entlast))          Prende l'ultima entità del disegno.
(if (not fname)
(princ "\nNessuna entità nel disegno. ")
(progn
(setq ourset (ssadd fname))      Crea un gruppo di selezione della prima entità.
(ssadd lname ourset)           Aggiunge l'ultima entità al gruppo di selezione.
)
)

```

```
)
)
```

L'esempio viene eseguito correttamente anche se nel database è presente una sola entità; in questo caso, sia **entnext** che **entlast** impostano i loro argomenti con lo stesso nome di entità. Se la funzione **ssadd** riceve il nome di un'entità già presente nel gruppo di selezione, ignora la richiesta e *non* restituisce alcun errore. La funzione che segue elimina la prima entità dal gruppo di selezione creato nell'esempio precedente.

```
(ssdel fname ourset)
```

Se nel disegno sono presenti più entità, cioè se `fname` e `lname` non sono uguali, il gruppo di selezione `ourset` contiene solo `lname`, corrispondente all'ultima entità del disegno.

La funzione **sslength** restituisce il numero di entità presenti in un gruppo di selezione e **ssmemb** verifica se una particolare entità fa parte di un gruppo di selezione. Infine, la funzione **ssname** restituisce il nome di una particolare entità di un gruppo di selezione, utilizzando un indice (le entità di un gruppo sono numerate a partire da 0).

Nel codice che segue vengono mostrate le chiamate per **ssname**.

```
(setq sset (ssget))           Crea il gruppo di selezione (con richiesta all'utente).
(setq ent1 (ssname sset 0))   Prende il nome della prima entità in sset.
(setq ent4 (ssname sset 3))   Prende il nome della quarta entità in sset.
(if (not ent4)
  (princ "\nSelezionare almeno quattro entità. ")
)
(setq ilast (sslength sset))   Trova l'indice dell'ultima entità in sset.
                               Prende il nome dell'ultima entità in sset.
(setq lastent (ssname sset (1- ilast)))
```

Indipendentemente dal modo in cui le entità sono state aggiunte ad un gruppo di selezione, quest'ultimo non contiene *mai* entità duplicate. Se la stessa entità viene aggiunta più volte, tutte le aggiunte successive alla prima entità vengono ignorate. Quindi, **sslength** restituisce in modo accurato il numero di entità *distinte* presenti nel gruppo di selezione specificato.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione dei gruppi di selezione

Passaggio di gruppi di selezione tra applicazioni AutoLISP e ADSRX

Quando si passano dei gruppi di selezione tra applicazioni AutoLISP e ADSRX, tenere presente quanto segue:

Se un gruppo di selezione viene creato in AutoLISP, memorizzato in una variabile AutoLISP e quindi sovrascritto con un valore restituito da un'applicazione ADSRX, il gruppo di selezione originale può essere ripulito. Verrà pertanto eliminato alla successiva ripulitura automatica o esplicita.

Questo avviene anche se il valore restituito dall'applicazione ADSRX era il gruppo di selezione originale. Nell'esempio che segue, se la funzione ADSRX **adsfunc** ritorna lo stesso gruppo di selezione che ha precedentemente ricevuto come argomento, questo gruppo di selezione può essere ripulito pur essendo ancora assegnato alla stessa variabile.

```
(setq var1 (ssget))
(setq var1 (adsfunc var1))
```

Per proteggere dalla ripulitura il gruppo di selezione originale, *non* assegnare il valore restituito dall'applicazione ADSRX alla stessa variabile AutoLISP che fa già riferimento al gruppo di selezione. Modificando l'esempio precedente nel modo indicato qui di seguito si impedisce che il gruppo di selezione di riferimento di `var1` venga ripulito:

```
(setq var1 (ssget))
(setq var2 (adsfunc var1))
```

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione degli oggetti

AutoLISP fornisce funzioni per la gestione degli oggetti. Di seguito sono riportate le funzioni di gestione degli oggetti.

entdel	entlast	entmod	entupd
entget	entmake	entnext	handent

Le funzioni per la gestione degli oggetti sono organizzate in due categorie: quelle che richiamano il nome entità di un particolare oggetto e quelle che richiamano o modificano i dati degli oggetti.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione degli oggetti

Funzioni riguardanti i nomi delle entità

Per operare su un oggetto, un'applicazione AutoLISP deve conoscere il suo nome entità per utilizzarlo nelle chiamate successive riguardanti i dati dell'entità stessa o nelle funzioni del gruppo di selezione. Due delle funzioni descritte in questa sezione, **entsel** e **nentsel**, oltre a restituire il nome dell'entità restituiscono informazioni aggiuntive per l'uso dell'applicazione.

Entrambe le funzioni richiedono all'utente AutoCAD di scegliere interattivamente un oggetto selezionando un punto sullo schermo di disegno. Tutte le altre funzioni riguardanti il nome entità possono richiamare un'entità anche se questa non è visibile sullo schermo o se si trova su un layer congelato. La funzione **entsel** richiede all'utente di scegliere un oggetto selezionando un punto sullo schermo di disegno e la funzione **nentsel** restituisce sia il nome dell'entità che il valore del punto selezionato. Alcune operazioni riguardanti le entità richiedono la conoscenza del punto mediante il quale l'oggetto è stato selezionato; alcuni esempi di comandi AutoCAD esistenti sono SPEZZA, TAGLIA e ESTENDI. La funzione **nentsel** viene descritta dettagliatamente in "Contesto dell'entità e dati per la trasformazione delle coordinate." Queste funzioni accettano le parole chiave se esse sono precedute da una chiamata per la funzione **initget**.

La funzione **entnext** richiama sequenzialmente i nomi delle entità. Se **entnext** viene chiamata senza argomenti, restituisce il nome della prima entità presente nel database dei disegni. Se il suo argomento è il nome di un'entità del disegno corrente, la funzione **entnext** restituisce il nome dell'entità successiva.

La parte di codice riportata di seguito mostra il modo in cui **ssadd** può essere utilizzata insieme a **entnext** per creare gruppi di selezione ed aggiungere membri ad un gruppo già esistente.

```
(setq e1 (entnext))
(if (not e1)                                Imposta e1 come nome della prima entità
    (princ "\nNessuna entità nel disegno. ")
    (progn
      (setq ss (ssadd))                     Imposta ss come gruppo di selezione nullo
      (ssadd e1 ss)                         Restituisce il gruppo di selezione ss con e1 aggiunto
      (setq e2 (entnext e1))                Prende l'entità che segue e1
      (ssadd e2 ss)                         Aggiunge e2 al gruppo di selezione ss
    )
  )
```

La funzione **entlast** richiama il nome dell'ultima entità del database, cioè l'entità principale creata per ultima. Quindi, la funzione **entlast** può essere chiamata per ottenere il nome di un'entità appena creata mediante una chiamata a **command**.

È possibile impostare il nome entità restituito da **entnext** con lo stesso nome di variabile passato a questa funzione. In questo modo, si otterrà il "passaggio" attraverso il database di una singola variabile di entità, come mostrato nell'esempio riportato di seguito.

```
(setq one_ent (entnext)) Prende il nome della prima entità
(while one_ent
  .
  .
  .
  (setq one_ent (entnext one_ent))
) Il valore di one_ent adesso è nil
```

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione degli oggetti

Funzioni riguardanti i nomi delle entità

Gestori di entità e loro uso

La funzione **handent** richiama il nome di un'entità con un *gestore* specifico. Come i nomi di entità, anche i gestori sono univoci nell'ambito di un disegno; tuttavia, il gestore di un'entità è costante per tutta la sua durata. Le applicazioni AutoLISP che gestiscono un database specifico possono usare la funzione **handent** per ottenere il nome corrente dell'entità che devono utilizzare. È possibile ottenere il gestore di un oggetto selezionato usando il comando DDMODIF.

La parte di codice riportata di seguito utilizza la funzione **handent** per ottenere e visualizzare il nome di un'entità.

```
(if (not (setq e1 (handent "5a2")))
  (princ "\nNessuna entità con quel gestore. ")
  (princ e1)
)
```

In una particolare sessione di modifica, la parte di codice sopra riportata potrebbe causare la visualizzazione di quanto riportato di seguito.

<Nome di entità: 60004722>

In un'altra sessione di modifica, con lo stesso disegno, tale parte di codice potrebbe causare la visualizzazione di un numero completamente diverso ma, in entrambi i casi, il codice accedrebbe alla stessa entità.

La funzione **handent**, descritta nella sezione che segue, ha un ulteriore uso. Le entità vengono eliminate dal database solo alla fine del disegno corrente. Ciò significa che **handent** può recuperare i nomi delle entità cancellate, che possono essere successivamente ripristinate nel disegno mediante una seconda chiamata a **entdel**.

Nota I gestori vengono forniti per le definizioni dei blocchi, comprese le sottoentità.

Le entità dei disegni che sono collegate mediante XRIF Attacca non fanno effettivamente parte del disegno corrente; i corrispondenti gestori rimangono immutati ma non è possibile accedervi mediante **handent**. Tuttavia, quando i disegni vengono combinati mediante INSER, INSER *, XRIF Unisci (XUNISCE) o DXFIN parziale, i gestori delle entità nel disegno in arrivo vengono persi e alle entità in arrivo vengono assegnati nuovi valori di gestore affinché ogni gestore del disegno corrente resti univoco.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione degli oggetti

Funzioni riguardanti i nomi delle entità

Contesto dell'entità e dati per la trasformazione delle coordinate

Le funzioni **nentsel** e **nentselp** sono simili a **entsel**, con l'eccezione che restituiscono due valori aggiuntivi per gestire entità nidificate nell'ambito dei riferimenti dei blocchi.

Un'altra differenza tra queste funzioni è che quando l'utente risponde ad una chiamata **nentsel** selezionando un'entità complessa o quando un'entità complessa viene selezionata mediante **nentselp**, queste funzioni restituiscono il nome di entità della sottoentità selezionata e *non* l'intestazione dell'entità complessa, come invece fa **entsel**.

Ad esempio, quando l'utente seleziona una polilinea, invece dell'intestazione di quest'ultima, **nentsel** restituisce una sottoentità vertice. Per richiamare l'intestazione di una polilinea, l'applicazione deve utilizzare **entnext** per arrivare alla sottoentità seqend e dopo ottenere il nome dell'intestazione dal gruppo -2 della sottoentità seqend. Lo stesso è valido quando l'utente seleziona gli attributi in un riferimento di blocco nidificato. È preferibile usare la funzione **nentselp** invece della funzione **nentsel**, in quanto restituisce una matrice di trasformazione dello stesso formato di quella restituita da **grvecs**.

Il primo degli elementi aggiuntivi restituiti da **nentsel** è la Matrice di trasformazione da Modello a Globale, che è un elenco composto da quattro elenchi secondari, ognuno dei quali contiene un gruppo di coordinate. Questa matrice può essere utilizzata per trasformare i punti dei dati di definizione dell'entità da un sistema di coordinate interno, chiamato Sistema di Coordinate di Movimento (SCM) nel sistema WCS. Il punto di inserimento del blocco (valido anche per i riferimenti esterni) contenente l'oggetto selezionato definisce l'origine del sistema SCM. L'orientamento del sistema di coordinate utente quando il blocco viene creato determina la direzione degli assi SCM.

Il secondo elemento aggiuntivo è un elenco contenente il nome entità del blocco in cui si trova l'oggetto selezionato. Se tale oggetto è contenuto in un blocco nidificato (un blocco situato all'interno di un altro blocco), l'elenco riporta anche i nomi entità di tutti i blocchi in cui l'entità selezionata è nidificata, cominciando dal blocco più interno e continuando verso l'esterno fino a che non viene riportato il nome del blocco più esterno inserito nel disegno.

```

(<Nome entità: ename1>      Nome dell'entità
(Px Py Pz)                Punto di selezione
( ( X0 Y0 Z0)              Matrice di trasformazione
  (X1 Y1 Z1)              da Modello a Globale
  (X2 Y2 Z2)
  (X3 Y3 Z3)
)
(<Nome di entità: ename2>  Nome del blocco nidificato più interno
.                           contenente l'oggetto selezionato
.
.
<Nome di entità: ename3>  Nome del blocco più esterno
)                           contenente l'oggetto selezionato

```

Nell'esempio riportato di seguito viene creato un blocco da utilizzare con la funzione **nentsel**.

```

Comando: linea
Dal punto: 1,1
Al punto: 3,1
Al punto: 3,3
Al punto: 1,3
Al punto: c
Comando: blocco
Nome del blocco (o ?) square
Punto base per inserimento: 2,2
Selezionare oggetti: Selezionare le quattro linee appena disegnate.
Selezionare oggetti: RETURN

```

Quindi il blocco viene inserito nel sistema di coordinate utente ruotato di 45 gradi intorno all'asse Z.

```

Comando: ucs
ORigine/Asse-z/3punti/OGgetto/Vista/X/Y/Z/Precedente/Ripristina/Memorizza/Cancella/?/
<Globale>: Z
Angolo di rotazione intorno all'asse Z <0>: 45
Comando: inser
Nome del blocco (o ?) square
Punto di inserimento: 7,0
Fattore di scala X <1> / Angolo / XYZ: RETURN
Fattore di scala per Y (standard=X): RETURN
Angolo di rotazione: RETURN

```

Usare **nentsel** per selezionare il lato inferiore sinistro del quadrato.

```
(setq ndata (nentsel))
```

Questo codice imposta il simbolo ndata uguale ad un elenco simile a quello riportato di seguito.

```

(<Nome entità: 400000a0>      Nome dell'entità
(6.46616 -1.0606 0.0)       Punto di selezione
((0.707107 0.707107 0.0)    Matrice di trasformazione
(-0.707107 0.707107 0.0)   da Modello a Globale
(0.0 -0.0 1.0)
(4.94975 4.94975 0.0)
)
(<Nome entità:6000001c>)    Nome del blocco contenente l'oggetto selezionato
)

```

Dopo aver ottenuto il nome dell'entità e la Matrice di trasformazione da Modello a Globale, è possibile trasformare i punti dei dati di definizione dell'entità da SCM in WCS. Usare **entget** e **assoc** sul nome dell'entità per ottenere i punti di definizione desiderati espressi in coordinate SCM. Quindi, passare i punti ed i dati della Matrice di trasformazione da Modello a Globale (ottenuti nella

prima chiamata a **nentsel**) alle formule riportate di seguito.

Se l'entità selezionata non è un'entità nidificata, la matrice di trasformazione viene impostata su quella di identità.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Le equazioni riportate di seguito mostrano come trasformare un punto o un vettore.

$$X' = XM_{00} + YM_{10} + ZM_{20} + M_{30}$$

$$Y' = XM_{01} + YM_{11} + ZM_{21} + M_{31}$$

$$Z' = XM_{02} + YM_{12} + ZM_{22} + M_{32}$$

M_{ij} , dove $0 \leq i, j \leq 2$, sono le coordinate della Matrice di trasformazione da Modello a Globale; X, Y e Z è il punto dei dati di definizione dell'entità espresso in coordinate SCM, mentre X', Y' e Z' è il punto dei dati di definizione risultante espresso in coordinate WCS.

Nota Per trasformare un *vettore* anziché un punto, non aggiungere il vettore di trasformazione [M30 M31 M32] (dalla quarta colonna della matrice di trasformazione).

Nell'esempio che segue viene mostrato come ottenere il punto iniziale SCM di una linea (codice di gruppo 10) contenuta nella definizione di un blocco. L'istruzione memorizza i dati riguardanti l'entità (usando il nome entità ottenuto precedentemente con la funzione **nentsel**) nel simbolo edata. Restituisce quanto riportato di seguito:

```
(setq edata (assoc 10 (entget (car ndata))))
(10 -1.0 1.0 0.0)
```

L'istruzione riportata di seguito memorizza l'elenco secondario della Matrice di trasformazione da Modello a Globale nel simbolo matrix.

```
(setq matrix (caddr ndata))
```

Restituisce quanto riportato di seguito.

```
((0.707107 0.707107 0.0) Trasformazione X
(-0.707107 0.707107 0.0) Trasformazione Y
(0.0 -0.0 1.0) Trasformazione Z
(4.94975 4.94975 0.0) Spostamento dall'origine WCS
)
```

Applicare la formula di trasformazione per X' per cambiare la coordinata X del punto iniziale della linea da coordinata SCM in coordinata WCS. Memorizzare i risultati nel simbolo answer:

```
(setq answer
(+
(* (car (nth 0 matrix)) (cadr edata)) aggiungere: M00 * X
(* (car (nth 1 matrix)) (caddr edata)) M10 * Y
(* (car (nth 2 matrix)) (caddr edata)) M20 * Z
(car (nth 3 matrix)) M30
)
)
```

Questa istruzione restituisce 3.53553, la coordinata X WCS del punto iniziale della linea selezionata.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione degli oggetti

Funzioni riguardanti i nomi delle entità

Funzioni per l'accesso all'entità

Le funzioni per l'accesso all'entità sono relativamente lente. È più opportuno richiamare una volta il contenuto di una particolare entità (o voce della tabella di simboli) e dopo mantenere tale informazione in memoria, piuttosto che richiedere ripetutamente ad AutoCAD gli stessi dati. Assicurarsi che i dati rimangano validi; se l'utente ha l'opportunità di alterare l'entità o la voce della tabella di simboli, per assicurare la validità dei dati sarà necessario inviare nuovamente la funzione per l'accesso all'entità.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione degli oggetti

Funzioni dei dati dell'entità

Le funzioni descritte in questa sezione operano sui dati dell'entità e possono essere utilizzate per modificare il database del disegno corrente.

La funzione **entdel** consente di cancellare un'entità specificata. L'entità viene effettivamente eliminata dal database solo alla fine della sessione di disegno corrente, quindi, se l'applicazione chiama la funzione **entdel** una seconda volta durante la sessione e specifica la stessa entità, la cancellazione dell'entità viene annullata.

Nota Gli attributi ed i vertici di polilinee *non* possono essere cancellati, indipendentemente dalle entità di livello superiore corrispondenti. La funzione **entdel** funziona solo sulle entità principali. Se è necessario cancellare un attributo o un vertice, è possibile usare **command** per richiamare il comando EDITATT o EDITPL di AutoCAD.

La funzione **entget** restituisce i dati di definizione di un'entità specificata. I dati vengono restituiti come un elenco. Ogni voce dell'elenco viene specificata mediante un codice di gruppo DXF. La prima voce dell'elenco contiene il nome corrente dell'entità.

Nell'esempio che segue, le condizioni riportate (quelle di default) riguardano il disegno corrente.

- Il layer è 0.
- Il tipo di linea è CONTINUOUS.
- L'elevazione è 0.

L'utente ha disegnato una linea con la sequenza di comandi riportata di seguito.

Comando: **linea**

Dal punto: **1,2**

Al punto: **6,6**

Al punto: **RETURN**

Un'applicazione AutoLISP può richiamare e stampare i dati di definizione della linea utilizzando la funzione AutoLISP riportata di seguito.

```
(defun C:PRINTDXF ( )
  (setq ent (entlast))           Imposta per ent l'ultima entità
  (setq entl (entget ent))      Imposta per entl l'elenco associativo
                                dell'ultima entità
  (setq ct 0)                   Imposta ct (un contatore) su 0
  (textpage)                   Passa allo schermo di testo
  (princ "\nRisultati da entget dell'ultima entità:")
  (repeat (length entl)        Ripete per il numero di membri dell'elenco
    (print (nth ct entl))      Stampa una nuova linea e ogni membro dell'elenco
    (setq ct (1+ ct))         Incrementa il contatore di uno
  )
  (princ)                      Esce senza apportare modifiche
)
```

Verrebbe stampato quanto riportato di seguito.

Risultati di entget dell'ultima entità:

```
(-1 . <Nome di entità: 60000014>)
(0 . "LINE")
(8 . "0")
(5 . "2D")
(10 1.0 2.0 0.0)
(11 6.0 6.0 0.0)
(210 0.0 0.0 1.0)
```

Il primo membro dell'elenco (-1) contiene il nome dell'entità rappresentata dall'elenco. La funzione **entmod** descritta in questa sezione lo usa per identificare l'entità da modificare.

I codici dei componenti delle entità sono quelli usati da DXF e descritti nell'appendice C "Codici di gruppo DXF". Come nel caso di DXF, le voci di intestazione dell'entità (il colore, il tipo di linea, lo spessore, il flag che segue gli attributi ed il gestore dell'entità) vengono restituite solo se hanno valori diversi da quelli di default. Contrariamente a quanto avviene nel caso di DXF, invece, i campi di definizione delle entità opzionali vengono restituiti indipendentemente dal fatto che corrispondano o meno ai loro valori di default; le coordinate X, Y e Z associate vengono restituite come singola variabile di punto piuttosto che come gruppi X (10), Y (20) e Z (30) separati.

La funzione **assoc** ricerca in un elenco un gruppo del tipo specificato. Il seguente codice restituisce il tipo di oggetto "LINE" (0) dall'elenco entl.

```
(cdr (assoc 0 entl))
```

Se il codice di gruppo DXF specificato non è presente nell'elenco (o se non è un gruppo DXF valido), **assoc** restituisce nil.

La funzione **entmod** modifica un'entità. Tale funzione fornisce un elenco avente lo stesso formato dell'elenco restituito da **entget**, ma con alcuni valori del gruppo dell'entità eventualmente modificati dall'applicazione. Questa funzione completa la funzione **entget**. Il meccanismo principale mediante il quale un'applicazione AutoLISP aggiorna il database è quello di richiamare un'entità con **entget**, di modificare l'elenco dell'entità e dopo di passare nuovamente l'elenco al database con **entmod**.

La parte di codice riportata di seguito richiama i dati di definizione della prima entità del disegno e cambia la sua proprietà di layer in MIOLAYER.

```
(setq en (entnext))           Imposta per en il nome della prima entità del disegno
(setq ed (entget en))        Imposta per ed i dati dell'entità per il nome di entità en
(setq ed
  (subst (cons 8 "MIOLAYER")
    (assoc 8 ed)             Cambia il gruppo layer in ed
    ed                       nel layer MIOLAYER
  )
)
(entmod ed)                 Modifica il layer en nel disegno
```

Esistono dei limiti per le modifiche al database che **entmod** può apportare; la funzione *non* può modificare quanto segue:

- Il tipo o il gestore dell'entità.
- I campi interni. I campi interni sono i valori che AutoCAD assegna a certi codici di gruppo: -2, riferimento al nome dell'entità; -1, nome dell'entità; 5, gestore dell'entità. Qualsiasi tentativo di modifica di un campo interno, ad esempio, il nome dell'entità principale in una sottoentità seqend (gruppo -2), viene ignorato.
- Le entità della finestra. Se si tenta di modificare l'entità di una finestra si verifica un errore.

Esistono altri limiti quando vengono modificate le quote e i modelli di tratteggio.

AutoCAD deve riconoscere tutti gli oggetti (ad eccezione dei layer) a cui l'elenco delle entità fa riferimento. *Prima* che l'elenco venga passato a **entmod**, è necessario definire nel disegno corrente il nome di ogni stile di testo, tipo di linea, forma o blocco che appare nell'elenco delle entità. Esiste un'eccezione: **entmod** accetta nuovi nomi di layer.

Se l'elenco delle entità fa riferimento ad un nome di layer che non è stato definito nel disegno corrente, **entmod** crea un nuovo layer. Gli attributi del nuovo layer sono i valori di default standard usati dall'opzione Nuovo del comando LAYER di AutoCAD.

La funzione **entmod** può modificare sottoentità quali i vertici di polilinea e gli attributi dei blocchi.

Nota Se si usa **entmod** per modificare un'entità nella definizione di un blocco, verranno influenzati tutti i riferimenti INSER o XRIF per tale blocco. Inoltre, le entità presenti nelle definizioni dei blocchi *non* possono essere cancellate da **entdel**.

Un'applicazione può anche aggiungere un'entità al database del disegno chiamando la funzione **entmake**. Analogamente a **entmod**, l'argomento per **entmake** è un elenco il cui formato è simile a quello restituito da **entget**. La nuova entità descritta dall'elenco viene aggiunta al database dei disegni e diventa l'ultima entità del disegno. Se l'entità è complessa (una polilinea o un blocco), viene aggiunta al database solo dopo essere stata completata.

La parte di codice riportata di seguito crea un cerchio sul layer MIOLAYER.

```
(entmake '( (0 . "CIRCLE")           Tipo di oggetto
            (8 . "MIOLAYER")        Layer
            (10 5.0 7.0 0.0)         Centro
            (40 . 1.0)               Raggio
          ) )
```

I limiti per **entmake** sono simili a quelli di **entmod**:

- Il primo o il secondo membro dell'elenco *deve* specificare il tipo di oggetto. Il tipo deve essere un codice di gruppo DXF valido. Se il primo membro non specifica il tipo, può specificare *solamente* il nome dell'entità: gruppo -1 (il nome non viene salvato nel database).
- AutoCAD *deve* riconoscere tutti gli oggetti (ad eccezione dei layer) a cui l'elenco delle entità fa riferimento. Esiste un'eccezione: **entmake** accetta nuovi nomi di layer.
- Tutti i campi interni passati a **entmake** vengono ignorati.
- La funzione **entmake** *non* può creare entità di finestra.

Sia **entmod** che **entmake** effettuano sui dati dell'entità che vengono loro passati gli stessi controlli di coerenza eseguiti dal comando DXFIN quando legge i file DXF. Questi controlli non hanno esito positivo se non possono creare entità di disegno valide.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione degli oggetti

Funzioni dei dati dell'entità

Uso dei blocchi

Non c'è nessun metodo diretto in cui l'applicazione può controllare se un blocco elencato nella tabella BLOCK è effettivamente un blocco a cui fa riferimento un oggetto insert del disegno. Per cercare occorrenze di un riferimento di blocco nel disegno è possibile utilizzare il seguente codice:

```
(ssget "x" '( (2 . "BLOCKNAME") ) )
```

Occorre inoltre scansionare ciascuna definizione di blocco in caso di blocchi nidificati.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione degli oggetti

Funzioni dei dati dell'entità

Blocchi anonimi

La tabella di definizione dei blocchi di un disegno (BLOCCO) può contenere blocchi anonimi che AutoCAD crea per supportare modelli di tratteggio e quotatura associativa. La funzione **entmake** può creare blocchi anonimi diversi da **Dnnn* (quote) e **Xnnn* (modelli di tratteggio). I blocchi anonimi privi di riferimenti vengono eliminati dalla tabella BLOCCO all'inizio di ogni sessione di disegno. I blocchi anonimi che presentano riferimenti (quelli che sono stati inseriti) *non* vengono eliminati. È possibile utilizzare **entmake** per creare il riferimento di un blocco (inserire un oggetto) in un blocco anonimo. Un blocco anonimo *non* può essere passato al comando INSER. La funzione **entmake** può anche essere usata per ridefinire il blocco. Mediante **entmod** è possibile modificare le entità di un blocco (ma non l'oggetto blocco).

Il formato del nome (gruppo 2) di un blocco anonimo creato da AutoLISP o ARX è **Unnn*, dove *nnn* è un numero generato da AutoCAD. Inoltre, il bit meno significativo di un *flag di tipo blocco* (gruppo 70) riguardante un blocco anonimo è impostato su 1. Quando **entmake** crea un blocco con il nome che inizia con * e con il bit anonimo impostato, AutoCAD considera tale blocco come anonimo e gli assegna un nome. Qualsiasi carattere che segue * nella stringa del nome passata a **entmake** viene ignorato.

Nota I nomi di blocchi anonimi *non* rimangono costanti. Sebbene un blocco anonimo con riferimenti diventi permanente, la parte

numerica del suo nome può variare da una sessione di disegno all'altra.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione degli oggetti

Funzioni dei dati dell'entità

Creazione di entità complesse

Per creare un'entità complessa (una polilinea o un blocco), effettuare più chiamate alla funzione **entmake**, usando una chiamata separata per ogni sottoentità. Quando **entmake** riceve per la prima volta il componente iniziale di un'entità complessa, crea un file temporaneo in cui raccogliere i dati della definizione e gli eventuali dati estesi. Vedere "Dati estesi"). Per ogni chiamata **entmake** successiva, la funzione controlla se il file temporaneo esiste. In caso affermativo, la nuova sottoentità viene aggiunta al file. Una volta completata la definizione dell'entità complessa, cioè quando **entmake** riceve una sottoentità `seqend` o `endblk` appropriata, viene verificata la coerenza dell'entità; se l'entità risulta valida, viene aggiunta al disegno. Quando l'entità complessa è completa, o dopo che la sua creazione è stata annullata, il file viene cancellato.

Nota L'entità non appare nel database del disegno finché a **entmake** non viene passata la sottoentità `seqend` o `endblk` finale. In particolare, **entlast non** può essere usato per richiamare l'ultima sottoentità creata di un'entità complessa che non è stata completata.

Da quanto si può dedurre dai paragrafi precedenti, **entmake** può costruire una sola entità complessa alla volta. Se viene creata un'entità complessa e la funzione **entmake** riceve dati non validi o un'entità che non è una sottoentità appropriata, vengono rifiutate sia l'entità non valida che l'intera entità complessa. È possibile annullare esplicitamente la creazione di un'entità complessa chiamando **entmake** senza argomenti.

L'esempio riportato di seguito contiene cinque funzioni **entmake** che creano una singola entità complessa, una polilinea. La polilinea ha un tipo di linea TRATTEGGIATO ed è di colore blu. ed ha tre vertici che si trovano alle coordinate (1,1,0), (4,6,0) e (3,2,0). Tutti gli altri dati di definizione opzionali assumono i valori di default. Perché l'esempio sia valido, è necessario aver caricato il tipo di linea TRATTEGGIATO.

```
(entmake '((0 . "POLYLINE")           Tipo di oggetto
          (62 . 5)                     Colore
          (6 . "dashed")                Tipo di linea
          (66 . 1)                      Seguono i vertici
        ) )
(entmake '((0 . "VERTEX")             Tipo di oggetto
          (10 1.0 1.0 0.0)              Punto iniziale
        ) )
(entmake '((0 . "VERTEX")             Tipo di oggetto
          (10 4.0 6.0 0.0)              Secondo punto
        ) )
(entmake '((0 . "VERTEX")             Tipo di oggetto
          (10 3.0 2.0 0.0)              Terzo punto
        ) )
(entmake '((0 . "SEQEND"))            Fine della sequenza)
```

Nota Quando si definiscono delle coppie punteggiate, come nell'esempio sopra, *deve* esserci uno spazio prima e dopo il punto, altrimenti viene visualizzato un messaggio di errore che informa della non validità della coppia punteggiata.

Le definizioni dei blocchi iniziano con un'entità blocco e terminano con una sottoentità di fine blocco. Tali definizioni non possono né essere nidificate, né fare riferimento a se stesse, ma *possono* contenere riferimenti per altre definizioni di blocchi.

Nota Prima di usare **entmake** per creare un blocco, usare **tblsearch** per assicurarsi che il nome del nuovo blocco sia univoco. La funzione **entmake** non controlla eventuali conflitti riguardanti i nomi nella tabella di definizione dei blocchi, quindi può definire nuovamente blocchi già esistenti.

I riferimenti dei blocchi possono includere un flag che segue gli attributi (gruppo 66). Se tale flag è presente ed è uguale ad 1, l'oggetto di tipo inserimento dovrebbe essere seguito da una serie di entità attributo (`attrib`). La sequenza di attributi viene conclusa da una sottoentità `seqend`.

Le entità polilinea includono sempre un flag che segue i vertici (gruppo 66). Il flag deve avere 1 come valore ed essere seguito da una sequenza di entità vertice terminate da una sottoentità di fine sequenza (`seqend`).

Le entità complesse possono esistere nello spazio modello o nello spazio carta, ma non in entrambi contemporaneamente. Se lo spazio corrente è stato modificato richiamando SPAZIOM o SPAZIOC (tramite **command**) durante la costruzione di un'entità

complessa, una chiamata successiva a **entmake** annulla l'entità complessa. Ciò può verificarsi anche se la sottoentità ha un gruppo 67 il cui valore non corrisponde al gruppo 67 dell'instestazione dell'entità.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione degli oggetti

Funzioni riguardanti i dati dell'entità e schermo di disegno

Le modifiche al disegno apportate dalle funzioni riguardanti i dati dell'entità si riflettono sullo schermo di disegno, sempre che l'entità cancellata, ripristinata, modificata o creata si trovi in un'area e su un layer correntemente visibili. Esiste un'eccezione: quando **entmod** modifica una sottoentità, non aggiorna l'immagine dell'intera entità (complessa). Se, ad esempio, un'applicazione modifica 100 vertici di una polilinea complessa con 100 chiamate a **entmod**, il tempo richiesto per ricalcolare e visualizzare nuovamente l'intera polilinea è eccessivo. Invece, un'applicazione può eseguire una serie di modifiche riguardanti le sottoentità e visualizzare nuovamente l'intera entità con una singola chiamata alla funzione **entupd**.

Considerare quanto segue: se la prima entità del disegno corrente è una polilinea con più vertici, il codice riportato di seguito modifica il secondo vertice della polilinea e rigenera la sua immagine sullo schermo.

```
(setq e1 (entnext))           Imposta e1 come nome entità della polilinea
(setq v1 (entnext e1))       Imposta v1 come primo vertice
(setq v2 (entnext v1))      Imposta v2 come secondo vertice
(setq v2d (entget v2))      Imposta v2d come dati del vertice
(setq v2d
  (subst
    '(10 1.0 2.0 0.0)
    (assoc 10 v2d)           Cambia la posizione del vertice in v2d
    v2d)                   al punto (1,2,0)
  )
)
(entmod v2d)                Sposta il vertice nel disegno
(entupd e1)                  Rigenera l'entità polilinea e1
```

L'argomento per **entupd** può specificare un'entità principale o una sottoentità. In entrambi i casi, **entupd** rigenera l'intera entità. Sebbene venga principalmente utilizzata per le entità complesse, **entupd** può rigenerare qualsiasi entità presente nel disegno corrente.

Nota Per assicurare che tutte le istanze dei riferimenti dei blocchi vengano aggiornate, è necessario rigenerare il disegno richiamando il comando RIGEN di AutoCAD (mediante **command**). La funzione **entupd** non è sufficiente se l'entità modificata è nella definizione di un blocco.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione degli oggetti

Polilinee e polilinee ottimizzate

L'entità `lwpolyline`, o "polilinea ottimizzata", è una novità della Release 14. Questa entità viene definita nel database dei disegni come singola entità grafica. Una polilinea ottimizzata è differente da una polilinea standard (definita come un gruppo di sottoentità) in quanto viene visualizzata più velocemente, occupa meno spazio su disco ed utilizza meno RAM.

Nella Release 14, le polilinee 3D vengono sempre create come entità polilinee standard. Le polilinee 2D vengono create come entità polilinee ottimizzate, a meno che non siano state curvate o adattate con il comando EDITPL. Quando nella Release 14 si apre un disegno di una release precedente, tutte le polilinee 2D vengono automaticamente convertite in polilinee ottimizzate, a meno che non siano state curvate o adattate oppure contengano dati estesi.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione degli oggetti

Polilinee e polilinee ottimizzate

Elaborazione di polilinee curve e spline

Quando un'applicazione AutoLISP utilizza la funzione **entnext** per spostarsi tra i vertici di una polilinea, potrebbe incontrarne alcuni non creati esplicitamente. I vertici ausiliari vengono inseriti automaticamente dalle opzioni Curva e Spline del comando EDITPL e possono essere tranquillamente ignorati, in quanto le modifiche apportate a tali vertici vengono ignorate la volta successiva in cui l'utente utilizza sulla polilinea tali opzioni del comando EDITPL.

I flag del gruppo 70 dell'entità polilinea indicano se questa è stata trasformata in una curva (valore di bit 2) o in una spline (valore di bit 4). Se nessuno di questi due bit risulta impostato, tutti i vertici della polilinea sono stati regolarmente definiti dall'utente. Tuttavia, se risulta impostato il bit riguardante la curva (2), un vertice su due della polilinea avrà valore di bit 1 nel gruppo 70 per indicare che tali vertici sono stati inseriti durante il procedimento di curvatura. Se si utilizza la funzione **entmod** per spostare i vertici di una polilinea di questo tipo, con l'intento di eseguire nuovamente la curvatura mediante il comando EDITPL, ignorare questi vertici.

Allo stesso modo, se è impostato il bit del flag riguardante la spline (bit 4), si troverà una varietà di vertici, alcuni con bit 1 (inseriti dal procedimento di curvatura se la variabile di sistema SPLINESEGS era negativa), alcuni con valore di bit 8 (inseriti durante la creazione della spline) ed altri con valore di bit 16 (punto di controllo della spline). Anche qui, se si utilizza la funzione **entmod** per spostare i vertici intendendo modificare in seguito la spline, spostare solamente i vertici dei punti di controllo.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione degli oggetti

Gestione degli oggetti non grafici

AutoCAD usa due tipi di oggetti non grafici, i dizionari e le tabelle di simboli. Sebbene questi due oggetti siano simili, vengono gestiti diversamente.

Tutti gli oggetti sono supportati dalle funzioni **entget**, **entmod**, **entdel** e **entmake**, sebbene gli oggetti siano diversificati e possano rifiutare qualsiasi elaborazione all'interno delle funzioni. Le seguenti regole possono essere applicate con rispetto agli oggetti incorporati di AutoCAD. Gli oggetti non grafici *non* possono essere passati alla funzione **entupd**.

Quando si utilizza **entmake**, il tipo di oggetto determina dove deve risiedere l'oggetto. Ad esempio, se viene passato un layer alla funzione **entmake**, esso automaticamente arriva nella tabella di simboli del layer. Se viene passato un oggetto grafico alla funzione **entmake**, esso risiederà nello spazio corrente (modello o carta).

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione degli oggetti

Gestione degli oggetti non grafici

Oggetti di tabelle di simboli

Le regole riportate di seguito vengono applicate agli oggetti incorporati di AutoCAD sia grafici che non grafici:

- Le voci delle tabelle di simboli possono essere create mediante **entmake** con delle limitazioni, purché siano rappresentazioni di record valide e si possono verificare conflitti tra nomi soltanto nella tabella VPORT. Non è possibile creare le voci *ACTIVE.
- Mediante la funzione **entget** è possibile accedere alle tabelle di simboli e alle voci delle tabelle di simboli. La funzione **tblobname** può essere usata per ricercare il nome dell'entità di una voce di una tabella di simboli.
- Le tabelle di simboli non possono essere create con **entmake**, ma è possibile creare gli oggetti delle tabelle di simboli.
- I gruppi del gestore (5, 105) non possono essere modificati nella funzione **entmod**, né specificati nella funzione **entmake**.

- Molti dei campi delle voci delle tabelle di simboli possono essere modificati con la funzione **entmod**. Per poter passare un elenco di record della tabella di simboli alla funzione **entmod**, è necessario inserire un nome di entità che si può ottenere utilizzando la funzione **entget**, ma non le funzioni **tblsearch** e **tblnext**. Il gruppo 70 delle voci presenti nella tabella di simboli verrà ignorato in **entmod** e **entmake**.

Non è possibile rinominare le voci delle tabelle di simboli con nomi duplicati, ad eccezione della tabella di simboli VPORT. Non è possibile modificare né rinominare le voci riportate di seguito, ad eccezione della maggior parte delle voci della tabella LAYER, che possono essere rinominate ed i cui dati possono essere modificati in tutte le voci delle tabelle di simboli.

Voci che non è possibile modificare o rinominare

Tabella	Nome voce
VPORT	*ACTIVE
LINETYPE	CONTINUOUS
LAYER	Le voci non possono essere modificate, ad eccezione dei dati estesi e delle operazioni che consentono di rinominare tali voci.

Le voci riportate di seguito non possono essere rinominate, ma possono essere modificate con delle limitazioni.

Voci che non è possibile rinominare

Tabella	Nome voce
LAYER	0
STYLE	STANDARD
DIMSTYLE	STANDARD
BLOCKS	*MODEL_SPACE
BLOCKS	*PAPER_SPACE
APPID	Non è possibile rinominare alcuna voce.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Gestione degli oggetti

Gestione degli oggetti non grafici

Oggetti dizionario

Le regole riportate di seguito vengono applicate agli oggetti dizionario:

- Gli oggetti dizionario possono essere esaminati mediante la funzione **entget**, mentre è possibile utilizzare la funzione **entmod** per modificare i relativi dati estesi e reattori costanti. Le corrispondenti voci *non* possono essere alterate con **entmod**. L'accesso alle voci viene effettuato con le funzioni **dictsearch** e **dictnext**.
- Il contenuto delle voci di dizionario *non* può essere modificato con **entmod**, sebbene sia possibile modificare dati esterni.
- Le voci di dizionario che cominciano con **ACAD*** *non* possono essere rinominate.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Dati estesi

Molte funzioni AutoLISP vengono fornite per gestire *dati estesi* creati dalle applicazioni scritte mediante ARX o AutoLISP. Se un'entità contiene dati estesi, questi seguono i dati di definizione regolari dell'entità.

Di seguito sono riportate le funzioni di gestione dei dati estesi.

regapp xdroom xdsiz

È possibile richiamare i dati estesi di un'entità chiamando la funzione **entget**. La funzione **entget** richiama i dati di definizione regolari di un'entità ed i dati estesi per le applicazioni specificate nella chiamata **entget**.

Quando i dati estesi vengono richiamati con la funzione **entget**, il loro inizio viene indicato dal codice -3. Tale codice si trova in un elenco che precede il primo gruppo 1001. Tale gruppo contiene il nome della prima applicazione richiamata, come mostrato nella tabella e come descritto nelle sezioni che seguono.

Campo gruppo	Codice	Campo	
		Nome entità)	Dati normali
		Campi di definizione dati)	definizione entità
		.	
		.	
		.	
)			
	(-3	Sentinella dati estesi	Dati estesi
	(1001	Nome dell'applicazione registrata 1)	
	(1000,		
	1002-1071	Campi dati estesi)	
		.	
		.	
		.	
	(1001	Nome dell'applicazione registrata 2)	
	(1000,		
	1002-1071	Campi dati estesi)	
		.	
		.	
		.	
	(1001	Nome dell'applicazione registrata 3)	
		.	
		.	

Dati estesi

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Dati estesi

Organizzazione dei dati estesi

I dati estesi sono costituiti da uno o più gruppi 1001, ognuno dei quali inizia con un nome di applicazione univoco.

I gruppi di dati estesi restituiti dalla funzione **entget** seguono i dati di definizione nell'ordine in cui questi vengono salvati nel database. Tutto questo viene mostrato in modo schematico nella figura.

Nell'ambito di ogni gruppo dell'applicazione, il contenuto, il significato e l'organizzazione dei dati vengono definiti dall'applicazione stessa. AutoCAD gestisce le informazioni ma non le utilizza. La tabella mostra anche che i codici di gruppo per i dati estesi sono compresi tra 1000-1071. Molti di questi codici di gruppo riguardano tipi di dati di uso comune, come riportato di seguito.

Stringa

1000. Le stringhe nei dati estesi possono avere una lunghezza massima di 255 byte (con il 256esimo byte riservato al carattere nullo).

Nome applicazione

1001 (anche valore di stringa). I nomi delle applicazioni possono essere composti da un massimo di 31 byte (il 32esimo byte è riservato al carattere nullo) e devono rispettare le regole riguardanti i nomi delle tabelle di simboli (come i nomi dei layer). Il nome di un'applicazione può contenere lettere, cifre ed i caratteri speciali \$ (dollaro), - (trattino) e _ (lineetta di sottolineatura), ma *non* può contenere spazi. Le lettere del nome vengono convertite in lettere maiuscole.

Nome layer

1003. Il nome di un layer associato ai dati estesi.

Gestore database

1005. Il gestore di un'entità nel database dei disegni.

punto 3D

1010. Tre valori reali contenuti in un punto.

Reale

1040. Un valore reale.

Numero intero

1070. Un numero intero a 16 bit (con e senza segno).

Lungo

1071. Un numero intero (lungo) con segno a 32 bit. Se il valore che appare in un gruppo 1071 è un intero breve o reale, viene convertito in un numero intero lungo; se non è valido (ad esempio, una stringa), viene convertito in uno zero lungo (0L).

Nota AutoLISP gestisce i gruppi 1071 come valori reali. Se si usa la funzione **entget** per richiamare l'elenco di definizione di un'entità che contiene un gruppo 1071 il valore viene restituito come reale, come mostrato nell'esempio riportato di seguito.

(1071 . 12.0)

Se si desidera creare un gruppo 1071 in un'entità con la funzione **entmake oentmod**, è possibile utilizzare un valore reale o un valore intero, come mostrato nell'esempio riportato di seguito.

```
(entmake '((..... (1071 . 12) .... )))
(entmake '((..... (1071 . 12.0) .... )))
(entmake '((..... (1071 . 65537.0) .... )))
(entmake '((..... (1071 . 65537) .... )))
```

AutoLISP restituisce ancora il valore del gruppo come reale.

```
(entmake '((..... (1071 . 65537) .... )))
```

L'istruzione precedente restituisce quanto riportato di seguito.

(1071 . 65537.0)

ARX gestisce sempre i gruppi 1071 come valori interi lunghi.

Molti altri gruppi di dati estesi hanno un significato speciale in questo contesto, se l'applicazione sceglie di utilizzarli.

Stringa di controllo

1002. Una stringa di controllo dei dati estesi può essere "{ o }". Queste parentesi graffe consentono all'applicazione di organizzare i propri dati suddividendoli in elenchi. La parentesi aperta inizia un elenco, mentre la parentesi chiusa termina l'ultimo elenco creato. Gli elenchi possono essere nidificati.

Nota Se in un elenco appare un gruppo 1001, tale gruppo viene trattato come stringa e *non* inizia un nuovo gruppo di applicazioni.

Dati binari

1004. I dati binari organizzati in *parti* di lunghezza variabile possono essere gestiti in ADSRX con la struttura ads_binary. Ogni parte può essere lunga, al massimo, 127 byte.

Nota AutoLISP *non* può gestire direttamente le parti binarie, quindi per i gruppi binari sono valide le stesse precauzioni riguardanti i gruppi lunghi (1071).

Posizione globale

1011. A differenza di un semplice punto tridimensionale, le coordinate WCS vengono spostate, messe in scala, ruotate e rese speculari insieme all'entità ad esse correlata a cui appartengono i dati estesi. La posizione WCS viene anche stirata quando il comando STIRA viene applicato all'entità correlata e quando questo punto si trova all'interno della finestra di selezione.

Spostamento globale

1012. Un punto tridimensionale che viene messo in scala, ruotato o reso speculare insieme all'entità ad esso correlata, ma che *non* viene stirato o spostato.

Direzione globale

1013. Un punto tridimensionale che viene ruotato o reso speculare insieme all'entità ad esso correlata, ma che *non* viene messo in scala, stirato o spostato. La direzione WCS è uno spostamento normalizzato che ha sempre lunghezza unitaria.

Distanza

1041. Un valore reale che viene scalato insieme all'entità correlata.

Fattore di scala

1042. Un altro valore reale che viene scalato insieme all'entità correlata.

I codici di gruppo DXF per i dati estesi vengono descritti anche nell'appendice C "Codici di gruppo DXF".

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Dati estesi

Registrazione di un'applicazione

Un'applicazione, per essere riconosciuta da AutoCAD, deve registrare il nome o i nomi che utilizza. I nomi delle applicazioni vengono salvati con i dati estesi di ogni entità che li utilizza ed anche nella tabella APPID. La registrazione viene effettuata con la funzione **regapp**. La funzione **regapp** specifica una stringa da utilizzare come nome dell'applicazione. Se tale funzione aggiunge con esito positivo il nome alla tabella APPID, restituisce il nome dell'applicazione; altrimenti, restituisce nil. Il risultato nil indica che il nome è già presente nella tabella di simboli. Questa non è una condizione di errore reale, ma un valore di ritorno previsto, in quanto il nome dell'applicazione deve essere registrato una sola volta per ogni disegno.

Per registrare se stessa, un'applicazione deve prima controllare che il suo nome non sia già presente nella tabella APPID. Se non è presente, l'applicazione deve registrarlo. Altrimenti, può semplicemente proseguire ed utilizzare i dati, come descritto successivamente in questa sezione.

La parte di codice riportata di seguito mostra l'uso tipico della funzione **regapp**. La funzione **tblsearch** viene descritta in "Accesso alle tabelle di simboli e ai dizionari."

```
(setq nomeapp "MIAPP_2356")
(if (tblsearch "appid" nomeapp)
    (princ (strcat
        "\n" nomeapp " già registrata. "))
    (if (= (regapp appname) nil)
        (princ (strcat
            "\nImpossibile registrare dati estesi per " nomeapp ". "))
    )
)
```

Nome applicazione univoco
Controlla se è già registrata

Altri problemi

Nota La funzione **regapp** fornisce una misura di sicurezza, ma non può garantire che due applicazioni separate non abbiano scelto lo stesso nome. Un modo per fare questo controllo è quello di adottare uno schema per i nomi che utilizza il nome dell'azienda o del prodotto ed un numero univoco, come ad esempio il numero telefonico oppure la data e l'ora correnti.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Dati estesi

Recupero dei dati estesi

La funzione **entget** può essere richiamata da un'applicazione per ottenere i dati estesi memorizzati. La funzione **entget** può restituire sia i dati di definizione che i dati estesi relativi alle applicazioni da essa richieste. È necessario specificare un ulteriore argomento, *applicazione*, che indichi i nomi delle applicazioni. I nomi passati a **entget** devono corrispondere alle applicazioni memorizzate da una precedente chiamata alla funzione **regapp**; in essi possono essere contenuti anche i caratteri jolly.

Per default, i modelli di tratteggio associativi contengono dati estesi. Con il codice riportato di seguito viene mostrato l'elenco delle associazioni di tali dati estesi.

Comando: **(entget (car (entsel)) ("ACAD"))**

Selezione oggetto: *seleziona un comando associativo*

L'immissione alla riga di comando del codice sopra riportato restituisce un elenco simile a quello descritto di seguito.

```
((-1 . <Nome di entità: 600000c0>) (0 . "INSER") (8 . "0") (2 . "*X0")
(10 0.0 0.0 0.0) (41 . 1.0) (42 . 1.0) (50 . 0.0) (43 . 1.0) (70 . 0) (71 . 0)
(44 . 0.0) (45 . 0.0) (210 0.0 0.0 1.0) (-3 ("ACAD" (1000 . "HATCH")
(1002 . "{") (1070 . 16) (1000 . "LINE") (1040 . 1.0) (1040 . 0.0)
(1002 . "}"))))
```

Questo frammento descrive una sequenza tipica per il recupero dei dati estesi relativi alle due applicazioni specificate. Notare che l'argomento *applicazione* passa i nomi delle applicazioni in forma di elenco.

```
(setq working_elist
  (entget ent_name
    ('("MIA_APP_1" "QUALCOSA_ALTRO")
  )
)
)
(if working_elist
  (progn
    ...
    (entmod working_elist)
  )
)

```

Vengono recuperati solo i dati estesi delle applicazioni "MIA_APP_1" e "QUALCOSA_ALTRO".

Aggiorna i gruppi di elementi di lavoro. Vengono modificati solo i dati estesi che si trovano nelle applicazioni memorizzate, ancora contenute nell'elenco working_elist

Come descritto nel codice di esempio, è possibile modificare i dati estesi recuperati dalla funzione **entget** utilizzando una chiamata successiva alla funzione **entmod**, nello stesso modo in cui **entmod** viene utilizzata per modificare i normali dati di definizione. È inoltre possibile creare dati estesi definendoli nell'elenco di entità passato a **entmake**.

Restituendo solamente i dati estesi delle applicazioni richieste esplicitamente, consente di evitare che una applicazione ne danneggi un'altra. Inoltre, viene controllata la quantità di memoria necessaria per un'applicazione e viene semplificata l'elaborazione dei dati estesi che l'applicazione deve eseguire.

Nota Poiché le stringhe passate mediante *applicazione* possono includere caratteri jolly, il nome "*" di un'applicazione provocherà la restituzione, da parte di **entget** di *tutti* i dati estesi ad una entità.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Dati estesi

Unione di dati estesi ad una entità

I dati estesi possono essere usati per memorizzare qualsiasi tipo di informazione. Di seguito viene riportato un esempio di unione di dati estesi ad una entità.

Per prima cosa è necessario disegnare una entità (come una linea o un cerchio), quindi immettere il seguente codice:

```
(setq lastent (entget (entlast)))
(regapp "NUOVIDATI")
(setq exdata
  '((-3 ("NUOVIDATI"
    (1000 . "Nuova stringa!")
  )))
)
(setq newent
  (append lastent exdata))
(entmod newent)
```

Prende l'elenco di associazioni dei dati di definizione relativi all'ultima entità. Memorizza il nome dell'applicazione. Imposta la variabile exdata nello stesso modo dei nuovi dati estesi; in questo caso una stringa di testo.

Aggiunge il nuovo elenco di dati all'elenco delle entità. Modifica l'elemento con i nuovi dati di definizione.

Per verificare che i nuovi dati estesi siano stati uniti all'entità, inserire il codice riportato di seguito e selezionare l'oggetto.

```
(entget (car (entsel)) ("NUOVIDATI"))
```

In questo esempio è stato descritto il metodo di base per unire i dati estesi ad una entità.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Dati estesi

Gestione dell'utilizzo della memoria da parte dei dati estesi

I dati estesi sono attualmente limitati a 16 K per entità. Poiché i dati estesi di un'entità possono essere creati e gestiti da più applicazioni, potrebbero verificarsi dei problemi quando la dimensione dei dati estesi è prossima al limite massimo. AutoLISP fornisce due funzioni, **xysize** e **xdroom**, per facilitare la gestione della memoria occupata dai dati estesi. Quando a **xysize** viene passato un elenco di dati estesi, la funzione restituisce la quantità di memoria in byte che sarà occupata dai dati; quando a **xdroom** viene passato il nome di una entità, la funzione restituisce il numero restante di byte liberi che possono ancora essere aggiunti all'entità.

Poiché la funzione **xysize** legge un elenco di dati estesi, che potrebbe anche essere molto grande, la sua esecuzione richiede più tempo. Si consiglia, pertanto, di non utilizzarla frequentemente. Il modo migliore è di utilizzarla insieme a **xdroom** in un gestore di errori. Se una chiamata alla funzione **entmod** ha esito negativo, è possibile usare **xysize** e **xdroom** per determinare se ciò si è verificato perché l'entità non disponeva di spazio sufficiente per i dati estesi.

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Dati estesi

Gestori nei dati estesi

I dati estesi possono contenere gestori (gruppo 1005) che consentono di salvare le strutture relazionali all'interno di un disegno. Un'entità può fare riferimento ad un'altra salvando nei suoi dati estesi il gestore di quest'ultima. Il gestore può essere successivamente recuperato dai dati estesi e passato alla funzione **handent** per ottenere l'altra entità. Poiché più di una entità può fare riferimento ad un'altra, i gestori dei dati estesi non sono necessariamente univoci. Il comando VERIFICA richiede che i gestori inclusi nei dati estesi siano NULL oppure gestori di entità validi, all'interno del disegno attuale. Il modo migliore per accertarsi che i gestori delle entità estese siano validi è di ottenere il gestore di una entità alla quale viene fatto riferimento direttamente dai suoi dati di definizione, per mezzo di **entget**. Il valore del gestore si trova nel gruppo 5.

Quando viene fatto riferimento ad entità contenute in altri disegni (ad esempio, entità che sono unite con XRIF), è possibile evitare l'esito negativo del comando VERIFICA utilizzando stringhe di entità estese (gruppo 1000) invece dei gestori (gruppo 1005), poiché i gestori di entità con riferimenti ad incrocio *non* sono validi nel disegno attuale oppure sono in conflitto con i gestori validi. Tuttavia, se un XRIF Attacca viene cambiato in XRIF Unisci oppure viene combinato con il disegno corrente in un altro modo, è compito dell'applicazione rivedere e correggere di conseguenza i riferimenti all'entità.

Quando i disegni vengono uniti tramite INSER, INSER*, XRIF Unisci (XUNISCE) o DXFIN parziale, i gestori vengono convertiti in modo tale da diventare validi nel disegno corrente. Se nel nuovo disegno non è stato utilizzato alcun gestore, ad esso verranno assegnati nuovi gestori. Quando questi comandi sono richiamati, vengono convertiti anche i gestori delle entità estese che fanno riferimento a nuove entità.

Quando una entità viene posizionata nella definizione di un blocco con il comando BLOCCO, all'entità all'interno del blocco vengono assegnati dei nuovi gestori. Se l'entità originaria viene memorizzata con OOPS, essa mantiene i suoi gestori originali. Il valore di tutti i gestori di dati estesi rimane invariato. Quando un blocco viene esploso con il comando ESPLODI, i gestori dei dati estesi vengono convertiti in un modo simile a quello con cui sono convertiti quando vengono combinati i disegni. Se il gestore dei dati estesi si riferisce ad un'entità che non si trova all'interno del blocco, esso rimane invariato; se invece il gestore dei dati estesi si riferisce ad una entità che si trova all'interno del blocco, ad esso viene assegnato il valore del gestore della nuova entità (quella decomposta).

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Oggetti xrecord

Gli oggetti xrecord vengono utilizzati per memorizzare e gestire dati arbitrari o liberi. Sono composti da codici di gruppo DXF con gruppi di "oggetti normali" ossia codici di gruppo di dati non estesi, compresi nell'intervallo da 1 a 369 per gli intervalli supportati.

Questo oggetto è concettualmente simile ai dati estesi, ma non ha limiti di ordine o di dimensioni.

Gli oggetti xrecord sono progettati in modo da non danneggiare le release da R13c0 a R13c3 di AutoCAD. Tuttavia, se letti in una release precedente a R13c4, gli oggetti xrecord scompaiono.

Seguono alcuni esempi che illustrano dei metodi per creare ed elencare i dati xrecord.

```
(defun C:MAKEXRECORD( / xrec xname )
  ; creazione dell'elenco di dati xrecord
  (setq xrec '((0 . "XRECORD")(100 . "AcDbXrecord")
    (1 . "Questo è un elenco xrecord di prova")
    (10 1.0 2.0 0.0) (40 . 3.14159) (50 . 3.14159)
    (62 . 1) (70 . 180))
  )
  ; creazione di un xrecord senza proprietario con entmakex
  (setq xname (entmakex xrec))
  ; aggiunta del nuovo xrecord al dizionario di oggetti denominati
  (dictadd (namedobjdict) "XRECLIST" xname)
  (princ)
)

(defun C:LISTXRECORD ( / xlist )
  ; ricerca di xrecord nel dizionario di oggetti denominati
  (setq xlist (dictsearch (namedobjdict) "XRECLIST"))
  ; stampa dell'elenco di dati xrecord
  (princ xlist)
  (princ)
)
```

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Accesso alle tabelle di simboli e ai dizionari

AutoLISP fornisce alcune funzioni che consentono di accedere alle voci presenti nelle tabelle di simboli e nei dizionari. Le funzioni per l'accesso alle tabelle di simboli e ai dizionari sono le seguenti:

dictsearch	namedobjdict	tblnext	tblsearch
dictnext	snvalid	tblobjname	

Esempi delle funzioni **tblnext** e **tblsearch** sono forniti in questa sezione. Vedere il capitolo 13 per informazioni sulle altre funzioni di accesso a dizionari e a tabelle di simboli. Per ulteriori informazioni su oggetti non grafici, vedere "Gestione degli oggetti non grafici."

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Accesso alle tabelle di simboli e ai dizionari

Tabelle di simboli

Le voci delle tabelle di simboli e dei dizionari possono essere modificate anche dalle seguenti funzioni:

entdel	entmake	handent
entget	entmod	

La funzione **tblnext** analizza in modo sequenziale le voci delle tabelle, mentre la funzione **tblsearch** recupera determinate voci. I nomi delle tabelle sono specificati da stringhe. I nomi validi sono "LAYER", "LTYPE", "VIEW", "STYLE", "BLOCK", "UCS", "VPORT", "DIMSTYLE" e "APPID". Entrambe le funzioni restituiscono elenchi con codici di gruppo DXF simili ai dati dell'entità restituiti da **entget**.

La prima chiamata a **tblnext** restituisce la prima voce contenuta nella tabella specificata. Le chiamate successive nelle quali viene specificata la stessa tabella restituiscono le voci successive, a meno che il secondo argomento di **tblnext(rewind)** non sia diverso da zero, nel qual caso **tblnext** restituirà di nuovo la prima voce.

Nell'esempio riportato di seguito, la funzione **PRENDI_BLOCCO** richiama la voce della tabella di simboli relativa al primo blocco (se presente) contenuto nel disegno corrente, quindi la visualizza come un elenco.

```
(defun C:PRENDI_BLOCCO (/ blk ct)
  (setq blk (tblnext "BLOCCO" 1)) Prende la prima immissione di BLOCK
  (setq ct 0) Imposta ct (un contatore) a 0
  (textpage) Passa allo schermo di testo
  (princ "\nRisultati da PRENDI_BLOCCO: ")
  (repeat (length blk) Esegue la stessa operazione per il numero di membri presenti nell'elenco
    (print (nth ct blk)) Stampa una riga nuova e poi ogni membro dell'elenco
    (setq ct (1+ ct)) Incrementa il contatore di 1
  )
  (princ) Esce senza apportare modifiche
)
```

Le voci richiamate dalla tabella BLOCCO contengono un gruppo -2 in cui si trova il nome della prima entità inclusa nella definizione di blocco. Se il blocco è vuoto, il nome è quello dell'entità ENDBLK del blocco, che non compare mai nei blocchi non vuoti. In un disegno con un solo blocco di nome BOX, la chiamata alla funzione **PRENDI_BLOCCO** visualizza quanto riportato di seguito. Il nome varia da sessione a sessione.

Risultati di PRENDI_BLOCCO:

```
(0 . "BLOCK")
(2 . "BOX")
(70 . 0)
(10 9.0 2.0 0.0)
(-2 . <Nome di entità: 40000126>)
```

Come con **tblnext**, il primo argomento di **tblsearch** è una stringa che contiene il nome di una tabella, ma il secondo argomento è una stringa che contiene il nome di un determinato simbolo incluso nella tabella. Se il simbolo viene trovato, **tblsearch** restituisce i dati ad esso relativi. Questa funzione ha un terzo argomento, *successivo*, che può essere usato per coordinare le operazioni con **tblnext**. Se *successivo* è nil, la chiamata **tblsearch** non ha alcun effetto su **tblnext**, ma se *successivo* non è nil, la successiva chiamata di **tblnext** restituirà la voce di tabella immediatamente successiva a quella trovata da **tblsearch**.

L'opzione *successivo* è utile per la gestione della tabella di simboli VPORT, poiché tutte le finestre di una particolare configurazione hanno lo stesso nome, come ad esempio *ACTIVE.

Se si accede alla tabella di simboli VPORT quando TILEMODE è disattivato, le modifiche non hanno effetto visibile fino a quando TILEMODE non sarà attivo. Non confondere VPORTS, che viene descritto dalla tabella di simboli VPORT, con le entità della finestra dello spazio carta.

Il codice seguente elabora tutte le finestre presenti nella configurazione 4VIEW.

```
(setq v (tblsearch "VPORT" "4VIEW" T)) Cerca la prima voce VPORT
(while (and v (= (cdr (assoc 2 v)) "4VIEW"))
  .
  . ... Elaboro la voce ...
  .
  (setq v (tblnext "VPORT")) Prende la successiva voce VPORT
)
```

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Accesso alle tabelle di simboli e ai dizionari

Dizionari

Per dizionario si intende un oggetto contenente dati, simile nella funzione alle tabelle di simboli. Le voci di dizionario possono essere richiamate con le funzioni **dictsearch** e **dictnext**. Ogni voce di dizionario consiste in una chiave per il nome del testo ed un riferimento del gestore all'oggetto relativo alla voce stessa. Le voci di dizionario possono essere eliminate passando direttamente il nome dell'oggetto alla funzione **entdel**. La chiave per il nome del testo utilizza la

Capitolo 9 -- Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli

Accesso alle tabelle di simboli e ai dizionari

Dizionari

Accesso ai gruppi AutoCAD

Il seguente esempio illustra uno dei metodi per accedere alle entità contenute in un gruppo. Nell'esempio si suppone che nel disegno corrente esista un gruppo chiamato G1.

```
(setq objdict (namedobjdict))
(setq grpdict (dictsearch objdict "ACAD_GROUP"))
```

Questo imposta la variabile grpdict sull'elenco di definizione delle entità del dizionario ACAD_GROUP e restituisce quanto segue:

```
((-1 . <Nome di entità: 8dc10468>) (0 . "DIZIONARIO") (5 . "D")
(102 . "{ACAD_REACTORS}") (330 . <Nome entità: 8dc10460>)
(102 . "}") (100 . "AcDbDictionary") (3 . "G1")
(350 . <Nome di entità: 8dc41240>))
```

Il seguente codice imposta la variabile group1 all'elenco di definizione delle entità del gruppo G1.

```
(setq group1 (dictsearch (cdar grpdict) "G1"))
```

Restituisce quanto riportato di seguito:

```
((-1 . <Nome di entità: 8dc10518>) (0 . "GROUP") (5 . "23")
(102 . "{ACAD_REACTORS}") (330 . <Nome entità: 8dc10468>)
(102 . "}") (100 . "AcDbGroup") (300 . "linea e cerchio") (70 . 0) (71 . 1)
(340 . <Nome di entità: 8dc10510>)(340 . <Nome entità: 8dc10550> )
```

I codici di gruppo 340 indicano le entità che appartengono al gruppo.

Capitolo 10 -- Gestione delle finestre di dialogo

Panoramica

AutoCAD consente di progettare ed utilizzare le finestre di dialogo in base alle applicazioni in uso. La visualizzazione delle finestre di dialogo viene definita tramite i file DCL (Dialog Control Language) che vengono descritti nella parte III della presente guida. La funzionalità delle finestre di dialogo viene gestita dall'applicazione AutoLISP o ADSRX. Il presente capitolo indica come gestire le finestre di dialogo con AutoLISP. Nonostante questo capitolo fornisca degli esempi di file DCL, può risultare utile leggere prima la parte III.

Capitolo 10 -- Gestione delle finestre di dialogo

Apertura e chiusura delle finestre di dialogo

AutoLISP fornisce le seguenti funzioni per l'apertura e la chiusura delle finestre di dialogo:

done_dialog	new_dialog	term_dialog
load_dialog	start_dialog	unload_dialog

Prima di poter utilizzare le funzioni per le finestre di dialogo, è necessario disporre di un file DCL che definisca la finestra. Salvare il seguente codice DCL in un file denominato *ciao.dcl* in una directory del percorso dei file di supporto. Questo file definisce una

finestra di dialogo denominata Esempio di finestra di dialogo che contiene una casella di testo ed un unico pulsante OK.

```
ciao: dialog {
  label = "Esempio di finestra di dialogo";
  : text { label = "Ciao a tutti."
  ok_only;
}
```

La visualizzazione di una finestra di dialogo comporta una serie di operazioni. La prima richiede l'uso della funzione `load_dialog` per caricare il file DCL in memoria ed ottenere il numero identificativo DCL. Una volta caricato il file DCL, è necessario richiamare la funzione `new_dialog` e passare il nome della finestra di dialogo ed il numero identificativo DCL come argomenti. Se la funzione `new_dialog` non restituisce nil, richiamare la funzione `start_dialog` per passare il controllo della finestra di dialogo ad AutoCAD ed all'utente. Poiché la casella `ok_only` (pulsante OK) è predefinita, anche l'azione che svolge lo è. In genere, prima di richiamare la funzione `start_dialog` vengono assegnate le azioni e vengono impostati gli stati ed i valori delle caselle (vedere "Espressioni delle azioni e funzioni di richiamo"). L'azione assegnata alla casella `ok_only` è `done_dialog`. Quando viene scelto il pulsante OK, AutoCAD passa la funzione `done_dialog` all'applicazione AutoLISP e chiude la finestra di dialogo. Quindi il numero identificativo DCL viene passato alla funzione `unload_dialog`, che scarica il file DCL dalla memoria. Per una descrizione completa della sequenza di funzioni delle finestre di dialogo, vedere "Sequenza di funzioni."

Se il file `ciao.dcl` è stato salvato in una directory del percorso dei file di supporto, è possibile visualizzarlo utilizzando la seguente funzione AutoLISP. Questa versione dispone di funzioni di controllo degli errori di base, quindi risulta più semplice vedere la relativa struttura.

```
(defun C:HELLO( / dcl_id )
  (setq dcl_id (load_dialog "ciao.dcl"))
  (if (not (new_dialog "ciao" dcl_id))
    (exit)
  )
  (start_dialog)
  (unload_dialog dcl_id)
  (princ)
)
```

Carica il file DCL
Inizializza il dialogo
Esce se non ha esito positivo

Visualizza finestra di dialogo
Scarica il file DCL

La funzione `start_dialog` rimane attiva fino a quando non viene selezionata una casella (in genere un pulsante) la cui *espressione azione* richiama `done_dialog`. La chiamata alla funzione `done_dialog` può essere effettuata esplicitamente tramite la casella o impostando su vero l'attributo `is_cancel` della casella selezionata.

Attenzione In teoria, la funzione della finestra di dialogo acquisisce il controllo dell'immissione nel momento in cui viene richiamata la funzione `start_dialog`, ma sotto Windows ed altre piattaforme essa acquisisce il controllo quando viene richiamata la funzione `new_dialog`. Ciò non incide sulla scrittura di programmi. Tuttavia, se queste funzioni vengono richiamate in modo interattivo, è necessario immetterle come istruzioni alla riga di comando di AutoCAD. Tali funzioni devono essere interne a `progn` o ad un'altra funzione, altrimenti la chiamata interattiva per la funzione `new_dialog` potrebbe congelare lo schermo. La chiamata interattiva di `new_dialog` e `start_dialog` può risultare utile durante il debug. Per un esempio dell'uso interattivo di queste funzioni, vedere "Gestione degli errori DCL."

Quando una finestra di dialogo è attiva, cioè durante la funzione `start_dialog`, non è possibile richiamare le funzioni AutoLISP elencate in seguito. Tali funzioni influiscono sulla visualizzazione, che non deve subire variazioni durante la visualizzazione della finestra di dialogo, oppure richiedono un intervento da parte dell'utente che non riguarda la finestra di dialogo.

command	getcorner	getstring	nentsel
entdel	getdist	graphscr	osnap
entmake	getint	grdraw	messaggio di
			richiesta[a]
entmod	getkword	grread	redraw
entsel	getorient	grtext	ssget [b]
entupd	getpoint	grvecs	textpage
getangle	getreal	menucmd	textscr

[a] Le funzioni per la scrittura di testo come `print`, `princ` e `prinl` sono utili per visualizzare le informazioni relative al debug durante la verifica di una finestra di dialogo. Non utilizzarle in prodotti finiti. Se la finestra di dialogo visualizzata si sovrappone all'area della riga di comando, l'output di queste funzioni sovrascriverà quella parte della finestra di dialogo.

[b] Le opzioni interattive `ssget` non sono consentite, mentre altre opzioni lo sono.

Se il programma in uso richiama una di queste funzioni tra le chiamate `start_dialog` e `done_dialog`, AutoCAD chiude tutte le finestre di dialogo e visualizza il seguente messaggio di errore:

la funzione è stata respinta da AutoCAD

Se il valore della variabile di sistema `CMDACTIVE` è maggiore di 7, è attiva una finestra di dialogo. La variabile di sistema `CMDACTIVE` ha valori codificati in base al sistema binario che indicano l'attività di comandi, script e finestre di dialogo. Vedere "Variabili di sistema," nella *Guida di riferimento dei comandi di AutoCAD*.

Come viene indicato nella seconda nota a piè pagina relativa alla tabella, le chiamate **ssget** sono consentite per tutte le opzioni, tranne per quelle interattive.

Se è necessario effettuare un'immissione in base allo schermo di disegno piuttosto che in base alla finestra di dialogo (ad esempio, per specificare un punto o selezionare un oggetto), occorre *nascondere* la finestra di dialogo. A tale scopo, è necessario richiamare la funzione **done_dialog** in modo che lo schermo di disegno sia visibile e quindi riattivare la finestra di dialogo dopo che l'utente ha eseguito la selezione. Per ulteriori informazioni, vedere "Come nascondere le finestre di dialogo."

La funzione **term_dialog** conclude tutte le finestre di dialogo correnti come se l'utente avesse cancellato ciascuna di esse. Tale funzione può essere utilizzata per eliminare una serie di finestre di dialogo nidificate.

Capitolo 10 -- Gestione delle finestre di dialogo

Gestione di caselle ed attributi

AutoLISP fornisce le seguenti funzioni per la gestione di caselle ed attributi: Le funzioni sono le seguenti:

action_tile	get_tile	set_tile
get_attr	mode_tile	

Esempi sull'uso di queste funzioni sono forniti nella sezione seguente. Queste funzioni sono inoltre descritte in capitolo 13, "AutoLISP: catalogo delle funzioni".

Capitolo 10 -- Gestione delle finestre di dialogo

Gestione di caselle ed attributi

Espressioni delle azioni e funzioni di richiamo

Per definire l'azione da effettuare quando viene selezionata una determinata casella in una finestra di dialogo, è opportuno associare un'espressione AutoLISP alla casella richiamando la funzione **action_tile**, conosciuta come *espressione di azione*. Nell'ambito dell'espressione di azione, risulta spesso necessario accedere agli attributi contenuti nel file DCL. Le funzioni **get_tile** e **get_attr** consentono tale accesso. La funzione **get_attr** richiama gli -attributi definiti dall'utente all'interno del file DCL. La funzione **get_tile** richiama il valore di *esecuzione* corrente di una casella, basandosi sull'input dell'utente in quella casella. Le espressioni delle azioni devono essere definite dopo la chiamata alla funzione **new_dialog** e prima della chiamata alla funzione **start_dialog**.

Le informazioni relative al modo in cui l'utente ha selezionato una casella o modificato il contenuto di una casella vengono restituite all'espressione delle azioni come *funzione di richiamo*. Nella maggior parte dei casi, ogni casella attiva all'interno di una finestra di dialogo può generare una funzione di richiamo. La risposta di una espressione di azione a una funzione di richiamo dovrebbe eseguire una verifica di validità sulla casella ad essa associata e dovrebbe aggiornare le informazioni contenute nella finestra di dialogo relative al valore della casella in questione. L'aggiornamento della finestra di dialogo può includere l'invio di un messaggio di errore, che comporta la disattivazione delle altre caselle, e la visualizzazione del testo appropriato in una casella di modifica o di riepilogo.

La richiesta di salvare in modo permanente le ultime impostazioni selezionate può essere inoltrata ai valori delle caselle soltanto tramite il pulsante OK (o pulsanti equivalenti). In altre parole, è necessario aggiornare le variabili associate ai valori delle caselle nell'ambito delle funzioni di richiamo per il pulsante OK e non aggiornare la funzione di richiamo per una singola casella. Se le variabili permanenti vengono aggiornate nell'ambito di funzioni di richiamo per singole caselle, non è possibile ripristinare i valori nel caso in cui venga selezionato il pulsante Annulla. Se la funzione di richiamo del pulsante OK rileva un errore, in genere viene visualizzato un messaggio di errore e viene riattivata la casella in questione, senza uscire dalla finestra di dialogo.

Quando una finestra di dialogo include diverse caselle la cui gestione è simile, può risultare utile associare tali caselle ad una singola funzione di richiamo. Le modifiche apportate dall'utente non vengono comunque applicate fino a quando non viene scelto il pulsante OK.

Oltre alla chiamata della funzione **action_tile**, vi sono altri due modi per definire le azioni. Infatti, è possibile definire un'azione di *default* per l'intera finestra di dialogo quando viene richiamata la funzione **new_dialog** e definire un'azione utilizzando un attributo dell'azione di una casella. La sezione "Azioni di default e DCL" fornisce una descrizione di queste alternative per la definizione delle azioni e dell'ordine in cui esse si verificano.

Capitolo 10 -- Gestione delle finestre di dialogo

Gestione di caselle ed attributi

Espressioni delle azioni e funzioni di richiamo

Espressioni delle azioni

L'espressione di un'azione può accedere alle variabili riportate nella seguente tabella, indicare quale casella è stata selezionata e descrivere lo stato della casella al momento dell'esecuzione dell'azione stessa. I nomi delle variabili sono riservati. I relativi valori sono a sola lettura e non hanno alcun significato a meno che non vi si acceda nell'ambito dell'espressione di un'azione.

Variabili delle espressioni delle azioni

Variabile	<i>Descrizione</i>
\$key	<i>Attributo chiave della casella selezionata.</i>
\$value)	<i>Questa variabile è valida per tutte le azioni. Formato della stringa del valore corrente della casella, come ad esempio la stringa di una casella di modifica oppure il valore "1" o "0" di un interruttore.</i>
	<i>Questa variabile è valida per tutte le azioni.</i>
\$data	Nota Se si tratta di una casella di riepilogo (o a comparsa) e non è selezionata alcuna casella, la variabile \$value è nil. <i>Dati (se esistenti) gestiti dall'applicazione impostati subito dopo l'esecuzione della funzione new_dialog tramite la funzione client_data_tile.</i>
\$reason	<i>Questa variabile è valida per tutte le azioni, ma non ha alcun significato a meno che non sia stata già inizializzata dall'applicazione in uso tramite la chiamata della funzione client_data_tile. Vedere "Dati specifici dell'applicazione." <i>Codice di causa che indica quale azione effettuata dall'utente ha causato l'azione. Usato con le caselle edit_box, list_box, image_button e slider.</i></i>
\$key	<i>Questa variabile indica il motivo per il quale si è verificata l'azione. Il relativo valore è impostato per qualunque tipo di azione, ma è necessario verificarlo solo quando l'azione è associata ad una casella edit_box, list_box, image_button o slider. Per informazioni dettagliate, vedere "Cause di funzioni di richiamo." <i>Attributo chiave della casella selezionata.</i></i>
	<i>Questa variabile è valida per tutte le azioni.</i>

Se edit1 è una casella di testo, l'espressione dell'azione nella seguente funzione **action_tile** viene valutata quando l'utente esce dalla casella di testo.

```
(action_tile "edit1" "(setq ns $value)")
```

La variabile \$value contiene la stringa immessa dall'utente e l'espressione salva tale stringa nella variabile ns.

Nel secondo esempio, viene salvato il nome della casella selezionata in modo che il programma possa farvi riferimento.

```
(action_tile "edit1" "(setq newtile $key)")
```

La variabile newtile viene impostata in base al nome chiave della casella selezionata, in questo caso "edit1". in questo caso "edit1". La variabile \$key risulta particolarmente utile nell'ambito di una funzione che gestisce le azioni da eseguire per diverse caselle separate.

Nota Quando ad una casella viene assegnato un nome in più di una chiamata alla funzione **action_tile**, soltanto l'ultima di queste chiamate (prima di **start_dialog**) ha effetto (come se si assegnassero più valori alla stessa variabile). La funzionalità PDB consente soltanto un'azione per casella.

Capitolo 10 -- Gestione delle finestre di dialogo

Gestione di caselle ed attributi

Espressioni delle azioni e funzioni di richiamo

Cause di funzioni di richiamo

La *causa* di una funzione di richiamo, restituita come variabile \$reason, specifica la causa del verificarsi dell'azione. Il relativo valore è impostato per qualunque tipo di azione, ma è necessario verificarlo solo quando l'azione è associata ad una casella edit_box, list_box, image_button o slider. La seguente tabella indica i possibili valori. Il testo che segue la tabella descrive i codici 2, 3 e 4 in modo più dettagliato, mentre il codice 1 è descritto completamente nella tabella.

Codici di causa delle funzioni di richiamo

Codice	Descrizione
1	Valore valido per la maggior parte delle caselle di azione. La casella è stata selezionata (premendo RETURN se si tratta della casella di default e se la piattaforma riconosce i tasti di scelta).
2	Caselle di modifica: l'utente è uscito dalla casella di modifica senza effettuare la selezione finale.
3	Dispositivi di scorrimento: l'utente ha modificato il valore del dispositivo di scorrimento trascinando l'indicatore, ma non ha effettuato la selezione finale.
4	Caselle di riepilogo e pulsanti immagine: questa causa della funzione di richiamo segue sempre un codice 1. In genere, significa che "risulta valida la selezione precedente". Non deve mai annullare la selezione precedente, altrimenti confonde l'utente.

Codice 2: Caselle di modifica

L'utente è uscito dalla casella di modifica premendo TAB oppure scegliendo una casella diversa, ma non ha effettuato la selezione finale. Se questa è la causa di una funzione di richiamo della casella di modifica, l'applicazione non dovrebbe aggiornare il valore della variabile associata, ma dovrebbe verificare la validità del valore contenuto nella casella di modifica.

Codice 3: Dispositivi di scorrimento

L'utente ha modificato il valore del dispositivo di scorrimento trascinando l'indicatore (oppure un'azione equivalente), ma non ha effettuato la selezione finale. Se questa è la causa della funzione di richiamo del dispositivo di scorrimento, l'applicazione non dovrebbe aggiornare il valore della variabile associata, ma del testo relativo allo stato del dispositivo stesso. Per ulteriori informazioni, vedere "Dispositivi di scorrimento:." Per esempi di codici, vedere "Gestione dei dispositivi di scorrimento."

Codice 4: Caselle di riepilogo

L'utente ha fatto doppio clic sulla casella di riepilogo. È possibile definire il significato del doppio clic nell'applicazione in uso. Se lo scopo primario della finestra di dialogo è la selezione di un elemento di un elenco, facendo doppio clic viene effettuata una selezione e, quindi, si esce dalla finestra di dialogo. In questo caso, l'attributo is_default della casella list_box è vero. Se la casella di riepilogo non è la casella primaria della finestra di dialogo, il doppio clic equivale ad una selezione (codice 1).

Le caselle di riepilogo che consentono di selezionare più elementi (multiple_select = true) non supportano il doppio clic.

Codice 4: Pulsanti immagine

L'utente ha fatto doppio clic sul pulsante immagine. È possibile definire il significato del doppio clic nell'applicazione in uso. In molti casi, il pulsante viene selezionato facendo un singolo clic; in altri casi, è meglio evidenziare il pulsante con un singolo clic (oppure utilizzando la tastiera) e selezionarlo utilizzando RETURN o facendo doppio clic.

Capitolo 10 -- Gestione delle finestre di dialogo

Gestione di caselle ed attributi

Espressioni delle azioni e funzioni di richiamo

Azioni di default e DCL

La funzione **action_tile** non rappresenta l'unico modo per specificare un'azione. In AutoLISP, la descrizione DCL di una casella può comprendere l'attributo di un'azione e la funzione **new_dialog** può specificare un'azione di default per l'intera finestra di dialogo. Una

casella può contenere solo una singola azione alla volta. Se nel DCL e nell'applicazione vengono specificate più azioni, esse avranno il seguente ordine di priorità.

- 1 L'azione di default specificata tramite la funzione **new_dialog** (usata solo se non vi è alcuna azione esplicitamente assegnata alla casella).
- 2 L'azione specificata tramite l'attributo contenuto nel file DCL.
- 3 L'azione assegnata tramite la **action_tile** (livello di priorità più alto).

Capitolo 10 -- Gestione delle finestre di dialogo

Gestione di caselle ed attributi

Gestione delle caselle

Il programma utilizzato esercita il proprio controllo sulle caselle contenute nella finestra di dialogo corrente al momento dell'inizializzazione e dell'esecuzione di un'azione (funzione di richiamo). La presente sezione introduce le funzioni di gestione delle caselle ed indica come inizializzare e modificare le modalità ed i valori delle caselle.

Capitolo 10 -- Gestione delle finestre di dialogo

Gestione di caselle ed attributi

Gestione delle caselle

Inizializzazione di modalità e valori

L'inizializzazione di una casella può:

- Rendere la casella il primo oggetto attivo e pronto per ricevere l'input dalla tastiera nella finestra di dialogo.
- Disattivare o attivare la casella stessa
- Evidenziarne il contenuto, se si tratta di una casella di modifica o di un'immagine.

Queste operazioni vengono effettuate tramite le chiamate **mode_tile**. Per impostare il valore di una casella, è possibile utilizzare la funzione **set_tile**.

Per visualizzare un valore di default in una casella di modifica, come un cognome, ed impostare come primo oggetto attivo della finestra di dialogo la casella stessa, utilizzare il seguente codice:

```
(setq name_str "Sartori")   Default
(set_tile "cognome" name_str)   Inizializza il campo
(mode_tile "cognome" 2)   2 rende la casella rif. primario
```

Un'ulteriore funzione **mode_tile** può evidenziare tutto il contenuto di una casella di modifica, consentendo di digitare immediatamente sopra il contenuto di default, come mostra il seguente esempio.

```
(mode_tile "cognome" 3)   3 seleziona il contenuto della casella
```

Nota In alcune piattaforme, l'impostazione del fuoco su una casella di modifica comporta l'evidenziazione del contenuto della casella stessa, rendendo inutile questo ulteriore passo (nonostante sia possibile comunque effettuarlo).

Capitolo 10 -- Gestione delle finestre di dialogo

Gestione di caselle ed attributi

Gestione delle caselle

Modifica di modalità e valori al momento dell'esecuzione di una funzione di richiamo

Al momento dell'esecuzione di una funzione di richiamo, è possibile verificare il valore di una casella. Se l'applicazione richiama la casella, è possibile utilizzare nuovamente la funzione **set_tile** per modificare questo valore. Durante le funzioni di richiamo, è possibile anche utilizzare la funzione **mode_tile** per modificare lo stato di una casella. La seguente tabella riporta i valori dell'argomento della *modalità* della funzione **mode_tile**.

Codici della modalità della casella per la funzione *mode_tile*

Valore	Descrizione
0	Abilita la casella.
1	Disabilita la casella.
2	Imposta l'evidenziazione sulla casella.
3	Seleziona il contenuto della casella di modifica.
4	Abilita e disabilita l'evidenziazione dell'immagine.

Quando si utilizza la funzione **mode_tile** per disattivare una casella impostata attualmente come prima casella attiva e pronta a ricevere l'input, è necessario richiamare nuovamente la funzione **mode_tile** per impostare un'altra casella come prima casella attiva (nella maggior parte dei casi, la successiva tabulazione all'interno della finestra di dialogo). Altrimenti, rimane impostata come prima casella attiva la casella disattivata causando un'incongruenza ed eventuali errori.

Un valido esempio di una casella che "disattiva se stessa" è rappresentato da una serie di "pagine" di una finestra di dialogo che l'utente scorre tramite il pulsante Successivo o Precedente. Quando viene scelto il pulsante Successivo sulla pagina che precede l'ultima, il pulsante viene disattivato. La stessa cosa avviene dopo aver scelto il pulsante Precedente sulla seconda pagina. In entrambi i casi, il codice deve disattivare il pulsante scelto e, quindi, impostare un'altra casella come riferimento.

Si supponga che la casella denominata "group_on" sia un interruttore che controlla un cluster denominato "gruppo". Quando l'interruttore è disattivato, le caselle contenute nel cluster non sono attive e non devono essere modificate. In questo caso, è possibile definire per l'interruttore la seguente azione. È da notare l'uso del carattere di controllo "\", che consente di inserire le virgolette all'interno di un argomento **action_tile**.

```
(action_tile "group_on" "(mode_tile \"gruppo\" (- 1 (atoi $value)))")
```

La sottrazione e la funzione **atoi** nell'espressione dell'azione impostano l'argomento *modalità* della funzione **mode_tile**. Dal momento che un interruttore è impostato su 0 quando è spento e su 1 quando è acceso, la sottrazione inverte il valore e la *modalità* controlla se il cluster è attivo.

Utilizzando la funzione **get_attr** è possibile verificare altri attributi oltre al valore della casella. Ad esempio, è possibile richiamare l'etichetta di un pulsante denominato "premimi".

```
(get_attr "premimi" "etichetta")
```

Nota Se viene utilizzata la funzione **get_attr** per richiamare l'attributo di un valore, viene richiamato l'attributo contenuto nel file DCL (il valore iniziale del file). Tuttavia, la funzione **get_tile** richiama il valore di esecuzione corrente della casella. I due valori non risultano necessariamente uguali.

La funzione **get_attr** restituisce il valore dell'attributo specificato sotto forma di stringa.

Capitolo 10 -- Gestione delle finestre di dialogo

Gestione di caselle ed attributi

Gestione delle caselle

Gestione di cluster di scelta

Nei cluster di scelta appaiono i pulsanti di scelta. Il valore di ogni pulsante è "1" quando è attivo oppure "0" quando non lo è. Il valore del cluster di scelta è l'attributo chiave del pulsante correntemente selezionato. I valori dei pulsanti di scelta all'interno di un cluster sono gestiti dal pacchetto PDB (Programmable Dialog Box, finestre di dialogo programmabili), che assicura che vi sia solo un pulsante attivo alla volta. È possibile assegnare un'azione a ogni pulsante di scelta, ma è consigliabile assegnare un'azione all'intero cluster di scelta e, quindi, controllare il valore del cluster per verificare quale pulsante è stato scelto.

Nel seguente esempio, un cluster di scelta controlla quale vista di un oggetto tridimensionale verrà visualizzata dopo l'uscita dalla finestra di dialogo. Il cluster contiene quattro pulsanti di scelta (è possibile avere un numero maggiore di pulsanti).

```
(action_tile "view_sel" "(pick_view $value)")
.
.
.
(defun pick_view (which)
  (cond
    ((= which "di fronte") (setq show_which 0))
    ((= which "in alto") (setq show_which 1))
    ((= which "sinistra") (setq show_which 2))
    ((= which "destra") (setq show_which 3))
  )
)
```

Questi esempi mostrano ogni pulsante di scelta associato ad una singola variabile che acquisisce valori diversi. Le variabili possono comportare anche ulteriori azioni, come ad esempio la disattivazione delle selezioni nella finestra di dialogo. Se il cluster di scelta è ampio, è possibile memorizzare in una tabella i relativi valori. In tal caso, è opportuno strutturare la tabella in modo che non dipenda dall'ordine dei pulsanti all'interno del cluster. Il pacchetto PDB non comporta questa limitazione e l'ordine può variare con le modifiche del DCL.

Capitolo 10 -- Gestione delle finestre di dialogo

Gestione di caselle ed attributi

Gestione delle caselle

Gestione dei dispositivi di scorrimento

Quando le azioni e le funzioni di richiamo vengono gestite dai dispositivi di scorrimento, l'applicazione dovrebbe verificare il codice di causa ricevuto con la funzione di richiamo. L'utente non deve necessariamente verificare il codice di causa, ma è consigliabile effettuare tale verifica che riduce l'elaborazione. La frequenza di funzioni di richiamo generate dai dispositivi di scorrimento dipende dalla piattaforma, ma alcune piattaforme generano una funzione di richiamo codice 1 per *ogni* movimento del mouse rilevato dal dispositivo.

La seguente funzione mostra lo schema di base che una funzione deve avere per gestire un dispositivo di scorrimento. Esso viene richiamato da un'espressione di azione associata alla casella del dispositivo. La casella slider_info utilizzata dalla funzione visualizza il valore corrente del dispositivo in formato decimale. Questa casella spesso è anche una casella di modifica che consente di manipolare il dispositivo di scorrimento oppure di immettere direttamente il relativo valore. Se invece il valore viene immesso nella slider_info, la funzione di richiamo della casella di modifica dovrebbe aggiornare il valore relativo al dispositivo di scorrimento.

```
(action_tile
  "mia_diapos"
  "(slider_action $value $reason)"
)
(action_tile
  "slider_info"
  "(ebox_action $value $reason)"
)
.
.
.
(defun slider_action(val why)
  (if (or (= why 2) (= why 1))
    (set_tile "info_dia" val) Mostra il risultato temporaneo
  )
)
(defun ebox_action(val why)
  (if (or (= why 2) (= why 1))
    (set_tile "mia_diapos" val) Mostra il risultato temporaneo
  )
)
```

Capitolo 10 -- Gestione delle finestre di dialogo

Gestione di caselle ed attributi

Gestione delle caselle

Gestione delle caselle di modifica

Le azioni e le funzioni di richiamo per la gestione delle caselle di modifica sono simili a quelle per i dispositivi di scorrimento. Tuttavia, dal momento che i caratteri all'interno delle caselle di modifica sono già visibili, non è necessario effettuare alcuna azione sui risultati temporanei. Le caselle di modifica restituiscono un codice di richiamo quando viene modificato il loro stato da attivo a disattivo.

Il seguente esempio di codice verifica il valore, ma non lo rivisualizza.

```
(action_tile "mia_casella_mod" "(edit_action $value $reason)")
.
.
.
(defun edit_action (val why)
  (if (or (= why 2) (= why 4))
      .
      .
      .
  )
)
```

Controlla se il valore temporaneo è compreso in un intervallo

Capitolo 10 -- Gestione delle finestre di dialogo

Gestione di caselle ed attributi

Finestre di dialogo nidificate

Le finestre di dialogo nidificate vengono create e gestite richiamando le funzioni **new_dialog** e **start_dialog** da un' espressione di azione oppure da una funzione di richiamo. Ad esempio, includendo la seguente istruzione, la funzione può visualizzare la casella "Ciao a tutti" nel momento in cui viene scelto il pulsante button_1.

```
(action_tile "button_1" "(c:ciao)")
```

Prima di utilizzare nuovamente la finestra di dialogo precedente, è necessario uscire dalla finestra nidificata.

Nota AutoCAD impone un limite di otto finestre di dialogo nidificate. Tuttavia, allo scopo di evitare confusione, si consiglia di non nidificare più di quattro finestre di dialogo.

Capitolo 10 -- Gestione delle finestre di dialogo

Gestione di caselle ed attributi

Come nascondere le finestre di dialogo

Quando è attiva una finestra di dialogo, non è possibile effettuare una selezione interattiva. Se si desidera effettuare una selezione dallo schermo grafico, è necessario *nascondere* la finestra di dialogo e, quindi, ripristinarla. Nascondere una finestra equivale a chiuderla utilizzando la funzione **done_dialog**, l'unica differenza è che la funzione di richiamo deve utilizzare l'argomento *stato della funzione done_dialog* per indicare che la finestra di dialogo è nascosta e non chiusa o annullata. Impostare l'argomento *stato* per un valore definito dall'applicazione. Quando la finestra di dialogo scompare, la funzione **start_dialog** restituisce l'argomento *stato* definito dall'applicazione. Il programma deve quindi esaminare lo stato restituito dalla funzione **start_dialog** per stabilire quale sarà l'azione successiva da intraprendere. Tale operazione implica la necessità di bloccare le risposte della funzione **start_dialog**

racchiudendo la funzione stessa in una iterazione, in modo da permettere di visualizzare nuovamente la finestra di dialogo dopo averla nascosta. Per i valori *distato* standard e definiti dall'applicazione, vedere "**done_dialog**" in capitolo 13,.

Il programma di esempio *bmake.lsp* dispone di un pulsante, Punto di selezione, che nasconde la finestra di dialogo in modo da permettere di specificare un punto sullo schermo grafico. La scelta di questo pulsante comporta la chiusura della finestra con uno *stato* speciale 4.

```
(action_tile "pick_pt" "(done_dialog 4)")
```

Dopo le risposte della funzione **start_dialog**, il programma verifica lo stato della risposta ed aggiorna, se necessario, una variabile *punto*.

```
(setq what_next (start_dialog))
(cond
  (Stabilisce l'azione successiva
   ; Se è selezionato un punto base . . .
   ((= what_next 4)
    (setq pick_pt (getpoint "Punto di inserimento base: "))
    (setq x_pt (rtos (car pick_pt) 2 4))
    (setq y_pt (rtos (cadr pick_pt) 2 4))
    (setq z_pt (rtos (caddr pick_pt) 2 4))
   )
  )
)
```

Il corpo della funzione principale è racchiuso in un'iterazione in modo che le funzioni **new_dialog** e **start_dialog** (insieme alle funzioni **action_tile**, **set_tile**, **start_list** e simili) vengano richiamate di nuovo fino a quando non viene scelto il pulsante OK o Annulla. Il seguente codice di esempio è parte dell'applicazione di esempio *bmake.lsp*.

```
(defun bmake_main ...
  (; Carica la finestra di dialogo ed avvial' inizializzazione globale
   ; Lo stato indica che la finestra è nascosta
   ; Inizializzazione di una singola finestra di dialogo come
   ; new_dialog, action_tile, set_tile,
   ; e start_list
   (while (< 2 what_next)
     (setq what_next
           (start_dialog)
          )
     (cond
       (; Verifica se WHAT_NEXT indica che la finestra
        ; è nascosta ed intraprende l'azione adeguata
        )
       .
       .
       .
     )
   )
)
```

Se la finestra di dialogo è nidificata, la procedura risulta più complicata ma essenzialmente la stessa.

```
(defun c:maindlg ( / what_next what_next1 )
  (setq what_next 5)
  (if
    (<
      (setq dcl_id
            (load_dialog "maindlg.dcl")
           )
      0)
    (exit)
  )
  (while (< 1 what_next)
    (; Rimane valida l'iterazione fino alla scelta di OK
     ; 0 Annulla
     ; Imposta azioni e set_tiles
     (new_dialog "maindlg" dcl_id)
     (action_tile "x" "(subdlg)")
     (cond
       (; Vera solo se viene restituito un punto specifico
        ; Richiama la routine SUBDLG
        ; Inizializza la finestra di dialogo principale
        ((= what_next1 3)
         (subdlg)
         (if (/= 3 what_next1)
           (setq what_next
                 (start_dialog)
                )
         )
        )
      )
    )
  )
)
```



```

(setq dcl_id (load_dialog "getpass.dcl"))
(if (new_dialog "passdlg" dcl_id)
  (progn
    (action_tile "parola d'ordine" "(setq pass $value)")
    (start_dialog)
    (unload_dialog dcl_id)
  )
  (princ "Errore: Impossibile caricare GETPASS.DCL. ")
)
pass
)

```

La funzione **GETPASS** ritorna la stringa immessa dall'utente.

Capitolo 10 -- Gestione delle finestre di dialogo

Caselle di riepilogo ed elenchi a comparsa

AutoLISP fornisce delle funzioni per la gestione delle caselle di riepilogo e degli elenchi a comparsa presenti nelle finestre di dialogo. Tali funzioni sono le seguenti:

`add_list` `(end_list)` `start_list`

Gli elenchi visualizzati nelle caselle di riepilogo e negli elenchi a comparsa vengono impostati utilizzando una sequenza di chiamate a queste tre funzioni. Una volta creato un elenco, è possibile revisionarlo. A tale scopo, vi sono tre operazioni possibili, ognuna delle quali viene specificata tramite l'argomento *operazione* della funzione **start_list**. I valori dell'argomento sono riportati tra parentesi.

- Creazione di un nuovo elenco (3). Questa è l'operazione di default.

Dopo la chiamata alla funzione **start_list**, è possibile richiamare ripetutamente la funzione **add_list**. Ogni chiamata alla funzione **add_list** aggiunge un nuovo elemento all'elenco. Terminare la gestione dell'elenco richiamando la funzione **end_list**.

- Modifica di un elemento dell'elenco (1).

Dopo la chiamata alla funzione **start_list**, richiamare la funzione **add_list** *una volta* per sostituire l'elemento per il quale è stato specificato l'indice nella funzione **start_list**; se la funzione **add_list** viene richiamata più di una volta, viene sostituito di nuovo lo stesso elemento. Terminare la gestione dell'elenco richiamando la funzione **end_list**. Il primo elemento nell'elenco ha un indice uguale a 0.

- Aggiunta di un elemento all'elenco (2).

Dopo la chiamata alla funzione **start_list**, richiamare la funzione **add_list** per aggiungere un elemento alla fine dell'elenco. Continuando a richiamare la funzione **add_list**, vengono aggiunti ulteriori elementi all'elenco fino a quando non viene richiamata la funzione **end_list**.

Indipendentemente dall'operazione che si sta effettuando sull'elenco, è necessario richiamare le tre funzioni rispettando la sequenza: **start_list**, **add_list** (possibilmente più volte) quindi **end_list**.

La funzione **mapcar** consente di trasformare un elenco AutoLISP "grezzo" nella visualizzazione di una casella di riepilogo. Nell'esempio seguente, l'elenco `appnames` contiene le stringhe che si desidera vengano visualizzate nella casella di riepilogo denominata `selezioni`; è possibile utilizzare questa parte di codice per impostare e visualizzare l'elenco.

```

(start_list "selezioni") ;Specifica il nome della casella di riepilogo
(mapcar 'add_list appnames) ;Specifica l'elenco AutoLISP
(end_list)

```

Dal momento che in AutoLISP l'operazione 3, relativa alla creazione di un elenco, è la soluzione di default, questo esempio non specifica tale operazione.

Il valore di una casella `list_box` è costituito dall'indice o dagli indici dell'elemento o degli elementi selezionati. Se il programma utilizzato richiede di conoscere il testo effettivo associato ad un indice, è necessario salvare l'elenco originale. Inoltre, il programma deve tenere traccia delle modifiche apportate utilizzando i metodi riportati nei seguenti esempi.

L'aggiunta di elementi ad un elenco è simile alla creazione di un nuovo elenco. Se, ad esempio, `nomiapp` contiene 12 elementi e si desidera aggiungere un altro elenco denominato `nuovinomi`, utilizzare il seguente codice:

```

(start_list "selezioni" 2)
(mapcar 'add_list nuovinomi)

```

```
(end_list)
```

Poiché l'aggiunta di elementi ad un elenco non è l'operazione di default, l'operazione sull'elenco (codice 2) deve essere specificata in modo esplicito.

La modifica di un singolo elemento richiede una sola chiamata alla funzione **add_list**. In questo caso, specificare l'indice dell'elemento da modificare:

```
(start_list "selezioni" 1 5);Modifica il sesto elemento dell'elenco
(add_list "SORPRESA!")          ;Ricordare che il primo indice è 0
(end_list)
```

Nota Non è possibile cancellare o inserire un elemento in un elenco senza dover costruire di nuovo l'intero elenco.

Poiché il valore di una casella list_box può contenere spazi iniziali (specialmente se vengono richiamati più elementi), non è necessario controllare il valore come un confronto di stringhe. In primo luogo, è necessario convertire il valore in un numero intero con la funzione **atoi**, prima di elaborare la casella di riepilogo. È possibile anche utilizzare la funzione **read**, che converte automaticamente un token in un numero intero. Se l'elenco denominato **justone** accetta soltanto una selezione singola, la seguente parte di codice verifica se è stato selezionato il terzo elemento dell'elenco. In primo luogo, è necessario verificare se la stringa è vuota, in quanto le funzioni **atoi** restituiscono 0 per una stringa vuota così come per una stringa "0".

```
(setq index (get_tile "solouno"))
(cond
  ((/= index "")
    (= 2 (atoi index))
    ; Elaborazione del terzo elemento
    ...
  )
)
```

Nota Il valore di un elenco a comparsa non contiene mai uno spazio iniziale, quindi non è necessario effettuare la conversione. Gli elenchi a comparsa non consentono selezioni multiple.

Se la casella di riepilogo prevede selezioni multiple, il programma utilizzato deve eseguire la conversione e scorrere i diversi valori contenuti nella stringa. La seguente definizione di **MK_LIST** restituisce un elenco contenente soltanto gli elementi che sono stati selezionati dal **displist** originale. Nel seguente esempio, l'elenco di visualizzazione **displist** viene gestito come variabile globale. Si prevede che la funzione **MK_LIST** venga richiamata con il valore \$value corrente della casella di riepilogo.

```
(defun MK_LIST (readlist / count item retlist)
  (setq count 1)
  (while (setq item (read readlist))
    (setq retlist (cons (nth item dispelist) retlist))
    (while (and (/= " " (substr readlist count 1))
      (/= "" (substr readlist count 1)))
      (setq count (1+ count))
    )
    (setq readlist (substr readlist count))
  )
  (reverse retlist)
)
```

Entrambi gli esempi precedenti sono validi anche nel caso di un'unica selezione.

Capitolo 10 -- Gestione delle finestre di dialogo

Pulsanti e gruppi di immagini

AutoLISP fornisce le seguenti funzioni per la gestione dei gruppi di immagini e dei pulsanti immagine.

dimx_tile	end_image	slide_image	vector_image
dimy_tile	fill_image	start_image	

Esempi sull'uso di queste funzioni sono forniti nella sezione seguente. Queste funzioni sono descritte anche nel capitolo 13.

Capitolo 10 -- Gestione delle finestre di dialogo

Pulsanti e gruppi di immagini

Creazione di immagini

La sequenza di richiamo per la creazione di immagini per gruppi di immagini e pulsanti immagine è simile alla sequenza di gestione degli elenchi. La funzione **start_image** avvia la creazione di un'immagine, la funzione **end_image** termina l'operazione. Tuttavia, le opzioni relative agli oggetti da disegnare vengono specificate da chiamate separate a funzioni invece che da argomenti.

vector_image

Disegna un vettore (una singola linea retta) nell'immagine corrente.

fill_image

Disegna un rettangolo riempito nell'immagine corrente.

slide_image

Disegna una diapositiva di AutoCAD nell'immagine.

I vettori ed i rettangoli riempiti sono utili per le immagini semplici, come colori campione (rettangoli riempiti) che la finestra di dialogo per la selezione dei colori di AutoCAD utilizza per visualizzare la scelta di colori effettuata. Per immagini complesse, conviene utilizzare le diapositive. Tuttavia, la visualizzazione delle diapositive richiede più tempo. Se vengono utilizzate le diapositive, è consigliabile fare in modo che siano semplici.

Nota Se si utilizzano le diapositive con oggetti riempiti (polilinee spesse, solidi e facce tridimensionali) in caselle di immagini, le immagini appariranno come contorni a meno che non si creino le immagini con il comando OMBRA.

La funzione per il disegno delle immagini, **vector_image**, richiede di specificare coordinate assolute, mentre le funzioni **fill_image** e **slide_image** richiedono di specificare una coordinata iniziale insieme ai relativi valori di larghezza ed altezza. Per fornire questi valori in modo corretto, è necessario conoscere le dimensioni esatte del gruppo di immagini o del pulsante immagine. Dal momento che tali dimensioni vengono, in genere, assegnate quando la finestra di dialogo è visualizzata, il pacchetto PDB fornisce funzioni che restituiscono i valori relativi alla larghezza ed all'altezza di una determinata casella. AutoLISP dispone di due funzioni relative a queste dimensioni: **dimx_tile** e **dimy_tile**. È consigliabile richiamare tali funzioni prima di creare un'immagine. L'origine di una casella, (0,0), è sempre il relativo angolo superiore sinistro.

I colori possono essere specificati come numeri di colore di AutoCAD oppure come uno dei numeri di colore "logici" riportati nella seguente tabella. I valori ed i caratteri mnemonici vengono definiti da ADI (Autodesk Device Interface).

Nomi simbolici per l'attributo colore

Numero colore	Mnemonico ADI	Significato
-2	BGLCOLOR	È lo sfondo corrente dello schermo grafico di AutoCAD.
-15	DBGLCOLOR	È il colore di sfondo corrente della finestra di dialogo.
-16	DFGLCOLOR	È il colore di primo piano corrente della finestra di dialogo (per testo).
-18	LINELCOLOR	È il colore della linea corrente della finestra di dialogo.

Nel seguente esempio, "cur_color" è un gruppo di immagini che si desidera riempire interamente di colore rosso.

```
(setq width (dimx_tile "cur_color")
      height (dimy_tile "cur_color"))
(start_image "cur_color")
(fill_image 0 0 width height 1)    1 = Rosso di AutoCAD
(end_image)
```

Le funzioni per il disegno delle immagini possono essere utilizzate insieme. Nel seguente esempio, il codice riempie un'immagine, quindi disegna una riga verticale sul disegno stesso.

```
(setq width (dimx_tile "stripe")
      height (dimy_tile "stripe"))
(start_image "stripe")
(fill_image 0 0 width height 3)    ;3 = Verde di AutoCAD
```

```
(setq x (/ width 2))           ; Centra il vettore verticalmente
(vector_image x 0 x height 4) ; 4 = Ciano di AutoCAD
(end_image)
```

Le diapositive visualizzate con la funzione **slide_image** possono essere file indipendenti (.sld) oppure possono fare parte di un file di libreria di diapositive (.slb). Se la diapositiva si trova in un file .sld, il nome del file può essere specificato senza l'estensione .sld (ad esempio, "frntview"). Se la diapositiva si trova in una libreria di diapositive, specificare prima il nome della libreria, senza l'estensione, seguito dal nome della diapositiva stessa, sempre senza l'estensione, racchiuso tra parentesi (ad esempio, "allviews(frntview)"). La funzione **slide_image** effettua la ricerca del file contenente la diapositiva o della libreria di diapositive in base al percorso di ricerca della libreria di AutoCAD corrente "**load_dialog**" nel capitolo 13,).

Nel seguente esempio, la diapositiva è un file singolo denominato *topview.sld*.

```
(setq x (dimx_tile "view")
      y (dimy_tile "view"))
(start_image "view")
(slide_image 0 0 x y "topview")
(end_image)
```

I vettori nelle diapositive vengono spesso disegnati in bianco (colore numero 7), che è il colore di sfondo di default per un'immagine. Se quando si visualizza per la prima volta una diapositiva il gruppo di immagini è vuoto, tentare di modificare l'attributo colore in **graphics_background**. È anche possibile modificare lo sfondo dell'immagine facendo precedere la funzione **fill_image** alla funzione **slide_image**.

Capitolo 10 -- Gestione delle finestre di dialogo

Pulsanti e gruppi di immagini

Gestione dei pulsanti immagine

Un pulsante immagine può essere gestito come un semplice pulsante, ovvero può essere utilizzato per avviare una singola azione. Tuttavia, è possibile utilizzare la funzione PDB anche per definire le regioni del pulsante in modo che l'azione intrapresa dipenda dalla *parte* del pulsante immagine selezionata. Il meccanismo che è alla base di questo concetto è evidente: l'azione di un pulsante immagine o di una funzione di richiamo restituisce la posizione (X,Y) selezionata. Le coordinate sono comprese nell'intervallo della casella specifica del pulsante immagine, così come sono state restituite dalle funzioni relative alle dimensioni. L'applicazione deve assegnare un significato alle posizioni di selezione definendo in modo implicito le regioni dell'immagine. La finestra di dialogo DDPUNTOV utilizza questa caratteristica.

Nel seguente esempio, il pulsante immagine ha due colori campione creati con la funzione **fill_image**. È possibile selezionare uno dei due colori in base alla regione selezionata. Se il pulsante immagine è diviso orizzontalmente (parte scura sopra e chiara sotto), l'azione deve verificare soltanto una dimensione.

```
(action_tile "image_sel" "(pick_shade $key $value $y)")
...
(defun pick_shade (key val y)
  (setq threshold
    (/ ( dimy_tile key) 2))           ; L'immagine è divisa orizzontalmente
  (if (> y threshold)                ; Ricordare che l'origine è nell'angolo
    (setq result "Chiaro")           ; superiore sinistro
    (setq result "Scuro") )
)
```

Capitolo 10 -- Gestione delle finestre di dialogo

Dati specifici dell'applicazione

AutoLISP fornisce la funzione **client_data_tile** per la gestione dei dati specifici dell'applicazione. La funzione **client_data_tile** assegna tali dati ad una casella. I dati sono disponibili al momento della funzione di richiamo come la variabile \$data e devono essere sotto forma di stringa. I dati client non vengono rappresentati in DCL, ma sono validi soltanto durante l'esecuzione dell'applicazione. L'uso dei dati client equivale all'uso degli attributi definiti dall'utente. La differenza principale è che gli attributi definiti dall'utente sono di sola lettura, mentre i dati client possono subire delle variazioni al momento dell'esecuzione. Inoltre, gli utenti finali possono verificare gli attributi definiti dall'utente contenuti nel file DCL dell'applicazione, mentre i dati client non sono visibili.

Dal momento che il programma utilizzato deve mantenere l'elenco visualizzato tramite una casella di riepilogo (o da un elenco a comparsa), i dati client risultano utili per la gestione di queste informazioni. La seguente modifica alla definizione della funzione **MK_LIST** trasforma l'elenco in un argomento (l'intera funzione è riportata in "Caselle di riepilogo ed elenchi a comparsa").

```
(defun MK_LIST ( readlist displist / )
```

Questo codice elimina la necessità di utilizzare una variabile elenco globale. Le seguenti chiamate contenute nella parte principale del gestore della finestra di dialogo associano un breve elenco alla casella richiamando la funzione **client_data_tile**, quindi passano l'elenco alla funzione **MK_LIST** tramite un'espressione dell'azione.

```
(client_data_tile
  "colorsyslist"
  "Red-Green-Blue Cyan-Magenta-Yellow Hue-Saturation
)
(action_tile
  "colorsyslist"
  "(setq usrchoice (mk_list $value $data))"
)
```

Capitolo 10 -- Gestione delle finestre di dialogo

Riepilogo delle funzioni delle finestre di dialogo

Questa sezione fornisce un riepilogo delle funzioni di gestione delle finestre di dialogo e descrive i concetti esposti precedentemente in questo capitolo. Viene inoltre descritta un'applicazione di esempio a cui fare riferimento quando si progettano funzioni per le finestre di dialogo.

Capitolo 10 -- Gestione delle finestre di dialogo

Riepilogo delle funzioni delle finestre di dialogo

Sequenza di funzioni

Il seguente elenco mostra la tipica sequenza delle funzioni.

- 1 Caricamento del file DCL con la funzione **load_dialog** .
- 2 Richiamo della funzione **new_dialog** per visualizzare una determinata finestra di dialogo.
È necessario verificare il valore restituito dalla funzione **new_dialog**. Se la funzione **start_dialog** viene richiamata quando la funzione **new_dialog** ha avuto esito negativo, i risultati sono imprevedibili.
- 3 Inizializzazione della finestra di dialogo tramite l'impostazione di valori di caselle, elenchi ed immagini. Questa operazione deve essere effettuata anche quando viene richiamata la funzione **action_tile** per impostare le espressioni delle azioni o delle funzioni di richiamo. Le altre funzioni che vengono, in genere, richiamate in questo caso sono **set_tile** e **mode_tile** per valori e stati di caselle generiche; **start_list**, **add_list** e **end_list** per le caselle di riepilogo e le funzioni per le dimensioni con **start_image**, **vector_image**, **fill_image**, **slide_image** e **end_image** per le immagini. A questo punto, è possibile anche richiamare la funzione **client_data_tile** per associare i dati specifici dell'applicazione alla finestra di dialogo ed ai relativi componenti.
- 4 Richiamo della funzione **start_dialog** per attivare il controllo sulla finestra di dialogo in modo da consentire all'utente di effettuare le immissioni.
- 5 Elaborazione dell'immissione effettuata dall'utente dall'interno delle azioni (funzioni di richiamo). Questa operazione deve essere eseguita quando si intende utilizzare le funzioni **get_tile**, **get_attr**, **set_tile** e **mode_tile**.
- 6 Quando viene premuto il pulsante di uscita, un'azione richiama la funzione **done_dialog**, che fa in modo che la funzione **start_dialog** restituisca un valore. A questo punto, è necessario scaricare il file DCL richiamando la funzione **unload_dialog**.

Questo schema gestisce soltanto una finestra di dialogo ed un file DCL alla volta. Le applicazioni in genere dispongono di più finestre di dialogo. Il modo più semplice e veloce per gestire queste finestre è di salvarle tutte in un unico file DCL. In questo modo, la funzione **load_dialog** carica tutte le finestre di dialogo contemporaneamente, consentendo di richiamare la funzione **new_dialog** per qualsiasi finestra. Tuttavia, se vi sono problemi di memoria insufficiente, è possibile che sia necessario creare più file DCL ed

utilizzare la funzione **unload_dialog** per rimuovere una serie di finestre dalla memoria prima di caricare un'altra serie tramite la funzione **load_dialog**.

Capitolo 10 -- Gestione delle finestre di dialogo

Riepilogo delle funzioni delle finestre di dialogo

Finestra di dialogo di definizione dei blocchi di esempio

L'applicazione di esempio *bmake.lsp* ed il relativo file *bmake.dcl* forniscono informazioni relative ad una serie di tecniche utilizzate per le finestre di dialogo. Questi file si trovano nella directory *sample*. L'applicazione *bmake* è un'interfaccia interattiva per la funzione **entmake** di AutoLISP. Può essere utilizzata per definire nuovi blocchi e visualizzare i nomi dei blocchi esistenti. Alcune delle tecniche riportate dall'applicazione *bmake* sono le seguenti:

- Nascondere le finestre di dialogo definendo dei codici di stato speciali per la funzione **done_dialog** per passare alla funzione **start_dialog**. Vedere l'iterazione principale della funzione **C:MAKE** (in seguito alle funzioni **load_dialog** ed **action_tile**).
- Utilizzare un interruttore per attivare o disattivare un'altra casella. Vedere la definizione della funzione **DO_UNNAMED**.
- Creare un elenco per una casella di riepilogo. Vedere le funzioni **PAT_MATCH** e **SORT**.
- Visualizzare la finestra di dialogo di aiuto standard di AutoCAD. Vedere la funzione **DO_HELP**.

Oltre a mostrare le tecniche utilizzate per le finestre di dialogo, *bmake* illustra un disegno.

Capitolo 11 -- Corso interattivo di AutoLISP

Panoramica

Questo corso interattivo mostra come aggiungere un nuovo comando ad AutoCAD, come funziona AutoLISP e come è possibile utilizzare al massimo le sue potenzialità. AutoLISP è stato originariamente creato per l'architettura paesaggistica, ma i concetti che si apprenderanno rimangono validi a prescindere dall'area di applicazione.

Questo corso interattivo è destinato ad utenti di AutoCAD già esperti. Per eseguire le operazioni richieste, sarà necessario un editor di testo in grado di produrre file ASCII.

Argomenti di questo capitolo

{button ,JI(','The_Goal_al_u0305')} Obiettivo

{ewc ,JI(','Getting_Started_al_u0305')} Introduzione

{button ,JI(','Getting_Input_al_u0305')} Come ottenere input

{button ,JI(','Getting_Oriented_al_u0305')} Orientamento

{ewc ,JI(','Drawing_the_Tiles_al_u0305')} Come disegnare le piastrelle

{button ,JI(','Adding_the_Command_to_AutoCAD_al_u0305')} Come aggiungere il comando ad AutoCAD

{button ,JI(','Adding_a_Dialog_Box_Interface_al_u0305')} Come aggiungere un'interfaccia a finestra di dialogo

Capitolo 11 -- Corso interattivo di AutoLISP

Obiettivo

Lo scopo di questo corso interattivo è quello di sviluppare per AutoCAD un nuovo comando che disegni un vialetto da giardino e lo riempia con piastrelle di cemento circolari. Il messaggio di richiesta del nuovo comando presenterà questa sequenza:

Comando: **path**

Punto iniziale del vialetto: *specificare il punto iniziale*

Punto finale del vialetto: *specificare il punto finale*

Mezza larghezza del vialetto: *specificare un numero*

Raggio delle piastrelle: *specificare un numero*

Spaziatura tra le piastrelle: *specificare un numero*

Viene digitato il *punto iniziale* e il *punto finale* per specificare la linea del centro di un vialetto. Poi viene digitata la mezza larghezza del vialetto ed il raggio delle piastrelle circolari. Infine, viene digitata la spaziatura fra le piastrelle. Viene specificata la mezza larghezza del vialetto invece della larghezza totale perché dal punto iniziale è più facile visualizzare la prima.

Capitolo 11 -- Corso interattivo di AutoLISP

Introduzione

Questa applicazione viene sviluppata dall'interno all'esterno o dal basso verso l'alto e vengono frequentemente utilizzati angoli. AutoLISP specifica gli angoli in radianti. I radianti misurano gli angoli da 0 a $2 * \pi$ (pi). Dal momento che la maggior parte degli utenti pensa agli angoli in termini di gradi, sarà necessaria una funzione che converta i gradi in radianti. Mediante il proprio editor di testo creare un file chiamato *gp.lsp*. Digitare il programma seguente:

```
; Convertire angolo da gradi a radianti

(defun dtr (a)
  (* pi (/ a 180.0))
)
```

In tal modo si definisce una funzione utilizzando la funzione **defun**. Questa funzione viene chiamata *dtr* (forma abbreviata per *degrees to radians*, gradi in radianti) ed ha un argomento, *a*, l'angolo in gradi. Il risultato è questa espressione:

$$\pi * (a / 180.0)$$

La funzione è espressa in notazione LISP e può essere letta come *il prodotto di p greco moltiplicato per il quoziente di A diviso per 180.0*. *p* è una costante predefinita da AutoLISP come 3.14159.... La riga che inizia con il punto e virgola è un *commento*; dopo un punto e virgola AutoLISP ignora tutto il testo presente sulla riga.

Salvare il file su un disco e poi avviare AutoCAD su un nuovo disegno; il nome non ha alcuna rilevanza perché il disegno non verrà salvato. Alla riga di comando, caricare la funzione digitando:

Comando: (load "gp")
DTR

AutoLISP carica questa funzione mantenendo il nome DTR assegnatole se *gp.lsp* è nel percorso di ricerca di AutoCAD. Da questo momento in poi, *avviare AutoCAD e caricare il programma* si riferirà alla sequenza appena descritta.

Verificare ora la funzione eseguendola con valori diversi. In base alla definizione di radianti, 0 gradi dovrebbe essere uguale a 0 radianti. Digitare quanto segue:

Comando: (dtr 0)

Digitando una riga che comincia con una parentesi aperta viene data ad AutoCAD l'istruzione di passare la valutazione della riga ad AutoLISP. In questo caso, viene valutata la funzione *dtr* appena definita ed alla stessa viene passato l'argomento 0. Dopo l'esecuzione del calcolo, AutoCAD visualizza il risultato, e l'input dovrebbe produrre questa risposta:

0.0

Provare ora con 180 gradi. Digitare quanto segue:

Comando: (dtr 180)

Verrà visualizzata la risposta:

3.14159

Questo significa che 180 gradi è uguale a π radianti. Se si esamina la funzione, è possibile vedere che corrisponde a quanto definito.

A questo punto, è necessario uscire da AutoCAD e ritornare all'editor di testo.

Capitolo 11 -- Corso interattivo di AutoLISP

Come ottenere input

Il comando "path" chiede all'utente la posizione del vialetto da disegnare, la sua larghezza, la dimensione delle piastrelle di cemento e la loro spaziatura. Verrà definita una funzione che chiede all'utente tutte queste informazioni e poi calcola i valori da utilizzare negli altri messaggi di richiesta del comando.

Utilizzando il proprio editor di testo, aggiungere le seguenti righe a *gp.lsp*.

```
; Acquisire le informazioni relative al vialetto da giardino

(defun gpuser ()
  (setq sp (getpoint "\nPunto iniziale del vialetto: "))
  (setq ep (getpoint "\nPunto finale del vialetto: "))
  (setq hwidth (getdist "\nMezza larghezza del vialetto: " sp))
  (setq trad (getdist "\nRaggio delle piastrelle: " sp))
  (setq tspac (getdist "\nSpaziatura fra le piastrelle: " sp))

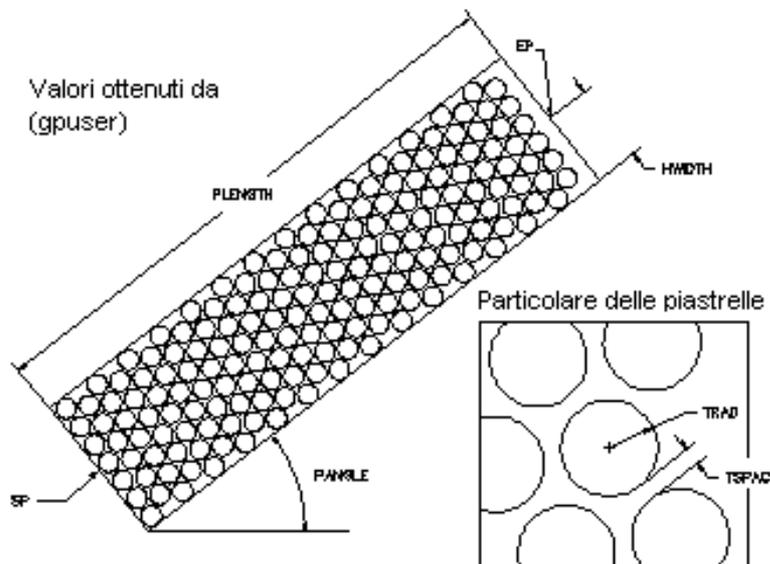
  (setq pangle (angle sp ep))
  (setq plength (distance sp ep))
  (setq width (* 2 hwidth))
  (setq angp90 (+ pangle (dtr 90))) ; Angolo del vialetto + 90 gradi
  (setq angm90 (- pangle (dtr 90))) ; Angolo del vialetto - 90 gradi
)
```

Non è necessario utilizzare i rientri per le espressioni che costituiscono le funzioni. Tuttavia, il rientro e le interruzioni di riga rendono più chiara la struttura del programma. L'allineamento delle parentesi aperte e chiuse delle espressioni principali consente inoltre di controllare il corretto bilanciamento delle parentesi. Per ottenere il rientro delle righe, è consigliabile utilizzare spazi invece di tabulazioni. In tal modo, viene garantito un rientro coerente fra sessioni di modifica ed elaboratori di testo.

In questo esempio è stata definita una funzione chiamata *gpuser* che non utilizza argomenti e chiede all'utente tutti gli elementi desiderati. La funzione *setq* imposta una variabile AutoLISP su un valore specifico. La prima *setq* imposta la variabile *sp* (punto iniziale) sul risultato della funzione *getpoint*, che richiede l'immissione di un punto da parte dell'utente. Una stringa specifica il messaggio di richiesta utilizzato da AutoCAD per ricavare il punto. La funzione *getdist* viene utilizzata per ottenere la mezza larghezza del vialetto, il raggio delle piastrelle e la spaziatura fra di esse. Il secondo argomento nella funzione *getdist*, *sp*, specifica il *punto di base* per la distanza. In questo modo, la distanza, se specificata in AutoCAD con un punto, sarà relativa al punto iniziale del vialetto, dal quale verrà visualizzata una linea elastica.

Dopo aver ottenuto l'input dall'utente, la funzione calcola diverse variabili di uso comune. La variabile *pangle* viene impostata in base all'angolo che va dal punto iniziale al punto finale del vialetto. Dati due punti, la funzione *angle* restituisce quest'angolo. La variabile *plength* è impostata sulla lunghezza del vialetto. La funzione *distance* calcola una distanza basata su due punti. Essendo stata specificata la mezza larghezza del vialetto, è possibile calcolare la larghezza totale raddoppiando quel valore. Infine viene calcolato e salvato l'angolo del vialetto più e meno 90 gradi, rispettivamente in *angp90* ed *angm90*. Dal momento che in AutoLISP gli angoli sono espressi in radianti, prima di effettuare questi calcoli è stata utilizzata la funzione *dtr* per convertire gli angoli in radianti.

La seguente figura mostra in che modo le variabili ottenute da *gpuser* specificano le dimensioni del vialetto.



Salvare su un disco questo programma aggiornato. Quindi avviare AutoCAD e caricarlo. A questo punto, verificare il corretto funzionamento di questa funzione di input. Attivare la funzione digitando:

Comando: **(gpuser)**

Rispondere ai messaggi di richiesta secondo quanto indicato qui di seguito:

Punto iniziale del vialetto: **2,2**

Punto finale del vialetto: **9,8**

Mezza larghezza del vialetto: **2**

Raggio delle piastrelle: **.2**

Spaziatura tra le piastrelle: **.1**

La funzione gpuser utilizza i valori forniti dall'utente per calcolare le altre variabili di cui necessita e poi visualizza il risultato dell'ultimo calcolo, che in questo caso è -0.86217, ovvero il valore di angm90 espresso in radianti. È possibile estrapolare i valori di tutte le variabili impostate dalla funzione gpuser digitando i rispettivi nomi preceduti da un punto esclamativo (!). In tal modo, AutoCAD calcola la variabile e stampa il risultato. Digitando i seguenti comandi, si dovrebbero ottenere i risultati mostrati fra parentesi.

Comando: **!sp**

(2.0 2.0 0.0)

Comando: **!ep**

(9.0 8.0 0.0)

Comando: **!hwidth**

2.0

Comando: **!width**

4.0

Comando: **!trad**

0.2

Comando: **!tspac**

0.1

Comando: **!pangle**

0.708626

Comando: **!plength**

9.21954

Comando: **!angp90**

2.27942

Comando: **!angm90**

-0.86217

Le variabili *sp* ed *ep* vengono restituite come punti tridimensionali (X, Y e Z); ignorare la componente Z in questo esercizio.

Anche *pangle*, *angp90* ed *angm90* sono rappresentate in radianti. Dopo avere verificato questi valori, uscire da AutoCAD, tornare all'editor di testo ed aprire *gp.lsp*.

Capitolo 11 -- Corso interattivo di AutoLISP

Orientamento

Dopo aver chiesto all'utente la posizione del vialetto, è possibile disegnarne il contorno. Aggiungere al proprio file *gp.lsp* le righe seguenti:

```
; Disegnare il contorno del vialetto

(defun drawout ()
  (command "pline"
    (setq p (polar sp angm90 hwidth))
    (setq p (polar p pangle plength))
    (setq p (polar p angp90 width))
    (polar p (+ pangle (dtr 180)) plength)
    "close"
  )
)
```

Questa aggiunta definisce una funzione chiamata *drawout*, che utilizza il punto iniziale, l'angolo e la lunghezza del vialetto ottenuti dalla funzione *gpuser* e disegna il contorno del vialetto come una polilinea. La funzione *drawout* utilizza la funzione *command* per inviare comandi e dati ad AutoCAD. La funzione *command* utilizza un numero qualsiasi di argomenti e li invia, uno alla volta, ad AutoCAD.

In questo caso, viene inviato il comando *PLINEA* ad AutoCAD e quindi vengono forniti i quattro angoli del vialetto. La funzione individua ogni angolo con la funzione *polar* e ne memorizza le coordinate nella variabile temporanea *p*. La funzione *polar* utilizza come primo argomento un punto, nonché un angolo ed una distanza forniti dal secondo e dal terzo argomento, e restituisce un punto alla distanza ed all'angolazione specificate rispetto al punto di origine. In questo caso si calcolano in modo geometrico i quattro punti che delimitano il vialetto dal punto iniziale del vialetto. Il comando viene completato inviando la stringa "close" al comando *PLINEA*, determinando in tal modo il quarto lato del vialetto e la visualizzazione della riga di comando.

Per verificare questa funzione, salvare il file *gp.lsp* aggiornato, avviare AutoCAD su un nuovo disegno e caricare il file AutoLISP come prima. Attivare la funzione di input utente come in precedenza.

Comando: (*gpuser*)

Fornire i valori come fatto nel passo precedente. Verificare quindi la nuova funzione *drawout* richiamandola.

Comando: (*drawout*)

La funzione fornisce i comandi ad AutoCAD per disegnare il contorno del vialetto che viene visualizzato sullo schermo. Dopo aver verificato la funzione, uscire da AutoCAD.

Capitolo 11 -- Corso interattivo di AutoLISP

Come disegnare le piastrelle

Dopo aver sviluppato e verificato la funzione di input utente e la funzione che disegna il contorno, è ora possibile riempire il vialetto con le piastrelle circolari. Questa operazione richiede alcune nozioni di geometria. Richiamare l'editor di testo ed aggiungere il seguente codice:

```
; Posizionare una fila di piastrelle alla distanza data lungo il
; vialetto e se possibile sfalsarla

(defun drow (pd offset)
  (setq pfirst (polar sp pangle pd))
  (setq pctile (polar pfirst angp90 offset))
```

```

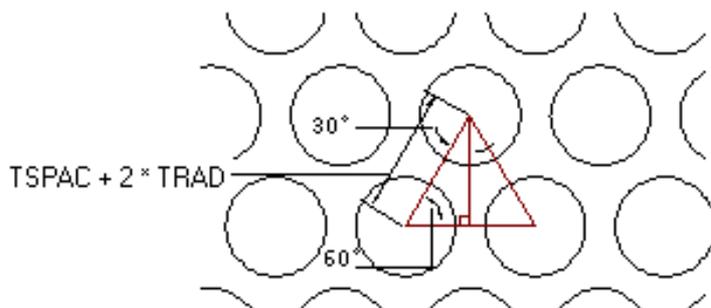
(setq pntile pntile)
(while (< (distance pfirst pntile) (- hwidth trad))
  (command "cerchio" pntile trad)
  (setq pntile
    (polar pntile angp90 (+ tspac trad trad)))
)
(setq pntile (polar pntile angm90 (+ tspac trad trad)))
(while (< (distance pfirst pntile) (- hwidth trad))
  (command "cerchio" pntile trad)
  (setq pntile
    (polar pntile angm90 (+ tspac trad trad)))
)
)
)

; Disegnare le file di piastrelle

(defun drawtiles ()
  (setq pdist (+ trad tspac))
  (setq off 0.0)
  (while (<= pdist (- plength trad))
    (draw pdist off)
    (setq pdist
      (+ pdist (* (+ tspac trad trad) (sin (dtr 60)))))
    (if (= off 0.0)
      (setq off (* (+ tspac trad trad) (cos (dtr 60))))
      (setq off 0.0)
    )
  )
)
)

```

Per comprendere come agiscono queste funzioni, vedere l'illustrazione seguente. La funzione `draw` disegna una file di piastrelle ad una data distanza lungo il vialetto specificato dal suo primo argomento, sfalsando la fila perpendicolarmente al vialetto in base alla distanza data dal suo secondo argomento. È possibile sfalsare le piastrelle su file alternate per coprire più spazio e disporle in maniera più piacevole.



La funzione `draw` trova la posizione per la prima piastrella nella fila utilizzando `polar` una prima volta per spostarsi lungo il vialetto per la distanza determinata dal primo argomento e una seconda per spostarsi perpendicolarmente al vialetto per lo sfalsamento. La funzione `draw` utilizza la funzione `while` per continuare a disegnare cerchi finché non incontra il limite del vialetto. `setq` alla fine dell'iterazione `while` si sposta sulla posizione della piastrella successiva con una spaziatura equivalente a due volte il raggio della piastrella più una spaziatura corrispondente allo spazio fra le piastrelle. Una seconda iterazione `while` disegna in seguito le piastrelle nella fila nell'altra direzione finché non incontra l'altro limite del vialetto.

La funzione `drawtiles` chiama ripetutamente `draw` per disegnare tutte le file di piastrelle. L'iterazione `while` richiama la funzione `draw` per ogni fila di piastrelle. Piastrelle in file adiacenti formano triangoli equilaterali, come mostrato nella figura. Gli spigoli di questi triangoli sono uguali a due volte il raggio della piastrella più la spaziatura fra le piastrelle. Quindi, in base alla trigonometria, la distanza fra le file lungo il vialetto è il seno di 60 gradi moltiplicato per questo valore, mentre lo sfalsamento per le file dispari è il coseno di 60 gradi moltiplicato per questo valore.

La funzione `if` viene utilizzata in `drawtiles` per sfalsare le altre file. La funzione `if` verifica il suo primo argomento ed esegue il secondo se è vero o altrimenti il terzo. In questo caso, se `OFF` è uguale a 0, impostarlo sulla spaziatura fra i centri delle piastrelle moltiplicata per il coseno di 60 gradi, come spiegato in precedenza. Se `OFF` è diverso da 0, impostarlo su 0, in modo da alternare a piacere lo sfalsamento delle file.

Per verificare questa funzione, salvare il file, avviare AutoCAD e caricare il programma. Digitare quanto segue:

Comando: **(gpuser)**

Fornire le informazioni relative al vialetto come fatto precedentemente. Digitare quanto segue:

Comando: **(drawout)**

Il contorno dovrebbe essere disegnato come prima. Infine digitare quanto segue:

Comando: **(drawtiles)**

Tutte le piastrelle dovrebbero essere disegnate entro i bordi del vialetto.

Capitolo 11 -- Corso interattivo di AutoLISP

Come aggiungere il comando ad AutoCAD

A questo punto, l'utente è in grado di combinare le funzioni in un comando di AutoCAD. Se in AutoLISP è stata definita una funzione con il nome C::XXX, digitando XXX (se XXX non è un comando di AutoCAD) la funzione viene richiamata. Per completare il comando PATH, definire una funzione chiamata C:PATH, che consente di digitare **path** dopo aver caricato *gp.lsp* per eseguire il comando che disegna un vialetto da giardino.

Quando il comando PATH è in esecuzione, i comandi che sottopone ad AutoCAD vengono visualizzati uno ad uno durante l'esecuzione nell'area della riga di comando ed i punti selezionati vengono indicati sullo schermo con delle crocette (contrassegni). Quando su una funzione di comando sono stati fissati tutti gli eventuali errori, è possibile disattivare l'output per far sì che questo comando implementato da AutoLISP venga visualizzato esattamente come un comando di AutoCAD.

Utilizzare il proprio editor di testo per aggiungere le seguenti righe a *gp.lsp*, quindi avviare AutoCAD e caricare il programma.

```
; Eseguire il comando richiamando le funzioni che lo costituiscono

(defun C:PATH ()
  (gpuser)
  (setq sblip (getvar "blipmode"))
  (setq scmde (getvar "cmdecho"))
  (setvar "blipmode" 0)
  (setvar "cmdecho" 0)
  (drawout)
  (drawtiles)
  (setvar "blipmode" sblip)
  (setvar "cmdecho" scmde)
  (princ)
)
```

Aggiungendo una funzione chiamata C:PATH, viene aggiunto un comando PATH ad AutoCAD. Utilizzare la funzione *getvar* per ottenere i valori correnti delle variabili di sistema BLIPMODE e CMDECHO e salvare questi valori con *setq* nelle variabili *sblip* e *scmde*. Quindi utilizzare la funzione *setvar* per impostare entrambe queste variabili di AutoCAD a 0, disattivando i contrassegni e la visualizzazione del comando mentre viene eseguito. Impostare queste variabili a 0 dopo aver ottenuto l'input dall'utente mediante *gpuser*. È possibile lasciare i contrassegni per confermare l'input dell'utente.

Dopo aver disegnato il vialetto, utilizzare la funzione *setvar* per ripristinare entrambe queste variabili ai loro valori originari.

L'aggiunta di una chiamata finale alla funzione *princ* consente alla funzione C:PATH di uscire senza problemi. Le funzioni AutoLISP restituiscono sempre il valore dell'ultima chiamata della funzione. In questo caso, se si omette la chiamata finale a *princ*, viene restituito 1 o 0.

Salvare il file, avviare AutoCAD e provare ad utilizzare il comando PATH. È possibile verificare il comando digitando quanto segue:

Comando: **path**

Punto iniziale del vialetto: **2,2**

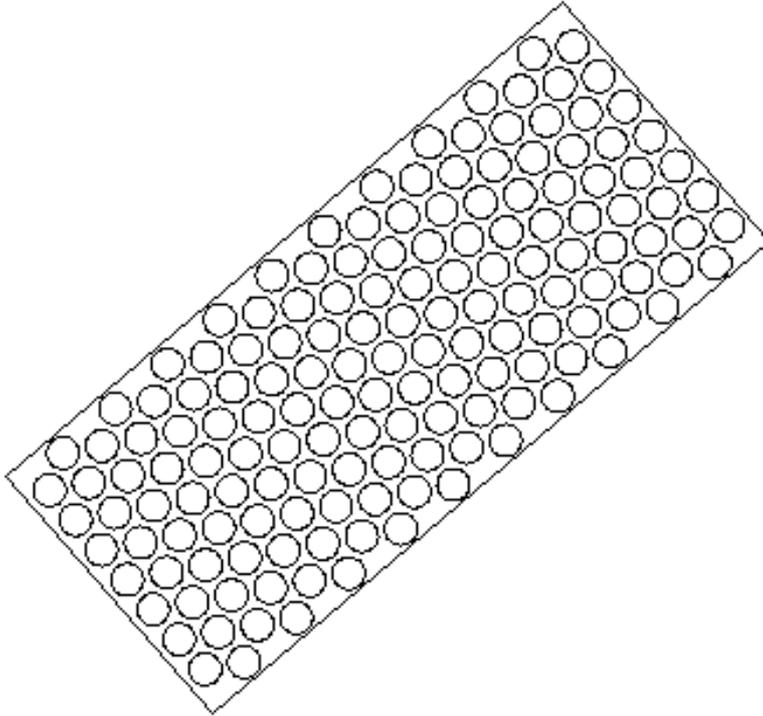
Punto finale del vialetto: **9,8**

Mezza larghezza del vialetto: **2**

Raggio delle piastrelle: **.2**

Spaziatura tra le piastrelle: **.1**

Questo esempio dovrebbe disegnare un vialetto da giardino come mostrato nella figura seguente.



È possibile esercitarsi con il comando PATH, specificando l'input mediante il dispositivo di puntamento o la tastiera.

Capitolo 11 -- Corso interattivo di AutoLISP

Come aggiungere un'interfaccia a finestra di dialogo

Il linguaggio DCL (Dialog Control Language) viene utilizzato per aggiungere finestre di dialogo definibili dall'utente ai programmi AutoLISP.

Il nuovo comando PATH accetta il suo input dalla riga di comando. È possibile aggiungere facilmente a questo comando un'interfaccia a finestra di dialogo utilizzando le funzioni di finestra di dialogo e creando un file *.dcl* che contenga la descrizione DCL della finestra di dialogo. Il linguaggio DCL viene descritto nella parte III di questa guida e le funzioni di finestra di dialogo AutoLISP sono descritte nel chapter 10, "Gestione delle finestre di dialogo."

Le finestre di dialogo sono utili per consentire agli utenti di scegliere fra varie opzioni. Poiché questo programma ha pochissime opzioni, si aggiungeranno ora altre caratteristiche.

Una nuova caratteristica che verrà aggiunta alla funzione C:PATH consentirà di specificare la forma delle piastrelle del vialetto. Verrà anche inserita una funzione di gestione degli errori.

Iniziare copiando la versione completa di *gp.lsp* in un altro file, *ddgp.lsp*. I nomi della maggior parte dei programmi AutoLISP forniti con AutoCAD e dotati di interfaccia a finestre di dialogo sono assegnati secondo il formato *ddxxx.lsp*). Verrà anche creato un nuovo file ASCII *ddgp.dcl* che contiene la descrizione DCL della finestra di dialogo.

Capitolo 11 -- Corso interattivo di AutoLISP

Come aggiungere un'interfaccia a finestra di dialogo

Il file DCL *ddgp.dcl*

La finestra di dialogo creata conterrà due pulsanti di scelta (la selezione di uno causa la deselegione automatica dell'altro) per scegliere la forma delle piastrelle: cerchio o poligono. Conterrà altresì tre caselle di modifica per digitare i valori numerici: raggio della piastrella, spaziatura fra le piastrelle e numero di lati, disponibile solo se è selezionato il pulsante di scelta Poligono.

I passi seguenti sono necessari per aggiungere una semplice finestra di dialogo alla funzione C:PATH. Inserire il seguente codice nel file *ddgp.dcl*.

```
/* DDGP.DCL, file DCL per DDGP.LSP */
gp_box1 : dialog {
  label = "Specifiche per le piastrelle del vialetto da giardino";
  : boxed_radio_row {          // definisce le aree dei pulsanti di scelta
    label = "Forma delle piastrelle";
    : radio_button {          // definisce il pulsante di scelta Poligono
      label = "&Poligono";
      key = "gp_poly";
    }
    : radio_button {          // definisce il pulsante di scelta Cerchio
      label = "&Cerchio";
      key = "gp_circ";
      value = "1";
    }
  }
  : edit_box {                // definisce la casella di modifica Raggio della piastrella
    label = "&Raggio della piastrella";
    key = "gp_trad";
    edit_width = 6;
  }
  : edit_box {                // definisce la casella di modifica Spaziatura fra le piastrelle
    label = "&Spaziatura fra le piastrelle:";
    key = "gp_spac";
    edit_width = 6;
  }
  : edit_box {                // definisce la casella di modifica Numero di lati
    label = "&Numero di lati";
    key = "gp_side";
    edit_width = 4;
  }
  : row {                      // definisce la riga di pulsanti OK/Annulla
    : spacer { width = 1; }
    : button {                 // definisce il pulsante OK
      label = "OK";
      key = "accept";
      width = 8;
      fixed_width = true;
    }
    : button {                 // definisce il pulsante Annulla
      label = "Annulla";
      is_cancel = true;
      key = "cancel";
      width = 8;
      fixed_width = true;
    }
    : spacer { width = 1;}
  }
}
```

Capitolo 11 -- Corso interattivo di AutoLISP

Come aggiungere un'interfaccia a finestra di dialogo

Funzioni di finestra di dialogo nel file AutoLISP *ddgp.lsp*

Ora che è stato creato il file DCL, è possibile modificare le righe necessarie nel file AutoLISP *ddgp.lsp*, creato da *daddgp.lsp*.

I nomi delle funzioni indicano l'azione da esse eseguita. La funzione `load_dialog` carica una finestra di dialogo. La funzione `set_tile` imposta un valore iniziale per la casella specificata (ogni pulsante, casella di modifica e così via, viene chiamata casella) su un valore di stringa, così come `action_tile` determina l'azione che verrà eseguita attivando la casella in questione.

Le nuove righe del codice sono contrassegnate dal commento `;<-NUOVA` dopo la riga di codice. Alcune delle vecchie righe devono essere commentate o rimosse da questo nuovo file; queste righe sono in grassetto, contrassegnate da due punti e virgola all'inizio della riga (`;;`) e terminano con la nota `;<-RIMUOVI`. Tutte le righe nuove e modificate sono in grassetto.

```
;; DDGP.LSP: una nuova idea per il vialetto da giardino.

; Convertire angolo da gradi a radianti

(defun dtr (a)
  (* pi (/ a 180.0))
)
; Acquisire le informazioni relative al vialetto da giardino

(defun gpuser ()
  (setq sp (getpoint "\nPunto iniziale del vialetto: "))
  (setq ep (getpoint "\nPunto finale del vialetto: "))
  (setq hwidth (getdist "\nMezza larghezza del vialetto: " sp))
  ;; (setq trad (getdist "\nRaggio delle piastrelle: " sp)) ;<-RIMUOVI
  ;; (setq tspac (getdist "\nSpaziatura fra le piastrelle: " sp) ;<-RIMUOVI
  (setq pangle (angle sp ep))
  (setq plength (distance sp ep))
  (setq width (* 2 hwidth))
  (setq angp90 (+ pangle (dtr 90)))
  (setq angm90 (- pangle (dtr 90)))
)

; Disegnare il contorno del vialetto

(defun drawout ()
  (command "pline"
    (setq p (polar sp angm90 hwidth))
    (setq p (polar p pangle plength))
    (setq p (polar p angp90 width))
    (polar p (+ pangle (dtr 180)) plength)
    "close"
  )
)
```

Aggiungere le seguenti righe di codice dopo la funzione `drawout`.

```
; Chiamare la finestra di dialogo per impostare le specifiche per le piastrelle

(defun gp_dialog ()
  (setq tshape "Cerchio"
    trad 0.5
    tspac 0.1
    tsides 8 )
  (setq dcl_id (load_dialog "ddgp.dcl"))
  (if (not (new_dialog "gp_box1" dcl_id)) (exit))
  (set_tile "gp_trad" "0.5")
  (set_tile "gp_spac" "0.1")
  (mode_tile "gp_side" 1)
  (set_tile "gp_side" "8")
  (action_tile "gp_circ"
    "(setq tshape \"Cerchio\") (mode_tile \"gp_side\" 1)")
  (action_tile "gp_poly"
    "(setq tshape \"Poligono\") (mode_tile \"gp_side\" 0)")
  (action_tile "cancel" "(done_dialog) (setq gperr \"\") (exit)")
  (action_tile "accept"
    (strcat
      "(progn (setq trad (atof (get_tile \"gp_trad\")))"
        "(setq tspac (atof (get_tile \"gp_spac\")))"
        "(setq tsides (atoi (get_tile \"gp_side\")))"
        " (done_dialog))")
  )
)
```

```

)
)
(start_dialog)
(unload_dialog dcl_id)
(if (= tshape "Cerchio")
  (defun gp_tile () (command "cerchio" ptile trad))
  (defun gp_tile () (command "poligono" tsides ptile "" trad))
)
)
; Definire il gestore degli errori

(defun gp_err (msg)
  (setq *error* olderr)
  (if (not gperr)
    (princ (strcat "\nErrore nel vialetto da giardino: " msg))
    (princ gperr))
  )
  (if sblip (setvar "blipmode" sblip))
  (if scmde (setvar "cmdecho" scmde))
  (princ)
)

```

Il seguente codice dovrebbe esistere già nel file *ddgp.lsp*, revisionato come indicato.

```

; Posizionare una fila di piastrelle alla distanza data lungo il
; vialetto e se possibile sfalsarla

(defun drow (pd offset)
  (setvar "snapang" pangle) ; <- NUOVA
  (setq pfirst (polar sp pangle pd))
  (setq pctile (polar pfirst angp90 offset))
  (setq ptile pctile)
  (while (< (distance pfirst ptile) (- hwidth trad))
    (gp_tile) ;<-NUOVA<
  ;; (command "cerchio" ptile trad) ;<-RIMUOVI
    (setq ptile (polar ptile angp90 (+ tspac trad trad)))
  )
  (setq ptile (polar pctile angm90 (+ tspac trad trad)))
  (while (< (distance pfirst ptile) (- hwidth trad))
    (gp_tile) ;<-NUOVA
  ;; (command "cerchio" ptile trad) ;<-RIMUOVI
    (setq ptile (polar ptile angm90 (+ tspac trad trad)))
  )
)
; Disegnare le file di piastrelle

(defun drawtiles ()
  (setq pdist (+ trad tspac))
  (setq off 0.0)
  (while (<= pdist (- plength trad))
    (drow pdist off)
    (setq pdist (+ pdist (* (+ tspac trad trad) (sin (dtr 60))))))
    (if (= off 0.0)
      (setq off (* (+ tspac trad trad) (cos (dtr 60))))
      (setq off 0.0))
  )
)
; Eseguire il comando richiamando le funzioni che lo costituiscono

(defun C:DDPATH () ; <- RIVEDERE, aggiungere "DD" a PATH
  (setq olderr *error* ; <- NUOVA
    *error* gp_err ; <- NUOVA
    sblip nil ; <- NUOVA
    scmde nil ; <- NUOVA
  )
)

```

```

    gperr nil ; <- NUOVA
  ) ; <- NUOVA
  (gpuser)
  (setq sblip (getvar "blipmode"))
  (setq scmde (getvar "cmdecho"))
  (setq sang (getvar "snapang")) ; <- NUOVA
  (setvar "blipmode" 0)
  (setvar "cmdecho" 0)
  (drawout)
  (gp_dialog) ;<-NUOVA
  (drawtiles)
  (setvar "blipmode" sblip)
  (setvar "cmdecho" scmde)
  (setvar "snapang" sang) ; <- NUOVA
  (setq *error* olderr) ; <- NUOVA
  (princ)
)

; Stampare il messaggio dopo il caricamento. ; <- NUOVA

(princ "\nDDGP.LSP Caricato. Digitare DDPATH per eseguirlo.") ; <- NUOVA
(princ) ; <- NUOVA

```

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Panoramica

Per trovare una funzione senza conoscerne il nome, utilizzare gli elenchi riportati nel presente capitolo. Tutte le funzioni AutoLISP sono descritte brevemente ed organizzate nei seguenti gruppi di funzioni:

- *Funzioni di base* (funzioni aritmetiche, di gestione delle stringhe, di uguaglianza e condizionali, di manipolazione degli elenchi e di gestione dei simboli, delle funzioni, degli errori e dell'applicazione)
- *Funzioni di utilità* (funzioni di richiesta e di comando, di controllo della visualizzazione, di input utente, funzioni geometriche, di conversione, di gestione dei file e di accesso ai dispositivi)
- *Funzioni di gruppi di selezione, di oggetti e di tabelle di simboli* (funzioni per la manipolazione dei gruppi di selezione, la gestione degli oggetti, la gestione estesa di dati e la gestione delle tabelle di simboli)
- *Funzioni per la programmazione delle finestre di dialogo* (funzioni per l'apertura e la chiusura delle finestre di dialogo, per la gestione delle caselle e degli attributi, delle caselle di riepilogo e degli elenchi a comparsa, dei gruppi di immagini e per la gestione di dati specifici dell'applicazione)
- *Funzioni di gestione della memoria.*

Le funzioni sono raggruppate in base al tipo di dati che utilizzano ed all'azione che svolgono. Le informazioni che seguono sono un compendio degli argomenti trattati dal capitolo 7 al capitolo 10 di questo manuale. Informazioni dettagliate per ogni funzione AutoLISP vengono fornite in ordine alfabetico nel capitolo 13, "AutoLISP: catalogo delle funzioni".

Argomenti di questo capitolo

{ewc ,JI(','Basic_Functions_al_u0306')} Funzioni di base

{button ,JI(','Utility_Functions_al_u0306')} Funzioni di utilità

{button ,JI(','Selection_Set44_Object44_and_Symbol_Table_Functions_al_u0306')} Funzioni di gruppi di selezione, di oggetti e di tabelle di simboli

{button ,JI(','Programmable_Dialog_Box_Functions_al_u0306')} Funzioni di programmazione delle finestre di dialogo

{button ,JI(','Memory_Management_Functions_al_u0306')} Funzioni di gestione della memoria

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di base

Le funzioni di base consistono in funzioni aritmetiche, di gestione delle stringhe, di uguaglianza e condizionali, di manipolazione degli elenchi e di gestione dei simboli, delle funzioni, degli errori e dell'applicazione. Per una descrizione dettagliata di queste funzioni, vedere il capitolo 7, "Nozioni fondamentali sull'uso di AutoLISP".

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di base

Funzioni aritmetiche

Di seguito sono riportate le funzioni aritmetiche di AutoLISP.

(+ [numero numero] . . .)

Restituisce la somma di tutti gli argomenti.

(-[numero numero] . . .)

Sottrae il secondo argomento e i seguenti dal primo e restituisce la differenza.

(* [numero numero] . . .)

Restituisce il prodotto di tutti gli argomenti.

(/ [numero numero] . . .)

Divide il primo argomento per il prodotto degli argomenti restanti e restituisce il resto.

(~ int)

Restituisce l'operatore a livello bit NOT (complemento ar1) dell'argomento.

(1+ numero)

Restituisce l'argomento aumentato di 1 (incrementato).

(1- numero)

Restituisce l'argomento diminuito di 1 (decrementato).

(abs numero)

Restituisce il valore assoluto dell'argomento.

(atan num1 [num2])

Restituisce l'arcotangente espresso in radianti di un argomento.

(cos angolo)

Restituisce il coseno di un angolo espresso in radianti.

(exp numero)

Restituisce la costante e elevata ad una determinata potenza (antilogaritmo naturale).

(expt base potenza)

Restituisce un numero elevato ad una determinata potenza.

(fix numero)

Restituisce la conversione di un numero reale nel numero intero inferiore che lo precede.

(float numero)

Restituisce la conversione di un numero in numero reale.

(gcd *int1 int2*)

Restituisce il massimo comune denominatore di due numeri interi.

(log *numero*)

Restituisce il logaritmo naturale di un numero esprimendolo come numero reale.

(logand *int int ...*)

Restituisce il risultato di un AND logico di un elenco di numeri interi.

(logior *int int ...*)

Restituisce il risultato di un operatore logico a livello bit OR inclusivo di un elenco di numeri interi.

(lsh *int bit_num*)

Restituisce lo spostamento logico di un numero intero per il numero di bit specificati.

(max *numero numero ...*)

Restituisce il maggiore dei numeri dati.

(min *numero numero ...*)

Restituisce il minore dei numeri dati.

(minusp *numero*)

Verifica che il numero sia negativo.

(rem *num1 num2 ...*)

Divide il primo argomento per il prodotto degli argomenti restanti e restituisce il resto

(sin *angolo*)

Restituisce il seno di un angolo espresso in radianti come numero reale.

(sqrt *numero*)

Restituisce la radice quadrata di un numero come numero reale.

(zerop *numero*)

Verifica che l' argomento sia uguale a zero.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di base

Funzioni di gestione delle stringhe

Di seguito sono riportate le funzioni di gestione delle stringhe.

(strcase *stringa [quale]*)

Restituisce la stringa i cui caratteri alfabetici sono stati convertiti in lettere maiuscole o minuscole.

(strcat *stringa1 [stringa2] ...*)

Restituisce una stringa ottenuta dalla concatenazione di più stringhe.

(strlen *[stringa] ...*)

Restituisce un numero intero che rappresenta il numero di caratteri contenuti in una stringa.

(substr *stringa inizio [lunghezza]*)

Restituisce una sottostringa di una stringa.

(wcmatch *stringa modello*)

Confronta un modello di caratteri jolly con una stringa.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di base

Funzioni di uguaglianza e funzioni condizionali

Di seguito sono riportate le funzioni di uguaglianza e le funzioni condizionali.

(= *str_num* [*str_num*] ...)

Restituisce T se tutti gli argomenti sono numericamente uguali, altrimenti restituisce nil.

(/= *str_num* [*str_num*] ...)

Restituisce T se gli argomenti non sono numericamente uguali, altrimenti restituisce nil.

(< *str_num* [*str_num*] ...)

Restituisce T se ogni argomento è numericamente minore rispetto a quello posizionato alla sua destra, altrimenti restituisce nil.

(<= *str_num* [*str_num*] ...)

Restituisce T se ogni argomento è numericamente minore rispetto a quello posizionato alla sua destra, altrimenti restituisce nil.

(> *str_num* [*str_num*] ...)

Restituisce T se ogni argomento è numericamente maggiore rispetto a quello posizionato alla sua destra, altrimenti restituisce nil.

(>= *str_num* [*str_num*] ...)

Restituisce T se ogni argomento è numericamente minore rispetto a quello posizionato alla sua destra, altrimenti restituisce nil.

(and *espressione* ...)

Restituisce l'AND logico di un elenco di espressioni.

(Boole *funz int1 int2* ...)

Funge da funzione booleana generale a livello di bit.

(cond (*verifica1 risultato1* ...) ...)

Funge da funzione condizionale primaria per AutoLISP.

(eq *espr1 espr2*)

Determina se due espressioni sono identiche.

(equal *espr1 espr2* [*approssimazione*])

Determina se due espressioni sono uguali.

(if *espr_verifica espr_then* [*espr_else*])

Valuta le espressioni in modo condizionale.

(or *espressione* ...)

Restituisce l'OR logico di un elenco di espressioni.

(repeat *int espr*...)

Valuta ogni espressione un numero specifico di volte e restituisce il valore dell'ultima espressione.

(while *espr_verifica espr* ...)

Valuta un'espressione di verifica e, se risulta diversa da nil, ne valuta altre; questo processo viene iterato finché l'espressione valutata non restituisce nil.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di base

Funzioni di manipolazione delle liste

Di seguito sono riportate le funzioni di manipolazione delle liste.

(acad_strlsort *lista*)

Dispone una lista di stringhe in ordine alfabetico.

(append *lista* ...)

Accoda un determinato numero di liste come se facessero parte di una singola lista.

(assoc *elem lista_assoc*)

Ricerca un elemento in una lista di associazioni e restituisce la voce associata.

(car *lista*) e (cdr *lista*)

Restituisce il primo elemento di una lista oppure una lista contenente tutti gli elementi tranne il primo.

(cons *nuovo-primo-elemento lista*)

Rappresenta il costruttore di base di liste.

(foreach *nome elenco espr* ...)

Valuta tutte le espressioni dei membri di una lista.

(last *elenco*)

Restituisce l'ultimo elemento di una lista.

(length *elenco*)

Restituisce un numero intero che indica il numero di elementi di una lista.

(list *espressione* ...)

Prende un numero qualsiasi di espressioni e le combina in una lista.

(listp *elem*)

Verifica che un elemento sia una lista.

(mapcar *funzione elenco1 ... elencon*)

Restituisce una lista come risultato dell'esecuzione di una funzione con i singoli elementi di una lista o più liste fornite come argomenti alla funzione.

(member *espr elenco*)

Restituisce una lista come risultato dell'esecuzione di una funzione con i singoli elementi di una lista o più liste fornite come argomenti alla funzione.

(nth *n elenco*)

Restituisce l'elemento n di una lista.

(reverse *elenco*)

Restituisce una lista con gli elementi invertiti.

(subst *nuovo_elem vecchio_elem elenco*)

Cerca un elemento esistente in una lista e restituisce una copia della lista con un nuovo elemento in sostituzione di quello precedente, per tutte le occorrenze del vecchio <XRefEnd >

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di base

Funzioni di gestione dei simboli

Di seguito sono riportate le funzioni di gestione dei simboli.

(atom *elem*)

Verifica che un elemento sia un atomo.

(atoms-family *formato [elenco_simb]*)

Restituisce un elenco dei simboli attualmente definiti.

(boundp *simb*)

Verifica se un valore è associato ad un simbolo.

(not *elem*)

Verifica se ad un elemento è stato assegnato il valore nil.

(null *elem*)

Verifica se un elemento è stato limitato solo al valore nil.

(numberp *elem*)

Verifica se un elemento è un numero reale o intero.

(quote *espressione*)

Restituisce un'espressione senza valutarla.

(read [*stringa*])

Restituisce il primo elenco o l'atomo restituito da una stringa.

(set *simb espr*)

Imposta il valore del nome di un simbolo racchiuso tra virgolette ad una espressione

(setq *simb1 espr1 [simb2 espr2]. . .*)

Imposta il valore di un simbolo o di simboli per le espressioni associate.

(type *elem*)

Restituisce il tipo di un determinato elemento

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di base

Funzioni di gestione delle funzioni

Di seguito sono riportate le funzioni per la gestione delle funzioni.

(apply *funzione elenco*)

Passa un elenco di argomenti ad una funzione specificata.

(defun *simb elenco-arg espr . . .*)

Definisce una funzione.

(eval *espressione*)

Restituisce il risultato della valutazione di un'espressione AutoLISP.

(lambda *argomenti espr* . . .)

Definisce una funzione anonima.

(progn [*espressione*] . . .)

Valuta ogni espressione in modo sequenziale e restituisce il valore dell'ultima espressione

(trace *funzione* . . .)

È di ausilio nelle operazioni di debug di AutoLISP.

(untrace *funzione* . . .)

Cancella il flag di trace per le funzioni specificate.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di base

Funzioni di gestione degli errori

Di seguito sono riportate le funzioni di gestione degli errori.

(alert *stringa*)

Visualizza una casella di avviso contenente l'errore o il messaggio di avvertimento assegnato sotto forma di stringa.

(*error* *stringa*)

È una funzione per la gestione degli errori che può essere definita dall'utente.

(exit)

Forza l'uscita dall'applicazione corrente.

(quit)

Forza l'uscita dall'applicazione corrente.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di base

Funzioni di gestione dell'applicazione

Di seguito sono riportate le funzioni di gestione dell'applicazione.

(ads)

Restituisce un elenco delle applicazioni ADS correntemente caricate.

(arx)

Restituisce un elenco delle applicazioni ARX correntemente caricate.

(arxload *applicazione [se_fallisce]*)

Carica un'applicazione ARX.

(arxunload *applicazione [se_fallisce]*)

Scarica un'applicazione ARX.

(autoarxload *nomefile cmdlist*)

Predefinisce i nomi dei comandi per caricare un file ARX associato.

(autoload *nomefile cmdlist*)

Predefinisce i nomi dei comandi per caricare un file AutoLISP associato.

(autoload *nomefile cmdlist*)

Predefinisce i nomi dei comandi per caricare un'applicazione ADS associata.

(load *nomefile [se_fallisce]*)

Valuta le espressioni AutoLISP in un file.

(startapp *cmdapp file*)

Avvia un'applicazione di Windows.

(xload *applicazione [se_fallisce]*)

Carica un'applicazione ADS.

(xunload *applicazione [se_fallisce]*)

Scarica un'applicazione ADS.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di utilità

Le funzioni di utilità consistono in funzioni di richiesta e di comando, di controllo della visualizzazione, di input utente, geometriche, di conversione, di gestione dei file e di accesso ai dispositivi. Per una descrizione dettagliata di queste funzioni, vedere il capitolo 8, "Funzioni di utilità generali"..

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di utilità

Funzioni di richiesta e di comando

Le funzioni di richiesta e di comando sono le seguenti:

(acad_colordlg *num_colore [flag]*)

Visualizza la finestra di dialogo standard per la selezione dei colori di AutoCAD.

(acad_helpdlg *file_guida argomento*)

Richiama la Guida (obsoleto).

(command *[argomenti] . . .*)

Esegue un comando di AutoCAD.

(getcfig *nome_cfg*)

Richiama i dati dell'applicazione dalla sezione AppData del file acad.cfg.

(getcname *nomec*)

Richiama il nome localizzato o inglese di un comando AutoCAD.

(getenv *nome-variabile*)

Restituisce il valore della stringa assegnato ad una variabile d'ambiente.

(getvar *nome_var*)

Richiama il valore di una variabile di sistema di AutoCAD.

(help *[file_guida [argomento [comando]]]*)

Richiama la Guida.

(setcfg nome_cfg val_cfg)

Scrivi i dati dell'applicazione nella sezione AppData del file acad.cfg.

(setfunhelp funzione [file_guida [argomento [comando]])]

Registra un comando definito dell'utente con la funzionalità Guida in modo che il file della Guida e l'argomento vengano richiamati quando l'utente richiede la guida <XRefEnd >

(setvar nome_var valore)

Imposta un valore specifico per una variabile di sistema di AutoCAD.

(ver)

Restituisce una stringa in cui è contenuto il numero della versione corrente di AutoLISP.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di utilità

Funzioni di controllo della visualizzazione

Di seguito sono riportate le funzioni di controllo della visualizzazione.

(graphscr)

Visualizza lo schermo grafico di AutoCAD.

(gdraw da a colore [evidenz])

Disegna un vettore tra due punti nella finestra corrente.

(grtext [casella testo [evidenz]])

Scrivi il testo sulla riga di stato o nelle aree del menu di schermo.

(grvecs elenco_vet [trasf])

Disegna vettori multipli sullo schermo grafico.

(menucmd stringa)

Invia comandi di menu o imposta e richiama lo stato di voci di menu.

(gruppmenu nomegruppo)

Verifica che sia caricato un gruppo menu

(prin1 [espr [desc-file]])

Stampa un'espressione sulla riga di comando oppure scrive un'espressione in un file aperto.

(princ [espr [desc-file]])

Stampa un'espressione sulla riga di comando oppure scrive un'espressione in un file aperto.

(print [espr [desc-file]])

Stampa un'espressione sulla riga di comando o scrive un'espressione in un file aperto.

(prompt msg)

Visualizza una stringa nell'area del messaggio di richiesta dello schermo.

(redraw [nome_ent [modalità]])

Ridisegna la finestra corrente o un oggetto specificato (entità) nella finestra corrente.

(terpri)

Visualizza una riga nuova nella riga di comando.

(textpage)

Passa dallo schermo grafico allo schermo di testo.

(textscr)

Passa dallo schermo grafico allo schermo di testo, come ad esempio il tasto funzione Grafico/Testo di AutoCAD.

(vports)

Restituisce un elenco di descrittori di finestre per la configurazione della finestra corrente.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di utilità

Funzioni di input utente

Di seguito sono riportate le funzioni di input utente.

(entselect [msg_richiesta])

Richiede all'utente di selezionare un singolo oggetto (entità) specificando un punto.

(getangle [pt] [msg_richiesta])

Sospende l'esecuzione per consentire all'utente di specificare un angolo e restituisce quell'angolo in radianti.

(getcorner pt [msg_richiesta])

Sospende l'esecuzione per consentire all'utente di specificare il secondo angolo di un rettangolo.

(getdist [pt] [msg_richiesta])

Sospende l'esecuzione per consentire all'utente di specificare una distanza.

(getfiled titolo default est flag)

Richiede il nome di un file con la finestra di dialogo standard AutoCAD e restituisce quel nome file.

(getint [msg_richiesta])

Sospende l'esecuzione per consentire all'utente di specificare un numero intero e restituisce quel numero intero.

(getkeyword [msg_richiesta])

Sospende l'esecuzione per consentire all'utente di specificare una parola chiave e restituisce quella parola chiave.

(getorient [pt] [msg_richiesta])

Sospende l'esecuzione per consentire all'utente di specificare un angolo e restituisce quell'angolo in radianti.

(getpoint [pt] [msg_richiesta])

Sospende l'esecuzione per consentire all'utente di specificare un punto e restituisce quel punto.

(getreal [msg_richiesta])

Sospende l'esecuzione per consentire all'utente di specificare un numero reale e restituisce quel numero reale.

(getstring [cr] [msg_richiesta])

Sospende l'esecuzione per consentire all'utente di specificare una stringa e restituisce tale stringa.

(initget [bit] [stringa])

Stabilisce parole chiave da utilizzare dalla successiva chiamata alla funzione di input utente.

(nentsel [msg_richiesta])

Richiede all'utente di selezionare un oggetto (entità) specificando un punto e fornisce l'accesso ai dati di definizione contenuti in un oggetto complesso.

(nentselp [msg_richiesta] [pt])

Fornisce una funzionalità simile a quella della funzione nentsel senza che sia necessario l'input dell'utente.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di utilità

Funzioni geometriche

Di seguito sono riportate le funzioni geometriche.

(angle pt1 pt2)

Restituisce un angolo misurato in radianti di una linea definita da due punti finali.

(distance pt1 pt2)

Restituisce la distanza tridimensionale tra due punti.

(inters pt1 pt2 pt3 pt4 [su_seg])

Trova l'intersezione di due linee.

(osnap pt modalità)

Restituisce un punto tridimensionale risultante dall'applicazione della modalità snap ad oggetto ad un punto specifico.

(polar pt angolo distanza)

Restituisce il punto tridimensionale del sistema di coordinate utente all'angolo indicato ed alla distanza specificata rispetto ad un punto.

(textbox elenco_ent)

Misura un determinato oggetto di testo e restituisce le coordinate diagonali della casella in cui è racchiuso il testo.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di utilità

Funzioni di conversione

Di seguito sono riportate le funzioni di conversione.

(angtof stringa [modalità])

Converte in un valore reale (a virgola mobile) una stringa che rappresenta un angolo misurato in radianti.

(angtos angolo [modalità [precisione]])

Converte un valore angolare espresso in radianti in una stringa.

(ascii stringa)

Restituisce la conversione del primo carattere di una stringa nel relativo codice ASCII (numero intero).

(atof stringa)

Restituisce la conversione di una stringa in un numero reale.

(atoi stringa)

Restituisce la conversione di una stringa in un numero intero.

(chr num_intero)

Restituisce la conversione di un numero intero che rappresenta un codice ASCII in una stringa costituita da un unico carattere.

(cvunit valore da a)

Converte un valore da un'unità di misura in un'altra.

(distof stringa [modalità])

Converte una stringa che contiene un valore reale (con virgola mobile) in un valore reale.

(itoa *int*)

Restituisce la conversione di un numero intero in una stringa.

(rtos *numero [modalità [precisione]]*)

Converte un numero in una stringa.

(trans *pt da a [spost]*)

Converte un punto o uno spostamento da un sistema di coordinate ad un altro.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di utilità

Funzioni di gestione dei file

Di seguito sono riportate le funzioni di gestione dei file.

(close *desc-file*)

Chiude un file aperto.

(findfile *nomefile*)

Ricerca il file specificato nel percorso di libreria di AutoCAD.

(open *nomefile modalità*)

Apre un file al quale possono accedere le funzioni AutoLISP di I/O.

(read-char *[desc-file]*)

Restituisce il codice decimale ASCII che rappresenta il carattere letto dal buffer di input della tastiera o da un file aperto.

(read-line *[desc-file]*)

Legge una stringa dalla tastiera o da un file aperto.

(write-char *num [desc-file]*)

Valuta un'espressione di verifica e, se risulta diversa da nil, ne valuta altre; questo processo viene iterato finché l'espressione valutata non restituisce nil.

(write-line *stringa [desc-file]*)

Scriva un carattere sullo schermo o in un file aperto.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di utilità

Funzioni di accesso ai dispositivi

Di seguito sono riportate le funzioni di accesso ai dispositivi.

(grread *[traccia] [tutti_tasti [tipocur]]*)

Legge i valori da qualsiasi dispositivo di input AutoCAD.

(tablet *codice [riga1 riga2 riga3 direzione]*)

Richiama ed imposta le graduazioni del digitalizzatore (tavoletta).

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di gruppi di selezione, di oggetti e di tabelle di simboli

Le funzioni di gruppi di selezione, di oggetti e di tabelle di simboli consistono in funzioni per la manipolazione dei gruppi di selezione, la gestione degli oggetti, la gestione estesa di dati e la gestione delle tabelle di simboli. Per una descrizione

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di gruppi di selezione, di oggetti e di tabelle di simboli

Funzioni per la manipolazione dei gruppi di selezione

Di seguito sono riportate le funzioni per la manipolazione dei gruppi di selezione.

(ssadd [nome_ent [gs]])

Aggiunge un oggetto (entità) ad un gruppo di selezione oppure crea un nuovo gruppo di selezione.

(ssdel nome_ent gs)

Cancella un oggetto (entità) da un gruppo di selezione.

(ssget [modalità] [pt1 [pt2]] [elenco-pt] [elenco-filtri])

Richiede all'utente di selezionare gli oggetti (entità) e restituisce un gruppo di selezione.

(ssgetfirst)

Individua gli oggetti selezionati e sottoposti a grip.

(sslength gs)

Restituisce un numero intero che indica il numero di oggetti (entità) contenuti in un gruppo di selezione.

(ssmemb nome_ent gs)

Verifica se un oggetto (entità) è un membro di un gruppo di selezione.

(ssname gs indice)

Restituisce il nome dell'oggetto (entità) dell'elemento indicizzato contenuto in un gruppo di selezione.

(ssnamex gs indice)

Recupera le informazioni relative alla creazione di un gruppo di selezione

(sssetfirst gruppogrip [grupposelezione])

Stabilisce quali oggetti vengono selezionati e sottoposti al grip.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di gruppi di selezione, di oggetti e di tabelle di simboli

Funzioni di gestione degli oggetti

Di seguito sono riportate le funzioni di gestione degli oggetti.

(entdel nome_ent)

Elimina gli oggetti (entità) oppure ripristina gli oggetti precedentemente eliminati.

(entget nome_ent [elenco_app])

Richiama i dati di definizione di un oggetto (entità).

(entlast)

Restituisce il nome dell'ultimo oggetto (entità) principale non eliminato nel disegno.

(entmake [elenco_ent])

Crea una nuova entità (oggetto grafico) nel disegno.

(entmakex [elenco_ent])

Crea una nuovo oggetto grafico, fornisce un gestore e un nome entità (ma non assegna un proprietario), quindi restituisce il nuovo nome entità.

(entmod [elenco_ent])

Modifica i dati di definizione di un oggetto (entità).

(entnext [nome_ent])

Restituisce il nome dell'oggetto (entità) successivo nel disegno.

(entupd nome_ent)

Aggiorna l'immagine visualizzata di un oggetto (entità).

(handent gestore)

Restituisce il nome di un oggetto (entità) basato sul gestore.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di gruppi di selezione, di oggetti e di tabelle di simboli

Funzioni di gestione dei dati estesi

Di seguito sono riportate le funzioni di gestione dei dati estesi.

(regapp applicazione)

Memorizza il nome di un'applicazione in cui il disegno AutoCAD corrente utilizzerà i dati estesi dell'oggetto.

(xdroom nome_ent)

Restituisce la quantità di spazio di dati estesi (Xdata) disponibile per un oggetto (entità).

(xdsiize elenco)

Restituisce la dimensione in byte occupata da un elenco quando è collegato ad un oggetto (entità) sotto forma di dati estesi.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di gruppi di selezione, di oggetti e di tabelle di simboli

Funzioni di gestione delle tabelle di simboli e di dizionari

Di seguito sono riportate le funzioni di gestione delle tabelle di simboli e di dizionari.

(dictadd nome_ent simbolo nuovo_oggetto)

Aggiunge un oggetto non grafico ad un dizionario specificato.

(dictdel nome_ent simbolo)

Cancella una voce dal dizionario specificato.

(dictnext nome_ent simbolo [primo])

Trova l'elemento successivo in un dizionario.

(dictrename *nome_ent vecchio_simbolo nuovo_simbolo*)

Rinomina la voce di un dizionario.

(dictsearch *nome_ent simbolo [successivo]*)

Ricerca un elemento in un dizionario.

(namedobjdict)

Restituisce il nome dell'entità del dizionario dell'oggetto del disegno corrente, che è la base di tutti gli oggetti non grafici del disegno.

(setview *descrizione_finestra [id_finestra]*)

Stabilisce una vista per una finestra specificata.

(snvalid *nome_simb*)

Verifica la validità dei caratteri nel nome della tabella di simboli.

(tblnext *nome_tabella [primo]*)

Richiama ed imposta le graduazioni del digitalizzatore (tavoletta).

(tblobjname *nome_tabella simbolo*)

Restituisce il nome dell'entità di una voce specifica della tabella di simboli.

(tblsearch *nome_tabella simbolo [successivo]*)

Cerca il nome di un simbolo in una tabella di simboli.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di programmazione delle finestre di dialogo

Le funzioni per la programmazione delle finestre di dialogo consistono in funzioni per l'apertura e la chiusura delle finestre di dialogo, per la gestione delle caselle e degli attributi, delle caselle di riepilogo e degli elenchi a comparsa, dei gruppi di immagini e per la gestione di dati specifici dell'applicazione. Per una descrizione dettagliata di queste funzioni, vedere il capitolo 10, "Gestione delle finestre di dialogo".

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di programmazione delle finestre di dialogo

Funzioni di apertura e chiusura delle finestre di dialogo

Di seguito sono riportate le funzioni per l'apertura e la chiusura delle finestre di dialogo.

(done_dialog *[stato]*)

Chiude una finestra di dialogo.

(load_dialog *file_dcl*)

Carica un file DCL.

(new_dialog *nomedlg id_dcl [azione [schermo_pt]]*)

Attiva e visualizza una nuova finestra di dialogo; è possibile specificare anche un'azione di default.

(start_dialog)

Visualizza una finestra di dialogo ed inizia ad accettare l'input dell'utente.

(term_dialog)

Chiude tutte le finestre di dialogo correnti come se ognuna di esse fosse stata annullata dall'utente.

(unload_dialog id_dcl)

Scarica un file DCL.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di programmazione delle finestre di dialogo

Funzioni di gestione di caselle ed attributi

Di seguito sono riportate le funzioni per la gestione delle caselle e degli attributi.

(action_tile chiave espressione-azione)

Assegna un'azione da valutare quando l'utente seleziona la casella specificata in una finestra di dialogo.

(get_attr chiave attributo)

Indica il valore DCL di un attributo di una finestra di dialogo.

(get_tile chiave)

Indica il valore corrente del tempo di esecuzione di una casella della finestra di dialogo.

(mode_tile chiave modalità)

Imposta la modalità di una casella di una finestra di dialogo.

(set_tile chiave valore)

Imposta il valore della casella di una finestra di dialogo.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di programmazione delle finestre di dialogo

Funzioni di gestione di caselle di riepilogo ed elenchi a comparsa

Di seguito sono riportate le funzioni di gestione delle caselle di riepilogo e degli elenchi a comparsa.

(add_list stringa)

Aggiunge o modifica una stringa nell'elenco delle finestre di dialogo correntemente attivo.

(end_list)

Termina la valutazione dell'elenco della finestra di dialogo correntemente attiva.

(start_list chiave [operazione [indice]])

Avvia l'elaborazione di un elenco nella casella di una finestra di dialogo relativa ad una casella di riepilogo o ad un elenco a comparsa.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di programmazione delle finestre di dialogo

Funzioni di gestione dei gruppi di immagini

Di seguito sono riportate le funzioni di gestione dei gruppi di immagini.

(dimx_tile *chiave*) e **(dimy_tile *chiave*)**

Richiama le dimensioni di una casella in base alle unità di misura della finestra di dialogo.

(end_image)

Termina la creazione dell'immagine della finestra di dialogo correntemente attiva.

(fill_image *x1 y1 larg altez colore*)

Disegna un rettangolo riempito nel gruppo di immagini attualmente attivo nella finestra di dialogo.

(slide_image *x1 y1 larg altez nome_dia*)

Visualizza una diapositiva di AutoCAD nel gruppo di immagini della finestra di dialogo correntemente attiva.

(start_image *chiave*)

Avvia la creazione di un'immagine nella casella di una finestra di dialogo.

(vector_image *x1 y1 x2 y2 colore*)

Disegna un vettore nell'immagine della finestra di dialogo attualmente attiva.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di programmazione delle finestre di dialogo

Funzioni di gestione di dati specifici dell'applicazione

Di seguito sono riportate le funzioni di gestione dei dati specifici dell'applicazione.

(client_data_tile *chiave daticliente*)

Associa i dati gestiti dall'applicazione ad una casella della finestra di dialogo.

Capitolo 12 -- Sommario delle funzioni di AutoLISP

Funzioni di gestione della memoria

Le funzioni di gestione della memoria riportate di seguito sono descritte più dettagliatamente nel capitolo 15, "Gestione della memoria"..

(alloc *int*)

Imposta la dimensione del segmento ad un determinato numero di nodi.

(expand *numero*)

Assegna lo spazio per i nodi richiedendo un determinato numero di segmenti.

(gc)

Libera nodi non utilizzati.

(mem)

Visualizza lo stato corrente della memoria AutoLISP.

Capitolo 13 -- AutoLISP: catalogo delle funzioni



Di seguito viene presentato un elenco, ordinato alfabeticamente, di tutte le funzioni standard AutoLISP definite da AutoCAD.

In questo capitolo ogni elenco contiene una breve descrizione dell'utilizzo della funzione e un'istruzione di sintassi che indica l'ordine ed il tipo di argomenti richiesti; per informazioni specifiche sulle istruzioni di sintassi, vedere "Sintassi delle funzioni AutoLISP." Se non si conosce il nome della funzione da ricercare, usare l'elenco nel capitolo 12, "Sommaro delle funzioni di AutoLISP".

È necessario specificare ulteriori informazioni sull'argomento *numero*. Un numero può essere reale, intero o rappresentato da un simbolo impostato su un valore reale o intero. Se tutti gli argomenti sono numeri interi, il risultato sarà un numero intero. Se invece gli argomenti sono numeri reali, i numeri interi vengono convertiti in reali ed il risultato sarà un numero reale.

La maggior parte delle funzioni AutoLISP sono sempre disponibili; alcune funzioni tuttavia vengono definite da un'applicazione AutoLISP o ARX. Il file che contiene le funzioni definite esternamente viene indicato dopo la dicitura "Funzione definita esternamente". Prima di utilizzare queste funzioni, è necessario verificarne la disponibilità.

Operatori

```
{ewc ,JI(','43_40addition41_al_u0307')} + (addizione)
{button ,JI(','45_40subtraction41_al_u0307')} - (sottrazione)
{ewc ,JI(','42_40multiplication41_al_u0307')} * (moltiplicazione)
{button ,JI(','47_40division41_al_u0307')} / (divisione)
{button ,JI(','61_40equal_to41_al_u0307')} = (uguale a)
{button ,JI(','4761_40not_equal_to41_al_u0307')} /= (diverso da )
{button ,JI(','60_40less_than41_al_u0307')} < (minore di)
{ewc ,JI(','6061_40less_than_or_equal_to41_al_u0307')} <= (minore di o uguale a )
{ewc ,JI(','62_40greater_than41_al_u0307')} > (maggiore di)
{button ,JI(','6261_40greater_than_or_equal_to41_al_u0307')} >= (maggiore di o uguale a )
{button ,JI(','126_40bitwise_NOT41_al_u0307')} ~ (NOT a livello bit)
{button ,JI(','143_40increment41_al_u0307')} 1+ (incremento)
{button ,JI(','145_40decrement41_al_u0307')} 1- (decremento)
```

A

```
{button ,JI(','abs_al_u0307')} abs
{button ,JI(','acad95colordlg_al_u0307')} acad_colordlg
{ewc ,JI(','acad95helpdlg_al_u0307')} acad_helpdlg
{button ,JI(','acad95strlsort_al_u0307')} acad_strlsort
{ewc ,JI(','action95tile_al_u0307')} action_tile
{ewc ,JI(','add95list_al_u0307')} add_list
{button ,JI(','ads_al_u0307')} ads
{button ,JI(','alert_al_u0307')} alert
{button ,JI(','alloc_al_u0307')} alloc
{button ,JI(','and_al_u0307')} and
```

{button ,JI(','angle_al_u0307')} angle
{button ,JI(','angtof_al_u0307')} angtof
{ewc ,JI(','angtos_al_u0307')} angtos
{ewc ,JI(','append_al_u0307')} append
{ewc ,JI(','apply_al_u0307')} apply
{button ,JI(','arx_al_u0307')} arx
{button ,JI(','arxload_al_u0307')} arxload
{button ,JI(','arxunload_al_u0307')} arxunload
{button ,JI(','ascii_al_u0307')} ascii
{button ,JI(','assoc_al_u0307')} assoc
{button ,JI(','atan_al_u0307')} atan
{ewc ,JI(','atof_al_u0307')} atof
{ewc ,JI(','atoi_al_u0307')} atoi
{ewc ,JI(','atom_al_u0307')} atom
{ewc ,JI(','atoms45family_al_u0307')} atoms-family
{button ,JI(','autoarxload_al_u0307')} autoarxload
{button ,JI(','autoload_al_u0307')} autoload
{ewc ,JI(','autoxload_al_u0307')} autoxload

B

{button ,JI(','Boole_al_u0307')} Boole
{button ,JI(','boundp_al_u0307')} boundp

C

{ewc ,JI(','car_and_cdr_al_u0307')} car e cdr
{button ,JI(','chr_al_u0307')} chr
{button ,JI(','client95data95tile_al_u0307')} client_data_tile
{ewc ,JI(','close_al_u0307')} close
{button ,JI(','command_al_u0307')} command
{button ,JI(','cond_al_u0307')} cond
{ewc ,JI(','cons_al_u0307')} cons
{ewc ,JI(','cos_al_u0307')} cos
{button ,JI(','cvunit_al_u0307')} cvunit

D

{ewc ,JI(','defun_al_u0307')} defun
{button ,JI(','dictadd_al_u0307')} dictadd
{button ,JI(','dictnext_al_u0307')} dictnext
{ewc ,JI(','dictremove_al_u0307')} dictremove
{ewc ,JI(','dictrename_al_u0307')} dictrename

```
{ewc ,JI('`,`dictsearch_al_u0307')} dictsearch  
{ewc ,JI('`,`dimx95tile_and_dimy95tile_al_u0307')} dimx_tile e dimy_tile  
{button ,JI('`,`distance_al_u0307')} distance  
{ewc ,JI('`,`distof_al_u0307')} distof  
{ewc ,JI('`,`done95dialog_al_u0307')} done_dialog  
{ewc ,JI('`,`end95image_al_u0307')} end_image
```

E

```
{ewc ,JI('`,`end95list_al_u0307')} end_list  
{button ,JI('`,`entdel_al_u0307')} entdel  
{ewc ,JI('`,`entget_al_u0307')} entget  
{ewc ,JI('`,`entlast_al_u0307')} entlast  
{button ,JI('`,`entmake_al_u0307')} entmake  
{button ,JI('`,`entmakex_al_u0307')} entmakex  
{ewc ,JI('`,`entmod_al_u0307')} entmod  
{ewc ,JI('`,`entnext_al_u0307')} entnext  
{button ,JI('`,`entsel_al_u0307')} entsel  
{ewc ,JI('`,`entupd_al_u0307')} entupd  
{ewc ,JI('`,`eq_al_u0307')} eq  
{ewc ,JI('`,`equal_al_u0307')} equal  
{button ,JI('`,`42error42_al_u0307')} *error*  
{button ,JI('`,`eval_al_u0307')} eval  
{ewc ,JI('`,`exit_al_u0307')} exit  
{ewc ,JI('`,`exp_al_u0307')} exp  
{button ,JI('`,`expand_al_u0307')} expand  
{ewc ,JI('`,`expt_al_u0307')} expt
```

F

```
{ewc ,JI('`,`fill95image_al_u0307')} fill_image  
{button ,JI('`,`findfile_al_u0307')} findfile  
{ewc ,JI('`,`fix_al_u0307')} fix  
{button ,JI('`,`float_al_u0307')} float  
{button ,JI('`,`foreach_al_u0307')} foreach
```

G

```
{button ,JI('`,`gc_al_u0307')} gc  
{ewc ,JI('`,`gcd_al_u0307')} gcd  
{button ,JI('`,`get95attr_al_u0307')} get_attr  
{button ,JI('`,`get95tile_al_u0307')} get_tile  
{button ,JI('`,`getangle_al_u0307')} getangle
```

```
{ewc ,JI('`,`getcfg_al_u0307')} getcfg  
{ewc ,JI('`,`getcname_al_u0307')} getcname  
{ewc ,JI('`,`getcorner_al_u0307')} getcorner  
{button ,JI('`,`getdist_al_u0307')} getdist  
{button ,JI('`,`getenv_al_u0307')} getenv  
{ewc ,JI('`,`getfiled_al_u0307')} getfiled  
{ewc ,JI('`,`getint_al_u0307')} getint  
{ewc ,JI('`,`getkword_al_u0307')} getkword  
{ewc ,JI('`,`getorient_al_u0307')} getorient  
{button ,JI('`,`getpoint_al_u0307')} getpoint  
{ewc ,JI('`,`getreal_al_u0307')} getreal  
{button ,JI('`,`getstring_al_u0307')} getstring  
{button ,JI('`,`getvar_al_u0307')} getvar  
{ewc ,JI('`,`graphscr_al_u0307')} graphscr  
{ewc ,JI('`,`grclear_al_u0307')} grclear  
{button ,JI('`,`grdraw_al_u0307')} grdraw  
{ewc ,JI('`,`grread_al_u0307')} grread  
{ewc ,JI('`,`grtext_al_u0307')} grtext  
{ewc ,JI('`,`grvecs_al_u0307')} grvecs
```

H

```
{ewc ,JI('`,`handent_al_u0307')} handent  
{button ,JI('`,`help_al_u0307')} help
```

I

```
{ewc ,JI('`,`if_al_u0307')} if  
{ewc ,JI('`,`initget_al_u0307')} initget  
{button ,JI('`,`inters_al_u0307')} inters  
{ewc ,JI('`,`itoa_al_u0307')} itoa
```

L

```
{ewc ,JI('`,`lambda_al_u0307')} lambda  
{ewc ,JI('`,`last_al_u0307')} last  
{ewc ,JI('`,`length_al_u0307')} length  
{ewc ,JI('`,`list_al_u0307')} list  
{ewc ,JI('`,`listp_al_u0307')} listp  
{button ,JI('`,`load95dialog_al_u0307')} load_dialog  
{ewc ,JI('`,`load_al_u0307')} load  
{ewc ,JI('`,`log_al_u0307')} log  
{button ,JI('`,`logand_al_u0307')} logand
```

{ewc ,JI(','logior_al_u0307')} logior

{ewc ,JI(','lsh_al_u0307')} lsh

M

{button ,JI(','mapcar_al_u0307')} mapcar

{ewc ,JI(','max_al_u0307')} max

{button ,JI(','mem_al_u0307')} mem

{button ,JI(','member_al_u0307')} member

{ewc ,JI(','menucmd_al_u0307')} menucmd

{button ,JI(','menugroup_al_u0307')} gruppomenu

{button ,JI(','min_al_u0307')} min

{button ,JI(','minusp_al_u0307')} minusp

{ewc ,JI(','mode95tile_al_u0307')} mode_tile

N

{ewc ,JI(','namedobjdict_al_u0307')} namedobjdict

{ewc ,JI(','nentsel_al_u0307')} nentsel

{button ,JI(','nentselp_al_u0307')} nentselp

{ewc ,JI(','new95dialog_al_u0307')} new_dialog

{button ,JI(','not_al_u0307')} not

{button ,JI(','nth_al_u0307')} nth

{button ,JI(','null_al_u0307')} null

{ewc ,JI(','numberp_al_u0307')} numberp

O

{button ,JI(','open_al_u0307')} open

{ewc ,JI(','or_al_u0307')} or

{ewc ,JI(','osnap_al_u0307')} osnap

P

{button ,JI(','polar_al_u0307')} polar

{button ,JI(','prin1_al_u0307')} prin1

{button ,JI(','princ_al_u0307')} princ

{button ,JI(','print_al_u0307')} print

{button ,JI(','progn_al_u0307')} progn

{button ,JI(','prompt_al_u0307')} prompt

Q

{ewc ,JI(','quit_al_u0307')} quit

{ewc ,JI(','quote_al_u0307')} quote

R

{button ,JI(','read_al_u0307')} read

{ewc ,JI(','read45char_al_u0307')} read-char

{ewc ,JI(','read45line_al_u0307')} read-line

{ewc ,JI(','redraw_al_u0307')} redraw

{ewc ,JI(','regapp_al_u0307')} regapp

{ewc ,JI(','rem_al_u0307')} rem

{button ,JI(','repeat_al_u0307')} repeat

{ewc ,JI(','reverse_al_u0307')} reverse

{button ,JI(','rtos_al_u0307')} rtos

S

{ewc ,JI(','set_al_u0307')} set

{button ,JI(','set95tile_al_u0307')} set_tile

{button ,JI(','setcfg_al_u0307')} setcfg

{ewc ,JI(','setfunhelp_al_u0307')} setfunhelp

{ewc ,JI(','setq_al_u0307')} setq

{ewc ,JI(','setvar_al_u0307')} setvar

{ewc ,JI(','sin_al_u0307')} sin

{button ,JI(','setview_al_u0307')} setview

{button ,JI(','slide95image_al_u0307')} slide_image

{ewc ,JI(','snvalid_al_u0307')} snvalid

{button ,JI(','sqrt_al_u0307')} sqrt

{ewc ,JI(','ssadd_al_u0307')} ssadd

{button ,JI(','ssdel_al_u0307')} ssdel

{button ,JI(','ssget_al_u0307')} ssget

{button ,JI(','ssgetfirst_al_u0307')} ssgetfirst

{ewc ,JI(','sslenght_al_u0307')} sslenght

{ewc ,JI(','ssmemb_al_u0307')} ssmemb

{ewc ,JI(','ssname_al_u0307')} ssname

{ewc ,JI(','ssnamex_al_u0307')} ssnamex

{button ,JI(','sssetfirst_al_u0307')} sssetfirst

{ewc ,JI(','startapp_al_u0307')} startapp

{ewc ,JI(','start95dialog_al_u0307')} start_dialog

{button ,JI(','start95image_al_u0307')} start_image

{ewc ,JI(','start95list_al_u0307')} start_list

{button ,JI(','strcase_al_u0307')} strcase

{button ,JI(','strcat_al_u0307')} strcat

{ewc ,JI(','strlen_al_u0307')} strlen
{ewc ,JI(','subst_al_u0307')} subst
{button ,JI(','substr_al_u0307')} substr

T

{ewc ,JI(','tablet_al_u0307')} tablet
{ewc ,JI(','tblnext_al_u0307')} tblnext
{button ,JI(','tblobjname_al_u0307')} tblobjname
{ewc ,JI(','tblsearch_al_u0307')} tblsearch
{button ,JI(','term95dialog_al_u0307')} term_dialog
{ewc ,JI(','terpri_al_u0307')} terpri
{ewc ,JI(','textbox_al_u0307')} textbox
{button ,JI(','textpage_al_u0307')} textpage
{ewc ,JI(','textscr_al_u0307')} textscr
{button ,JI(','trace_al_u0307')} trace
{button ,JI(','trans_al_u0307')} trans
{button ,JI(','type_al_u0307')} type

U

{ewc ,JI(','unload95dialog_al_u0307')} unload_dialog
{ewc ,JI(','untrace_al_u0307')} untrace

V

{button ,JI(','vector95image_al_u0307')} vector_image
{button ,JI(','ver_al_u0307')} ver
{button ,JI(','vports_al_u0307')} vports

W

{ewc ,JI(','wcmatch_al_u0307')} wcmatch
{button ,JI(','while_al_u0307')} while
{button ,JI(','write45char_al_u0307')} write-char
{button ,JI(','write45line_al_u0307')} write-line

X

{button ,JI(','xdroom_al_u0307')} xdroom
{ewc ,JI(','xdsiize_al_u0307')} xdsiize
{ewc ,JI(','xload_al_u0307')} xload
{button ,JI(','xunload_al_u0307')} xunload

Z

```
{button ,JI('`zerop_al_u0307')} zerop
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni **+ (addizione)**

Restituisce la somma di tutti gli argomenti.

(+ [numero numero] . . .)

Se si fornisce solo un argomento *numero*, questa funzione restituisce come risultato il numero aggiunto a zero, quindi il numero stesso. Se non viene specificato alcun argomento, la funzione restituisce 0.

(+ 1 2)	restituisce	3
(+ 1 2 3 4.5)	restituisce	10.5
(+ 1 2 3 4.0)	restituisce	10.0

Capitolo 13 -- AutoLISP: catalogo delle funzioni **- (sottrazione)**

Sottrae il secondo argomento e i seguenti dal primo e restituisce la differenza.

(-[numero numero] . . .)

Se si forniscono più di due argomenti *numero*, la somma dei numeri a partire dal secondo viene sottratta dal primo e viene restituito il valore finale. Se si specifica solo un argomento *numero*, viene restituito il risultato dato dalla sottrazione di questo argomento da zero; la funzione restituisce il numero stesso. Se non viene specificato alcun argomento, la funzione restituisce 0.

(- 50 40)	restituisce	10
(- 50 40.0)	restituisce	10.0
(- 50 40.0 2.5)	restituisce	7.5
(- 8)	restituisce	-8

Capitolo 13 -- AutoLISP: catalogo delle funzioni *** (moltiplicazione)**

Restituisce il prodotto di tutti gli argomenti.

(* [numero numero] . . .)

Se si specifica solo un argomento *numero*, viene restituito il risultato dato dalla divisione di questo argomento per uno; la funzione restituisce il numero stesso. Se non viene specificato alcun argomento, la funzione restituisce 0.

(* 2 3)	restituisce	6
(* 2 3.0)	restituisce	6.0
(* 2 3 4.0)	restituisce	24.0
(* 3 -4.5)	restituisce	-13.5
(* 3)	restituisce	3

Capitolo 13 -- AutoLISP: catalogo delle funzioni

/ (divisione)

Divide il primo argomento per il prodotto degli argomenti restanti e restituisce il resto.

(/ [numero numero] . . .)

Se si forniscono più di due argomenti *numero*, il primo numero viene diviso per il prodotto dei numeri a partire dal secondo e viene restituito il quoziente finale. Se si specifica solo un argomento *numero*, viene restituito il risultato dato dalla divisione di questo argomento per uno; la funzione restituisce il numero stesso. Se non viene specificato alcun argomento, la funzione restituisce 0.

(/ 100 2)	<i>restituisce</i>	50
(/ 100 2.0)	<i>restituisce</i>	50.0
(/ 100 20.0 2)	<i>restituisce</i>	2.5
(/ 100 20 2)	<i>restituisce</i>	2
(/ 4)	<i>restituisce</i>	4

Capitolo 13 -- AutoLISP: catalogo delle funzioni

= (uguale a)

Restituisce T se tutti gli argomenti sono numericamente uguali, altrimenti restituisce nil.

(= *str_num* [*str_num*] . . .)

Ogni argomento *str_num* può essere un numero o una stringa.

(= 4 4.0)	<i>restituisce</i>	T
(= 20 388)	<i>restituisce</i>	nil
(= 2.4 2.4 2.4)	<i>restituisce</i>	T
(= 499 499 500)	<i>restituisce</i>	nil
(= "io" "io")	<i>restituisce</i>	T
(= "io" "tu")	<i>restituisce</i>	nil

Vedere anche

le funzioni `eq` e `equal` .

Capitolo 13 -- AutoLISP: catalogo delle funzioni

/= (diverso da)

Restituisce T se gli argomenti non sono numericamente uguali, altrimenti restituisce nil.

(/= *str_num* [*str_num*] . . .)

Ogni argomento *str_num* può essere un numero o una stringa.

(/= 10 20)	<i>restituisce</i>	T
(= "tu" "tu")	<i>restituisce</i>	nil
(/= 5.43 5.44)	<i>restituisce</i>	T
(/= 10 20)	<i>restituisce</i>	T

Capitolo 13 -- AutoLISP: catalogo delle funzioni

< (minore di)

Restituisce T se ogni argomento è numericamente minore rispetto a quello posizionato alla sua destra, altrimenti restituisce nil.

(< *str_num* [*str_num*] . . .)

Ogni argomento *str_num* può essere un numero o una stringa.

(< 10 20)	<i>restituisce</i>	T
(< "b" "c")	<i>restituisce</i>	T
(< 357 33.2)	<i>restituisce</i>	nil
(< 2 3 88)	<i>restituisce</i>	T
(< 2 3 4 4)	<i>restituisce</i>	nil

Capitolo 13 -- AutoLISP: catalogo delle funzioni

<= (minore di o uguale a)

Restituisce T se ogni argomento è numericamente minore rispetto a quello posizionato alla sua destra, altrimenti restituisce nil.

(<= *str_num* [*str_num*] . . .)

Ogni argomento *str_num* può essere un numero o una stringa.

(<= 10 20)	<i>restituisce</i>	T
(<= "b" "b")	<i>restituisce</i>	T
(<= 357 33.2)	<i>restituisce</i>	nil
(<= 2 9 9)	<i>restituisce</i>	T
(<= 2 9 4 5)	<i>restituisce</i>	nil

Capitolo 13 -- AutoLISP: catalogo delle funzioni

> (maggiore di)

Restituisce T se ogni argomento è numericamente maggiore rispetto a quello posizionato alla sua destra, altrimenti restituisce nil.

(> *str_num* [*str_num*] . . .)

Ogni argomento *str_num* può essere un numero o una stringa.

(> 120 17)	<i>restituisce</i>	T
(> "c" "b")	<i>restituisce</i>	T
(> 3.5 1792)	<i>restituisce</i>	nil
(> 77 4 2)	<i>restituisce</i>	T
(> 77 4 4)	<i>restituisce</i>	nil

Capitolo 13 -- AutoLISP: catalogo delle funzioni

>= (maggiore di o uguale a)

Restituisce T se ogni argomento è numericamente minore rispetto a quello posizionato alla sua destra, altrimenti

restituisce nil.

(>= *str_num* [*str_num*] ...)

Ogni argomento *str_num* può essere un numero o una stringa.

(>= 120 17)	<i>restituisce</i>	T
(>= "c" "c")	<i>restituisce</i>	T
(>= 3.5 1792)	<i>restituisce</i>	nil
(>= 77 4 4)	<i>restituisce</i>	T
(>= 77 4 9)	<i>restituisce</i>	nil

Capitolo 13 -- AutoLISP: catalogo delle funzioni

~ (NOT a livello bit)

Restituisce l'operatore a livello bit NOT (complemento ar1) dell'argomento.

(~ <i>int</i>)		
(~ 3)	<i>restituisce</i>	-4
(~ 100)	<i>restituisce</i>	-101
(~ -4)	<i>restituisce</i>	3

Capitolo 13 -- AutoLISP: catalogo delle funzioni

1+ (incremento)

Restituisce l'argomento aumentato di 1 (incrementato).

(1+ <i>numero</i>)		
(1+ 5)	<i>restituisce</i>	6
(1+ -17.5)	<i>restituisce</i>	-16.5

Capitolo 13 -- AutoLISP: catalogo delle funzioni

1- (decremento)

Restituisce l'argomento diminuito di 1 (decrementato).

(1- <i>numero</i>)		
(1- 5)	<i>restituisce</i>	4
(1- -17.5)	<i>restituisce</i>	-18.5

Capitolo 13 -- AutoLISP: catalogo delle funzioni

abs

Restituisce il valore assoluto dell'argomento.

(abs *numero*)

(abs 100)	<i>restituisce</i> 100
(abs -100)	<i>restituisce</i> 100
(abs -99.25)	<i>restituisce</i> 99.25

Capitolo 13 -- AutoLISP: catalogo delle funzioni

acad_colordlg

Visualizza la finestra di dialogo standard per la selezione dei colori di AutoCAD.

(acad_colordlg num_colore [flag])

L'argomento *num_colore* è un numero intero compreso nell'intervallo 0-256 (inclusi) ed indica il numero del colore di AutoCAD da visualizzare come valore iniziale di default. Se *num_colore* è uguale a 0, viene impostato per default su DABLOCCO; se è uguale a 256 viene impostato per default su DALAYER. L'impostazione dell'argomento opzionale *flag* su nil comporta la disabilitazione dei pulsanti DALAYER e DABLOCCO. L'omissione di tale argomento o l'impostazione su un valore diverso da nil abilita i pulsanti DALAYER e DABLOCCO.

La funzione **acad_colordlg** restituisce il numero del colore selezionato dall'utente. Se si annulla una finestra di dialogo, **acad_colordlg** restituisce nil.

Il codice seguente richiede all'utente di selezionare un colore ed indica verde come colore di default:

```
(acad_colordlg 3)
```

Funzione definita esternamente applicazione ARX *acadapp*

Capitolo 13 -- AutoLISP: catalogo delle funzioni

acad_helpdlg

Richiama la Guida (obsoleto).

(acad_helpdlg file_guida argomento)

Questa funzione definita esternamente è stata sostituita dalla funzione incorporata **help** per consentire la compatibilità con le release precedenti di AutoCAD. Vedere "**help**" per la descrizione completa di questa funzione.

Funzione definita esternamente applicazione AutoLIST *acadr14.lsp*

Capitolo 13 -- AutoLISP: catalogo delle funzioni

acad_strlsort

Dispone una lista di stringhe in ordine alfabetico.

(acad_strlsort lista)

L'argomento *lista* rappresenta la lista delle stringhe da ordinare. La funzione **acad_strlsort** restituisce una lista delle stesse stringhe disposte in ordine alfabetico. Se la lista di argomenti *lista* non è valida o se non vi è memoria sufficiente per eseguire l'ordinamento, **acad_strlsort** restituisce nil.

Il codice seguente ordina la lista dei nomi abbreviati dei mesi:

```
(setq mos '("Gen" "Feb" "Mar" "Apr" "Mag" "Giu"
            "Lug" "Ago" "Set" "Ott" "Nov" "Dic"))
(acad_strlsort mos)
```

Viene restituita la lista seguente:

```
("Ago" "Apr" "Dic" "Feb" "Gen" "Giu"
 "Lug" "Mag" "Mar" "Nov" "Ott" "Set")
```

Funzione definita esternamente applicazione ARX *acadapp*

Capitolo 13 -- AutoLISP: catalogo delle funzioni

action_tile

Assegna un'azione da valutare quando l'utente seleziona la casella specificata in una finestra di dialogo.

(action_tile *chiave espressione-azione*)

Gli argomenti *chiave* ed *espressione-azione* sono stringhe. L'argomento *chiave* indica il nome della casella che attiva l'azione, specificata come attributo chiave. È sensibile a maiuscolo/minuscolo. L'argomento *espressione-azione* viene valutato al momento della selezione della casella.

L'azione assegnata dalla funzione **action_tile** sostituisce l'azione di default della finestra di dialogo, assegnata da **new_dialog**, o l'attributo azione della casella, se specificati. L'espressione si può riferire al valore corrente della casella (l'attributo valore) come \$value, al relativo nome come \$key, ai dati specifici dell'applicazione, in base a quanto impostato dalla funzione **client_data_tile**, come \$data, al motivo del richiamo come \$reason e se la casella è un pulsante di immagine, alle coordinate dell'immagine come \$x e \$y.

Nota Non è possibile richiamare la funzione AutoLISP **command** dalla funzione **action_tile**.

Vedere anche

"Azioni di default e DCL"

Capitolo 13 -- AutoLISP: catalogo delle funzioni

add_list

Aggiunge o modifica una stringa nell'elenco delle finestre di dialogo correntemente attivo.

(add_list *stringa*)

Prima di utilizzare **add_list**, è necessario aprire l'elenco ed inizializzarlo con una chiamata alla funzione **start_list**. L'attributo *string* viene aggiunto all'elenco corrente oppure sostituisce l'elemento di tale elenco a seconda dell'operazione specificata in **start_list**.

Si supponga che il file DCL attualmente attivo contenga un `popup_list` o un `list_box` con la chiave `elencolungo`; la parte di codice seguente inizializza l'elenco aggiungendovi le stringhe di testo presenti in `l1ist`.

```
(setq l1ist '("prima riga" "seconda riga" "terza riga"))
(start_list "elencolungo")
(mapcar 'add_list l1ist)
(end_list)
```

Una volta definito l'elenco, la parte di codice di seguito riportata cambia il testo presente sulla seconda riga in "2a riga".

```
(start_list "elencolungo" 1 0)
(add_list "2a riga")
(end_list)
```

Vedere anche

le funzioni **start_list** e **end_list** .

Capitolo 13 -- AutoLISP: catalogo delle funzioni

ads

Restituisce un elenco delle applicazioni ADS correntemente caricate.

(ads)

Tutte le applicazioni e il relativo percorso sono stringhe tra virgolette.

```
(ads) restituisce ad esempio ("files/progs/PROG1" "PROG2")
```

Vedere anche

le funzioni **xload** e **xunload** .

Capitolo 13 -- AutoLISP: catalogo delle funzioni

alert

Visualizza una casella di avviso contenente l'errore o il messaggio di avvertimento assegnato sotto forma di stringa.

(alert *stringa*)

Una casella di avviso è una finestra di dialogo contenente un unico pulsante OK.

```
(alert "Questa funzione non è disponibile.")
```

È possibile visualizzare più righe utilizzando il carattere di riga nuova nell'argomento *stringa*.

```
(alert "Questa funzione\nnon è disponibile.")
```

Nota La lunghezza della riga ed il numero di righe di una casella di avviso dipendono dalla piattaforma, dal dispositivo e dalla finestra utilizzata. AutoCAD tronca le stringhe che sono troppo lunghe per poter essere visualizzate in una casella di avviso.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

alloc

Imposta la dimensione del segmento ad un determinato numero di nodi.

(alloc *int*)

Questa funzione restituisce la dimensione precedente del segmento.

Vedere anche

"Allocazione manuale"

Capitolo 13 -- AutoLISP: catalogo delle funzioni

and

Restituisce l'AND logico di un elenco di espressioni.**(and espressione ...)**

Questa funzione interrompe ogni ulteriore valutazione e restituisce nil se una delle espressioni produce nil; altrimenti restituisce T.

Ad esempio, date le assegnazioni

```
(setq a 103 b nil c "stringa")
```

si avrà che

```
(and 1.4 a c)           restituisce  T
(and 1.4 a b c)        restituisce  nil
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni **angle****Restituisce un angolo misurato in radianti di una linea definita da due punti finali.****(angle pt1 pt2)**

L'angolo viene misurato in radianti dall'asse X del corrente piano di costruzione e viene incrementato in senso antiorario. Se vengono forniti punti tridimensionali, essi vengono proiettati sul piano corrente di costruzione.

```
(angle '(1.0 1.0) '(1.0 4.0))  restituisce  1.5708
(angle '(5.0 1.33) '(2.4 1.33)) restituisce  3.14159
```

Vedere anche

"Conversione angolare."

Capitolo 13 -- AutoLISP: catalogo delle funzioni **angtof****Converte in un valore reale (a virgola mobile) una stringa che rappresenta un angolo misurato in radianti.****(angtof stringa [modalità])**

L'argomento *stringa* descrive un angolo nel formato specificato dall'argomento *modalità*. L'argomento *modalità*, riportato nella tabella seguente, indica le unità di misura in cui viene formattata la stringa. Il valore deve corrispondere ai valori consentiti per la variabile di sistema AUNITS di AutoCAD. Se si omette *modalità*, **angtof** utilizza il valore corrente di AUNITS.

Valori di unità angolari

Valore per modalità	Formato stringa
0	Gradi
1	Gradi/minuti/secondi
2	Gradi centesimali
3	Radianti
4	Unità topografiche

La *stringa* deve essere una stringa che la funzione **angtof** può analizzare correttamente in base alla *modalità* specificata. Può avere lo stesso formato che viene restituito dalla funzione **angtos** oppure il formato consentito da AutoCAD per l'immissione da tastiera.

Le funzioni **angtof** e **angtos** sono complementari: se alla funzione **angtof** viene passata una stringa creata da **angtos**, **angtof** restituirà un valore valido e viceversa (presumendo che i valori di *modalità* corrispondano).

Se **angtof** viene eseguita con esito positivo, restituisce un valore reale espresso in radianti, altrimenti restituisce nil.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

angtos

Converte un valore angolare espresso in radianti in una stringa.

(*angtos* *angolo* [*modalità* [*precisione*]])

La funzione **angtos** valuta *angolo*, un numero reale espresso in radianti, e lo restituisce modificato in una stringa in base alle impostazioni di *modalità*, di *precisione*, della variabile di sistema `UNITMODE` di AutoCAD e della variabile di quotatura `DIMZIN`. Gli argomenti *modalità* e *precisione* sono numeri interi che indicano la modalità e la precisione delle unità angolari.

Valori di unità angolari

Valore per modalità	Formato stringa
0	Gradi
1	Gradi/minuti/secondi
2	Gradi centesimali
3	Radianti
4	Unità topografiche

L'argomento *precisione* è un numero intero che seleziona il numero di decimali per la precisione desiderata. Gli argomenti *modalità* e *precisione* corrispondono alle variabili di sistema `AUNITS` e `AUPREC` di AutoCAD. Se si omettono questi argomenti, **angtos** utilizzerà rispettivamente le impostazioni correnti di `AUNITS` e `AUPREC`.

La funzione **angtos** accetta un argomento *angolo* negativo; tuttavia, prima di eseguire la conversione specificata, lo riduce sempre ad un valore positivo compreso tra zero e 2 pi radianti.

```
(angtos 0.785398 0 4)      restituisce "45.0000"
(angtos -0.785398 0 4)    restituisce "315.0000"
```

La variabile `UNITMODE` interessa la stringa restituita quando si selezionano le unità topografiche, se il valore dell'attributo *modalità* è 4. Se `UNITMODE` è uguale a 0, nella stringa vengono inseriti degli spazi (ad esempio, "N 45d E"); se `UNITMODE` è uguale a 1, la stringa non contiene spazi (ad esempio, "N45dE").

Nota Le routine che utilizzano la funzione **angtos** per visualizzare gli angoli arbitrari (ossia non relativi al valore di `ANGBASE`) devono controllare e considerare il valore di `ANGBASE`.

Vedere anche

"Conversioni di stringhe."

Capitolo 13 -- AutoLISP: catalogo delle funzioni

append

Accoda un determinato numero di liste come se facessero parte di una singola lista.

(*append* *lista* . . .)

```
(append '(a b) '(c d))      restituisce (A B C D)
(append '((a) (b)) '((c) (d))) restituisce ((A) (B) (C) (D))
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

apply

Passa un elenco di argomenti ad una funzione specificata.

(apply 'funzione elenco)

La funzione **apply** può essere utilizzata sia con le funzioni incorporate (subroutine) che con le funzioni definite dall'utente, vale a dire quelle create utilizzando **defun** o **lambda**.

```
(apply '+ '(1 2 3)) restituisce 6
(apply 'strcat '("a" "b" "c")) restituisce "abc"
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni **arx**

Restituisce un elenco delle applicazioni ARX correntemente caricate.

(arx)

Tutte le applicazioni e il relativo percorso sono stringhe tra virgolette.

```
(arx) potrebbe restituire ("files/progs/PROG1" "PROG2")
```

Vedere anche

le funzioni **arxload** e **arxunload**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni **arxload**

Carica un'applicazione ARX.

(arxload applicazione [se_fallisce])

L'argomento *applicazione* viene immesso come stringa racchiusa tra virgolette o come variabile contenente il nome di un file eseguibile. Nel momento del caricamento del file viene verificata la validità dell'applicazione ARX.

Se l'operazione **arxload** fallisce, si determina un errore AutoLISP. Comunque se viene fornito l'argomento *se_fallisce*, in caso di errore la funzione **arxload** restituisce il valore dell'argomento invece di un messaggio di errore.

Se il caricamento dell'applicazione ha esito positivo, viene restituito il nome dell'applicazione.

```
(arxload "/myapps/appx") se riesce, restituisce "/myapps/appx"
```

Se si tenta di caricare un'applicazione già caricata, la funzione **arxload** visualizza il seguente messaggio e restituisce il nome dell'applicazione.

```
Applicazione "applicazione" già caricata.
```

Le applicazioni ARX correntemente caricate con la funzione **arx** possono essere controllate prima di usare **arxload**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni **arxunload**

Scarica un'applicazione ARX.

(arxunload applicazione [se_fallisce])

Se l'applicazione viene scaricata con successo, viene restituito il nome dell'applicazione, altrimenti viene visualizzato un messaggio di errore.

Digitare *applicazione* come stringa tra virgolette o come variabile contenente il nome di un'applicazione caricata con la funzione **arxload**. Il nome dell'applicazione deve essere inserito nello stesso modo in cui è stato inserito per la funzione **arxload**. Se è stato digitato un percorso (nome della directory) o un'estensione per l'applicazione in **arxload**, esso può essere omesso nella funzione **arxunload**.

Ad esempio, il seguente codice scarica l'applicazione caricata precedentemente con la funzione **arxload**:

```
(arxunload "appx") se riesce, restituisce "appx"
```

Se l'operazione **arxunload** fallisce, generalmente si determina un errore AutoLISP. Comunque, se viene fornito l'argomento *se_fallisce*, **arxunload** restituisce il valore di questo argomento invece di un messaggio di errore. Questa caratteristica di **arxunload** è simile a quella della funzione **arxload**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

ascii

Restituisce la conversione del primo carattere di una stringa nel relativo codice ASCII (numero intero).

(**ascii** *stringa*)

```
(ascii "A")           restituisce 65
(ascii "a")           restituisce 97
(ascii "BIG")         restituisce 66
```

Simile alla funzione **ASC** in linguaggio BASIC®.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

assoc

Ricerca un elemento in una lista di associazioni e restituisce la voce associata.

(**assoc** *elem lista_assoc*)

Ricerca *elem* nella lista di associazioni *lista_assoc* per l'argomento come elemento chiave e restituisce la voce associata. Se **assoc** non individua *elem* come chiave in *lista_assoc*, restituisce nil.

Ad esempio, dato il codice:

```
(setq al '((name box) (width 3) (size 4.7263) (depth 5)))
```

si avrà che

```
(assoc 'size al)           restituisce (SIZE 4.7263)
(assoc 'weight al)        restituisce nil
```

Le liste di associazioni vengono usate generalmente per memorizzare dati ai quali si può accedere mediante una *chiave*. La funzione **subst** risulta molto utile quando si deve sostituire il valore associato con una chiave in una lista di associazioni.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

atan

Restituisce l'arcotangente espresso in radianti di un argomento.

(**atan** *num1 [num2]*)

Se si fornisce un solo argomento alla funzione **atan**, questa restituisce l'arcotangente di *num1* espresso in radianti. Se si specificano entrambi gli argomenti *num1* e *num2*, **atan** restituisce l'arcotangente di *num1/num2* espresso in radianti. Se *num2* è uguale a zero, viene restituito un angolo di più o meno 1.570796 radianti (+90 gradi° oppure -90 gradi), in base al segno di *num1*. I valori degli angoli restituiti sono compresi tra $-\pi/2$ to $+\pi/2$ radianti.

(atan 0.5)	<i>restituisce</i>	0.463648
(atan 1.0)	<i>restituisce</i>	0.785398
(atan -1.0)	<i>restituisce</i>	0.785398
(atan 2.0 3.0)	<i>restituisce</i>	0.588003
(atan 2.0 -3.0)	<i>restituisce</i>	2.55359
(atan 1.0 0.0)	<i>restituisce</i>	1.5708

Nota la funzione **angtos** converte in un valore di stringa il valore in radianti restituito dalla funzione **atan**.

(angtos (atan -1.0) 0 4)	<i>restituisce</i>	"315.0000"
(angtos (atan 2.0 -3.0) 0 4)	<i>restituisce</i>	"33.6901"
(angtos (atan 2.0 -3.0) 0 4)	<i>restituisce</i>	"146.3099"
(angtos (atan 1.0 0.0) 0 4)	<i>restituisce</i>	"90.0000"

Capitolo 13 -- AutoLISP: catalogo delle funzioni

atof

Restituisce la conversione di una stringa in un numero reale.

(atof <i>stringa</i>)		
(atof "97.1")	<i>restituisce</i>	97.1
(atof "3")	<i>restituisce</i>	3.0
(atof "3.9")	<i>restituisce</i>	3.9

Capitolo 13 -- AutoLISP: catalogo delle funzioni

atoi

Restituisce la conversione di una stringa in un numero intero.

(atoi <i>stringa</i>)		
(atoi "97")	<i>restituisce</i>	97
(atoi "3")	<i>restituisce</i>	3
(atoi "3.9")	<i>restituisce</i>	3

Capitolo 13 -- AutoLISP: catalogo delle funzioni

atom

Verifica che un elemento sia un atomo.

(**atom** *elem*)

Questa funzione restituisce nil se *elem* è un elenco, altrimenti restituisce T. Se un elemento non è un elenco, viene considerato un atomo.

Ad esempio, date le assegnazioni

```
(setq a '(x y z))
(setq b 'a)
```

si avrà che

```
(atom 'a)           restituisce T
(atom a)           restituisce nil
(atom 'b)          restituisce T
(atom b)           restituisce T
(atom '(a b c))    restituisce nil
```

Alcune versioni di LISP interpretano in modo diverso la funzione `atom`; pertanto, è necessario prestare attenzione nell'utilizzare il codice convertito.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

`atoms-family`

Restituisce un elenco dei simboli attualmente definiti.

(atoms-family *formato* [*elenco_simb*])

L'argomento *formato* è un valore intero uguale a 0 o 1. Se il valore di *formato* è 0, la funzione **atoms-family** restituisce i nomi dei simboli sotto forma di elenco. Se *formato* è 1, i nomi dei simboli vengono restituiti sotto forma di elenco di stringhe. La funzione **atoms-family** ricerca un determinato elenco di nomi di simboli qualora si fornisca l'argomento *elenco_simb*. Quest'ultimo argomento è un elenco di stringhe che specificano i nomi dei simboli. La funzione **atoms-family** restituisce un elenco del tipo specificato in base al formato (simboli o stringhe) contenente i nomi dei simboli definiti e nil per quelli non definiti.

```
(atoms-family 0) restituisce un elenco dei simboli attualmente definiti
```

Il codice di seguito riportato verifica che siano stati definiti i simboli CAR, CDR e XYZ e restituisce l'elenco sotto forma di stringhe:

```
(atoms-family 1
  ("CAR" "CDR" "XYZ")) restituisce ("CAR" "CDR" nil)
```

Il valore restituito, illustrato nell'esempio, indica che il simbolo XYZ non è stato definito.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

`autoarxload`

Predefinisce i nomi dei comandi per caricare un file ARX associato.

(autoarxload *nomefile cmdlist*)

L'argomento *nome_file* è una stringa che specifica il file *arx* caricato quando uno dei comandi definiti dall'argomento *elenco_cmd* viene immesso alla riga di comando. L'argomento *elenco_cmd* deve essere un elenco di stringhe. La funzione **autoarxload** restituisce nil.

Il codice seguente definisce le funzioni C:APP1, C:APP2 e C:APP3 per caricare il file *bounsapp.arx*. La prima volta che viene immesso il comando APP1, APP2 o APP3 alla riga di comando, l'applicazione ARX viene caricata e il comando continua.

```
(autoarxload "BONUSAPP" '("APP1" "APP2" "APP3"))
```

Attenzione I comandi elencati dall'argomento *elenco_cmd* devono essere definiti come comandi dal file specificato dall'argomento *nome_file*.

Funzione definita esternamente applicazione AutoLIST *acadr14.lsp*

Capitolo 13 -- AutoLISP: catalogo delle funzioni

autoload

Predefinisce i nomi dei comandi per caricare un file AutoLISP associato.

(autoload *nomefile cmdlist*)

L'argomento *nome_file* è una stringa che specifica il file *.lsp* caricato quando viene immesso uno dei comandi definiti dall'argomento *elenco_cmd* dalla riga di comando. L'argomento *elenco_cmd* deve essere un elenco di stringhe. La funzione **autoload** restituisce nil.

Il codice seguente definisce le funzioni C:APP1, C:APP2 e C:APP3 per caricare il file *bounsapp.lsp*. La prima volta che viene immesso il comando APP1, APP2 o APP3 alla riga di comando, viene caricato il file AutoLISP e il comando continua.

```
(autoload "BONUSAPP" ' ("APP1" "APP2" "APP3"))
```

Attenzione I comandi elencati dall'argomento *elenco_cmd* devono essere definiti come comandi dal file specificato dall'argomento *nome_file*.

Funzione definita esternamente applicazione AutoLIST *acadr14.lsp*

Capitolo 13 -- AutoLISP: catalogo delle funzioni

autoload

Predefinisce i nomi dei comandi per caricare un'applicazione ADS associata.

(autoload *nomefile cmdlist*)

L'argomento *nome_file* è una stringa che specifica l'applicazione ADS caricata quando viene immesso uno dei comandi definiti dall'argomento *elenco_cmd* alla riga di comando. L'argomento *elenco_cmd* deve essere un elenco di stringhe. La funzione **autoload** restituisce nil.

Il codice seguente definisce le funzioni C:APP1, C:APP2 e C:APP3 per caricare l'applicazione ADS *bounsapp*. La prima volta che viene immesso il comando APP1, APP2 o APP3 alla riga di comando viene caricata l'applicazione ADS e il comando continua.

```
(autoload "BONUSAPP" ' ("APP1" "APP2" "APP3"))
```

Attenzione I comandi elencati dall'argomento *elenco_cmd* devono essere definiti come comandi dal file specificato dall'argomento *nome_file*.

Funzione definita esternamente applicazione AutoLIST *acadr14.lsp*

Capitolo 13 -- AutoLISP: catalogo delle funzioni

Boole

Funge da funzione booleana generale a livello di bit.

(Boole *funz int1 int2 . . .*)

L'argomento *funz* è un numero intero compreso tra 0 e 15 che rappresenta una delle 16 funzioni booleane possibili su due variabili. Gli argomenti interi successivi sono combinati logicamente a livello bit in base a questa funzione ed alla tabella contenente i valori booleani validi di seguito riportati.

Tabella dei valori booleani validi

Int1	Int2	Bit funz
0	0	8
0	1	4

1	0	2
1	1	1

Ogni bit di *int1* viene associato al bit corrispondente *int2*, indicando una riga orizzontale della tabella. Il bit risultante è 0 o 1, a seconda dell'impostazione del bit *funz* che corrisponde alla riga interessata.

Se il bit appropriato è impostato nell'argomento *funz*, il bit risultante è 1, altrimenti è 0. Alcuni dei valori relativi all'attributo *funz* equivalgono alle operazioni booleane standard: AND, OR, XOR e NOT.

Valori dei bit di funzione per operazioni booleane

Funzione	Operazione	Il bit di risultante è 1 se...
1	AND	Entrambi i bit di input sono 1.
6	XOR	Solo uno dei due bit di input è 1.
7	OR	Uno o entrambi i bit di input sono 1.
8	NOR	Entrambi i bit di input sono 0 (complemento di 1).

Di seguito è riportata un'operazione logica AND per i valori 12 e 5:

```
(Boole 1 12 5)          restituisce 4
```

Allo stesso modo, nell'esempio seguente viene mostrata un'operazione logica XOR per i valori 6 e 5:

```
(Boole 6 6 5)          restituisce 3
```

È possibile utilizzare altri valori dell'attributo *funz* per eseguire altre operazioni booleane per le quali non sono definiti nomi standard. Ad esempio, se *funz* è uguale a 4, i bit risultanti vengono impostati nel caso in cui i bit corrispondenti siano impostati in *int2*, ma non in *int1*. Pertanto, si avrà che:

```
(Boole 4 3 14)         restituisce 12
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

boundp

Verifica se un valore è associato ad un simbolo.

```
(boundp simb)
```

Restituisce T se all'argomento *simb* è associato un valore. Se invece non è associato alcun valore oppure se è associato a nil, la funzione **boundp** restituisce nil. Se l'argomento *simb* è un simbolo non definito, viene creato automaticamente e viene associato a nil.

Ad esempio, date le assegnazioni

```
(setq a 2 b nil)
```

si avrà che

```
(boundp 'a)          restituisce T
(boundp 'b)          restituisce nil
```

La funzione **atoms-family** offre un metodo alternativo per stabilire l'esistenza di un simbolo senza crearlo automaticamente.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

car e cdr

Restituisce il primo elemento di una lista oppure una lista contenente tutti gli elementi tranne il primo.

```
(car lista) e (cdr lista)
```

Se l'argomento *lista* è vuoto, la funzione **car** restituisce nil.

```
(car '(a b c))           restituisce A
(car '((a b) c))        restituisce (A B)
(car '())                restituisce nil
```

Se l'argomento *lista* è vuoto, la funzione **cdr** restituisce nil.

```
(cdr '(a b c))           restituisce (B C)
(cdr '((a b) c))        restituisce (C)
(cdr '())                restituisce nil
```

Quando l'argomento *lista* è rappresentato da una coppia puntata (vedere "**cons**"), la funzione **cdr** restituisce il secondo elemento senza inserirlo in un elenco.

```
(cdr '(a . b))           restituisce B
(cdr '(1 . "Testo"))     restituisce "Testo"
```

AutoLISP prevede concatenazioni di **car** e **cdr** fino a quattro livelli. Di seguito sono riportate le funzioni valide:

caaar	cadaar	cdaaar	cddaar
caadr	cadadr	cdadr	cddadr
caaar	caaar	caaar	caaar
caadar	caddar	cdadar	cdddar
caaddr	caddr	cdaddr	cddddr
caadr	cadr	cdadr	cdddr
caar	car	cdar	cddr

Queste concatenazioni equivalgono alle chiamate nidificate alle funzioni **car** e **cdr**. Ogni *a* rappresenta una chiamata a **car**, mentre ogni *d* rappresenta una chiamata a **cdr**. Ad esempio:

```
(caar x)   equivale a (car (car x))
(cdar x)   equivale a (cdr (car x))
(cadar x)  equivale a (car (cdr (car x)))
(cadr x)   equivale a (car (cdr x))
(cddr x)   equivale a (cdr (cdr x))
(caddr x)  equivale a (car (cdr (cdr x)))
```

In AutoLISP, la funzione **cadr** viene di solito usata per ottenere la coordinata Y di un punto bidimensionale o tridimensionale, vale a dire il secondo elemento di un elenco composto da due o tre valori reali. Allo stesso modo, la funzione **caddr** può essere utilizzata per ottenere la coordinata Z di un punto tridimensionale. Ad esempio, date le assegnazioni

```
(setq pt2 '(5.25 1.0))      un punto bidimensionale
(setq pt3 '(5.25 1.0 3.0))  un punto tridimensionale
```

si avrà che

```
(car pt2)           restituisce 5.25
(cadr pt2)          restituisce 1.0
(caddr pt2)         restituisce nil
(car pt3)           restituisce 5.25
(cadr pt3)          restituisce 1.0
(caddr pt3)         restituisce 3.0
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni



chr

Restituisce la conversione di un numero intero che rappresenta un codice ASCII in una stringa costituita da un unico carattere.

(chr num_intero)

```
(chr 65)           restituisce "A"
```

```
(chr 66)          restituisce "B"
(chr 97)          restituisce "a"
```

Simile alla funzione **chr\$** in linguaggio BASIC.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

client_data_tile

Associa i dati gestiti dall'applicazione ad una casella della finestra di dialogo.

```
(client_data_tile chiave daticlente)
```

L'argomento *chiave* è una stringa che indica una casella, ed è sensibile al maiuscolo/minuscolo. I dati rappresentano una stringa specificata dall'argomento *daticlente*. Un'espressione di azione o una funzione di richiamo possono fare riferimento ad una stringa come \$data.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

close

Chiude un file aperto.

```
(close desc_file)
```

L'argomento *desc_file* è un descrittore di file ottenuto dalla funzione **open**. Dopo aver eseguito una funzione **close**, il descrittore di file rimane invariato, ma non è più valido. I dati aggiunti ad un file aperto vengono di fatto scritti solo dopo che il file viene chiuso. La funzione **close** restituisce nil se *desc-file* è valido, altrimenti restituisce un messaggio di errore.

Ad esempio, il seguente codice conta il numero di righe nel file *tafile.txt* e imposta la variabile *ct* su questo numero.

```
(setq fil "TALEFILE.TXT")
(setq x (open fil "r") ct 0)
(while (read-line x)
  (setq ct (1+ ct))
)
(close x)
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

command

Esegue un comando di AutoCAD.

```
(command [argomenti] . . .)
```

L'argomento *argomenti* rappresenta i comandi di AutoCAD e le opzioni corrispondenti. Gli argomenti della funzione **command** possono essere stringhe, numeri reali, numeri interi o punti come previsto dalla sequenza di richiesta del comando eseguito. Una stringa nulla ("") equivale a premere RETURN sulla tastiera. Il richiamo di **command** senza argomenti equivale a premere ESC ed annulla la maggior parte dei comandi AutoCAD. La funzione **command** restituisce nil.

La funzione **command** valuta ogni argomento e lo invia ad AutoCAD in risposta a messaggi di richiesta successivi. Inoltre i nomi dei comandi e le opzioni vengono sottomessi come stringhe, i punti bidimensionali come elenchi di due numeri reali ed i punti tridimensionali come elenchi di tre numeri reali. I nomi dei comandi vengono riconosciuti da AutoCAD solo quando digitati alla riga di comando.

Nota se la funzione **command** viene utilizzata in un file *acad.lsp* o *.mnl*, deve essere richiamata solo dall'interno di una istruzione **defun**. Utilizzare la funzione **S::STARTUP** per definire i comandi che devono essere eseguiti immediatamente all'inizio di una sessione di disegno.

L'esempio seguente imposta due variabili, *pt1* e *pt2*, uguali a due valori punto, 1,1 e 1,5, ed usa la funzione **command** per eseguire il comando LINEA e passa i valori dei due punti.

```
(setq pt1 '(1 1) pt2 '(1 5))
(command "line" pt1 pt2 "")
```

Se l'applicazione deve essere usata con versioni AutoCAD in lingue differenti, i nomi dei comandi devono essere preceduti da un trattino di sottolineatura (*_*) in modo da essere tradotti. Se si sta utilizzando il prefisso punto, per evitare di usare comandi ridefiniti, è possibile posizionare il punto ed il carattere di sottolineatura in qualsiasi ordine; sia *."_riga"* che *"_riga"* sono sintassi valide.

I comandi eseguiti dalla funzione **command** non vengono ripetuti sulla riga di comando se la variabile di sistema CMDECHO, accessibile dalle funzioni **setvar** e **getvar**, è impostata su 0.

Le funzioni **getxxx** che richiedono l'input dell'utente (**getangle**, **getstring**, **getint**, **getpoint** e così via) non possono essere utilizzate nella funzione **command**. Se si tenta di utilizzarle, verrà visualizzato il messaggio riportato di seguito e la funzione in esecuzione verrà interrotta.

errore: funzione rifiutata da AutoCAD

Se è necessario l'input dell'utente, inviare prima le funzioni **getxxx** oppure posizionarle tra chiamate successive della funzione **command**.

Per i comandi di AutoCAD che richiedono la selezione di un oggetto, come i comandi SPEZZA e TAGLIA, è possibile fornire un elenco ottenuto mediante la funzione **entsel**, anziché specificare un punto per selezionare l'oggetto. Per esempi, vedere "Trasmissione di punti di selezione a comandi di AutoCAD."

I comandi TESTODIN e SCHIZZO di AutoCAD leggono direttamente l'input dalla tastiera e dal digitalizzatore; pertanto, non è possibile utilizzarli con la funzione **command** di AutoLISP. Se si utilizza il comando 'SCRIPT con la funzione **command**, deve essere l'ultima chiamata di funzione nella routine AutoLISP.

ANNULLA Gruppo viene esplicitamente creato attorno a ciascun comando usato con la funzione **command**. Se un utente digita A (o ANNULLA) dopo aver eseguito una routine AutoLISP, verrà annullato solo l'ultimo comando eseguito. Con le successive esecuzioni di ANNULLA vengono annullati a ritroso i vari comandi usati nella routine. Se si desidera che un gruppo di comandi, o l'intera routine, siano considerati come gruppo, usare le opzioni ANNULLA Inizio e ANNULLA Fine.

Uso del simbolo PAUSA

Se un comando di AutoCAD è attivo e si rileva la presenza del simbolo predefinito PAUSA come argomento della funzione **command**, la funzione viene sospesa per consentire all'utente di immettere direttamente i dati.

Il simbolo PAUSA viene definito come una stringa composta da un'unica barra rovesciata. La barra rovesciata è l'unico carattere che è possibile utilizzare al posto del simbolo PAUSA. Tuttavia, se la funzione **command** viene richiamata da una voce di menu, la barra rovesciata sospende la lettura di tale voce con la conseguente valutazione parziale dell'espressione AutoLISP. Il meccanismo di pausa potrebbe inoltre richiedere un valore di attivazione diverso nelle versioni future di AutoLISP; si consiglia, quindi, di utilizzare sempre il simbolo PAUSA anziché una barra rovesciata esplicita.

Quando in una stringa viene utilizzata la barra rovesciata (**), questa deve essere preceduta da un'altra barra rovesciata (**).

Se si rileva la presenza del simbolo PAUSA quando è prevista l'immissione di una stringa di testo o un valore di attributo, AutoCAD sospende l'esecuzione per consentire all'utente di immettere dati soltanto se la variabile di sistema TEXTEVAL ha un valore diverso da zero. Altrimenti, viene utilizzato come testo il valore del simbolo PAUSA (un'unica barra inversa) e non viene eseguita alcuna sospensione.

La funzione **command** viene considerata attiva quando si esegue la sospensione per consentire all'utente di immettere dei dati. Pertanto, l'utente può specificare un'altra espressione AutoLISP da valutare.

L'esempio di seguito riportato mostra come utilizzare il simbolo PAUSA (devono essere presenti il layer NEW_LAY ed il blocco MIO_BLOCCO nel disegno prima di effettuare la prova con questo codice):

```
(setq blk "MIO_BLOCCO")
(setq old_layer (getvar "layer"))
(command "layer" "set" "NEW_LAY" "")
(command "insert" blk pause "" "" pause)
(command "layer" "set" old_layer "")
```

La parte di codice sopra descritta imposta il layer corrente su NEW_LAY, effettua una prima sospensione per consentire all'utente di selezionare un punto di inserimento per il blocco MIO_BLOCCO, che presenta i fattori di scala X e Y uguali a 1, ed una seconda sospensione per la selezione di un angolo di rotazione. Infine, il layer corrente viene reimpostato sul layer originale.

Se la funzione **command** specifica un simbolo PAUSA per il comando SELEZ ed è attivo PICKFIRST, il comando SELEZ ottiene il gruppo PICKFIRST senza effettuare alcuna sospensione.

Attenzione I comandi secondari Raggio e Diametro del messaggio di richiesta Dim in alcune situazioni inviano altri messaggi di richiesta. Ciò può causare un malfunzionamento dei programmi AutoLISP, precedenti alla Release 11, che utilizzano questi comandi.

Vedere anche

"Sottomissione di comandi" per ulteriori informazioni sulla funzione **command**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

cond

Funge da funzione condizionale primaria per AutoLISP.

```
(cond (verifica1 risultato1 ...) ...)
```

La funzione **cond** accetta un numero qualsiasi di elenchi come argomenti e valuta il primo elemento in ogni elenco (nell'ordine fornito) fin quando uno di questi elementi non restituisce un valore diverso da nil. A questo punto la funzione valuta quelle espressioni che seguono la verifica che ha avuto esito positivo e restituisce il valore dell'ultima espressione presente nell'elenco secondario. Se l'elenco secondario contiene soltanto un'espressione, ossia se manca l'argomento *risultato*, viene restituito il valore dell'espressione dell'argomento *verifica*.

Nell'esempio di seguito riportato la funzione **cond** viene utilizzata per calcolare un valore assoluto:

```
(cond
  ((minusp a) (- a))
  (t a)
)
```

Se la variabile *a* è impostata sul valore -10, viene restituito 10.

Come mostrato nell'esempio, la funzione **cond** può essere utilizzata come funzione *sensibile al maiuscolo/minuscolo*. Solitamente il valore T viene usato come ultima espressione di *verifica* di default. Di seguito è riportato un altro semplice esempio: data una stringa di risposta dell'utente nella variabile *s*, questa funzione verifica la risposta e restituisce 1 se tale risposta è Y o y, 0 se la risposta è N o n; negli altri casi, restituisce nil.

```
(cond
  ((= s "Y") 1)
  ((= s "y") 1)
  ((= s "N") 0)
  ((= s "n") 0)
  (t nil)
)
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

cons

Rappresenta il costruttore di base di liste.

```
(cons nuovo-primo-elemento lista)
```

La funzione **cons** valuta un elemento (*nuovo-primo-elemento*) ed una *lista* e restituisce una lista in cui l'elemento aggiunto viene posizionato all'inizio. Il primo elemento può essere un atomo o una lista.

```
(cons '(a) '(b c d))      restituisce (A) B C D)
(cons '(a) '(b c d))      restituisce ((A) B C D)
```

La funzione **cons** accetta anche un atomo al posto dell'argomento *lista*, creando in tal caso una struttura conosciuta come coppia puntata. Quando si visualizza una coppia di questo tipo, AutoLISP inserisce un punto tra il primo ed il secondo elemento. È possibile utilizzare la funzione **cdr** per restituire il secondo atomo di una coppia puntata.

```
(cons 'a 2)          restituisce (A . 2)
(car (cons 'a 2))   restituisce A
(cdr (cons 'a 2))   restituisce 2
```

Una coppia puntata è un tipo particolare di lista che non viene accettato come argomento da alcune funzioni che gestiscono liste di tipo normale.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

cos

Restituisce il coseno di un angolo espresso in radianti.

```
(cos angolo)

(cos 0.0)          restituisce 1.0
(cos pi)           restituisce -1.0
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

cvunit

Converte un valore da un'unità di misura in un'altra.

(cvunit *valore da a*)

L'argomento *valore* corrisponde al valore numerico da convertire. Può anche essere un elenco contenente due o tre numeri da convertire (un punto bidimensionale o tridimensionale). L'argomento *da* rappresenta il valore da cui si esegue la conversione, mentre l'argomento *a* rappresenta l'unità in cui il valore verrà convertito. Gli argomenti *da* ed *a* possono definire qualsiasi tipo di unità presente nel file *acad.unt*.

Se l'esito è positivo, la funzione **cvunit** restituisce il valore convertito. Se invece il nome dell'unità risulta sconosciuto (vale a dire non è contenuto nel file *acad.unt*), oppure se due unità non sono compatibili a livello di dimensione (come nel caso della conversione da grammi in anni), la funzione **cvunit** restituisce nil.

```
(cvunit 1 "minute" "second") restituisce 60.0
(cvunit 1 "gallon" "furlong") restituisce nil
(cvunit 1.0 "inch" "cm") restituisce 2.54
(cvunit 1.0 "acre" "sq yard") restituisce 4840.0
(cvunit '(1.0 2.5) "ft" "in") restituisce (12.0 30.0)
(cvunit '(1 2 3) "ft" "in") restituisce (12.0 24.0 36.0)
```

Nota Se si desidera eseguire lo stesso tipo di conversione per diversi valori, risulta più conveniente convertire il valore 1.0 una volta e poi applicare il valore risultante come fattore di scala nella funzione o calcolo in uso. Tale operazione può essere effettuata per tutte le unità predefinite ad eccezione della temperatura, poiché quest'ultima comporta degli sfalsamenti.

Vedere anche

"Conversione di unità."

Capitolo 13 -- AutoLISP: catalogo delle funzioni

defun

Definisce una funzione.

```
(defun simb elenco-arg espr . . .)
```

La funzione **defun** definisce una funzione con il nome *simb* ed il nome della funzione viene automaticamente racchiuso tra virgolette. Il nome della funzione è seguito da un elenco di argomenti, possibilmente vuoto, che a sua volta può essere opzionalmente seguito da una barra e dai nomi di uno o più simboli locali per la funzione. La barra deve essere separata dal primo simbolo locale e dall'ultimo argomento, se presenti, da almeno uno spazio. Se non si specificano argomenti o simboli locali, è necessario indicare una coppia di parentesi vuote dopo il nome della funzione.

Gli esempi di *elenco-arg* di seguito riportati mostrano i valori validi e quelli non validi:

```
(defun myfunc (x y) . . .)      La funzione valuta due argomenti
(defun myfunc (/ a b) . . .)   La funzione presenta due simboli locali
(defun myfunc (x / temp) . . .) Un argomento ed un simbolo locale
(defun myfunc () . . .)       Nessun argomento o simbolo locale
```

Non è possibile definire una funzione con più argomenti con lo stesso nome. È possibile, invece, disporre di un argomento che definisce una variabile locale con lo stesso nome di un'altra variabile locale o di uno degli argomenti.

```
(defun fubar (a a / b) . . .)   Non valida
(defun fubar (a b / a a b) . . .) Valida ma inutile
```

Se l'elenco di argomenti e/o simboli contiene voci duplicate, viene utilizzata la prima occorrenza di ogni nome, mentre le successive occorrenze vengono ignorate.

Quando si esegue la funzione, vengono valutate una o più espressioni che seguono l'elenco di argomenti e di simboli locali.

La funzione **defun** restituisce il nome della funzione definita. Quando quest'ultima viene richiamata, i relativi argomenti vengono valutati ed associati ai simboli corrispondenti. È possibile utilizzare i simboli locali all'interno della funzione senza modificare i loro legami ai livelli esterni. La funzione restituisce il risultato dell'ultima espressione valutata. Tutte le espressioni precedenti hanno solo effetti secondari.

Negli esempi di seguito riportati vengono definite nuove funzioni mediante la funzione **defun** e vengono mostrati i valori restituiti da tali funzioni:

```
(defun add10 (x)
  (+ 10 x)
)
(add10 5)
(add10 -7.4)
```

restituisce ADD10
restituisce 15
restituisce 2.6

and

```
(defun dots (x y / temp)
  (setq temp (strcat x "..."))
  (strcat temp y)
)
(dots "a" "b")
(dots "from" "to")
```

restituisce DOTS
restituisce "a...b"
restituisce "from...to"

Attenzione Non utilizzare mai il nome di una funzione incorporata o di un simbolo come *simb*, perché ciò non consentirebbe l'accesso alla funzione. Per ottenere un elenco delle funzioni incorporate e di quelle definite in precedenza, vedere "**atoms-family**".

Vedere anche

"Gestione di simboli e funzioni."

Capitolo 13 -- AutoLISP: catalogo delle funzioni

Aggiunge un oggetto non grafico ad un dizionario specificato.

```
(dictadd nome_ent simbolo nuovo_oggetto)
```

Aggiunge l'oggetto *nuovo_oggetto* al dizionario *nome_ent*. L'argomento *simbolo* è il nome chiave dell'oggetto da aggiungere al dizionario; questo nome deve essere univoco ed ancora inesistente nel dizionario. L'oggetto specificato da *nuovo_oggetto* corrisponde soltanto ad un oggetto non grafico.

Come regola generale, ogni oggetto aggiunto ad un dizionario deve essere presente solo in quel dizionario. Questa regola può non essere rispettata soprattutto quando si aggiungono oggetti raggruppati ad un dizionario per gruppi. L'aggiunta dello stesso oggetto raggruppatto utilizzando nomi chiave differenti genera nomi di gruppi duplicati ed il rischio che la funzione **dictnext** esegua una iterazione infinita.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

dictnext

Trova l'elemento successivo in un dizionario.

(dictnext nome_ent simbolo [primo])

L'argomento *nome_ent* è il nome di un'entità che specifica il dizionario su cui operare.

Quando **dictnext** è usata ripetutamente, restituisce ogni volta la voce successiva nel dizionario specificato. La funzione **dictsearch** specifica un'entità da indicare. Se l'argomento *primo* è presente e valuta un valore non -nil, viene indicato il primo inserimento del dizionario. Quando non vi sono più inserimenti nel dizionario, viene restituito nil. Gli inserimenti cancellati non vengono mai restituiti.

Vedere la descrizione di "**namedobjdict**" per il nome dell'entità del dizionario principale.

Nota Una volta iniziato il lavoro all'interno del dizionario, passando ad un nome diverso di dizionario con **dictnext** causa la perdita del posto nel dizionario originario. In altre parole, in questa funzione viene mantenuto soltanto un iter globale.

Se viene trovata una voce, questa viene restituita come elenco di coppie puntate di codici e valori di tipo DXF.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

dictremove

Cancella una voce dal dizionario specificato.

(dictremove nome_ent simbolo)

Elimina la voce di dizionario specificata dal *simbolo* dal dizionario specificato da *nome_ent*.

Se l'argomento *nome_ent* non è valido o l'argomento *simbolo* non viene trovato, **dictremove** restituisce nil. Se **dictremove** ha esito positivo, restituisce il nome entità della voce cancellata.

Per default, se si cancella una voce da un dizionario, la stessa non viene cancellata dal database. Questa operazione può essere eseguita con una chiamata a **entdel**. Ciò non vale per i gruppi e gli stili multilinea. Il codice che implementa queste funzioni richiede che il database e questi dizionari siano aggiornati e quindi elimina automaticamente l'entità nel momento in cui questa viene rimossa dal dizionario con **dictremove**.

La funzione **dictremove** non consente l'eliminazione di uno stile multilinea dal dizionario relativo se a quello stile viene fatto riferimento nel database mediante una multilinea.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

dictrename

Rinomina la voce di un dizionario.

(dictrename *nome_ent vecchio_simbolo nuovo_simbolo*)

Cambia il nome chiave di una voce di un dizionario da *vecchio_simbolo* a *nuovo_simbolo*. Il dizionario è specificato dall'argomento *nome_ent*.

Se *vecchio_nome* non è presente nel dizionario, *nome_ent* non è valido, *nuovo_nome* non è valido oppure *nuovo_nome* è già presente nel dizionario. La funzione **dictrename** restituisce nil.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

dictsearch

Ricerca un elemento in un dizionario.

(dictsearch *nome_ent simbolo [successivo]*)

L'argomento *nome_ent* è il nome di un'entità che specifica il dizionario su cui operare. L'argomento *simbolo* è una stringa che specifica l'elemento all'interno del dizionario.

Se la funzione **dictsearch** trova un inserimento per l'elemento dato, restituisce quell'inserimento nel formato descritto per la funzione **dictnext**. Se non viene trovata alcuna voce di quel tipo, la funzione restituisce nil.

Generalmente la funzione **dictsearch** non ha effetti sull'ordinamento degli inserimenti indicati da **dictnext**. Tuttavia, se **dictsearch** ha esito positivo e l'argomento *setnext* è presente e non è -nil, il contatore dell'inserimento di **dictnext** viene adeguato in modo che la chiamata successiva a **dictnext** **restituisca l'inserimento dopo quello restituito dalla chiamata a dictsearch**.

L'esempio che segue utilizza **dictsearch** per indicare l'elenco di definizioni del gruppo G2 (questo codice presuppone l'esistenza di un gruppo denominato G2 nel disegno corrente).

```
(setq grp (dictsearch (namedobjdict) "ACAD_GROUP"))
(setq g2 (dictsearch (cdr (assoc -1 grp)) "G2"))
```

Vedere "**namedobjdict**" per il nome dell'entità del dizionario principale.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

dimx_tile e dimy_tile

Richiama le dimensioni di una casella in base alle unità di misura della finestra di dialogo.

(dimx_tile *chiave*) e **(dimy_tile *chiave*)**

La funzione **dimx_tile** restituisce la larghezza della casella, mentre la funzione **dimy_tile** restituisce l'altezza. Per entrambe le funzioni, l'argomento *chiave* è una stringa che specifica la casella ed è sensibile al maiuscolo/minuscolo.

Le coordinate restituite indicano i valori massimi consentiti all'interno della casella; poiché le coordinate sono basate su zero, queste funzioni restituiscono il valore totale di X o Y (X-1 e Y-1). Le funzioni **dimx_tile** e **dimy_tile** vengono utilizzate con le funzioni **vector_image**, **fill_image** e **slide_image**, per le quali è necessario specificare le coordinate assolute della casella.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

distance

Restituisce la distanza tridimensionale tra due punti.

(distance *pt1 pt2*)

```
(distance '(1.0 2.5 3.0) '(7.7 2.5 3.0)) restituisce 6.7
(distance '(1.0 2.0 0.5) '(3.0 4.0 0.5)) restituisce 2.82843
```

Se uno o entrambi i punti forniti sono bidimensionali, la funzione **distance** ignora la coordinata Z dei punti tridimensionali specificati e restituisce la distanza bidimensionale tra i punti come se fossero proiettati sul piano corrente di costruzione.

Vedere anche

"Utilità geometriche"

Capitolo 13 -- AutoLISP: catalogo delle funzioni

distof

Converte una stringa che contiene un valore reale (con virgola mobile) in un valore reale.

(**distof** *stringa* [*modalità*])

L'argomento *modalità* indica le unità in cui viene formattata la stringa. Il valore deve corrispondere ai valori consentiti per la variabile di sistema LUNITS di AutoCAD, come indicato nella tabella di seguito riportata. Se si omette *modalità*, **angtof** utilizza il valore corrente di LUNITS.

Valori delle unità lineari

Valore per modalità	Formato stringa
1	Scientifica
2	Decimale
3	Ingegneristica (piedi e pollici decimali)
4	Architettonica (piedi e pollici frazionari)
5	Frazionario

L'argomento *stringa* deve essere una stringa che la funzione **distof** può analizzare correttamente in base alla modalità *specificata*. Può avere lo stesso formato che viene restituito dalla funzione **rtos** oppure il formato consentito da AutoCAD per l'immissione da tastiera. Le funzioni **distof** e **rtos** sono complementari: se alla funzione **distof** viene passata una stringa creata da **rtos**, **distof** restituirà un valore valido e viceversa (presumendo che i valori della modalità corrispondano).

Nota la funzione **distof** considera la modalità 3 e 4 nello stesso modo. Ciò significa che, se *modalità* è uguale a 3 (unità ingegneristica) oppure a 4 (unità architettonica) e *stringa* è *uno qualsiasi* di questi formati, la funzione **distof** restituisce il valore reale valido.

Se la funzione **distof** ha esito positivo, restituisce un numero reale, altrimenti restituisce nil.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

done_dialog

Chiude una finestra di dialogo.

(**done_dialog** [*stato*])

È necessario richiamare la funzione **done_dialog** all'interno di un'espressione di azione o di una funzione di richiamo (vedere "action_tile").

Se si specifica l'argomento opzionale *stato*, questo deve essere un numero intero positivo, che verrà restituito da **start_dialog** al posto di 1 per OK oppure 0 per Annulla. Il significato di qualsiasi valore di *stato* maggiore di 1 dipende dall'applicazione utilizzata.

La funzione **done_dialog** restituisce un elenco di punti bidimensionali che rappresentano la posizione (X,Y) della finestra di dialogo nel momento in cui l'utente è uscito da essa. È possibile passare questo punto ad una successiva chiamata alla funzione **new_dialog** per fare in modo che la finestra di dialogo venga aperta nuovamente nella posizione selezionata dall'utente.

Nota Se si usa una funzione di richiamo per il pulsante, la cui parola chiave è "accept" o "cancel" (di solito i pulsanti OK e Annulla),

tale funzione deve richiamare in modo esplicito la funzione **done_dialog**. In caso contrario, l'utente potrebbe rimanere nella finestra di dialogo senza poter eseguire alcuna operazione. Se per i suddetti pulsanti non si fornisce una funzione di richiamo esplicita e non si utilizzano i pulsanti di uscita standard, essi verranno gestiti in modo automatico da AutoCAD. È necessario inoltre che lo stato di un'azione esplicita AutoLISP per il pulsante "accept" sia uguale a 1 oppure un valore definito dall'applicazione, altrimenti la funzione **start_dialog** restituisce il valore di default 0 che comporta l'annullamento della finestra di dialogo.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

end_image

Termina la creazione dell'immagine della finestra di dialogo correntemente attiva.

(end_image)

È complementare alla funzione **start_image**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

end_list

Termina la valutazione dell'elenco della finestra di dialogo correntemente attiva.

(end_list)

È complementare alla funzione **start_list**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

entdel

Elimina gli oggetti (entità) oppure ripristina gli oggetti precedentemente eliminati.

(entdel *nome_ent*)

L'entità specificata dall'argomento *nome_ent* viene eliminata se è attualmente contenuta nel disegno. La funzione **entdel** ripristina l'entità nel disegno se è stata eliminata precedentemente in questa sessione di modifica. Quando si esce dal disegno, le entità eliminate vengono cancellate dal disegno. La funzione **entdel** può eliminare sia le entità grafiche che quelle non grafiche.

```
(setq e1 (entnext)) ;Assegna ad e1 il nome della prima
                    ;entità del disegno
(entdel e1)         ;Elimina l'entità e1
(entdel e1)         ;Ripristina l'entità e1 eliminata
```

La funzione **entdel** funziona solo sulle entità principali. Gli attributi ed i vertici di polilinee non possono essere cancellati, indipendentemente dalle entità di livello superiore corrispondenti. È possibile utilizzare la funzione **command** per modificare le sottoentità mediante il comando EDITATT o EDITPL.

Non è possibile eliminare entità all'interno della definizione di un blocco. Tuttavia, è possibile utilizzare la funzione **entmake** per ridefinire completamente il blocco, ad eccezione dell'entità che si desidera eliminare.

Capitolo 13 -- AutoLISP: catalogo delle funzioni



Richiama i dati di definizione di un oggetto (entità).

(entget nome_ent [elenco_app])

La funzione **entget** restituisce un elenco in cui sono contenuti i dati di definizione dell'entità *nome_ent*. Ciò è valido sia per le entità grafiche che per quelle non grafiche. Se si fornisce l'argomento *elenco_app*, vale a dire un elenco opzionale dei nomi delle applicazioni registrate, la funzione **entget** restituisce inoltre i dati estesi associati alle applicazioni specificate.

I dati restituiti dalla funzione **entget** vengono codificati come elenco di associazioni da cui è possibile estrarre elementi utilizzando la funzione **assoc**. Agli oggetti riportati nell'elenco sono assegnati codici di gruppo DXF di AutoCAD per ogni parte dei dati relativi all'entità.

Si supponga che l'ultimo oggetto creato nel disegno sia una linea passante dal punto (1,2) al punto (6,5). È possibile richiamare il nome dell'ultimo oggetto con la funzione **entlast** e passare tale nome alla funzione **entget**.

```
(entget (entlast))
```

Si potrebbe ottenere il seguente risultato:

```
((-1 . <Nome di entità: 60000014>)  Nome di entità
(0 . "LINE")                      Tipo di oggetto
(8 . "0")                          Layer
(10 1.0 2.0 0.0)                   Punto iniziale
(11 6.0 6.0 0.0)                   Punto finale
)
```

I codici di gruppo DXF utilizzati da AutoLISP differiscono leggermente dai codici di gruppo presenti in un file DXF e vengono descritti nell'appendice C, "Codici di gruppo DXF".

Allo stesso modo di DXF, gli elementi di intestazione delle entità (colore, tipo di linea, altezza, il flag "seguono attributi" ed il gestore di entità) vengono esportati soltanto se non hanno valori di default. A differenza di DXF, i campi opzionali di definizione delle entità vengono esportati indipendentemente dal fatto che corrispondano o meno ai valori di default. In tal modo, la valutazione viene semplificata in quanto i programmi possono sempre assumere che questi campi siano presenti per calcolare gli algoritmi generali che agiscono su di essi. Inoltre diversamente da DXF, le coordinate associate X, Y e Z vengono raggruppate in un singolo elenco di punti, ad esempio in (10 1.0 2.0 3.0), piuttosto che essere visualizzate come gruppi separati, ad esempio 10, 20 e 30.

L'elemento -1 all'inizio dell'elenco contiene il nome di questa entità. È possibile estrarre mediante la funzione **assoc** le singole coppie puntate, che rappresentano i valori, utilizzando la funzione **cdr** per trovare tali valori.

Gli elenchi secondari relativi ai punti non sono coppie puntate. Per convenzione, la funzione **cdr** di tale elenco è il valore del gruppo. Poiché un punto è un elenco composto da due (o tre) numeri reali, l'intero gruppo sarà un elenco costituito da tre (o quattro) entità. In tal modo, viene mantenuta la convenzione in base alla quale la funzione **cdr** restituisce sempre il valore.

Quando si scrivono funzioni per valutare questi elenchi di entità, accertarsi che non rispettino l'ordine degli elenchi secondari; a tale scopo, usare la funzione **assoc**. Il gruppo -1, in cui è contenuto il nome dell'entità, consente di effettuare operazioni di modifica in modo che l'elenco delle entità venga accettato ed in modo da evitare la necessità di conservare il nome dell'entità in una struttura parallela. Un'entità sequend posta alla fine di una polilinea o un gruppo di attributi contiene un gruppo -2, di cui la funzione **cdr** rappresenta il nome entità dell'intestazione. In tal modo, è possibile individuare l'intestazione presente in una sottoentità, avanzando fino all'entità Seqend ed utilizzando successivamente la funzione **cdr** del gruppo -2 come nome di entità per richiamare l'entità principale associata.

Attenzione Prima di eseguire una funzione **entget** sulle entità che dispongono di un vertice, è necessario leggere o scrivere l'intestazione dell'entità polilinea. Se l'entità polilinea valutata più recentemente differisce dall'entità a cui appartiene il vertice, si potrebbero perdere le informazioni relative alla larghezza (i gruppi 40 e 41).

Tutti i punti associati ad un oggetto vengono espressi in base al Sistema di Coordinate Oggetto (OCS). Per punti, linee, linee tridimensionali, facce tridimensionali, polilinee tridimensionali, mesh tridimensionali ed oggetti di quota, l'OCS equivale al Sistema di Coordinate Globali (WCS), in cui i punti dell'oggetto sono espressi in coordinate globali. Per tutti gli altri oggetti, l'OCS può derivare dal WCS e dalla direzione d'istruzione dell'oggetto (il gruppo 210). Quando si utilizzano oggetti disegnati utilizzando sistemi di coordinate diversi dal WCS, potrebbe essere necessario convertire i punti in WCS o nel sistema UCS corrente utilizzando la funzione **trans**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

entlast

Restituisce il nome dell'ultimo oggetto (entità) principale non eliminato nel disegno.

(entlast)

La funzione **entlast** viene spesso usata per ottenere il nome di una nuova entità che è stata appena aggiunta utilizzando la funzione **command**. Per selezionare un'entità non è necessario che sia visualizzata o che si trovi in un layer scongelato.

```
(setq e1 (entlast))           Assegna ad e1 il nome dell'ultima
                              entità principale del disegno
(setq e2 (entnext e1))       Imposta e2 su nil o sul nome di una
                              sottoentità attributo o vertice
```

Se l'applicazione richiede il nome dell'ultima entità non eliminata (l'entità principale o la sottoentità), definire una funzione uguale a quella di seguito riportata e richiamarla al posto della funzione **entlast**.

```
(defun lastent (/ a b)
  (if (setq a (entlast))      Ottiene l'ultima entità principale
      (while (setq b (entnext a)) Se seguono delle sottoentità,
                                  esegue delle iterazioni finché
                                  non vi sono più sottoentità
            (setq a b)
          )
      )
  a
)
```

Restituisce l'ultima entità principale o la sottoentità

Capitolo 13 -- AutoLISP: catalogo delle funzioni

entmake

Crea una nuova entità (oggetto grafico) nel disegno.

(entmake [elenco_ent])

L'argomento *elenco_ent* deve essere un elenco dei dati che definiscono l'entità in un formato simile a quello restituito dalla funzione **entget**. Deve contenere inoltre tutte le informazioni necessarie per definire l'entità. La funzione **entmake** può definire sia entità grafiche che non grafiche. Se si omette uno qualsiasi dei dati di definizione richiesti, la funzione **entmake** restituisce nil e l'oggetto viene rifiutato. Se si omettono i dati di definizione opzionali, ad esempio il layer, la funzione **entmake** utilizza il valore di default.

Se la funzione **entmake** riesce a creare una nuova entità, restituisce l'elenco dei dati definizione dell'entità stessa. In caso contrario, restituisce nil.

Un metodo per la creazione di una nuova entità consiste nell'ottenere i dati di definizione di un'entità tramite la funzione **entget**, nel modificarli e nel passarli successivamente alla funzione **entmake**. Prima di creare una nuova entità, la funzione **entmake** verifica che il nome del layer, il nome del tipo di linea ed il colore forniti siano validi. Se si specifica un nuovo nome di layer, la funzione **entmake** lo crea automaticamente. La funzione **entmake** verifica inoltre i nomi dei blocchi, i nomi dello stile di quota, il nome dello stile di testo ed i nomi delle forme qualora siano richiesti dal tipo di entità utilizzato.

Il tipo di entità, ad esempio CIRCLE o LINE, deve essere il primo o il secondo campo di *elenco_ent*. Se è il secondo campo, viene preceduto soltanto dal nome dell'entità. Questo rappresenta il formato restituito dalla funzione **entget**. In tali casi, il nome dell'entità viene ignorato quando si crea una nuova entità. Se nell' *elenco_ent* è contenuto un gestore di entità, anche quest'ultimo viene ignorato.

Il codice di seguito riportato crea un cerchio rosso con il centro in corrispondenza del punto (4,4) e con il raggio pari a 1. I campi opzionali relativi al layer ed al tipo di linea sono stati omissi e pertanto vengono utilizzati i valori di default.

```
(entmake
  '( (0 . "CIRCLE")           Tipo di entità
      (62 . 1)                Colore
      (10 4.0 4.0 0.0)        Centro
    )
```

```

    )
  (40 . 1.0)      Raggio
)
)

```

Nota Gli oggetti creati su un layer congelato non vengono rigenerati fin quando tale layer non viene scongelato.

Entità complesse

È possibile creare un'entità complessa (definizione di un blocco, polilinea o riferimento di un blocco contenente attributi) eseguendo diverse chiamate alla funzione **entmake** per definirne le sottoentità (attributi o vertici). Quando si utilizza la funzione **entmake** per creare un'entità complessa, viene creato un file temporaneo in cui vengono raccolti tutti i dati di definizione. Per ogni funzione **entmake**, viene verificata l'esistenza del file temporaneo; in caso positivo, i nuovi dati vengono aggiunti al file. Una volta terminata la definizione dell'entità complessa, aggiungendo l'entità `seqend` o `endblk` appropriata, dati forniti vengono di nuovo verificati e l'entità complessa viene aggiunta al disegno. Una volta terminata la definizione di un blocco (la funzione **entmake** di un elemento `endblk`) viene restituito il nome del blocco e non l'elenco dei dati dell'entità.

Nota Non utilizzare la funzione **entmake** per creare oggetti della finestra.

Se si ricevono dati non validi per il tipo di entità, l'entità viene rifiutata così come l'intera entità complessa. Una definizione di blocco non può essere nidificata né essere riferita a se stessa. Tuttavia può contenere riferimenti a definizioni di altri blocchi. Le entità complesse possono esistere nello spazio modello o nello spazio carta, ma non in entrambi.

Il codice di gruppo 66 viene accettato soltanto per gli oggetti da inserire (che significa *seguono attributi*); al contrario, tale codice viene impostato su 1 (che significa *seguono vertici*) per le entità polilinea, mentre per tutte le altre entità viene impostato al valore di default 0. L'unica entità che può seguire un'entità polilinea è un'entità vertice.

Nessuna parte di un'entità complessa viene visualizzata sul disegno finché la definizione non è stata completata. È possibile annullare la creazione di un'entità complessa, digitando **entmake** senza specificare argomenti. In questo modo, il file temporaneo viene cancellato e viene restituito `nil`.

Le entità blocco e `endblk` possono essere utilizzate per creare la definizione di un nuovo blocco. I blocchi appena creati vengono automaticamente immessi nella tabella dei simboli nella quale è possibile fare riferimento ad essi.

Alcune applicazioni possono rappresentare poligoni con un numero qualsiasi di lati nelle mesh poliedriche. Tuttavia, la struttura di entità di AutoCAD stabilisce un limite di vertici da specificare per una determinata entità faccia. È possibile rappresentare più poligoni complessi scomponendoli in cunei triangolari. Quest'ultimi vengono rappresentati da AutoCAD come facce a quattro vertici, di cui due vertici adiacenti presentano lo stesso valore. È necessario rendere invisibili gli spigoli per impedire che vengano disegnate le parti visibili di questa suddivisione. Il comando `POLIMESH` esegue questa suddivisione automaticamente, ma quando le applicazioni generano direttamente mesh poliedriche, è necessario che le applicazioni eseguano tale suddivisione.

Il numero di vertici per faccia è il parametro chiave di questa suddivisione. La variabile di sistema `PFACEVMAX` fornisce ad un'applicazione un determinato numero di vertici per l'elemento faccia. Questo valore è a sola lettura ed è impostato a 4.

Attenzione Quando la funzione **entmake** crea un blocco, può sovrascrivere un blocco esistente. Tale funzione non verifica se vi sono conflitti a livello di nomi nella tabella di definizione dei blocchi; pertanto, prima di utilizzare la funzione **entmake**, usare la funzione **tblsearch** (descritta) per accertarsi che il nome del nuovo blocco sia univoco. Tuttavia, può essere utile utilizzare la funzione **entmake** per ridefinire blocchi anonimi, come vengono descritti nella sezione seguente.

Blocchi anonimi

La tabella delle definizioni di blocchi in un disegno può contenere blocchi senza nome. Tali blocchi vengono creati per supportare i modelli di tratteggio e la quotatura associativa. È possibile inoltre crearli dalla funzione **entmake** per i fini specifici dell'applicazione, generalmente per fare in modo che contengano entità a cui l'utente non può accedere direttamente.

Il codice di gruppo 2 (nome del blocco) di un'entità di dimensione è opzionale per la funzione **entmake**. Se il nome del blocco viene ommesso dall'elenco di definizione delle entità, AutoCAD ne crea uno nuovo. Altrimenti, AutoCAD crea la dimensione con il nome indicato.

Il nome (gruppo 2) di un blocco anonimo è `*Unnn`, dove `nnn` rappresenta un numero generato da AutoCAD. Inoltre, il bit meno significativo di un *flag di tipo blocco* (gruppo 70) riguardante un blocco anonimo è impostato su 1. Quando **entmake** crea un blocco con il nome che inizia con `*` e con il bit anonimo impostato, AutoCAD considera tale blocco come anonimo e gli assegna un nome. Qualsiasi carattere che segue `*` nella stringa del nome passata a **entmake** viene ignorato. Una volta creato il blocco, la funzione **entmake** ne restituisce il nome. Se si crea il blocco eseguendo più chiamate alla funzione **entmake**, viene restituito il nome dopo che la chiamata di seguito riportata ha avuto esito positivo:

```
(entmake "endblk")
```

Ogni volta che viene visualizzato un disegno, tutti i blocchi anonimi a cui non si fa riferimento vengono cancellati dalla tabella delle definizioni dei blocchi. Non vengono eliminati, invece, i blocchi a cui si fa riferimento (quelli inseriti). È possibile utilizzare la funzione **entmake** per creare il riferimento di un blocco (`Inser`) per un blocco anonimo; non è possibile tuttavia passare un blocco anonimo al comando `INSER`. La funzione **entmake** può anche essere usata per ridefinire il blocco. Le entità di un blocco, ma non l'entità blocco, possono essere modificati mediante la funzione **entmod**.

Nota Sebbene un blocco anonimo con riferimenti diventi permanente, la parte numerica del suo nome può variare da una sessione

di disegno all'altra. Le applicazioni non possono basarsi sui nomi dei blocchi anonimi che rimangono costanti.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

entmakex

Crea un nuovo oggetto grafico, fornisce un gestore e un nome entità (ma non assegna un proprietario), quindi restituisce il nuovo nome entità.

(entmakex [elenco_ent])

L'argomento *elenco_ent* deve essere un elenco dei dati che definiscono l'entità in un formato simile a quello restituito dalla funzione **entget**. Deve contenere inoltre tutte le informazioni necessarie per definire l'entità o l'oggetto. La funzione **entmakex** può definire sia oggetti grafici che non grafici. Se si omette uno qualsiasi dei dati di definizione richiesti, la funzione **entmakex** restituisce nil e l'oggetto viene rifiutato. Se si omettono i dati di definizione opzionali, ad esempio il layer, la funzione **entmakex** utilizza i valori di default.

Se la funzione **entmakex** riesce a creare una nuova entità, ne restituisce il nome; In caso contrario, restituisce nil.

Attenzione Gli oggetti e le entità senza proprietario *non* vengono scritte nei file *.dwg* o *.dxf*. Dopo aver utilizzato **entmakex**, accertarsi di aver specificato un proprietario. Ad esempio, è possibile utilizzare **dictadd** per impostare un dizionario come proprietario di un oggetto.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

entmod

Modifica i dati di definizione di un oggetto (entità).

(entmod [elenco_ent])

Alla funzione **entmod** viene passato un elenco (*elenco_ent*) nel formato restituito dalla funzione **entget** ed aggiorna le informazioni del database per il nome dell'entità specificata dal gruppo -1 nell'*elenco_ent*. Il meccanismo principale mediante il quale un'applicazione AutoLISP aggiorna il database è quello di richiamare un'entità con **entget**, di modificare l'elenco che definisce un'entità ed aggiornare quest'ultima nel database utilizzando la funzione **entmod**, che può modificare sia oggetti grafici che non grafici.

```
(setq en (entnext))           Imposta en sul nome della prima entità del disegno
(setq ed (entget en))        Imposta ed sui dati dell'entità en
(setq ed
  (subst (cons 8 "0")
    (assoc 8 ed)              Sostituisce il gruppo di layer in ed con il layer 0
    ed
  )
)
(entmod ed)                  Modifica il layer dell'entità en nel disegno
```

La funzione **entmod** impone alcune restrizioni relative alle modifiche da apportare. In primo luogo, non è possibile modificare il tipo ed il gestore di entità. Se si desidera modificarli, è necessario eliminare l'entità con la funzione **entdel**, quindi crearne una nuova utilizzando le funzioni **command** o **entmake**. Tutti gli oggetti a cui l'elenco di entità fa riferimento devono essere riconosciuti da AutoCAD prima che venga eseguita la funzione **entmod**. Pertanto, è necessario definire in un disegno lo stile di testo, il tipo di linea, la forma ed i nomi dei blocchi prima che possano essere utilizzati in un elenco di entità con la funzione **entmod**. I nomi di layer rappresentano un'eccezione: entmod crea infatti un nuovo layer con i valori standard di default utilizzati dall'opzione Nuovo di LAYER se nell'elenco di entità si fa riferimento da un layer definito in precedenza.

Per i campi di entità che contengono valori a virgola mobile, come l'altezza, la funzione **entmod** accetta i valori interi e li converte in valori a virgola mobile. In modo analogo, se si assegna un valore a virgola mobile ad un campo di entità con valore intero, come ad esempio il numero del colore, tale valore viene troncato e convertito in un valore intero.

La funzione **entmod** esegue il controllo di coerenza sull'elenco fornito. Se viene rilevato un grave errore, il database non viene aggiornato e viene restituito il valore nil. In caso contrario, la funzione **entmod** restituisce l'elenco fornito sotto forma di argomento.

Questa funzione non può modificare i campi interni, ad esempio il nome dell'entità nel gruppo -2 di un'entità seqend. Pertanto, qualsiasi tentativo di modifica dei campi verrà ignorato.

Quando la funzione **entmod** aggiorna un'entità principale, modifica l'entità ed aggiorna la relativa immagine sullo schermo (comprese le sottoentità). Quando invece la funzione aggiorna una sottoentità (il vertice di una polilinea o l'attributo di un blocco), la sottoentità viene aggiornata nel database, ma non viene visualizzata di nuovo. Dopo aver apportato tutte le modifiche alle sottoentità di una determinata entità, è possibile utilizzare la funzione **entupd** per aggiornare l'immagine sullo schermo.

Nota non è possibile utilizzare la funzione **entmod** per modificare un'entità finestra. È possibile invece sostituire il valore del campo visibilità di un'entità con 0 o 1 (tranne per gli oggetti della finestra). Se tale funzione viene utilizzata per modificare un'entità all'interno della definizione di un blocco, la modifica viene apportata a tutti i blocchi di questo tipo presenti nel disegno.

Prima di eseguire una funzione **entmod** sulle entità che dispongono di un vertice, è necessario leggere o scrivere l'intestazione dell'entità polilinea. Se l'elemento polilinea valutato più recentemente differisce dall'elemento a cui appartiene il vertice, si potrebbero perdere le informazioni relative alla larghezza (i gruppi 40 e 41).

Attenzione È possibile utilizzare la funzione **entmod** per modificare le entità all'interno della definizione di un blocco; tale operazione può creare un blocco che si riferisce a se stesso, con la conseguente interruzione di AutoCAD.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

entnext

Restituisce il nome dell'oggetto (entità) successivo nel disegno.

(entnext [nome_ent])

Se **entnext** viene chiamata senza argomenti, restituisce il nome della prima entità presente nel database dei disegni. Se la funzione **entnext** viene richiamata con l'argomento *nome_ent*, restituisce il nome della prima entità non eliminata che segue *nome_ent* nel database. Se nel database non esiste un'entità successiva, viene restituito nil. La funzione **entnext** restituisce sia le entità principali che le sottoentità.

L'entità selezionata mediante la funzione **ssget** sono entità principali e non attributi di blocchi o vertici di polilinee. È possibile accedere alla struttura interna di queste entità complesse esaminando le sottoentità mediante la funzione **entnext**. Una volta ottenuto il nome di una sottoentità, è possibile utilizzarlo come le altre entità. Se si ottiene il nome di una sottoentità con la funzione **entnext**, è possibile individuare l'entità a livello superiore corrispondente esaminando le entità mediante la funzione **entnext** fin quando non si incontra un'entità seqend, estraendo quindi il gruppo -2 da tale entità, che rappresenta il nome dell'entità principale.

```
(setq en (entnext))           Imposta en sul nome della prima entità del disegno
(setq e2 (entnext e1))       Imposta e2 sul nome dell'entità che segue e1
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

entsel

Richiede all'utente di selezionare un singolo oggetto (entità) specificando un punto.

(entse1 [msg_richiesta])

La funzione **entse1** restituisce un elenco in cui il primo elemento è il nome dell'oggetto scelto, mentre il secondo elemento rappresenta le coordinate (in base all'UCS corrente) del punto utilizzato per selezionare l'oggetto. Se si specifica una stringa per l'argomento *msg_richiesta*, tale stringa viene utilizzata per richiedere all'utente di immettere l'oggetto. In caso contrario, verrà visualizzato il messaggio di richiesta di default: Selezionare oggetti.

Nota Il punto di selezione restituito da **entse1** non rappresenta un punto sull'oggetto selezionato, ma la posizione del puntatore a croce al momento della selezione. Il rapporto tra il punto di selezione e l'oggetto varia a seconda delle dimensioni della casella di selezione e la scala di ingrandimento corrente.

È possibile fornire ad AutoCAD un elenco delle forme restituito dalla funzione **entse1** in risposta ad un qualsiasi messaggio di richiesta per la selezione di un oggetto. Tale elenco viene interpretato dal programma come punto di selezione di un oggetto specificato.

La sequenza di comandi di AutoCAD di seguito riportata mostra come usare la funzione **entse1** e l'elenco che viene restituito.

Comando: **linea**
 Dal punto: **1,1**
 Al punto: **6,6**
 Al punto: ENTER
 Comando: **(setq e (entsel "Selezionare oggetti: "))**
 Scegliere un oggetto: **3,3**
 (<Nome di entità: 60000014> (3.0 3.0 0.0))

Talvolta quando si eseguono operazioni su oggetti, è preferibile selezionare contemporaneamente un oggetto ed il punto con il quale è stato selezionato, come ad esempio, nello snap ad oggetti e nei comandi SPEZZA, TAGLIA e ESTENDI. La funzione **entsel** consente ai programmi AutoLISP di eseguire questa operazione. Essa seleziona un unico oggetto, richiedendo la selezione mediante un punto di selezione. L'impostazione corrente di Osnap viene ignorata da questa funzione (nessuno snap ad oggetti) a meno che non venga specificatamente richiesta durante l'uso della funzione. La funzione **entsel** rispetta eventuali parole chiave provenienti da una precedente chiamata alla funzione **initget**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

entupd

Aggiorna l'immagine visualizzata di un oggetto (entità).

(entupd nome_ent)

Quando si modifica il vertice di una polilinea o l'attributo di un blocco mediante la funzione **entmod**, l'entità complessa non viene aggiornata sullo schermo. A tale scopo, è possibile utilizzare la funzione **entupd**. Questa funzione può essere richiamata con il nome di entità di una parte qualsiasi della polilinea o del blocco; non è necessario che sia l'entità di intestazione. Anche se la funzione **entupd** è prevista per polilinee e blocchi con attributi, essa può essere richiamata per qualsiasi altra entità. Essa rigenera sempre l'entità sullo schermo, comprese tutte le sottoentità.

Nota Se la funzione **entupd** viene utilizzata su un'entità nidificata, ossia un'entità all'interno di un blocco, oppure su un blocco in cui sono contenute entità nidificate, alcune entità potrebbero non essere rigenerate. Per garantire una completa rigenerazione, è necessario richiamare il comando RIGEN.

Ad esempio, supponendo che la prima entità del disegno sia una polilinea con diversi vertici, si avrà che:

```
(setq e1 (entnext))           Imposta e1 sul nome entità della polilinea
(setq e2 (entnext e1))       Imposta e2 sul primo vertice
(setq ed (entget e2))        Imposta ed sui dati del vertice
(setq ed
  (subst '(10 1.0 2.0)
    (assoc 10 ed)           Cambia la posizione del vertice in ed
    ed)                   nel punto (1,2)
  )
)
(entmod ed)                 Sposta il vertice nel disegno
(entupd e1)                 Rigenera l'entità polilinea e1
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

eq

Determina se due espressioni sono identiche.

(eq espressione1 espressione2)

La funzione **eq** determina se gli argomenti *espressione1* e *espressione2* sono associati allo stesso oggetto, ad esempio mediante la funzione **setq**. Restituisce T se le due espressioni sono identiche, altrimenti restituisce nil.

È possibile utilizzare questa funzione per stabilire se due elenchi sono uguali. Ad esempio, date le assegnazioni

```
(setq f1 '(a b c))
(setq f2 '(a b c))
(setq f3 f2)
```

si avrà che

```
(eq f1 f3)           restituisce nil, f1 e f3 non sono uguali
(eq f3 f2)           restituisce T, f3 e f2 sono perfettamente uguali
```

Vedere anche

le funzioni = (uguale a) e **equal**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

equal

Determina se due espressioni sono uguali.

(equal espressione1 espressione2 [approssimazione])

La funzione **equal** determina se la valutazione degli argomenti *espressione1* ed *espressione2* dà lo stesso risultato. Quando si comparano due numeri reali oppure due elenchi di numeri reali, come nel caso dei punti, due numeri *identici* possono differire leggermente se sono stati utilizzati metodi diversi per calcolarli. Pertanto, è possibile utilizzare un argomento numerico opzionale, *approssimazione*, per specificare la differenza massima che può intercorrere tra *espressione1* e *espressione2* senza che questi argomenti possano essere considerati diversi.

Ad esempio, date le assegnazioni

```
(setq f1 '(a b c))
(setq f2 '(a b c))
(setq f3 f2)
(setq a 1.123456)
(setq b 1.123457)
```

si avrà che

```
(equal f1 f3)           restituisce T
(equal f3 f2)           restituisce T
(equal a b)             restituisce nil
(equal a b 0.000001)    restituisce T
```

Due elenchi considerati identici dalla funzione **equal** potrebbero non esserlo per la funzione **eq**; al contrario, gli atomi considerati identici dalla funzione **equal** risultano sempre uguali per la funzione **eq**. Inoltre due elenchi o due atomi considerati identici da **eq**, lo sono anche per la funzione **equal**.

Vedere anche

le funzioni **e** = (uguale a) **eq**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

error

È una funzione per la gestione degli errori che può essere definita dall'utente.

(*error* stringa)

Se la funzione ***error*** è diversa da nil, viene eseguita come funzione ogni volta che si verifica una condizione di errore AutoLISP. AutoCAD passa un argomento a ***error*** che rappresenta una stringa in cui è contenuta la descrizione dell'errore.

La funzione di seguito riportata si comporta nello stesso modo del gestore standard degli errori AutoLISP. L'errore e la descrizione vengono visualizzati.

```
(defun *error* (msg)
  (princ "error: ")
  (princ msg)
  (princ)
)
```

Nella funzione ***error*** è possibile includere chiamate alla funzione **command** senza specificare argomenti, ad esempio (command). Ciò comporta l'annullamento di un precedente comando di AutoCAD richiamato con la funzione **command**.

Vedere anche

"Gestione degli errori" per un esempio del gestore di errori che verifica la stringa restituita dalle funzioni **exit** e **quit**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

eval

Restituisce il risultato della valutazione di un'espressione AutoLISP.

(eval espressione)

Ad esempio, date le assegnazioni

```
(setq a 123)
(setq b 'a)
```

si avrà che

(atan 4.0)	<i>restituisce</i>	4.0
(eval (abs -10))	<i>restituisce</i>	10
(eval a)	<i>restituisce</i>	123
(eval b)	<i>restituisce</i>	123

Capitolo 13 -- AutoLISP: catalogo delle funzioni

exit

Forza l'uscita dall'applicazione corrente.

(exit)

Se si richiama la funzione **exit**, viene restituito il messaggio di errore esci./continua e si torna alla riga di comando di AutoCAD.

Vedere anche

la funzione **quit**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

exp

Restituisce la costante e elevata ad una determinata potenza (antilogaritmo naturale).

(**exp** *numero*)

(exp 1.0)	<i>restituisce</i>	2.71828
(exp 2.2)	<i>restituisce</i>	9.02501
(exp -0.4)	<i>restituisce</i>	0.67032

Capitolo 13 -- AutoLISP: catalogo delle funzioni

expand

Assegna lo spazio per i nodi richiedendo un determinato numero di segmenti.

(**expand** *int*)

Vedere anche

"Allocazione manuale" per ulteriori informazioni sulla funzione **expand**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

expt

Restituisce un numero elevato ad una determinata potenza.

(**expt** *numero potenza*)

Se entrambi gli argomenti sono numeri interi, il risultato sarà un numero intero, altrimenti, sarà un numero reale. Ad esempio:

(expt 2 4)	<i>restituisce</i>	16
(expt 3.0 2.0)	<i>restituisce</i>	9.0

Capitolo 13 -- AutoLISP: catalogo delle funzioni

fill_image

Disegna un rettangolo riempito nel gruppo di immagini attualmente attivo nella finestra di dialogo.

(**fill_image** *x1 y1 larg altez colore*)

La funzione **fill_image** deve essere utilizzata tra le chiamate **start_image** e **end_image**. Il parametro *colore* è un numero di colore di AutoCAD oppure uno dei numeri di colore logici riportati nella tabella seguente.

Nomi simbolici per l'attributo colore

Numero colore	Mnemonico ADI	Descrizione
-2	BGLCOLOR	È lo sfondo corrente dello schermo grafico di AutoCAD.

-15	<i>DBGLCOLOR</i>	È il colore di sfondo corrente della finestra di dialogo.
-16	<i>DFGLCOLOR</i>	È il colore di primo piano corrente della finestra di dialogo (per testo).
-18	<i>LINELCOLOR</i>	È il colore della linea corrente della finestra di dialogo.

Il primo angolo, superiore sinistro, del rettangolo viene posizionato ($x1,y1$), il secondo, inferiore destro, si trova alla distanza relativa (*larg,altez*) dal primo angolo (*larg* e *altez* devono essere valori positivi). L'origine (0,0) è l'angolo superiore sinistro dell'immagine. È possibile ottenere le coordinate dell'angolo inferiore destro richiamando le funzioni di quota **dimx_tile** e **dimy_tile**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

findfile

Ricerca il file specificato nel percorso di libreria di AutoCAD.

(findfile nomefile)

La funzione **findfile** non considera il tipo di file o l'estensione di *nomefile*. Se non viene specificato un prefisso per un'unità o una directory, la funzione effettua la ricerca nella libreria di AutoCAD. In caso contrario, la funzione **findfile** ricerca il file soltanto in quella directory. La funzione **findfile** restituisce sempre il nome completo dell'unità, della directory e del file o restituisce nil se il file non viene trovato.

Considerare i seguenti esempi. Se la directory corrente è */acad* e contiene il file *abc.lsp*.

```
(findfile "abc.lsp") restituisce "/acad/abc.lsp"
```

Se si sta modificando un disegno nella directory */acad/drawings*, la variabile d'ambiente ACAD è impostata su */acad/support*, e il file *xyz.txt* esiste solo nella directory */acad/support*.

```
(findfile "xyz.txt") restituisce "/acad/support/xyz.txt"
```

Se il file *nosuch* non è presente in nessuna directory del percorso di ricerca della libreria.

```
(findfile "nosuch") restituisce nil
```

Il nome completo restituito dalla funzione **findfile** può essere utilizzato con la funzione **open**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

fix

Restituisce la conversione di un numero reale nel numero intero inferiore che lo precede.

(fix numero)

La funzione **fix** tronca *numero* all'intero più prossimo eliminando la parte frazionaria.

```
(fix 3) restituisce 3
```

```
(fix 3.7) restituisce 3
```

Nota Se **numero** è più grande del maggiore numero intero possibile (+2,147,483,647 o -2,147,483,648 su una piattaforma a 32 bit), la funzione **fix** restituisce un numero reale troncato (sebbene i numeri interi trasferiti tra AutoLISP e AutoCAD sono limitati a valori a 16 bit).

Capitolo 13 -- AutoLISP: catalogo delle funzioni

float

Restituisce la conversione di un numero in numero reale.

(float *num*)

(float 3)	restituisce	3.0
(float 3.75)	restituisce	3.75

Capitolo 13 -- AutoLISP: catalogo delle funzioni

foreach

Valuta tutte le espressioni dei membri di una lista.

(foreach *nome lista espressione...*)

Esamina *lista*, assegnando ogni elemento a *nome*, e valuta tutte le *espressioni* per ogni elemento della lista. Può essere specificato qualsiasi numero di *espressione*. La funzione **foreach** restituisce il risultato dell'ultima *espressione* valutata.

```
(foreach n '(a b c) (print n))
```

è equivalente a

```
(print a)
(print b)
(print c)           e restituisce c
```

tranne per il fatto che la funzione **foreach** restituisce soltanto il risultato dell'ultima espressione valutata.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

gc

Libera nodi non utilizzati.

(gc)

Vedere anche

"Spazio per i nodi" per ulteriori dettagli.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

gcd

Restituisce il massimo comune denominatore di due numeri interi.

(gcd *int1 int2*)

Gli argomenti *int1* e *int2* devono essere numeri interi maggiori di 0.

```
(gcd 81 57)          restituisce 3
(gcd 12 20)         restituisce 4
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

get_attr

Indica il valore DCL di un attributo di una finestra di dialogo.

(**get_attr** *chiave attributo*)

L'argomento *chiave* è una stringa che specifica la casella e riconosce i caratteri maiuscoli e minuscoli. L'argomento *attributo* specifica il nome dell'attributo così come appare nella descrizione DCL della casella. Entrambi gli argomenti *chiave* ed *attributo* sono stringhe. Il valore restituito è il valore iniziale dell'attributo come specificato nella descrizione DCL; esso *non* riflette le modifiche apportate allo stato della casella dall'utente o da chiamate alla funzione **set_tile**. Il valore dell'attributo viene restituito come stringa.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

get_tile

Indica il valore corrente del tempo di esecuzione di una casella della finestra di dialogo.

(**get_tile** *chiave*)

L'argomento *chiave* è una stringa che specifica la casella e riconosce i caratteri maiuscoli e minuscoli. Restituisce il valore della casella in forma di stringa.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

getangle

Sospende l'esecuzione per consentire all'utente di specificare un angolo e restituisce quell'angolo in radianti.

(**getangle** [*pt*] [*msg_richiesta*])

L'argomento *pt* è un punto base bidimensionale dell'UCS corrente e *msg_richiesta* è una stringa visualizzata come messaggio di richiesta. L'argomento *pt*, se specificato, deve essere il primo di due punti, in modo che l'utente possa indicare ad AutoLISP l'angolo puntando su un altro punto. È possibile fornire un punto base tridimensionale, ma l'angolo viene sempre misurato nel corrente piano di costruzione.

La funzione **getangle** misura angoli con direzione radiante zero (impostata dalla variabile di sistema ANGBASE) con angoli crescenti in senso antiorario. L'angolo restituito è espresso in radianti, rispetto al corrente piano di costruzione (il piano XY dell'UCS corrente, all'elevazione corrente).

L'utente può specificare un angolo digitando un numero nel formato di unità di angolo corrente di AutoCAD. Sebbene tale formato possa essere in gradi, gradi centesimali o altre unità, la funzione restituisce comunque il valore in radianti. L'utente può inoltre indicare ad AutoLISP l'angolo puntando le due posizioni bidimensionali sullo schermo grafico. AutoCAD disegna una linea elastica dal primo punto alla posizione corrente del puntatore a croce per facilitare la visualizzazione dell'angolo.

È importante comprendere la differenza tra l'angolo di input e quello restituito dalla funzione **getangle**. Gli angoli passati alla funzione sono basati sulle impostazioni correnti di ANGDIR e ANGBASE. Comunque, una volta dato l'angolo, esso viene misurato in direzione antioraria (ignorando ANGDIR), con zero radianti come impostazione corrente di ANGBASE. I seguenti esempi di codici mostrano come possono essere utilizzati argomenti diversi.

```
(setq ang (getangle))
(setq ang (getangle '(1.0 3.5)))
(setq ang (getangle "Quale direzione? "))
(setq ang (getangle '(1.0 3.5) "Quale direzione? "))
```

L'utente non può immettere un'altra espressione AutoLISP in risposta ad una richiesta alla funzione **getangle**.

Vedere anche

l'illustrazione e il confronto con la funzione **getorient**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

getcfig

Richiama i dati dell'applicazione dalla sezione AppData del file *acad.cfg*.

(getcfig nomecfg)

L'argomento *nomecfg* è una stringa di al massimo 347 caratteri che assegna il nome alla sezione e al valore del parametro da indicare. Se l'argomento *nomecfg* non è valido, **getcfig** restituisce nil. L'argomento *nomecfg* deve essere una stringa nella forma:

```
"AppData/nome_applicazione/nome_sezione/.../nome_parametro"
```

Ad esempio, se il parametro SpessMur nella sezione AppData/Parametri_arco ha valore 8, si avrà che:

```
(getcfig "AppData/Parametro_arco/SpessMur")      restituisce  "8"
```

Vedere anche

la funzione **setcfig**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

getcname

Richiama il nome localizzato o inglese di un comando AutoCAD.

(getcname cnome)

L'argomento *cnome* specifica il nome del comando localizzato o il nome inglese con il segno di sottolineatura, che deve essere composto al massimo da 64 caratteri. Se *cnome* non è preceduto da un segno di sottolineatura (quindi interpretato come nome di comando localizzato), **getcname** restituisce il nome inglese del comando preceduto da un segno di sottolineatura. Se *cnome* è preceduto da un segno di sottolineatura, **getcname** restituisce il nome localizzato del comando. Questa funzione restituisce nil se *cnome* non è un nome di comando valido.

Viene di seguito riportato un esempio relativo alla versione italiana di AutoCAD.

```
(getcname "ETIRER")          restituisce  "_STRETCH"
(getcname "_STRETCH")       restituisce  "ETIRER"
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

getcorner

Sospende l'esecuzione per consentire all'utente di specificare il secondo angolo di un rettangolo.

(getcorner *pt* [*msg_richiesta*])

La funzione **getcorner** richiede un argomento relativo al punto base, *pt*, basato sull'UCS corrente, e traccia un rettangolo a partire dal punto in cui l'utente sposta il puntatore a croce sullo schermo. L'argomento *msg_richiesta* è una stringa che deve essere visualizzata come messaggio di richiesta. La funzione restituisce un punto nell'UCS corrente simile alla funzione **getpoint**. Se l'utente fornisce un punto tridimensionale, la coordinata Z viene ignorata, e viene considerata come tale l'elevazione corrente.

L'utente non può immettere un'altra espressione AutoLISP in risposta ad una richiesta alla funzione **getcorner**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

getdist

Sospende l'esecuzione per consentire all'utente di specificare una distanza.

(getdist [*pt*] [*msg_richiesta*])

L'argomento *pt* è un punto base bidimensionale o tridimensionale nell'UCS corrente. Se viene fornito, *pt* è utilizzato come il primo dei due punti, e all'utente viene richiesto di specificare soltanto il secondo punto. L'argomento *msg_richiesta* è una stringa che deve essere visualizzata come messaggio di richiesta.

L'utente può specificare la distanza selezionando due punti o il secondo punto se è stato fornito un punto base. La distanza può anche essere fornita digitando un numero nel formato di unità di distanza corrente di AutoCAD. Sebbene il formato possa essere espresso in piedi e pollici (architettonico) la funzione **getdist** restituisce sempre la distanza come numero reale.

La funzione **getdist** disegna una linea elastica dal primo punto alla posizione corrente del puntatore a croce per facilitare la visualizzazione della distanza.

Se viene fornito un punto tridimensionale il valore restituito è una distanza tridimensionale. L'impostazione della funzione **initget** a 64 bit fa sì che la funzione **getdist** ignori la componente Z dei punti tridimensionali e restituisca una distanza bidimensionale.

```
(setq dist (getdist))
(setq dist (getdist '(1.0 3.5)))
(setq dist (getdist "Lontano "))
(setq dist (getdist '(1.0 3.5) "Lontano? "))
```

L'utente non può immettere un'altra espressione AutoLISP in risposta ad una richiesta alla funzione **getdist**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

getenv

Restituisce il valore della stringa assegnato ad una variabile d'ambiente.

(getenv *nome-variabile*)

L'argomento *nome_variabile* è una stringa che specifica il nome della variabile da leggere. Se tale variabile non esiste, **getenv** restituisce nil.

Per esempio, se la variabile d'ambiente ACAD è impostata su */acad/support* e la variabile NOSUCH non esiste, allora

```
(getenv "ACAD")      restituisce  "/acad/support"
(getenv "NOSUCH")   restituisce  nil
```

Nota Su sistemi UNIX, ACAD e acad fanno riferimento a due variabili diverse, poiché questi sistemi operativi sono sensibili al maiuscolo/minuscolo.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

getfiled

Richiede il nome di un file con la finestra di dialogo standard AutoCAD e restituisce quel nome file.

(**getfiled** *titolo default est flag*)

L'argomento *titolo* specifica l'etichetta della finestra di dialogo, *default* specifica il nome di un file di default da utilizzare (che può essere una stringa nulla [""]), e *est* è l'estensione del nome del file di default. Se *est* viene passato come stringa nulla [""], il default è * (tutti i tipi di file). Se nell'argomento *est* viene incluso il tipo di file **dwg**, la funzione visualizza l'anteprima di un'immagine nella finestra di dialogo. L'argomento *flag* è un valore intero (campo codificato a bit) che controlla il comportamento della finestra di dialogo. Per impostare più di una condizione alla volta, aggiungere i valori per creare un valore *flag* tra 0 e 15.

Valore flag = 1 (bit 0)

Impostare questo bit quando si desidera *creare* un nuovo file. Non impostare questo bit se si desidera *aprire* un file esistente; in questo caso, se l'utente digita il nome di un file inesistente, la finestra di dialogo visualizza un messaggio di errore alla base della casella.

Se viene impostato questo bit e l'utente sceglie un file esistente, AutoCAD visualizza una casella di allarme, dando la possibilità di procedere o di annullare l'operazione.

Valore flag = 2 (bit 1)

Disabilita il pulsante Digita. Questo bit risulta impostato se la funzione **getfiled** viene richiamata mentre è attiva un'altra finestra di dialogo. In caso contrario provoca la scomparsa dell'altra finestra.

Il pulsante Digita è abilitato se questo bit non è impostato. Se l'utente seleziona il pulsante, la finestra di dialogo scompare e la funzione **getfiled** restituisce il valore 1.

Valore flag = 4 (bit 2)

Consente all'utente di digitare un'estensione di file diversa da quella di default, o nessuna estensione.

Se il bit non è impostato, la funzione **getfiled** accetta *solo* l'estensione specificata nell'argomento *est*, aggiungendola al nome del file nel caso in cui l'utente non l'abbia digitata nella casella di testo File.

Valore flag = 8 (bit 3)

Se questo bit è impostato ed il bit 0 *non* è impostato, la funzione **getfiled** esegue una ricerca del nome del file specificato all'interno della libreria. Se vengono trovati il file e la relativa directory nel percorso di ricerca della libreria, il percorso viene rimosso e viene restituito solo il nome del file. Se viene trovato un file con lo stesso nome in una directory diversa, il nome del percorso non viene rimosso.

Se questo bit non viene impostato, la funzione **getfiled** restituisce il nome del file e il percorso.

Impostare questo bit se si utilizza la finestra di dialogo per aprire un file esistente che si desidera salvare nel disegno o in altri database.

Se viene digitato il nome di un file nella finestra di dialogo, la funzione **getfiled** restituisce una stringa che specifica il nome del file; in caso contrario, la funzione restituisce nil.

La seguente chiamata alla funzione **getfiled** visualizza la finestra di dialogo Seleziona file Lisp:

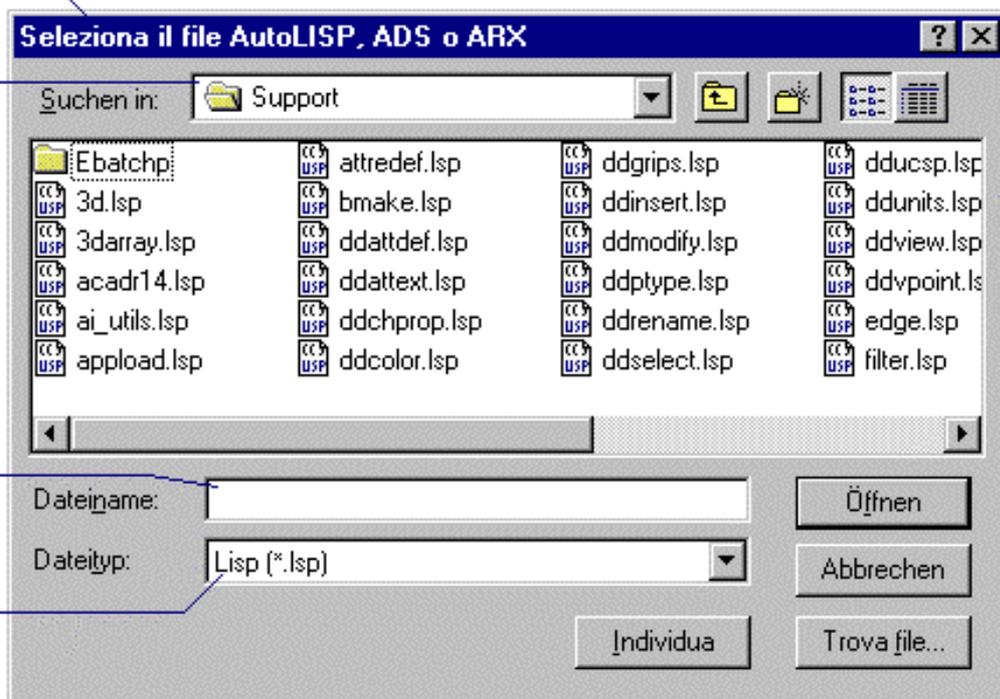
```
(getfiled "Seleziona file Lisp" "/acadr14/support/" "lsp" 8)
```

Impostato dall'argomento titolo

impostato dal nome del percorso dell'argomento default. (Se default non specifica un percorso, questa è inizialmente la directory corrente

impostato dal nome file dell'argomento default.

impostato dall'argomento est



Esempio di finestra di dialogo getfiled

La funzione **getfiled** visualizza una finestra di dialogo contenente un elenco di file del tipo di estensione specificata disponibili. È possibile usare tale finestra per sfogliare le diverse unità e directory, per selezionare un file esistente o specificare il nome di un nuovo file.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

getint

Sospende l'esecuzione per consentire all'utente di specificare un numero intero e restituisce quel numero intero.

`(getint [msg_richiesta])`

L'argomento *msg_richiesta* è una stringa opzionale che deve essere visualizzata come messaggio di richiesta. La funzione **getint** restituisce il valore intero o nil.

```
(setq num (getint))
(setq num (getint "Digitare un numero: "))
```

I valori passati alla funzione **getint** possono essere compresi tra -32,768 e +32,767. L'utente non può immettere un'altra espressione AutoLISP in risposta ad una richiesta alla funzione **getint**.

Vedere anche

"Funzioni getxxx" e la funzione if.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

getkword

Sospende l'esecuzione per consentire all'utente di specificare una parola chiave e restituisce quella parola chiave.

(**getkword** [*msg_richiesta*])

Le parole chiave valide vengono impostate prima della chiamata alla funzione **getkword** con la funzione **initget**. L'argomento *msg_richiesta* è una stringa che deve essere visualizzata come messaggio di richiesta. La funzione **getkword** restituisce la parola chiave corrispondente all'inserimento dell'utente come stringa. Se l'inserimento da parte dell'utente non è una parola chiave, AutoCAD effettua un nuovo tentativo. Se l'inserimento da parte dell'utente è nullo (RETURN), **getkword** restituisce nil (nel caso in cui sia consentito un inserimento nullo). La funzione restituisce nil anche nel caso in cui non sia stata preceduta da una chiamata alla funzione **initget** che ha stabilito una o più parole chiave.

L'esempio seguente mostra una chiamata iniziale alla funzione **initget** che imposta un elenco di parole chiave (Si e No) e non consente l'inserimento nullo (il valore (*bits* è pari a 1) alla seguente chiamata alla funzione **getkword**:

```
(initget 1 "Si No")
(setq x (getkword "Sicuro? (Si o No) "))
```

Questo codice richiede all'utente un inserimento, e il simbolo x viene impostato su Si o No, a seconda della risposta. Se la risposta non corrisponde ad alcuna parola chiave o se la risposta è nulla, AutoCAD chiede nuovamente di digitare la parola chiave usando la stringa fornita nell'argomento *msg_richiesta*. Se non viene fornito alcun argomento *msg_richiesta*, AutoCAD visualizza il seguente messaggio di richiesta:

Riprovare:

L'utente non può immettere un'altra espressione AutoLISP in risposta ad una richiesta alla funzione **getkword**.

Vedere anche

"Funzioni getxxx" e la funzione if.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

getorient

Sospende l'esecuzione per consentire all'utente di specificare un angolo e restituisce quell'angolo in radianti.

(**getorient** [*pt*] [*msg_richiesta*])

Questa funzione è simile alla funzione **getangle**, tranne per il fatto che le variabili di sistema ANGBASE e ANGDIR non influenzano il valore dell'angolo restituito dalla funzione **getorient**. Tuttavia, l'angolo specificato dall'utente è ancora basato sulle impostazioni correnti di ANGDIR e ANGBASE.

L'argomento *pt* è un punto base bidimensionale dell'UCS corrente e *msg_richiesta* è una stringa visualizzata come messaggio di richiesta. L'argomento *pt*, se specificato, deve essere il primo di due punti, in modo che l'utente possa indicare ad AutoLISP l'angolo puntando su un altro punto. È possibile fornire un punto base tridimensionale, ma l'angolo viene sempre misurato nel corrente piano di costruzione.

La funzione **getorient** misura gli angoli con direzione zero radianti verso destra (est) e gli angoli crescenti in senso antiorario. Come la funzione **getangle**, anche la funzione **getorient** esprime l'angolo restituito in radianti, secondo il corrente piano di costruzione. Gli angoli passati alla funzione sono basati sulle IMPOSTAZIONI CORRENTI DI ANGDIR e ANGBASE. Una volta fornito l'angolo esso viene comunque misurato in senso antiorario, con direzione zero radianti a destra (ignorando ANGDIR and ANGBASE). Se viene selezionato un orientamento di base differente o una diversa direzione per aumentare l'angolo, dovrà essere effettuata una conversione usando il comando UNITA o le variabili di sistema ANGBASE and ANGDIR.

Usare la funzione **getangle** quando è necessario il valore di rotazione (angolo relativo). Usare la funzione **getorient** per ottenere un orientamento (angolo assoluto).

L'utente non può immettere un'altra espressione AutoLISP in risposta ad una richiesta alla funzione **getorient**.

Vedere anche

"Funzioni getxxx," la funzione **getangle** e la funzione **if**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

getpoint

Sospende l'esecuzione per consentire all'utente di specificare un punto e restituisce quel punto.

```
(getpoint [pt] [msg_richiesta])
```

L'argomento *pt* è un punto base bidimensionale o tridimensionale dell'UCS corrente e *msg_richiesta* è una stringa visualizzata come messaggio di richiesta. L'utente può specificare un punto mediante un puntamento o digitando una coordinata nel formato di unità corrente. Se è presente l'argomento *pt*, AutoCAD traccia una linea elastica da quel punto alla posizione corrente del puntatore a croce. Il valore restituito è un punto tridimensionale espresso in termini di UCS corrente.

```
(setq p (getpoint))  
(setq p (getpoint "Dove? "))  
(setq p (getpoint '(1.5 2.0) "Secondo punto: "))
```

L'utente non può immettere un'altra espressione AutoLISP in risposta ad una richiesta alla funzione **getpoint**.

Vedere anche

"Funzioni getxxx," e le funzioni **getcorner** e **if**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

getreal

Sospende l'esecuzione per consentire all'utente di specificare un numero reale e restituisce quel numero reale.

```
(getreal [msg_richiesta])
```

L'argomento *msg_richiesta* è una stringa che deve essere visualizzata come messaggio di richiesta.

```
(setq val (getreal))  
(setq val (getreal "Fattore di scala: "))
```

L'utente non può immettere un'altra espressione AutoLISP in risposta ad una richiesta alla funzione **getreal**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

getstring

Sospende l'esecuzione per consentire all'utente di specificare una stringa e restituisce tale stringa.

```
(getstring [cr] [msg_richiesta])
```

Se l'argomento *cr* viene fornito o non è nil, la stringa di input può contenere spazi vuoti e deve terminare con un PETYPN. In caso contrario, la stringa di input viene terminata dallo spazio o da RETURN. L'argomento *msg_richiesta* è una stringa che deve essere visualizzata come messaggio di richiesta.

Se la stringa supera 132 caratteri, vengono restituiti soltanto i primi 132. Se la stringa di input contiene una barra inversa, (\), essa verrà convertita in due barre inverse (\\). In questo modo il valore restituito può contenere i percorsi dei file che possono essere utilizzati da altre funzioni.

```
(setq s (getstring "Qual è il tuo nome? "))
```

rispondendo **Gianni** , la funzione imposta s su "Gianni".

```
(setq s (getstring T "Qual è il tuo nome completo? "))
```

rispondendo **Gianni Rossi** , la funzione imposta s su "Gianni Rossi".

```
(setq s (getstring "Immettere il nome del file: "))
```

rispondendo **\\acad\\mydwg** , la funzione imposta s su "\\acad\\mydwg".

L'utente non può immettere un'altra espressione AutoLISP in risposta ad una richiesta alla funzione **getstring**.

Vedere anche

la funzione **getkeyword** per una routine che richiede all'utente di immettere una delle opzioni note (parole chiave).

Capitolo 13 -- AutoLISP: catalogo delle funzioni

getvar

Richiama il valore di una variabile di sistema di AutoCAD.

```
(getvar nomevar)
```

L'argomento *nomevar* è una stringa che assegna un nome alla variabile di sistema. Se *nomevar* non è una variabile di sistema valida, la funzione **getvar** restituisce nil.

Per esempio, se il raggio di raccordo è impostato su 0.25 unità,

```
(getvar "FILLETRAD")      restituisce 0.25
```

Nella Guida di riferimento dei comandi di AutoCAD è disponibile l'elenco delle variabili di sistema correnti di AutoCAD.

Vedere anche

la funzione **setvar**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

graphscr

Visualizza lo schermo grafico di AutoCAD.

```
(graphscr)
```

La funzione **graphscr** restituisce sempre nil.

Questa funzione è equivalente al comando SCHGRAF o all'attivazione del tasto funzione Grafico/Testo. La funzione **textscr** è complementare alla funzione **graphscr**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

grclear

Cancella la finestra corrente (obsoleta).

(**grclear**)

La funzione **grclear** restituisce sempre nil.

Questa funzione non modifica le aree di comando di messaggio di richiesta, di stato e di menu. È possibile utilizzare la funzione **redraw** per ripristinare il contenuto precedente dello schermo grafico. Su sistemi che prevedono un solo schermo la funzione **grclear** passa allo schermo grafico prima di cancellare la finestra corrente.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

grdraw

Disegna un vettore tra due punti nella finestra corrente.

(**grdraw** *da a colore [evidenziazione]*)

Gli argomenti *da* e *a* sono punti bidimensionali o tridimensionali (elenchi di due o tre numeri reali) che specificano le estremità del vettore nei termini dell'UCS corrente. AutoCAD ritaglia il vettore per adattarlo allo schermo. La funzione **grdraw** disegna il vettore con il colore specificato dell'argomento *colore* che è un numero intero; con -1 viene specificato *inchiostro XOR*, cioè un inchiostro che complementa qualsiasi colore su cui si disegna e non altera il colore di ciò che viene sovrascritto. Se l'argomento *evidenziazione* che è un numero intero fornito è diverso da zero, il vettore viene disegnato usando il metodo di evidenziazione di default del dispositivo video (generalmente tratteggiato). Se l'argomento *evidenziazione* viene omissso o è zero, la funzione **grdraw** usa la modalità di visualizzazione normale. e restituisce sempre nil.

Vedere anche

la funzione **grvecs** per la routine che disegna più vettori.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

grread

Legge i valori da qualsiasi dispositivo di input AutoCAD.

(**grread** [*traccia*] [*tutti_tasti*] [*tipo_cur*])

Questa funzione è richiesta soltanto da routine AutoLISP specializzate. Per la maggior parte dei casi sono sufficienti le varie funzioni **getxxx**.

Se l'argomento *traccia* è fornito ed è diverso da nil, viene abilitata la restituzione delle coordinate dal movimento di un dispositivo di puntamento quando questo viene spostato. Se è presente l'argomento *tutti_tasti*, **grread** esegue le funzioni in base al codice fornito. L'argomento *tipo_cur* può essere usato per controllare il tipo di cursore visualizzato. Gli argomenti *tutti_tasti* e *tipo_cur* sono numeri interi che vengono descritti di seguito:

tutti_tasti

I valori di codice del bit *tutti_tasti* possono essere aggiunti insieme per funzionalità combinate.

1 (bit 0) Restituisce le coordinate di *modalità trascinalamento*. Se questo bit viene impostato e l'utente sposta il dispositivo di puntamento invece di selezionare un pulsante o di premere un tasto, **grread** restituisce un elenco in cui il primo membro è di tipo 5 e il secondo membro è formato dalle coordinate (X,Y) della posizione del dispositivo di puntamento corrente: mouse o digitalizzatore. In questo modo AutoCAD attua il trascinalamento.

2 (bit 1) Restituisce *tutti* i valori dei tasti, inclusi i codici dei tasti funzione e del cursore e *non* sposta il cursore quando l'utente preme un tasto cursore.

4 (bit 2) Usa il valore passato all'argomento *tipo_cur* per controllare la visualizzazione del cursore.

8 (bit 3) Non visualizza il messaggio di errore: di interruzione della console quando si preme CTRL+C .

tipo_cur

Il valore *tutti_tasti* per il bit 2 deve essere impostato affinché i valori *tipo_cur* diventino effettivi.

- 0 Visualizza il normale puntatore a croce.
- 1 Non visualizza il cursore (nessun puntatore a croce).
- 2 Visualizza il cursore di "destinazione" della selezione di oggetti.

Nota l'argomento *tipo_cur* ha effetto sul tipo di cursore soltanto durante la chiamata corrente della funzione **gread**.

Nota nelle release future di AutoCAD potrebbero essere definiti ulteriori bit di controllo.

La funzione **gread** restituisce un elenco il cui primo elemento è un codice che specifica il tipo di input. Il secondo elemento dell'elenco può essere un numero intero o un punto, a seconda del tipo di input. I valori restituiti sono elencati nella seguente tabella.

Valori restituiti da *gread*

Primo elemento	Tipo di input	Secondo elemento	Descrizione
Valore		Valore	
2	Input da tastiera	varia	È il codice di carattere.
3	Punto selezionato	punto 3D	Coordinate del punto.
4	Schermo/voce di menu a discesa (da dispositivo di puntamento)	0 a 999 1001 a 1999 2001 a 2999 3001 a 3999 ... e così via, fino a 16001 a 16999	N. casella di menu di schermo N. casella di menu POP1 N. casella di menu POP2 N. casella di menu POP3 ...
5	Dispositivo di puntamento (restituito solo se è abilitata la tracciatura)	punto 3D	-N. casella di menu POP16 Coordinata modalità trascinamento.
6	voce di menu BUTTONS	0 a 999 1000 a 1999 2000 a 2999 3000 a 3999	N. pulsante di menu BUTTONS1 N. pulsante di menu BUTTONS2 N. pulsante di menu BUTTONS3 N. pulsante di menu BUTTONS4
7	Voce di menu TABLET1	0 a 32767	N. casella digitalizzata.
8	Voce di menu TABLET2	0 a 32767	N. casella digitalizzata.
9	Voce di menu TABLET3	0 a 32767	N. casella digitalizzata.
10	Voce di menu TABLET4	0 a 32767	N. casella digitalizzata.
11	Voce di menu AUX	0 a 999 1000 a 1999 2000 a 2999 3000 a 3999	N. pulsante di menu AUX1 N. pulsante di menu AUX2 N. pulsante di menu AUX3 N. pulsante di menu AUX4
12	Pulsante di puntamento (segue la restituzione del tipo 6 o 11)	punto 3D	Coordinate del punto.

Premere ESC quando è attivo **gread** provoca l'interruzione del programma AutoLISP da tastiera (a meno che l'argomento *allkeys* ne abbia disattivato il funzionamento). Qualsiasi altro input viene passato direttamente alla funzione **gread**, fornendo all'applicazione il controllo completo sui dispositivi di input.

Se l'utente preme il pulsante del puntatore all'interno di una casella di un menu di schermo o di un menu a discesa, la funzione **gread** restituisce il codice di tipo 6 o 11, ma in una chiamata successiva non ne restituisce uno di tipo 12: tale codice infatti segue il tipo 6 o 11 solo quando il pulsante del puntatore viene premuto all'interno dell'area grafica dello schermo.

È importante cancellare i dati del codice 12 dal buffer prima di eseguire un'altra operazione con un pulsante del puntatore o con un pulsante ausiliario. Per cancellare i dati, eseguire una funzione **gread** nidificata in questo modo:

```
(setq code_12 (gread (setq code (gread))))
```

Tale sequenza cattura il valore dell'elenco del codice 12 come input dalla periferica.

Nota Poiché l'input viene gestito diversamente a seconda delle varie piattaforme supportate da AutoCAD, la funzione **gread** può restituire risultati inattesi.

- Il dispositivo di puntamento di default su piattaforme che utilizzano il mouse restituisce il codice 11 e non il codice 6.
- Sulla piattaforma Macintosh i menu a comparsa restituiscono il codice 11 e non il codice 4. Anche in Macintosh un doppio clic restituisce il codice 11 (non il codice 6) ed è seguito da una coppia di coordinate di codice 5, se la selezione avviene nella finestra corrente. Un doppio clic in una finestra diversa da quella corrente restituisce una coppia di coordinate di codice 3, seguite dal codice 11.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

grtext

Scrivere il testo sulla riga di stato o nelle aree del menu di schermo.

`(grtext [casella testo [evidenz]])`

L'argomento *casella* è un numero intero che specifica la posizione in cui scrivere il testo. L'argomento *testo* è una stringa che specifica il testo da scrivere sul menu di schermo o la posizione della riga di stato. Se *testo* risulta troppo lungo rispetto all'area disponibile, viene troncato. L'argomento *evidenziazione* è un numero intero che seleziona o deselecta una posizione del menu di schermo. I valori di questi argomenti variano a seconda della posizione dello schermo su cui si sta scrivendo. Questa funzione visualizza il testo fornito nell'area di menu, e non modifica la voce di menu sottostante. La funzione **grtext** può essere richiamata senza argomenti per ripristinare i valori di default di tutte le aree di testo. Se l'esito della funzione è positivo, essa restituisce la stringa passata all'argomento *testo*; in caso contrario, restituisce nil.

Area menu di schermo

Impostando un valore positivo o zero per la *casella* viene specificata una posizione del menu di schermo. L'intervallo di valori validi dell'argomento *casella* è compreso tra 0 e il numero massimo del menu di schermo meno 1. La variabile di sistema SCREENBOXES riporta il numero massimo di caselle di menu di schermo. Se il valore fornito per l'argomento *evidenziazione* è un numero intero positivo, la funzione **grtext** evidenzia il testo nella casella designata. Quando si evidenzia una casella, l'evidenziazione di qualsiasi altra casella viene annullata. Se *evidenziazione* è zero, l'evidenziazione della voce di menu viene annullata. Se *evidenziazione* è un numero negativo, l'argomento viene ignorato. Su alcune piattaforme, il testo deve prima essere scritto senza l'argomento *evidenziazione* e deve essere evidenziato successivamente. L'evidenziazione della posizione di un menu dello schermo funziona solo se il cursore non è in quell'area.

Area riga di stato

Se la funzione **grtext** viene richiamata con un valore *casella* pari a -1, il testo verrà scritto nella modalità indicata dalla riga di stato. La lunghezza di tale riga differisce da un monitor a un altro (la maggior parte dei monitor accetta almeno 40 caratteri). Il seguente codice usa l'espressione **\$(linelen)** DIESEL per riportare la lunghezza dell'area di stato.

```
(setq modelen (menucmd "M=$(linelen)"))
```

Se viene usato un valore di *casella* -2, la funzione **grtext** scrive il testo all'interno della riga di stato delle coordinate. Se è attivata la tracciatura delle coordinate, i valori scritti in questo campo vengono sovrascritti non appena il puntatore invia un altro gruppo di coordinate. Per i valori -1 o -2, l'argomento *evidenziazione* viene ignorato.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

grvecs

Disegna vettori multipli sullo schermo grafico.

`(grvecs elenco_vett [trasf])`

L'argomento *elenco_vett* è un elenco di vettori comprendente una serie di numeri interi opzionali di colore e due elenchi di punti. L'argomento *trasf* è una matrice di trasformazione che è possibile utilizzare per cambiare la posizione o la proporzione dei vettori definiti nell'elenco dei vettori. Questa matrice è un elenco formato da quattro numeri reali.

Il formato di *elenco_vett* è il seguente:

```
([colore1] da 1 a 1 [colore2] da 2 a 2 ...)
```

Il valore del colore interessa tutti i vettori successivi finché *elenco_vett* non specifica un altro colore. L'intervallo dei colori di AutoCAD varia da 0 a 255. Se il valore del colore è maggiore di 255, i vettori successivi vengono disegnati in *inchiostro XOR*, cioè un inchiostro che complementa qualsiasi colore su cui si disegna e non altera il colore di ciò che viene sovrascritto. Se il valore del colore è minore di zero, il vettore viene evidenziato. L'evidenziazione dipende dal dispositivo di visualizzazione. La maggior parte dei driver indicano l'evidenziazione con una linea tratteggiata, ma alcuni utilizzano un colore particolare.

Una coppia di elenchi di punto, *da* e *a*, specifica i punti finali dei vettori, espressi nell'UCS corrente, e possono essere

bidimensionali o tridimensionali. È necessario passare questi punti come coppie, in due elenchi successivi, altrimenti la chiamata alla funzione **grvecs** non ha esito positivo.

AutoCAD ritaglia i vettori per farli entrare nello schermo. Se la chiamata alla funzione **grvecs** ha esito positivo, essa restituisce nil.

Il seguente codice disegna cinque linee verticali sullo schermo grafico, ognuna di un colore diverso:

```
(grvecs ' (1 (1 2) (1 5)      Disegna una linea rossa da (1,2) a (1,5)
          2 (2 2) (2 5)      Disegna una linea gialla da (2,2) a (2,5)
          3 (3 2) (3 5)      Disegna una linea verde da (3,2) a (3,5)
          4 (4 2) (4 5)      Disegna una linea ciano da (4,2) a (4,5)
          5 (5 2) (5 5)      Disegna una linea blu da (5,2) a (5,5)
) )
```

La matrice seguente rappresenta una scala uniforme di 1.0 e una inclinazione di 5.0,5.0,0.0. Se questa matrice viene applicata al precedente elenco di vettori, essi saranno spostati di 5.0,5.0,0.0.

```
' ( (1.0 0.0 0.0 5.0)
    (0.0 1.0 0.0 5.0)
    (0.0 0.0 1.0 0.0)
    (0.0 0.0 0.0 1.0)
)
```

Vedere anche

la funzione **nentselp** per ulteriori informazioni sulle matrici di trasformazione.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

handent

Restituisce il nome di un oggetto (entità) basato sul gestore.

(**handent** *gestore*)

Data una stringa di gestore di entità come argomento *gestore*, la funzione **handent** restituisce il nome dell'entità associato a quel gestore nella sessione di modifica corrente. La funzione **handent** restituisce il nome di entrambe le entità grafiche e non grafiche.

Se alla funzione **handent** viene passato un gestore non valido o un gestore non usato da alcuna entità nel disegno corrente, viene restituito nil. La funzione **handent** restituisce entità cancellate durante la sessione di modifica corrente; è possibile annullare la cancellazione con la funzione **entdel**.

Il nome di un'entità può cambiare da una sessione di modifica all'altra, mentre il gestore dell'entità rimane costante. In una particolare sessione di modifica, il codice:

```
(handent "5A2") potrebbe restituire <Nome entità: 60004722>
```

Usata con lo stesso disegno ma in un'altra sessione di modifica, la stessa chiamata potrebbe restituire un diverso nome di entità. Il nome dell'entità ottenuto può essere usato per manipolare l'entità stessa con qualsiasi funzione di entità.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

help

Richiama la Guida.

(**help** [*file_guida* [*argomento* [*comando*]])

L'argomento *file_guida* è una stringa che specifica un file della guida. Se si specifica un file della Guida di AutoCAD (*.ahp*), la funzione **help** usa la funzione per la visualizzazione della Guida di AutoCAD per visualizzare il contenuto di quel file. Se si specifica un file della Guida di Windows (*.hlp*), la funzione **help** usa il programma WinHelp per visualizzare il file. Se l'argomento *file_guida* è

una stringa vuota ("") o viene omessa, AutoCAD usa il file della Guida di AutoCAD di default. L'argomento *argomento* è una parola chiave che specifica l'argomento visualizzato all'inizio della funzione della guida. Se l'argomento *argomento* è una stringa vuota (""), la funzione della guida visualizza la parte introduttiva del file della guida. L'argomento *comando* è una stringa che specifica lo stato iniziale della finestra della Guida, come indica la tabella seguente.

Valori per l'argomento *comando*

Stringa	Descrizione
HELP_CONTENTS	Visualizza il primo argomento presente nel file della Guida.
HELP_HELPONHELP	Visualizza informazioni di aiuto circa l'uso della Guida.
HELP_PARTIALKEY	Visualizza la finestra di dialogo per la ricerca utilizzando la stringa trasmessa come argomento per il testo iniziale per la ricerca.

Se si sta specificando un file della Guida di Windows, l'argomento *comando* può anche essere una stringa utilizzata dall'argomento *fuCommand* della funzione WinHelp() come definito da API di WinHelp in SDK di Microsoft Windows.

Con l'argomento *fileguida* non è richiesta l'estensione di file. Se viene fornita un'estensione di file, AutoCAD cerca solo quel determinato file. Se non ne viene fornita nessuna, la ricerca viene eseguita in base alle seguenti regole: append *.hlp*, altrimenti ricerca *.ahp*.

La sola condizione di errore che la funzione **help** restituisce all'applicazione è la presenza del file specificato da *file_guida*. Tutte le altre condizioni di errore vengono segnalate all'utente mediante una finestra di dialogo. La funzione **help** restituisce la stringa *fileguida* se ha esito positivo e nil in caso contrario. Se si usa la funzione **help** senza argomenti, essa restituisce una stringa vuota ("") in caso di successo, altrimenti restituisce nil.

Il codice seguente richiama la funzione **help** per visualizzare le informazioni sul comando MIOCOMANDO nel file della guida *achelp.ahp*:

```
(help "achelp.ahp" "mio_comando")
```

Vedere anche

"Personalizzazione della documentazione in linea" per informazioni relative alla creazione di file della Guida di AutoCAD. La funzione **setfunhelp** associa la guida contestuale (quando l'utente preme F1) con un comando definito dall'utente.

Capitolo 13 -- AutoLISP: catalogo delle funzioni



Valuta le espressioni in modo condizionale.

```
(if espr_verifica espr_then [espr_else])
```

Se *espr_verifica* è diversa da nil, valuta *espr_then*, altrimenti valuta *espr_else*. La funzione **if** restituisce il valore dell'espressione selezionata. Se *espr_else* manca e *espr_verifica* è nil, la funzione **if** restituisce nil.

```
(if (= 1 3) "SI!!" "no.") restituisce "no."
(if (= 2 (+ 1 1)) "SI!!") restituisce "SI!!"
(if (= 2 (+ 3 4)) "SI!!") restituisce nil
```

Vedere anche

la funzione **progn**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni



Stabilisce parole chiave da utilizzare dalla successiva chiamata alla funzione di input utente.

```
(initget [bit] [stringa])
```

Le funzioni che utilizzano le parole chiave sono **getint**, **getreal**, **getdist**, **getangle**, **getorient**, **getpoint**, **getcorner**, **getkeyword**, **entsel**, **nentsel** e **nentselp**. La funzione **getstring** è l'unica funzione di input utente che non utilizza parole chiave.

L'argomento *bit* è un numero intero in bit che abilita o disabilita determinati tipi di input da parte dell'utente. L'argomento *stringa* definisce un elenco di parole chiave controllate dalla successiva chiamata alla funzione di input utente quando l'utente non immette il tipo di input richiesto (per esempio, un punto per la funzione **getpoint**). Se l'input dell'utente corrisponde ad una parola chiave dell'elenco, la funzione la restituisce come risultato in forma di stringa. L'applicazione può verificare le parole chiave ed eseguire l'azione associata ad ognuna di esse. Se l'input dell'utente non è del tipo richiesto e non corrisponde ad una parola chiave, AutoCAD chiede all'utente di ritentare. I valori di bit della funzione **initget** e le parole chiave vengono applicati soltanto alla chiamata della successiva funzione di input utente.

La funzione **initget** restituisce sempre nil.

Se la funzione **initget** imposta un bit di controllo e l'applicazione richiama una funzione di input utente per la quale il bit non ha valore, esso viene ignorato. I bit possono essere sommati in qualsiasi combinazione per formare un valore compreso tra 0 e 255. Se non viene fornito l'argomento *bit*, si presume zero (nessuna condizione). Se l'input da parte dell'utente non corrisponde ad una o più delle condizioni specificate (per esempio un valore zero quando tale valore non è ammesso), AutoCAD visualizza un messaggio e chiede all'utente di ritentare.

Opzioni di input impostate da iniget

Valore del bit	<i>Descrizione</i>
1 (bit 0)	<i>Impedisce all'utente di rispondere alla richiesta immettendo solo RETURN.</i>
2 (bit 1)	<i>Impedisce all'utente di rispondere alla richiesta immettendo zero.</i>
4 (bit 2)	<i>Impedisce all'utente di rispondere alla richiesta immettendo solo un valore negativo.</i>
8 (bit 3)	<i>Consente all'utente di immettere un punto al di fuori dei limiti del disegno corrente. Questa condizione viene applicata alla successiva funzione di input utente anche se è impostata la variabile di sistema LIMCHECK di AutoCAD.</i>
16 (bit 4)	<i>Attualmente non usato.</i>
32 (bit 5)	<i>Usa linee tratteggiate per il disegno di linee elastiche o caselle. Per quelle funzioni che permettono all'utente di specificare un punto selezionando una posizione sullo schermo grafico, il valore di questo bit fa sì che la linea elastica o la casella siano tratteggiate invece che solide. I driver di alcuni monitor usano un particolare colore al posto delle linee tratteggiate. Se la variabile di sistema POPUPS è 0, AutoCAD ignora questo bit.</i>
64 (bit 6)	<i>Impedisce l'input di una coordinata Z per la funzione getdist; permette ad una applicazione di assicurare che tale funzione restituisca una distanza bidimensionale.</i>
128 (bit 7)	<i>Permette l'input arbitrario come se fosse una parola chiave, rispettando qualsiasi altro bit di controllo e parola chiave presenti nell'elenco. Questo bit ha la precedenza sul bit 0; se il bit 7 ed il bit 0 sono impostati e l'utente immette RETURN, verrà restituita una stringa nulla.</i>

Nota in versioni future AutoLISP potrebbero essere definiti ulteriori bit di controllo della funzione **initget**, quindi è preferibile non impostare i bit non indicati nella tabella.

I valori di controllo speciali sono rispettati solo dalle funzioni **getxxx** per le quali hanno significato.

Funzioni input utente e bit di controllo applicabili

Funzione	<i>Parole chiave rispettate</i>	<i>Valore dei bit di controllo</i>						
		<i>Nessun null (1)</i>	<i>Nessun zero (2)</i>	<i>Nessun negativo (4)</i>	<i>Nessun limite (8)</i>	<i>Usa tratteggio (32)</i>	<i>2D distance (64)</i>	<i>Input arbitrario (128)</i>
getint	■	■	■	■				■
getreal	■	■	■	■				■
getdist	■	■	■	■		■	■	■
getangle	■	■	■					■
getorient	■	■	■					■
getpoint	■	■			■			■
getcorner	■	■			■	■		■
getkeyword	■	■						■
entsel	■							
nentsel	■							
nentselp	■							

Specifiche delle parole chiave

L'argomento *stringa* viene interpretato secondo le seguenti regole:

Ogni parola chiave è separata da quella successiva da uno o più spazi. Per esempio "Larghezza Altezza Profondità" definisce tre parole chiave.

Ogni parola chiave può contenere soltanto lettere, numeri e trattini di sillabazione (-).

È possibile abbreviare le parole chiave in due modi diversi:

- La parte obbligatoria della parola chiave è specificata in caratteri maiuscoli, la parte rimanente in caratteri minuscoli. L'abbreviazione in maiuscolo può trovarsi in un punto qualsiasi della parola chiave (ad esempio, "Tipolinea", "eSci" o "altO").
- La parola chiave *intera* è specificata in caratteri maiuscoli, ed è seguita da una virgola, seguita a sua volta dai caratteri obbligatori (ad esempio, "TIPOLINEA,TI"). In questo caso i caratteri della parola chiave devono includere la prima lettera della stessa, quindi "ESCI,S" non è valido.

I due esempi, "Tipolinea" e "TIPOLINEA,TI", sono equivalenti: se l'utente immette **TI** (in maiuscolo o minuscolo), la parola chiave verrà identificata. L'utente può immettere caratteri che *seguono* la parte di parola chiave obbligatoria, a condizione che non siano in conflitto con la specifica. Nell'esempio l'utente potrebbe anche immettere **TLI** o **TIPL**, ma **T** non sarebbe sufficiente.

Se l'argomento *stringa* mostra interamente la parola chiave in caratteri maiuscoli o minuscoli senza virgola seguita dalla parte obbligatoria, AutoCAD riconosce la parola chiave solo se l'utente la immette per intero.

La funzione **initget** fornisce il supporto per le parole chiave localizzate. La seguente sintassi della stringa della parola chiave consente di immettere la parola chiave localizzata pur restituendo la parola chiave che non dipende dalla lingua:

```
"local1 local2 localn _indep1 indep2 indepn"
```

dove gli argomenti da *local1* a *localn* rappresentano le parole chiave localizzate, mentre quelli da *indep1* a *indepn* sono le parole chiave che non dipendono dalla lingua.

Le parole chiave localizzate devono essere sempre in numero uguale alle parole chiave che non dipendono dalla lingua, e la prima parola chiave che non dipende dalla lingua è preceduta da un segno di sottolineatura, come indicato nell'esempio seguente:

```
(initget "Abc Def _Ghi Jkl")
(getkeyword "\nImmettere un'opzione (Abc/Def): ")
```

A restituisce Ghi e **_J** restituisce Jkl.

Vedere anche

"Controllo delle condizioni delle funzioni di input utente."

Capitolo 13 -- AutoLISP: catalogo delle funzioni



inters

Trova l'intersezione di due linee.

```
(inters pt1 pt2 pt3 pt4 [su_seg])
```

Gli argomenti *pt1* e *pt2* sono i punti finali della prima linea, mentre *pt3* e *pt4* sono quelli della seconda linea. Se l'argomento *su_seg* è presente ed è nil, le linee definite dai quattro argomenti *pt* sono considerate di lunghezza infinita, e la funzione **inters** restituisce il punto in cui si intersecano, anche se quel punto si trova oltre l'estremità di una o di entrambe le linee. Se l'argomento *su_seg* viene ommesso o è diverso da nil, il punto di intersezione deve trovarsi su entrambe le linee; in caso contrario, **inters** restituisce nil. Se le due linee non si intersecano, la funzione **inters** restituisce nil.

Tutti i punti sono espressi in base all'UCS corrente. Se tutti gli argomenti *pt* sono tridimensionali, la funzione **inters** controlla se esistono intersezioni tridimensionali. Se uno dei punti è bidimensionale, **inters** proietta le linee sul corrente piano di costruzione e controlla solo le intersezioni bidimensionali.

```
(setq a '(1.0 1.0) b '(9.0 9.0))
(setq c '(4.0 1.0) d '(4.0 2.0))
(inters a b c d)           restituisce nil
(inters a b c d T)        restituisce nil
(inters a b c d nil)      restituisce (4.0 4.0)
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

itoa

Restituisce la conversione di un numero intero in una stringa.

(**itoa** *int*)

L'argomento *int* specifica un numero intero.

```
(itoa 33)                restituisce "33"
(itoa -17)               restituisce "-17"
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

lambda

Definisce una funzione anonima.

(**lambda** *argomenti espressione...*)

Usare la funzione **lambda** quando non è giustificata la definizione di una nuova funzione; rende inoltre più chiare le intenzioni del programmatore, posizionando la definizione della funzione nel punto in cui deve essere utilizzata. Questa funzione restituisce il valore della sua ultima *espressione*, ed è spesso usata insieme alle funzioni **apply** e/o **mapcar** per eseguire una funzione su un elenco.

```
(apply '(lambda (x y z)
          (* x (- y z)))
        '(5 20 14))                restituisce 30
```

and

```
(setq counter 0)
(mapcar '(lambda (x)
           (setq counter (1+ counter))
           (* x 5))
        '(2 4 -6 10.2))            restituisce (10 20 -30 51.0)
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

last

Restituisce l'ultimo elemento di una lista.

(**last** *elenco*)

```
(last '(a b c d e))            restituisce E
(last '(a b c (d e)))          restituisce (D E)
```

Come mostrato, la funzione **last** può restituire un atomo o una lista.

Nota A prima vista la funzione **last** sembra la maniera migliore per ottenere la coordinata Y di un punto. Sebbene ciò sia vero per punti bidimensionali (elenchi di due numeri reali), **last** restituirà la coordinata Z di un punto tridimensionale. Affinché la funzione utilizzi correttamente i punti bidimensionali e tridimensionali, si consiglia di utilizzare la funzione **cadr** per ottenere le coordinate Y e

la funzione **caddr** per ottenere la coordinate Z.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

length

Restituisce un numero intero che indica il numero di elementi di una lista.

```
(length elenco)

(length ' (a b c d))      restituisce 4
(length ' (a b (c d)))   restituisce 3
(length ' ())            restituisce 0
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

list

Prende un numero qualsiasi di espressioni e le combina in una lista.

```
(list espressione...)

(list 'a 'b 'c)          restituisce (A B C)
(list 'a '(b c) 'd)     restituisce (A (B C) D)
(list 3.9 6.7)          restituisce (3.9 6.7)
```

In AutoLISP, questa funzione è usata spesso per definire una variabile di punto bidimensionale o tridimensionale (lista di due o tre numeri reali).

Come alternativa alla funzione **list** è possibile racchiudere una lista tra virgolette con la funzione **quote**, se nella lista non esistono variabili o elementi non definiti. Il carattere (') è definito come la funzione **quote**.

```
' (3.9 6.7)              ha lo stesso significato di (list 3.9 6.7)
```

Ciò può essere utile per creare liste associative e definire punti.

Vedere anche

la funzione **quote**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

listp

Verifica che un elemento sia una lista.

```
(listp elem)
```

Questa funzione restituisce T se *elem* è un elenco, altrimenti restituisce nil.

```
(listp ' (a b c))      restituisce T
(listp 'a)             restituisce nil
(listp 4.343)         restituisce nil
```

Nota poiché nil è sia un atomo che una lista, la funzione **listp** restituisce T quando viene passato nil.

```
(listp nil)                restituisce T
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

load_dialog

Carica un file DCL.

```
(load_dialog file_dcl)
```

L'argomento *file_dcl* è una stringa che specifica il file DCL da caricare. Se l'argomento *file_dcl* non specifica un'estensione, *viene assunta l'estensione .dcl*. La funzione restituisce un numero intero positivo (*dcl_id*) se ha esito positivo, se non riesce ad aprire il file restituisce un numero intero negativo. *dcl_id* viene usato come gestore nelle successive chiamate alle funzioni **new_dialog** e **unload_dialog**.

La funzione **load_dialog** ricerca i file secondo il percorso di ricerca di libreria di AutoCAD.

Questa funzione è complementare alla funzione **unload_dialog**. Un'applicazione può caricare file DCL multipli con chiamate multiple alla funzione **load_dialog**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

load

Valuta le espressioni AutoLISP in un file.

```
(load nome_file [se_fallisce])
```

L'argomento *nome_file* è una stringa che rappresenta il nome del file. Se questo argomento non specifica un'estensione, *viene assunta l'estensione .lsp*. Se la funzione **load** non ha esito positivo, restituisce il valore dell'argomento *se_fallisce*; tuttavia, se tale parametro non viene fornito, un esito negativo della funzione **load** causa un errore AutoLISP. Se l'operazione ha esito positivo, **load** restituisce il valore dell'ultima espressione del file.

L'argomento *nome_file* può includere il prefisso di una directory, come */function/test1*. Su sistemi DOS inoltre è possibile specificare una lettera di unità. Una barra (/) o due barre inverse (\) sono delimitatori di directory validi. Se nella stringa *nome_file*, non viene incluso il prefisso della directory, **load** ricerca il file specificato nel percorso di libreria di AutoCAD. Una volta trovato, il file viene caricato dalla funzione **load**.

Se l'argomento **se_fallisce** è una funzione AutoLISP valida, esso viene valutato. Nella maggior parte dei casi tale argomento dovrebbe essere una stringa o un atomo; in questo modo un'applicazione AutoLISP che richiama la funzione **load** può eseguire un'azione alternativa dopo un esito negativo.

Per esempio, se il file */fred/test1.lsp* contiene

```
(defun MIA-FUNZ1 (x)
...corpo funzione...
)
(defun MIA-FUNZ2 (x)
...corpo funzione...
```

e il file *test2.lsp* non esiste,

```
(load "/fred/test1")      restituisce MIA-FUNZ2
(load "\\fred\\test1")   restituisce MIA-FUNZ2
(load "/fred/test1" "bad") restituisce MIA-FUNZ2
(load "test2" "bad")     restituisce "bad"
(load "test2")           causa un errore AutoLISP
```

La funzione **load** può essere usata dall'interno di un'altra funzione AutoLISP o anche in modo ricorrente (dall'interno del file che viene caricato).

Vedere anche

la funzione **defun** e "Gestione di simboli e funzioni."

Capitolo 13 -- AutoLISP: catalogo delle funzioni **log**

Restituisce il logaritmo naturale di un numero esprimendolo come numero reale.

(log *num*)

(log 4.5)	<i>restituisce</i>	1.50408
(log 1.22)	<i>restituisce</i>	0.198851

Capitolo 13 -- AutoLISP: catalogo delle funzioni **logand**

Restituisce il risultato di un AND logico di un elenco di numeri interi.

(logand *int int...*)

(logand 7 15 3)	<i>restituisce</i>	3
(logand 2 3 15)	<i>restituisce</i>	2
(logand 8 3 4)	<i>restituisce</i>	0

Capitolo 13 -- AutoLISP: catalogo delle funzioni **logior**

Restituisce il risultato di un operatore logico a livello bit OR inclusivo di un elenco di numeri interi.

(logior *int int...*)

(logior 1 2 4)	<i>restituisce</i>	7
(logior 9 3)	<i>restituisce</i>	11

Capitolo 13 -- AutoLISP: catalogo delle funzioni **lsh**

Restituisce lo spostamento logico di un numero intero per il numero di bit specificati.

(lsh *int bit_num*)

Se l'argomento *bit_num* è positivo, *int* viene spostato a sinistra, se è negativo viene spostato a destra. In entrambi i casi i bit zero vengono spostati verso l'interno e quelli spostati verso l'esterno vengono eliminati. Il valore restituito è positivo se il bit significativo (n. 31) contiene uno 0 dopo lo spostamento, altrimenti esso è negativo.

(lsh 2 1)	<i>restituisce</i>	4
(lsh 2 1)	<i>restituisce</i>	1
(lsh 40 2)	<i>restituisce</i>	160

Capitolo 13 -- AutoLISP: catalogo delle funzioni

mapcar

Restituisce una lista come risultato dell'esecuzione di una funzione con i singoli elementi di una lista o più liste fornite come argomenti alla funzione.

(mapcar *funzione lista1* . . . *listan*)

Il numero di liste deve corrispondere al numero di argomenti richiesti dalla *funzione*.

```
(setq a 10 b 20 c 30)
(mapcar '1+ (list a b c))           restituisce (11 21 31)
```

è equivalente a

```
(1+ a)
(1+ b)
(1+ c)
```

tranne per il fatto che la funzione **mapcar** restituisce una lista di risultati.

La funzione **lambda** può specificare che una funzione anonima deve essere eseguita dalla funzione **mapcar**. Ciò è utile quando alcuni argomenti della funzione sono costanti o sono forniti in altro modo.

```
(mapcar      '(lambda (x)
              (+ x 3)
              )
              '(10 20 30))
           restituisce (13 23 33)
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

max

Restituisce il maggiore dei numeri dati.

(max *numero numero* . . .)

```
(max 4.07 -144)           restituisce 4.07
(max -88 19 5 2)         restituisce 19
(max 2.1 4 8)            restituisce 8.0
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

mem

Visualizza lo stato corrente della memoria AutoLISP.

(mem)

Visualizza lo stato corrente della memoria di AutoLISP, e restituisce nil. Tale funzione visualizza le seguenti informazioni:

Nodi è il numero totale di nodi assegnati fino a quel momento, che dovrebbe corrispondere alla dimensione del segmento del nodo moltiplicato il numero dei segmenti.

Nodi liberi è il numero di nodi correntemente sull'elenco *liberi* come risultato di una pulizia di nodi.

Segmenti è il numero dei segmenti dei nodi allocati.

Allocare è la dimensione del segmento corrente.

Raccolta è il conteggio delle raccolte di pulizia dei nodi, sia essa automatica che forzata.

Vedere anche

capitolo 15, "Gestione della memoria".

Capitolo 13 -- AutoLISP: catalogo delle funzioni

member

Cerca in una lista le occorrenze di un'espressione e restituisce il resto della lista, a partire dalla prima occorrenza dell'espressione.

(**member** *espr elenco*)

Se non vengono trovate occorrenze di *espressione* in *lista*, la funzione **member** restituisce nil.

```
(member 'c '(a b c d e))           restituisce (C D E)
(member 'q '(a b c d e))           restituisce nil
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

menucmd

Invia comandi di menu o imposta e richiama lo stato di voci di menu.

(**menucmd** *stringa*)

L'argomento *stringa* specifica la sezione di menu (e possibilmente l'elemento di menu) seguito dal sottomenu o dal comando di menu. L'argomento *stringa* presenta i parametri seguenti:

```
"area_menu=valore"
```

I valori permessi di *area_menu*, indicati nella tabella seguente, sono uguali a quelli presenti nei riferimenti del sottomenu del file di menu. Per ulteriori informazioni, vedere "Riferimento di menu a discesa ed a cursore."

Valori di *stringa* per area di menu

Stringa area di menu	Sezione di menu
B1-B4	Per menu BUTTONS da 1 a 4
A1-A4	Per menu AUX da 1 a 4
P0-P16	Per menu a discesa (POP) da 0 a 16
I	Per menu gruppo di immagini
S	Per menu SCREEN
T1-T4	Per menu TABLET da 1 a 4
M	Per espressioni di stringa DIESEL
Ggruppomenu.tagnome	Specifica un nome di gruppo e un contrassegno di nome.

Il parametro *valore* specifica il valore assegnato a *area_menu*.

La funzione **menucmd** può passare da un sottomenu all'altro in un menu AutoCAD e può forzare la visualizzazione dei menu. In questo modo i programmi AutoLISP possono usare menu a gruppo di immagini e visualizzare altri menu da cui l'utente può effettuare delle selezioni. I programmi AutoLISP possono inoltre abilitare, disabilitare e contrassegnare gli elementi di menu.

Il seguente codice visualizza il menu a gruppo di immagini **MOREICONS**.

```
(menucmd "I=moreicons")           Carica il menu a gruppo di immagini MOREICONS
(menucmd "I=*")                   Visualizza il menu
```

Il seguente codice controlla lo stato del terzo elemento di menu del menu a discesa **POP11**. Se l'elemento è abilitato, la funzione

menucmd lo disabilita.

```
(setq s (menucmd "P11.3=?"))
(if (= s "")
  (menucmd "P11.3=~")
)
```

*Controlla lo stato dell'elemento di menu
Se lo stato è una stringa vuota,
disabilita l'elemento di menu*

Il codice precedente non è sicuro. Oltre ad essere abilitati o disabilitati, gli elementi di menu possono essere contrassegnati. Il codice (menucmd "P11.3=?") potrebbe restituire "!.", indicando che quell'elemento di menu è stato controllato. In questo modo, il codice presume che l'elemento sia disabilitato e continua senza disabilitarlo. Se il codice comprendeva una chiamata alla funzione **wcmatch**, potrebbe controllare lo stato per un'occorrenza del carattere ~ (tilde) e successivamente intraprendere l'azione appropriata.

La funzione **menucmd** consente inoltre ai programmi AutoLISP di usufruire del linguaggio per DIESEL. Alcune azioni possono essere eseguite più semplicemente con DIESEL piuttosto che con il codice AutoLISP equivalente. Il seguente codice restituisce una stringa contenente il giorno e la data correnti:

```
(menucmd "M=$(edtime,$(getvar,date),DDDD\\",\\\" D MONTH YYYY) ")
restituisce "Domenica, 16 Luglio 1995"
```

Vedere anche

il capitolo 6, "Interfacce di programmazione", per ulteriori informazioni sull'uso di AutoLISP per accedere allo stato delle etichette dei menu, e il capitolo 5, "Linguaggio DIESEL e configurazione della riga di stato", per informazioni sull'uso di DIESEL

Capitolo 13 -- AutoLISP: catalogo delle funzioni

gruppomenu

Verifica che sia caricato un gruppo menu

```
(gruppomenu nomegruppo)
```

L'argomento *nomegruppo* è una stringa che specifica il nome del gruppo menu. Se corrisponde a un gruppo menu caricato, la funzione restituisce la stringa *nomegruppo*; in caso contrario, restituisce nil.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

min

Restituisce il minore dei numeri dati.

```
(min numero numero...)

(min 683 -10.0)      restituisce -10.0
(min 73 2 48 5)     restituisce 2
(min 2 4 6.7)       restituisce 2.0
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

minusp

Verifica che il numero sia negativo.

```
(minusp numero)
```

Restituisce T se *numero* è negativo, altrimenti restituisce nil.

```
(minusp -1)           restituisce T
(minusp -4.293)       restituisce T
(minusp 830.2)        restituisce nil
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

mode_tile

Imposta la modalità di una casella di una finestra di dialogo.

(mode_tile *chiave modalità*)

L'argomento *chiave* è una stringa che specifica la casella e riconosce i caratteri maiuscoli e minuscoli. L'argomento *modalità* è un valore intero; i valori di tale argomento sono descritti nella seguente tabella.

Valori argomento *modalità*

Valore	Descrizione
0	Abilita la casella.
1	Disabilita la casella.
2	Imposta l'evidenziazione sulla casella.
3	Seleziona il contenuto della casella di modifica.
4	Abilita e disabilita l'evidenziazione dell'immagine.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

namedobjdict

Restituisce il nome dell'entità del dizionario dell'oggetto del disegno corrente, che è la base di tutti gli oggetti non grafici del disegno.

(namedobjdict)

Utilizzando il nome restituito da questa funzione e le funzioni di accesso al dizionario, un'applicazione può accedere agli oggetti non grafici del disegno.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

nentsel

Richiede all'utente di selezionare un oggetto (entità) specificando un punto e fornisce l'accesso ai dati di definizione contenuti in un oggetto complesso.

(nentsel [*msg_richiesta*])

L'argomento *msg_richiesta* è una stringa che deve essere visualizzata come messaggio di richiesta. Se viene omissso, viene richiesto di selezionare un oggetto.

La funzione **nentsel** richiede all'utente di selezionare un oggetto. La modalità corrente di Osnap viene ignorata a meno che l'utente non la richieda espressamente. Per fornire un supporto aggiuntivo alla riga di comando, **nentsel** rispetta le parole chiave definite da una precedente chiamata alla funzione **initget**.

Quando l'oggetto selezionato non è complesso (per esempio una polilinea o un blocco), **nentsel** restituisce le stesse informazioni di **entsel**. Se comunque l'oggetto selezionato è una polilinea, **nentsel** restituisce un elenco contenente il nome delle sottoentità (vertice) ed il punto di selezione, simile all'elenco restituito da **entsel**, tranne per il fatto che viene restituito il nome del vertice selezionato invece dell'intestazione della polilinea. La funzione **nentsel** restituisce sempre il vertice iniziale del segmento di polilinea selezionato. Selezionando il terzo segmento di una polilinea, per esempio, viene restituito il terzo vertice. La sottoentità *Seqend on* viene mai restituita da **nentsel** per una polilinea.

Nota Una polilinea ottimizzata (entità *lwpolyline*) viene definita nel database dei disegni come singola entità e non contiene sottoentità.

Selezionando un attributo all'interno di un riferimento di un blocco viene restituito il nome dell'attributo e il punto di selezione. Quando l'oggetto selezionato è il componente del riferimento di un blocco diverso da un attributo, la funzione **nentsel** restituisce un elenco contenente quattro elementi.

Il primo elemento dell'elenco restituito dalla selezione di un oggetto all'interno di un blocco è il nome dell'entità selezionata. Il secondo elemento è un elenco contenente le coordinate del punto usate per selezionare l'oggetto.

Il terzo elemento è un modello di matrice di trasformazione globale, che è un elenco composto da quattro elenchi secondari, ognuno dei quali contiene un gruppo di coordinate. Questa matrice può essere usata per trasformare i punti dei dati di definizione dell'entità di un sistema di coordinate interno, Model Coordinate System (MCS), al sistema di coordinate globale. Il punto di inserimento del blocco contenente l'entità selezionata definisce l'origine dell'MCS. L'orientamento del sistema di coordinate utente quando il blocco viene creato determina la direzione degli assi MCS.

Il quarto elemento è un elenco contenente il nome dell'entità del blocco contenente l'oggetto selezionato. Se tale oggetto è in un blocco nidificato (un blocco all'interno di un blocco), l'elenco contiene anche i nomi delle entità di tutti i blocchi in cui è nidificato l'oggetto selezionato, a partire dal blocco più interno e proseguendo fino al nome di quello inserito nel disegno.

```

(<Nome entità: ename1>      Nome dell'entità

      (Px Py Pz)              Punto di selezione
      ( (X0 Y0 Z0)           Matrice di trasformazione globale
        (X1 Y1 Z1)
        (X2 Y2 Z2)
        (X3 Y3 Z3)
      )
      (<Nome entità: ename2>  Nome del blocco nidificato più interno
        .                      contenente l'entità selezionata
        .
        .
      (<Nome entità: enamen>) Nome del blocco più esterno
        .                      contenente l'entità selezionata
      )
    
```

Una volta ottenuti il nome dell'entità e la matrice di trasformazione globale, è possibile trasformare i punti dei dati di definizione dell'entità da MCS a WCS. Usare le funzioni **entget** ed **assoc** sul nome dell'entità per ottenere i punti di definizione espressi nelle coordinate MCS. La matrice di trasformazione globale restituita dalla funzione **nentsel** ha lo stesso scopo di quella restituita dalla funzione **nentselp**, ma è una matrice 4x3, passata come una serie di quattro punti che usa la convenzione secondo la quale un punto è una riga e non una colonna. La trasformazione viene descritta dalla seguente moltiplicazione di matrice:

$$\begin{bmatrix} X' & Y' & Z' & 1.0 \end{bmatrix} = \begin{bmatrix} X & Y & Z & 1.0 \end{bmatrix} \cdot \begin{bmatrix} M_{00} & M_{01} & M_{02} \\ M_{10} & M_{11} & M_{12} \\ M_{20} & M_{21} & M_{22} \\ M_{30} & M_{31} & M_{32} \end{bmatrix}$$

Le equazioni per derivare le nuove coordinate sono:

$$\begin{aligned}
 X' &= XM_{00} + YM_{10} + ZM_{20} + M_{30} \\
 Y' &= XM_{01} + YM_{11} + ZM_{21} + M_{31} \\
 Z' &= XM_{02} + YM_{12} + ZM_{22} + M_{32}
 \end{aligned}$$

M_{ij}, dove 0 *i, j* 2, sono le coordinate della Matrice di trasformazione globale; X, Y, Z è il punto di dati di definizione dell'entità

espresso in coordinate MCS, e X', Y', Z' è il punto risultante di dati di definizione di entità espresso in coordinate WCS.

Nota questa è l'unica funzione AutoLISP che usa una matrice di questo tipo; la funzione **nentselp** restituisce una matrice simile a quelle usate da altre funzioni AutoLISP e ADSRX.

Vedere anche

"Funzioni riguardanti i nomi delle entità" e le funzioni **entsel** e **if**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

nentselp

Fornisce una funzionalità simile a quella della funzione **nentsel** senza che sia necessario l'input dell'utente.

(nentselp [msg_richiesta] [pt])

Oltre all'argomento opzionale *msg_richiesta*, la funzione **nentselp** accetta come argomento opzionale un punto di selezione; ciò consente la selezione di un oggetto senza l'input da parte dell'utente. La funzione **nentselp** restituisce una matrice di trasformazione 4x4, definita nel seguente modo:

$$\begin{bmatrix} M_{00} & M_{01} & M_{02} & M_{03} \\ M_{10} & M_{11} & M_{12} & M_{13} \\ M_{20} & M_{21} & M_{22} & M_{23} \\ M_{30} & M_{31} & M_{32} & M_{33} \end{bmatrix}$$

Le prime tre colonne della matrice specificano la scala e la rotazione. La quarta è un vettore di trasformazione.

Le funzioni che utilizzano questo tipo di matrici considerano un punto come un vettore di colonna di quota 4. Il punto è espresso in *coordinate omogenee*, dove il quarto elemento del vettore del punto è un *fattore di scala* generalmente impostato su 1.0. L'ultima riga della matrice, il vettore [M30 M31 M32 M33], ha il valore nominale [0 0 0 1]; viene ignorato dalle funzioni che usano questo formato di matrice. Secondo questa convenzione, ad applicare una trasformazione a un punto è una moltiplicazione di matrice che appare come segue:

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1.0 \end{bmatrix} = \begin{bmatrix} M_{00} & M_{01} & M_{02} & M_{03} \\ M_{10} & M_{11} & M_{12} & M_{13} \\ M_{20} & M_{21} & M_{22} & M_{23} \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1.0 \end{bmatrix}$$

Questa moltiplicazione restituisce le singole coordinate del punto come segue:

$$\begin{aligned} X' &= XM_{00} + YM_{01} + ZM_{02} + M_{03}(1.0) \\ Y''' &= XM_{10} + YM_{11} + ZM_{12} + M_{13}(1.0) \\ Z' &= XM_{20} + YM_{21} + ZM_{22} + M_{23}(1.0) \end{aligned}$$

Come mostrato da queste equazioni, il fattore di scala e l'ultima riga della matrice non hanno alcun effetto e vengono ignorate.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

new_dialog

Attiva e visualizza una nuova finestra di dialogo; è possibile specificare anche un'azione di default.

`(new_dialog nomedlg id_dcl [azione [schermo_pt])`

L'argomento *nomedlg* è una stringa che indica la finestra di dialogo mentre *id_dcl* identifica il file DCL (è necessario ottenere questo valore dalla funzione **load_dialog**).

L'argomento *azione*, che deve essere indicato se viene specificato *pt-schermo*, è una stringa contenente una espressione AutoLISP da utilizzare come azione di default. Se non si intende definire un'azione di default, passare *azione* come stringa vuota ("").

L'argomento *pt-schermo* rappresenta un punto bidimensionale che indica la posizione X,Y della finestra di dialogo sullo schermo. Il punto generalmente individua l'angolo in alto a sinistra della finestra di dialogo, ma tale posizione dipende dalla piattaforma ed anche dall'unità di misura in cui la posizione viene specificata. Se viene passato il punto '(-1 -1), la finestra di dialogo viene visualizzata nella posizione di default, cioè nella parte centrale dello schermo grafico di AutoCAD.

Se l'esito è positivo, **new_dialog** restituisce T; altrimenti restituisce nil.

È necessario che all'interno dell'applicazione, la funzione **new_dialog** venga richiamata prima di **start_dialog**. L'inizializzazione completa della finestra di dialogo, che comprende l'impostazione dei valori delle caselle, la creazione delle immagini o degli elenchi per le caselle di riepilogo e l'associazione di azioni a determinate caselle (con l'utilizzo della funzione **action_tile**) deve aver luogo dopo la funzione **new_dialog** e prima della funzione **start_dialog**.

L'azione di default viene valutata quando l'utente seleziona una casella attiva alla quale non è stata assegnata un'azione o una funzione di richiamo in modo esplicito tramite la funzione **action_tile** o in DCL.

Nota Verificare sempre lo stato restituito da **new_dialog**. Richiamare la funzione **start_dialog** quando **new_dialog** ha avuto esito negativo potrebbe provocare risultati imprevedibili.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

not

Verifica se ad un elemento è stato assegnato il valore nil.

`(not elem)`

Restituisce T se ad *elem* è stato assegnato il valore nil, altrimenti restituisce nil.

```
(setq a 123 b "string" c nil)
(not a)           restituisce nil
(not b)           restituisce nil
(not c)           restituisce T
(not c)           restituisce T
```

Di solito, la funzione **null** viene usata per gli elenchi, mentre **not** viene utilizzata per altri tipi di dati insieme ad alcuni tipi di funzioni di controllo.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

nth

Restituisce l'elemento n di una lista.

`(nth n elenco)`

L'argomento *n* è il numero dell'elemento da restituire (zero è il primo elemento). Se *n* è superiore rispetto al numero dell'elemento

maggiore contenuto in *lista*, *nth* restituisce nil.

```
(nth 3 '(a b c d e))      restituisce D
(nth 0 '(a b c d e))      restituisce A
(nth 5 '(a b c d e))      restituisce nil
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

null

Verifica se un elemento è stato limitato solo al valore nil.

(null *elem*)

Restituisce T se ad *elem* è stato assegnato il valore nil, altrimenti restituisce nil.

```
(setq a 123 b "string" c nil)
(null a)          restituisce nil
(null b)          restituisce nil
(null c)          restituisce T
(null '())        restituisce T
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

numberp

Verifica se un elemento è un numero reale o intero.

(numberp *elem*)

Restituisce T se *elem* è un numero reale o intero, altrimenti restituisce nil.

```
(setq a 123 b 'a)
(numberp 4)          restituisce T
(numberp 3.8348)     restituisce T
(numberp "Salve")    restituisce nil
(numberp a)          restituisce T
(numberp b)          restituisce nil
(numberp (eval b))   restituisce T
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

open

Apri un file al quale possono accedere le funzioni AutoLISP di I/O.

(open *nomefile modalità*)

L'argomento *nomefile* è una stringa che indica il nome e l'estensione del file da aprire. L'argomento *modalità* è il flag di lettura/scrittura. Deve essere una stringa contenente una sola lettera minuscola. Nella tabella riportata di seguito vengono descritte le lettere valide per *modalità*.

Opzioni per la *modalità* della funzione *open*

Modalità di apertura	<i>Descrizione</i>
"r"	<i>Apertura per la lettura.</i> Se <i>nomefile</i> non esiste, open restituisce nil.
"w"	<i>Apertura per la scrittura.</i> Se <i>nomefile</i> non esiste, viene creato ed aperto un nuovo file. If Se <i>nomefile</i> è un file esistente, i dati in esso contenuti verranno sostituiti.
"a"	<i>Apertura per l'aggiunta.</i> Se <i>nomefile</i> non esiste, viene creato ed aperto un nuovo file. Se <i>nomefile</i> è un file esistente, tale file viene aperto ed il puntatore posizionato alla fine dei dati in esso contenuti, in modo tale che i nuovi dati inseriti nel file vengano aggiunti a quelli esistenti.

I dati passati ad un file aperto vengono in effetti scritti solo dopo che il file è stato chiuso con la funzione close.

Nota sui sistemi DOS, alcuni programmi ed editor di testo inseriscono nei file di testo un contrassegno di fine file (CTRL Z, codice decimale ASCII 26) alla fine del testo. Durante la lettura di un file di testo, se viene incontrato il contrassegno CTRL Z, DOS restituisce lo stato di fine file anche se tale contrassegno è seguito da ulteriori dati. Se si intende utilizzare la modalità "a" del comando APRI per aggiungere dati ai file prodotti da un altro programma, assicurarsi che il programma non abbia inserito i contrassegni CTRL Z alla fine dei file di testo.

La funzione **open** restituisce un descrittore di file che può essere utilizzato da altre funzioni di I/O. Tale descrittore deve essere assegnato ad un simbolo che utilizza la funzione **setq**.

```
(setq a (open "file.est" "r"))
```

Negli esempi riportati di seguito, si suppone che i file elencati *non* esistano.

```
(setq f (open "nuovo.tst" "w"))      restituisce <File #nnn>
(setq f (open "notipo.fil" "r"))     restituisce nil
(setq f (open "filereg" "a"))        restituisce <File #nnn>
```

L'argomento *nomefile* può includere il prefisso di una directory, come in */test/func3*. Sui sistemi DOS, è consentito includere anche la lettera dell'unità ed è possibile utilizzare la barra inversa (\) invece della barra (/), ma per ottenere una barra inversa in una stringa è necessario utilizzare due barre inverse (\\).

```
(setq f (open "/x/nuovo.tst" "w"))  restituisce <File #nnn>
(setq f (open "notipo.fil" "r"))     restituisce nil
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni



or

Restituisce l'OR logico di un elenco di espressioni.

(or espressione...)

La funzione **or** esegue la valutazione delle espressioni da sinistra a destra, ricercando una espressione che non sia nil. Se ne viene incontrata una, la funzione **or** non prosegue la valutazione e restituisce T. Se tutte le espressioni sono nil, **or** restituisce nil.

```
(or nil 45 '())      restituisce T
(or nil '())         restituisce nil
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni



osnap

Restituisce un punto tridimensionale risultante dall'applicazione della modalità snap ad oggetto ad un punto specifico.

(osnap pt modalità)

L'argomento *modalità* è una stringa che consiste di uno o più identificatori validi di snap ad oggetto come *mez*, *cen* e *così* via, separati da virgole.

```
(setq pt1 (getpoint))
```

```
(setq pt2 (osnap pt1 "cen"))
(setq pt3 (osnap pt1 "fin,int"))
```

Il punto restituito da **osnap** dipende dalla vista tridimensionale corrente e dall'impostazione della variabile di sistema APERTURE.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

polar

Restituisce il punto tridimensionale del sistema di coordinate utente all'angolo indicato ed alla distanza specificata rispetto ad un punto.

(polar *pt angolo distanza*)

L'argomento *angolo* è espresso in radianti relativi all'asse X, con angoli crescenti in senso antiorario. Sebbene *pt* possa essere un punto tridimensionale, *angolo* è sempre relativo al piano della costruzione corrente.

```
(polar '(1 1 3.5) 0.785398 1.414214) restituisce (2.0 2.0 3.5)
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

prin1

Stampa un'espressione sulla riga di comando oppure scrive un'espressione in un file aperto.

(prin1 [*espr [desc-file]*])

Non è necessario che l'argomento *espr* sia una stringa. Se *desc-file* è presente e si tratta di un descrittore di un file aperto per la scrittura, *espr* viene scritta nel file nello stesso modo in cui appare sullo schermo. Viene stampata solo l'*espr* specificata; non vengono inclusi spazi o linee nuove.

```
(setq a 123 b '(a))
(prin1 'a)           stampa A           e restituisce A
(prin1 a)           stampa 123         e restituisce 123
(prin1 b)           stampa (A)         e restituisce (A)
(prin1 "Ciao")      stampa "Ciao"      e restituisce "Ciao"
```

Ognuno dei precedenti esempi viene visualizzato sullo schermo poiché non è stato specificato alcun *desc-file*. Presupponendo che *f* sia un descrittore di file valido per un file aperto per la scrittura, la funzione:

```
(prin1 "Ciao" f)
```

scriverà "Ciao" nel file specificato e restituirà "Ciao".

Se *espr* è una stringa contenente caratteri di controllo, **prin1** espande questi caratteri con una \ iniziale, come mostrato nella tabella riportata di seguito.

Codici di controllo

Codice	Descrizione
\\	Carattere \
\"	Carattere "
\e	Carattere Escape
\n	Carattere di riga nuova
\r	Carattere di ritorno
\t	Carattere di tabulazione
\nnn	Character whose octal code is nnn
\U+XXXX	Unicode sequence
\M+NXXXX	Unicode sequence

Negli esempi riportati di seguito viene descritto come utilizzare i caratteri di controllo.

```
(prin1 (chr 2))      stampa "\002" e restituisce "\002"
(prin1 (chr 10))    stampa "\n"   e restituisce "\n"
```

La funzione **prin1** può essere richiamata senza alcun argomento ed in tal caso restituisce e stampa la stringa nulla. Se **prin1** viene usata senza argomenti come ultima espressione in una funzione definita dall'utente, quando la funzione viene completata, verrà stampata solo una riga bianca, consentendo l'uscita dall'applicazione senza apportare modifiche.

Vedere anche

"Visualizzazione di messaggi sulla riga di comando."

Capitolo 13 -- AutoLISP: catalogo delle funzioni

princ

Stampa un'espressione sulla riga di comando o scrive un'espressione in un file aperto.

```
(princ [espr [desc-file]])
```

Questa funzione è uguale a **prin1**, tranne per il fatto che il carattere di controllo in *espr* viene stampato senza espansione.

In generale, **prin1** è stata progettata per stampare espressioni in modo che siano compatibili con **load**, mentre **princ** le stampa in modo che siano leggibili per funzioni come, ad esempio, **read-line**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

print

Stampa un'espressione sulla riga di comando o scrive un'espressione in un file aperto.

```
(print [espr [desc-file]])
```

Questa funzione è uguale a **prin1**, tranne per il fatto che stampa un carattere di riga nuova prima di *espr*, ed uno spazio dopo *espr*.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

progn

Valuta ogni espressione in modo sequenziale e restituisce il valore dell'ultima espressione.

```
(progn [espressione] . . .)
```

La funzione **progn** può essere utilizzata per valutare diverse espressioni quando invece ne era prevista una solamente.

```
(if (= a b)
  (progn
    (princ "\nA = B ")
    (setq a (+ a 10) b (- b 10))
  )
)
```

La funzione **if** esegue di solito la valutazione di un'espressione *then* se la verifica dell'espressione restituisce un risultato diverso da nil. Nell'esempio riportato precedentemente, la funzione **progn** viene utilizzata per valutare due espressioni invece di una.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

prompt

Visualizza una stringa nell'area del messaggio di richiesta dello schermo.

(*prompt msg*)

Nelle configurazioni di AutoCAD a schermo doppio, **prompt** visualizza *msg_richiesta* su entrambi gli schermi; questa funzione è quindi preferibile a **princ**.

(prompt "Valore nuovo: ") *visualizza* Valore nuovo: *sullo schermo(i)*

La funzione **prompt** restituisce nil.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

quit

Forza l'uscita dall'applicazione corrente.

(*quit*)

Quando richiamata, la funzione **quit** restituisce il messaggio di errore *esci./continua* e ritorna alla riga di comando di AutoCAD.

Vedere anche

la funzione **exit**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

quote

Restituisce un'espressione senza valutarla.

(*quote espressione*)

Questa funzione può anche essere scritta nel seguente modo.

'expr		
(quote a)	<i>restituisce</i>	A
(quote cat)	<i>restituisce</i>	CAT
(quote (a b))	<i>restituisce</i>	(A B)
'a	<i>restituisce</i>	A
'cat	<i>restituisce</i>	CAT
'(a b)	<i>restituisce</i>	(A B)

Gli ultimi tre esempi non potranno essere eseguiti se immessi direttamente dalla tastiera in risposta ad un messaggio di richiesta di AutoCAD. Tenere presente che input di questo tipo devono iniziare con l'espressione del carattere (o ! per essere riconosciuti come espressione AutoLISP.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

read

Restituisce il primo elenco o l'atomo restituito da una stringa.

(read [stringa])

L'argomento *stringa* non può contenere spazi vuoti, a meno che non siano posizionati all'interno di un elenco o di una stringa. La funzione **read** restituisce gli argomenti convertiti nel corrispondente tipo di dati:

(read "ciao")	restituisce l'atomo	CIAO
(read "ciao a tutti")	restituisce l'atomo	CIAO
(read "\"Salve a tutti\"")	restituisce la stringa	"Salveatutti"
(read "(a b c)")	restituisce l'elenco	(A B C)
(read "(a b c) (d)")	restituisce l'elenco	(A B C)
(read "1.2300")	restituisce il numero reale	1.23
(read "87")	restituisce il numero intero	87
(read "87 3.2")	restituisce il numero intero	87

Capitolo 13 -- AutoLISP: catalogo delle funzioni

read-char

Restituisce il codice decimale ASCII che rappresenta il carattere letto dal buffer di input della tastiera o da un file aperto.

(read-char [desc-file])

Se *desc-file* non viene specificato e nel buffer di input della tastiera non è contenuto alcun carattere, **read-char** aspetta l'immissione dalla tastiera, seguita da RETURN). Ad esempio, presupponendo che il buffer di input della tastiera sia vuoto, la funzione:

```
(read-char)
```

attende che venga digitato qualcosa. Se viene immesso ABC seguito da RETURN, **read-char** restituisce 65, il codice decimale ASCII per la lettera A. Le successive tre chiamate alla funzione **read-char** restituiscono rispettivamente 66, 67 e 10 (nuova riga). Se viene eseguita un'altra funzione **read-char**, essa resterà in attesa dell'input successivo.

I vari sistemi operativi sotto i quali è possibile eseguire AutoCAD fanno uso di convenzioni diverse per segnalare la fine di un riga in un file di testo ASCII. I sistemi UNIX, ad esempio, utilizzano un unico carattere di riga nuova (nuova riga [LF], codice ASCII 10), mentre i sistemi DOS utilizzano una coppia di caratteri (ritorno a capo [CR]/LF, codici ASCII 13 e 10) per lo stesso scopo. Per facilitare lo sviluppo di programmi AutoLISP, **read-char** accetta tutte queste convenzioni, restituendo un unico carattere di riga nuova (codice ASCII 10) ogni volta che viene rilevato un carattere di fine riga o di fine sequenza.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

read-line

Legge una stringa dalla tastiera o da un file aperto.

(read-line [desc-file])

Se **read-line** rileva la fine del file, restituisce nil; altrimenti restituisce la stringa in corso di lettura.

Ad esempio, presupponendo che f sia un puntatore di file aperto valido, la funzione:

```
(read-line f)
```

restituisce la riga di input successiva presente nel file oppure nil se è stata raggiunta la fine del file.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

redraw

Ridisegna la finestra corrente o un oggetto specificato (entità) nella finestra corrente.

(redraw [nome_ent [modalità]])

L'effetto della funzione **redraw** dipende dagli argomenti specificati. Se viene richiamata senza argomenti, la funzione ridisegna la finestra corrente. Se invece viene richiamata con l'argomento *nome_ent* (nome dell'entità), la funzione ridisegna l'entità specificata. L'argomento *modalità* è un valore intero che controlla il modo in cui l'entità viene ridisegnata.

Modalità per la funzione redraw

Modalità redraw	Azione
1	Ridisegna l'entità.
2	Annulla il disegno ell'entità (ne annulla la visualizzazione.
3	Evidenzia l'entità.
4	Annulla l'evidenziazione dell'entità

Se *nome_elem* è l'intestazione di una entità complessa (una polilinea o il riferimento ad un blocco con attributi), esso elabora l'entità principale e tutte le sue sottoentità se l'argomento *modalità* è positivo. Se l'argomento *modalità* è negativo, **redraw** ha effetto solo sull'entità dell'intestazione.

La funzione **redraw** restituisce sempre nil.

Nota L'uso dell'evidenziazione dell'entità (modalità 3) deve essere bilanciato dall'annullamento dell'evidenziazione (modalità 4).

Il comando RIDIS non ha effetto sulle entità nascoste e evidenziate, mentre il comando RIGEN rivisualizza le entità nel loro formato normale.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

regapp

Memorizza il nome di un'applicazione in cui il disegno AutoCAD corrente utilizzerà i dati estesi dell'oggetto.

(regapp applicazione)

L'argomento *applicazione* è una stringa composta al massimo da 31 caratteri, conforme alle convenzioni per l'assegnazione dei nomi ai simboli. Il nome di un'applicazione può contenere lettere, cifre ed i caratteri speciali, quali il segno del dollaro (\$), il trattino (-) ed il carattere di sottolineatura (_), ma non può contenere spazi vuoti. Le lettere del nome vengono convertite in lettere maiuscole.

Se è già stata memorizzata un'applicazione con lo stesso nome, questa funzione restituisce returns nil; altrimenti restituisce il nome dell'applicazione.

Se la memorizzazione ha esito positivo, il nome dell'applicazione viene inserito nella tabella di simboli APPID nella quale sono elencate le applicazioni che utilizzano i dati estesi nel disegno.

```
(regapp "ADESK_4153322344")
(regapp "DESIGNER-v2.1-124753")
```

Nota Si consiglia di selezionare un nome univoco per l'applicazione. Per accertarsi di ciò, adottare uno schema per l'assegnazione dei nomi che utilizzi il nome della società o del prodotto ed un numero univoco, come il numero di telefono o l'ora e la data corrente. Il numero della versione del prodotto può essere incluso nel nome dell'applicazione o memorizzato dall'applicazione stessa in un campo particolare in cui può essere inserito un numero reale o intero come, ad esempio, (1040 2.1).

Capitolo 13 -- AutoLISP: catalogo delle funzioni

rem

Divide il primo argomento per il prodotto degli argomenti restanti e restituisce il resto.

(rem numero numero...)

Se vengono forniti più di due numeri, **rem** divide il primo per il secondo; il risultato della divisione viene poi diviso per il terzo numero, e così via.

(rem 42 12)	<i>restituisce</i>	6
(rem 12.0 16)	<i>restituisce</i>	12.0
(rem 26 7)	<i>restituisce</i>	5
(rem 5 2)	<i>restituisce</i>	1
(rem 26 7 2)	<i>restituisce</i>	1

Capitolo 13 -- AutoLISP: catalogo delle funzioni

repeat

Valuta ogni espressione un numero specifico di volte e restituisce il valore dell'ultima espressione.

(repeat int espr...)

L'argomento *int* deve essere positivo.

```
(setq a 10 b 100)
(repeat 4
  (setq a (+ a 10))
  (setq b (+ b 100))
)
```

Imposta a su 50, imposta b su 500 e restituisce 500

Capitolo 13 -- AutoLISP: catalogo delle funzioni

reverse

Restituisce una lista con gli elementi invertiti.

(reverse elenco)

(reverse '(a) b c)) *restituisce* (C B (A))

Capitolo 13 -- AutoLISP: catalogo delle funzioni

rtos

Converte un numero in una stringa.

(rtos numero [modalità [precisione]])

La funzione **rtos** restituisce una stringa che è la rappresentazione del *numero* conforme alle impostazioni di *modalità*, *precisione* e delle variabili di sistema UNITMODE e DIMZIN. Gli argomenti *modalità* e *precisione* sono numeri interi che selezionano la modalità

e la precisione delle unità lineari, come mostrato nella tabella riportata di seguito.

Valore delle unità lineari

Valore della modalità	Formato della stringa
1	Scientifica
2	Decimale
3	Ingegneristica (piedi e pollici decimali)
4	Architettonica (piedi e pollici frazionari)
5	Frazionario

Gli argomenti *modalità* e *precisione* corrispondono alle variabili di LUNITS e LUPREC. Se si omettono questi argomenti, **rtos** utilizza le impostazioni correnti di LUNITS e LUPREC. La variabile UNITMODE determina la stringa restituita quando vengono selezionate le unità di ingegneria, architettónica o frazionale (il valore di *modalità* è 3, 4 o 5).

Vedere anche

"Conversioni di stringhe," per la continuazione della descrizione della funzione **rtos**

Capitolo 13 -- AutoLISP: catalogo delle funzioni

set

Imposta il valore del nome di un simbolo racchiuso tra virgolette ad una espressione

(set *simb espr*)

Restituisce il valore di una espressione.

```
(set 'a 5.0)           restituisce 5.0 ed imposta il simbolo A
(set (quote b) 'a)    restituisce A ed imposta il simbolo B
```

Se **set** viene utilizzata con un nome di simbolo non racchiuso tra virgolette, la funzione può assegnare indirettamente un valore nuovo ad un altro simbolo. Ad esempio, riesaminando gli esempi precedenti, la funzione:

```
(set b 640)           restituisce 640
```

ed assegna il valore 640 al simbolo a, poiché quest'ultimo è il contenuto del simbolo b.

Vedere anche

la funzione **setq**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

set_tile

Imposta il valore della casella di una finestra di dialogo.

(set_tile *chiave valore*)

L'argomento *chiave* è una stringa che indica la casella, mentre *valore* è una stringa che attribuisce un nome al nuovo valore da assegnare (inizialmente impostato dall'attributo value).

Capitolo 13 -- AutoLISP: catalogo delle funzioni

setcfg

Scrive i dati dell'applicazione nella sezione AppData del file acad.cfg.

```
(setcfg nome_cfg val_cfg)
```

L'argomento *nome_cfg* è una stringa, composta da un massimo di 132 caratteri, che indica la sezione ed il parametro per impostare il valore di *val_cfg* (massimo di 347 caratteri). Se l'argomento *nome_cfg* non è valido, **setcfg** restituisce nil. L'argomento *nome_cfg* deve essere una stringa nel formato seguente:

```
"AppData/nome_applicazione/nome_sezione/. . ./nome_parametro"
```

Il codice riportato di seguito imposta 8 per il parametro SpessMur nella sezione AppData/GrupArch e restituisce la stringa "8":

```
(setcfg "AppData/GrupArch/SpessMur" "8") restituisce "8"
```

Vedere anche

la funzione **getcfg**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

setfunhelp

Registra un comando definito dall'utente con la funzionalità Guida in modo che il file della Guida e l'argomento vengano richiamati quando l'utente richiede la guida di quel comando.

```
(setfunhelp c:nome_funzione [fileguida [argomento [comando]])
```

L'argomento *c:nome_funzione* è una stringa che specifica il comando definito dall'utente (la funzione **C:XXX**) e deve includere il prefisso **C:**. Al contrario del nome della funzione passato alla funzione **defun**, l'argomento *funzione* per **setfunhelp** deve essere una stringa tra virgolette. Gli altri tre argomenti descrivono la chiamata da effettuare alla guida. Anche tali argomenti sono stringhe come gli argomenti passati alla funzione **help**. Se ha esito positivo la funzione **setfunhelp** restituisce la stringa passata come *c:nome_funzione*, altrimenti restituisce nil.

Con l'argomento *fileguida* non è richiesta l'estensione di file. Se viene fornita un'estensione di file, AutoCAD cerca solo quel determinato file. Se non ne viene fornita nessuna, la ricerca viene eseguita in base alle seguenti regole: in AutoCAD per Windows/NT, ricerca dei file *.hlp*, altrimenti ricerca *.ahp*. Se nessun file *<nomefile>.ahp* viene trovato, ricerca *<nomefile>* senza alcuna estensione. Il file senza estensione viene cercato per ultimo, quindi in UNIX si cerca *acad.ahp* prima di *acad*.

È da considerare che questa funzione verifica solo se l'argomento *c:nome_funzione* ha il prefisso **ma non** se la funzione *c:nome_funzione* esiste o se gli altri argomenti sono corretti.

Quando si utilizza la funzione **defun** per definire una funzione **C:XXX**, essa rimuove il nome di tale funzione da quelli registrati da **setfunhelp** (se esistono). Comunque la funzione **setfunhelp** dovrebbe essere richiamata solo dopo la chiamata a **defun**, che definisce il comando definito dall'utente.

L'esempio seguente usa **defun** per definire il comando definito dall'utente MIAFUNZ. La funzione **setfunhelp** registra il nome della funzione **C:MIAFUNZ** ed associa quel nome all'argomento miafunz nel file *miohelp.ahp*.

```
(defun c:miafunz ()
  ...
  (getint "dammi: ")
  ...
)
(setfunhelp "c:miafunz" "miohelp.ahp" "miafunz")
```

Comando: **miafunz**

dammi: **'help**

Presumendo che il file di Guida di AutoCAD *miohelp.ahp* esiste nel percorso di supporto, AutoCAD visualizza la finestra di dialogo Guida con l'argomento miafunz del file *miapp.ahp*.

Vedere anche

le funzioni **defun** e **help** .

Capitolo 13 -- AutoLISP: catalogo delle funzioni**setq**

Imposta il valore di un simbolo o di simboli per le espressioni associate.

(setq *simb1 espr1 [simb2 espr2] . . .)*

Si tratta della funzione AutoLISP di base per eseguire operazioni di assegnazione. In una chiamata alla funzione, **setq** può assegnare più simboli, ma restituisce solo l'ultima *espr*.

(setq a 5.0) *restituisce* 5.0

ed imposta il simbolo a su 5.0. Ogni volta che a viene valutato, esso restituirà il numero reale 5.0.

(setq b 123 c 4.7) *restituisce* 4.7
 (setq s "it") *restituisce* "it"
 (setq x '(a b)) *restituisce* (A B)

Vedere anche

"Variabili AutoLISP."

Capitolo 13 -- AutoLISP: catalogo delle funzioni**setvar**

Imposta un valore specifico per una variabile di sistema di AutoCAD.

(setvar *nome_var valore)*

Se l'esito è positivo, **setvar** restituisce il valore della variabile di sistema. Il nome della variabile deve essere racchiuso tra virgolette.

(setvar "FILLETRAD" 0.50) *restituisce* 0.5

ed imposta il raggio di raccordo di AutoCAD su 0.5 unità. Per variabili di sistema con valori interi, il *valore* deve essere compreso tra -32,768 e +32,767.

Alcuni comandi di AutoCAD ottengono i valori delle variabili di sistema prima di inviare qualsiasi messaggio di richiesta. Se la funzione **setvar** viene utilizzata per impostare un nuovo valore durante l'esecuzione di un comando, il nuovo valore potrebbe non avere alcun effetto fino al successivo comando di AutoCAD.

Nota Quando la funzione **setvar** viene utilizzata per modificare la variabile di sistema ANGBASE di AutoCAD, l'argomento relativo al valore viene interpretato in radianti. Il comando MODIVAR di AutoCAD, invece, interpreta questo argomento in gradi. Quando la funzione **setvar** viene utilizzata per modificare la variabile di sistema SNAPANG di AutoCAD, l'argomento del valore viene interpretato in radianti relativi alla direzione di default di AutoCAD per l'angolo 0, che è *est* oppure *le ore 3*. Il comando MODIVAR, invece, interpreta questo argomento in gradi relativi all'impostazione di ANGBASE.

Nota Il comando ANNULLA non annulla le modifiche apportate alla variabile di sistema CVPORT mediante la funzione **setvar**.

È possibile trovare l'elenco delle variabili di sistema correnti di AutoCAD in "Variabili di sistema," nella *Guida di riferimento dei comandi di AutoCAD*.

Vedere anche

la funzione **getvar**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

sin

Restituisce il seno di un angolo espresso in radianti come numero reale.

(*sin* *angolo*)

L'argomento *angolo* deve essere un angolo espresso in radianti.

```
(sin 1.0)           restituisce 0.841471
(sin 0.0)           restituisce 0.0
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

setview

Stabilisce una vista per una finestra specificata.

(*setview* *descrizione_vista* [*id_finestra*])

L'argomento *descrizione_vista* è un elenco di definizione di entità simile a quello restituito dalla funzione **tblsearch** quando applicata alla tabella di simboli VIEW. L'argomento opzionale *id_finestra* indica la finestra che deve visualizzare la nuova vista. Il numero dell'argomento *id_finestra* può essere richiamato dalla variabile di sistema CVPORT. Se l'argomento *vport_id* è 0, la nuova vista viene visualizzata dalla finestra corrente. Se la funzione **setview** ha esito positivo, restituisce l'argomento *descrizione_vista*.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

slide_image

Visualizza una diapositiva di AutoCAD nel gruppo di immagini della finestra di dialogo correntemente attiva.

(*slide_image* *x1* *y1* *larg* *altez* *nome_dia*)

La diapositiva può essere un file (.*sld*) oppure una diapositiva contenuta nel file della libreria di diapositive (.*slb*): l'argomento *nome_dia* specifica la diapositiva nello stesso modo in cui sarebbe stato necessario indicarla per il comando VISDIA o per un file di menu (vedere "Creazione di immagini"):

```
nome_dia oppure nome_lib(nome_dia)
```

Il punto di inserimento del primo angolo superiore sinistro della diapositiva si trova in corrispondenza delle coordinate (*x1,y1*), mentre il secondo angolo, inferiore destro, si trova alla distanza relativa (*larg,altez*) rispetto al primo (*larg* ed *altez* devono essere valori positivi). L'origine (0,0) è l'angolo superiore sinistro dell'immagine. È possibile ottenere le coordinate dell'angolo inferiore destro richiamando le funzioni di quota **dimx_tile** e **dimy_tile**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

svalid

Verifica la validità dei caratteri nel nome della tabella di simboli.

(snvalid nome_simb [flag])

L'argomento *nome_simb* è una stringa che specifica il nome di una tabella di simboli. L'argomento opzionale *flag* specifica se il carattere barra verticale è consentito nell'argomento *nome_simb*. L'argomento *flag* può essere 1 o 0 (il default è 0). La funzione **snvalid** restituisce T se *nome_simb* è un nome di tabella valido, altrimenti restituisce nil.

I nomi delle tabelle di simboli devono essere formati soltanto da caratteri alfanumerici e da caratteri speciali quali il segno del dollaro (\$), il carattere di sottolineatura (_) ed il trattino di sillabazione (-). Una stringa vuota non è un nome valido

Capitolo 13 -- AutoLISP: catalogo delle funzioni

sqrt

Restituisce la radice quadrata di un numero come numero reale.

(sqrt num)

(sqrt 4)	restituisce	2.0
(sqrt 2.0)	restituisce	1.41421

Capitolo 13 -- AutoLISP: catalogo delle funzioni

ssadd

Aggiunge un oggetto (entità) ad un gruppo di selezione oppure crea un nuovo gruppo di selezione.

(ssadd [nome_ent [gs]])

La funzione **ssadd**, se viene richiamata senza alcun argomento, crea un nuovo gruppo di selezione senza alcun membro. Se tale funzione viene richiamata solamente con l'argomento relativo al nome dell'entità *nome_ent*, crea un nuovo gruppo di selezione che conterrà solo l'entità specificata. Se la funzione viene richiamata con il nome di una entità ed il gruppo di selezione *gs*, **ssadd** aggiunge l'entità indicata al gruppo di selezione. La funzione **ssadd** restituisce sempre il gruppo di selezione nuovo o modificato.

Quando un'entità viene aggiunta ad un gruppo, la nuova entità viene aggiunta al gruppo esistente ed il gruppo passato come *ss* viene restituito come risultato. Perciò, se il gruppo viene assegnato ad altre variabili, l'entità sarà aggiunta anche ad esse. Se l'entità specificata è già presente nel gruppo, la funzione **ssadd** viene ignorata e non viene riportato alcun errore.

(setq e1 (entnext))	<i>Imposta e1 per il nome della prima entità nel disegno</i>
(setq ss (ssadd))	<i>Imposta ss come gruppo di selezione nullo</i>
(ssadd e1 ss)	<i>Restituisce ss con il nome dell'entità e1 aggiunto</i>
(setq e2 (entnext e1))	<i>Prende l'entità che segue e1</i>
(ssadd e2 ss)	<i>Restituisce ss con il nome dell'entità e2 aggiunto</i>

Capitolo 13 -- AutoLISP: catalogo delle funzioni

ssdel

Cancella un oggetto (entità) da un gruppo di selezione.

(ssdel nome_ent gs)

La funzione **ssdel** cancella l'entità *nome_ent* dal gruppo di selezione *gs* e restituisce il nome del gruppo di selezione *gs*. Tenere presente che l'entità viene effettivamente cancellata dal gruppo di selezione, nonostante venga restituito un nuovo gruppo di selezione con l'elemento cancellato. Se l'entità non si trova nel gruppo, viene restituito nil.


```
(ssget "L" elenco-pt)
```

Crea un gruppo di selezione di tutti gli oggetti che intersecano la delimitazione definita da elenco-pt.

```
(ssget "FP" elenco-pt elenco-filtri)
```

Crea un gruppo di selezione di tutti gli oggetti all'interno del poligono definito da elenco-pt che corrispondano all'elenco-filtri

Gli oggetti selezionati vengono evidenziati solo quando la funzione **ssget** viene utilizzata senza argomenti. I gruppi di selezione occupano gli alloggiamenti temporanei riservati ai file di AutoCAD, pertanto ad AutoLISP non è consentito averne più di 128 aperti contemporaneamente. Se viene raggiunto questo limite, AutoCAD non crea altri gruppi di selezione e restituisce nil a tutte le chiamate alla funzione **ssget**. Per chiudere la variabile di un gruppo di selezione non più necessario, impostare nil per tale variabile.

La variabile di un gruppo di selezione può essere passata ad AutoCAD in risposta a tutti i messaggi di richiesta Scegliere un oggetto per i quali è valida la selezione mediante la modalità Ultimo. In tale modo vengono selezionati tutti gli oggetti contenuti nella variabile del gruppo di selezione.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

ssget

Filtri del gruppo di selezione

Gli elenchi dei filtri per i gruppi di selezione possono essere usati con tutte le modalità di selezione. È possibile ottenere un gruppo di selezione che comprende tutti gli oggetti di un determinato tipo, su un layer specifico oppure di un determinato colore.

Nell'esempio riportato di seguito, viene restituito un gruppo di selezione composto solamente da linee blu che fanno parte del gruppo di selezione Implicito (gli oggetti selezionati mentre PICKFIRST era attiva).

```
(ssget "M" '((0 . "LINE") (62 . 5)))
```

Utilizzando l'elenco di filtri **ssget**, è possibile selezionare tutti gli oggetti che contengono i dati estesi per una determinata applicazione. Ciò può essere eseguito utilizzando il codice di gruppo -3, come mostrato nel seguente esempio.

```
(ssget "P" '((0 . "CIRCLE") (-3 ("NOMEAPP"))))
```

In questo modo vengono selezionati tutti i cerchi che contengono dati per l'applicazione "NOMEAPP".

Per ulteriori informazioni, vedere "Elenchi dei filtri dei gruppi di selezione" in "Filtraggio di dati estesi."

Capitolo 13 -- AutoLISP: catalogo delle funzioni

ssget

Filtri del gruppo di selezione

Verifiche relazionali

A meno che non sia stato specificato diversamente, per ogni voce presente nell'*elenco-filtri* è implicita un'equivalenza. Per i gruppi numerici (numeri interi, reali, punti e vettori), è possibile specificare altre relazioni includendo un codice di gruppo speciale -4 che specifica un operatore relazionale. Il valore di un gruppo -4 è una stringa che indica l'operatore di verifica che deve essere applicato al successivo gruppo contenuto nell'elenco di filtri. La funzione:

```
(ssget "X" '((0 . "CIRCLE") (-4 . ">=") (40 . 2.0)))
```

seleziona tutti i cerchi il cui raggio (codice di gruppo 40) è maggiore o uguale a 2.0.

Nella tabella riportata di seguito vengono descritti gli operatori che è possibile utilizzare.

Operatori relazionali per gli elenchi di filtri dei gruppi di selezione

Operatore	Descrizione
-----------	-------------

"*"	Qualsiasi condizione è sempre vera.
"="	È uguale a.
"!="	È diverso da.
"/>"	È diverso da.
"<>"	È diverso da.
"<"	È minore di.
"<="	È minore di o uguale a.
">"	È maggiore di.
">="	È maggiore di o uguale a.
"&"	È l'operatore AND a livello bit (solo gruppi di numeri interi).
"&="	È l'operatore uguale mascherato a livello bit (solo gruppi di numeri interi).

L'uso di operatori relazionali dipende dal tipo di gruppo sul quale si intende eseguire la verifica:

- Tutti gli operatori tranne quelli a livello bit ("&" e "&=") sono validi sia per gruppi valutati come numeri reali che interi.
- Gli operatori a livello bit "&" e "&=" sono validi solo per i gruppi valutati come numeri interi. L'operatore a livello bit AND, "&", è vero se $((\text{gruppo_num_int} \& \text{filtro}) \neq 0)$, cioè se uno dei bit contenuti nella maschera è stato impostato anche nel gruppo di numeri interi. L'operatore uguale mascherato a livello bit, "&=", è vero se $((\text{gruppo_num_int} \& \text{filtro}) = \text{filtro})$, ovvero se tutti i bit impostati nella maschera sono impostati anche nel *gruppo_num_int* (altri bit possono essere impostati nel *gruppo_num_int*, ma non vengono verificati).
- Per gruppi di punti, le verifiche di X, Y e Z possono essere combinate in una singola stringa, con gli operatori separati da virgole (ad esempio, ">, >, *"). Se un operatore viene ommesso dalla stringa (ad esempio, "=", "<>" non include la verifica di Z), si presume l'operatore "*" che sancisce come vera qualsiasi condizione.
- I vettori di direzione (tipo di gruppo 210) possono essere confrontati solo con gli operatori "*", "=", "!=" (oppure con una delle equivalenti stringhe "diverso da").
- Non è possibile utilizzare gli operatori relazionali con gruppi di stringhe; al loro posto utilizzare le verifiche dei caratteri jolly.

Capitolo 13 -- AutoLISP: catalogo delle funzioni



Filtri del gruppo di selezione

Raggruppamento logico delle verifiche dei filtri

Gli operatori relazionali descritti precedentemente sono operatori binari. È possibile eseguire la verifica di gruppi anche creando espressioni booleane nidificate che utilizzino gli operatori di raggruppamento mostrati nella tabella riportata di seguito. Tali operatori vengono specificati dai gruppi -4, come gli operatori relazionali. Essi vengono accoppiati e devono essere correttamente bilanciati nell'elenco dei filtri, altrimenti la chiamata **ssget** non avrà esito positivo. Il numero degli operandi che possono essere racchiusi da questi operatori dipende da tipo di operazione.

Operatori di raggruppamento per gli elenchi di filtri dei gruppi di selezione

Operatore iniziale	comprende	Operatore finale
"<AND"	Uno o più operandi	"AND>"
"<OR"	Uno o più operandi	"OR>"
"<XOR"	Due operandi	"XOR>"
"<NOT"	Un operando	"NOT>"

Per gli operatori di raggruppamento, un *operando* può essere un gruppo di campi dell'entità, un operatore relazionale seguito da un gruppo di campo dell'entità oppure una espressione nidificata creata da questi operatori. Di seguito viene riportato un esempio di operatori di raggruppamento in un elenco di filtri.

```
(ssget "X" '((-4 . "<OR")
  (-4 . "<AND")
    (0 . "CIRCLE")
    (40 . 1.0)
  (-4 . "<AND>")
  (-4 . "<AND")
    (0 . "LINE")
    (8 . "ABC")
  (-4 . "<AND>")
```

```
(-4 . "OR>") )
```

In questo modo vengono selezionati tutti i cerchi con raggio 1.0 e tutte le linee sul layer "ABC".

Poiché gli operatori di raggruppamento non sono sensibili al maiuscolo/minuscolo, è possibile utilizzare anche i loro equivalenti con lettere minuscole: "<and", "and>", "<or", "or>", "<xor", "xor>", "<not" e "not>".

Capitolo 13 -- AutoLISP: catalogo delle funzioni

ssgetfirst

Individua gli oggetti selezionati e sottoposti a grip.

```
(ssgetfirst)
```

Restituisce un elenco di due gruppi di selezione simili a quelli passati a **sssetfirst**. Il primo elemento dell'elenco è un gruppo di selezione di entità sottoposte a grip, ma non selezionate. Il secondo elemento è una selezione di entità sottoposte a grip e selezionate. Uno degli elementi (o entrambi) dell'elenco può essere il valore nil.

Nota Questa funzione può analizzare solo le entità che si trovano nello spazio modello e nello spazio carta del disegno corrente e non le entità o gli oggetti non grafici presenti nelle definizioni di altri blocchi.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

sslength

Restituisce un numero intero che indica il numero di oggetti (entità) contenuti in un gruppo di selezione.

```
(sslength gs)
```

Se maggiore di 32,767, il numero restituito sarà un numero reale. I gruppi di selezione non contengono mai selezioni duplicate della stessa entità.

```
(setq sset (ssget "L"))      Posiziona l'ultimo oggetto nel gruppo di selezione sset
(sslength sset)             restituisce 1
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

ssmemb

Verifica se un oggetto (entità) è un membro di un gruppo di selezione.

```
(ssmemb nome_ent gs)
```

Se un oggetto è un membro di un gruppo di selezione, **ssmemb** restituisce il nome dell'entità (*nome_ent*). In caso contrario, restituisce nil. Ad esempio, supponendo che l'entità e1 sia un membro del gruppo di selezione gs1 e che l'entità e2 non lo sia, si avrà:

```
(ssmemb e1 ss1)             restituisce il nome dell'entità e1
(ssmemb e2 ss1)             restituisce nil
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

ssname

Restituisce il nome dell'oggetto (entità) dell'elemento indicizzato contenuto in un gruppo di selezione.

(ssname *gs* *indice*)

L'argomento *indice* deve essere un numero intero. Se *indice* è negativo o maggiore rispetto al numero più grande assegnato ad una entità nel gruppo di selezione, viene restituito nil. Il primo elemento contenuto nel gruppo ha indice zero. I nomi delle entità contenute nei gruppi di selezione e restituiti dalla funzione **ssget** saranno sempre nomi di entità principali. Le sottoentità (attributi e vertici di polilinea) non vengono restituite. La funzione **entnext** consente invece di accedere a tali sottoentità. Vedere "**entnext**".

```
(setq sset (ssget))           Crea un gruppo di selezione chiamato sset
(setq ent1 (ssname sset 0))   Prende il nome della prima entità in sset
(setq ent4 (ssname sset 3))   Prende il nome della quarta entità in sset
```

Per accedere ad entità oltre la 32767esima in un gruppo di selezione, è necessario fornire l'argomento *indice* come numero reale. Ad esempio:

```
(setq entx (ssname sset 50843.0)) Prende il nome della 50844esima entità di sset
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

ssnamex

Recupera le informazioni relative alla creazione di un gruppo di selezione

(ssnamex *gs* [*indice*])

Questa funzione restituisce il nome dell'entità dell'elemento specificato da *indice* dal gruppo di selezione *gs* insieme ai dati che descrivono il modo in cui l'entità è stata selezionata. Se l'argomento *indice* non viene specificato, questa funzione restituisce un elenco contenente i nomi delle entità di tutti gli elementi del gruppo di selezione insieme ai dati che descrivono in che modo ciascuna entità è stata selezionata.

I dati restituiti da **ssnamex** diventano un elenco degli elenchi che contengono le informazioni che descrivono un'entità ed il relativo metodo di selezione oppure un poligono che è stato utilizzato per selezionare una o più entità. Ciascun elenco secondario che descrive la selezione di una determinata entità è composto da tre parti: l'ID del metodo di selezione (un intero ≥ 0), il nome dell'entità selezionata ed i dati specifici del metodo di selezione che descrivono in che modo l'entità è stata selezionata.

```
( (sel_id1 ename1 (data) ) (sel_id2 ename2 (data) ) . . . )
```

Fare riferimento all'argomento **ssnamex** nel Manuale di personalizzazione stampato per una lista degli ID ed una descrizione.

Ciascun elenco secondario che descrive un poligono utilizzato durante la selezione di entità diventa un ID poligono (un intero < 0), seguito dalle descrizioni dei punti.

```
(polygon_id point_description_1 point_description_n . . . )
```

La numerazione dell'ID poligono comincia da -1 e ciascun poligono aggiuntivo viene incrementato di -1. A seconda della posizione visualizzata, un punto viene rappresentato in uno dei seguenti modi: una linea tendente all'infinito, un raggio o un segmento di linea. Un descrittore di punti è composto da tre parti: un ID descrittore del punto (il tipo di voce descritta), il punto di inizio della voce ed un vettore di unità opzionale che descrive la direzione verso cui la linea tende all'infinito o un vettore che descrive lo sfalsamento rispetto all'altro lato del segmento di linea.

```
(id_poligono punto_descrizione_1 punto_descrizione_n . . .
```

La seguente tabella elenca gli ID descrittore di punti validi.

ID *descrittori di punti validi*

ID	<i>Descrizione</i>
0	<i>Linea infinita</i>
1	<i>Raggio</i>
2	<i>Segmento di linea</i>

Il *vettore_sfalsamento_o_unità* viene restituito quando il punto di vista è diverso da 0,0,1.

I *dati* associati alla selezione di entità Selezione (tipo 1) rappresentano la descrizione di un singolo punto. Ad esempio, il seguente record viene restituito per la selezione di un'entità selezionata al punto 1,1 nella vista piana del WCS:

```
(1 <Nome di entità: 60000064> (0 (1.0 1.0 0.0) ) )
```

I *dati* associati ad un'entità selezionata con il metodo Finestra, FPoligono, Interseca o IPoligono rappresenta l'ID intero del poligono che ha selezionato l'entità. È l'applicazione che associa gli identificatori dei poligoni ed effettua il collegamento tra il poligono e le entità da questo selezionate. Ad esempio, il seguente record viene restituito per un'entità selezionata da Interseca (tenere presente che l'ID poligono è -1).

```
((3 <Nome di entità: 60000024> -1) (-1 (0 (5.14828 7.05067 0.0) )
(0 (7.13676 7.05067 0.0) ) (0 (7.13676 4.62785 0.0) )
(0 (5.14828 4.62785 0.0) ) ) )
```

I dati associati alle selezioni di tipo Intercetta sono un elenco delle descrizioni dei punti in cui l'intercettazione e l'entità si intersecano visivamente. Ad esempio, il seguente record viene restituito per una linea verticale intersecata tre volte da un'intercettazione a forma di Z.

```
((4 <Nome di entità: 60000024> (0 (5.28135 6.25219 0.0) )
(0 (5.61868 2.81961 0.0) ) (0 (5.52688 3.75381 0.0) ) ) )
```

Nota Questa funzione può richiamare solo i gruppi di selezione con entità che si trovano nello spazio modello e nello spazio carta del disegno corrente e *non* gli oggetti o le entità non grafici presenti nelle definizioni di altri blocchi.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

sssetfirst

Stabilisce quali oggetti vengono selezionati e sottoposti al grip.

```
(sssetfirst gruppogrip [grupposelezione])
```

Il gruppo di selezione di oggetti specificato dall'argomento *gruppogrip* viene sottoposto a grip e il gruppo di selezione di oggetti specificato da *grupposelezione* viene sottoposto a grip e selezionato. Se vi sono degli oggetti comuni ad entrambi i gruppi di selezione, **sssetfirst** sottopone a grip e seleziona solo il gruppo di selezione specificato da *grupposelezione* (non sottopone a grip il gruppo *gruppogrip*). Se *gruppogrip* è nil **sssetfirst** sottopone a grip e seleziona il gruppo specificato da *grupposelezione*. La funzione **sssetfirst** restituisce un elenco delle due variabili passate al gruppo di selezione.

Nota Non richiamare **ads_sssetfirst()** mentre AutoCAD esegue un comando.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

startapp

Avvia un'applicazione di Windows.

```
(startapp cmdapp [file])
```

L'argomento *cmdapp* è una stringa che indica l'applicazione da eseguire. Se *cmdapp* non include il nome completo di un percorso, la funzione ricerca la variabile d'ambiente PATH nell'applicazione. L'argomento *file* è una stringa che indica il nome da file che deve essere aperto. Se l'esito è positivo restituisce un numero maggiore di 0; altrimenti restituisce 0.

Con il codice riportato di seguito viene avviato il Blocco note di Windows ed aperto il file *acad.lsp*.

```
(startapp "notepad" "acad.lsp")
```

Se un argomento include degli spazi, deve essere racchiuso tra virgolette. Ad esempio, per modificare il file *mio lavoro.txt* con Blocco note, utilizzare la sintassi seguente:

```
(startapp "notepad.exe" "\"mio lavoro.txt\"")
```

Funzione definita esternamente applicazione ARX *acadapp*.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

start_dialog

Visualizza una finestra di dialogo ed inizia ad accettare l'input dell'utente.

```
(start_dialog)
```

È necessario che la finestra di dialogo sia stata precedentemente inizializzata con la funzione **new_dialog**. La finestra di dialogo rimane attiva fino a quando una espressione di azione o una funzione di richiamo non richiama la funzione **done_dialog**. Generalmente, **done_dialog** è associata alla casella il cui key è "accept" (di solito, il pulsante OK) ed alla casella il cui key è "cancel" (di solito il pulsante Annulla).

La funzione **start_dialog** non ha argomenti e restituisce lo *stato* opzionale passato a **done_dialog**. Il valore di default è 1 se l'utente ha premuto OK, 0 se ha premuto Annulla oppure -1 se tutte le finestre di dialogo sono state chiuse con **term_dialog**; ma, se a **done_dialog** è stato passato *stato* come numero intero maggiore di 1, **start_dialog** restituisce questo valore il cui significato dipende dall'applicazione.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

start_image

Avvia la creazione di un'immagine nella casella di una finestra di dialogo.

```
(start_image chiave)
```

Le successive chiamate alle funzioni **fill_image**, **slide_image** e **vector_image** hanno effetto su questa immagine fino a quando l'applicazione non chiama la funzione **end_image**. L'argomento *chiave* è una stringa che indica la casella della finestra di dialogo ed è sensibile al maiuscolo/minuscolo.

Nota Non utilizzare la funzione **set_tile** tra le chiamate alle funzioni **start_image** ed **end_image**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

start_list

Avvia l'elaborazione di un elenco nella casella di una finestra di dialogo relativa ad una casella di riepilogo o ad un elenco a comparsa.

```
(start_list chiave [operazione [indice] ])
```

L'argomento *chiave* è una stringa che indica la casella della finestra di dialogo ed è sensibile al maiuscolo/minuscolo. L'argomento *operazione* è un numero intero il cui significato è descritto nella tabella riportata di seguito.

Codici della casella di riepilogo per *start_list*

Valore	Descrizione
1	Modifica il contenuto dell'elenco selezionato
2	Aggiunge la nuova voce all'elenco
3	Elimina l'elenco precedente e ne crea uno nuovo (il valore di default)

L'argomento *indice* viene ignorato a meno che la funzione **start_list** non attivi l'operazione di modifica (1); in questo caso, *indice* indica la voce dell'elenco che deve essere modificata dalla successiva funzione **add_list**. Il valore iniziale di *indice* è zero. Se

l'argomento *operazione* non viene specificato, viene assunto il valore di default 3 (creazione di un nuovo elenco); se invece viene specificato solo l'argomento *operazione* e non *indice*, verrà assunto l'indice di default 0.

Le successive chiamate alla funzione **add_list** hanno effetto sull'elenco avviato da **start_list** fino a quando l'applicazione non richiama la funzione **end_list**.

Nota Non utilizzare la funzione **set_tile** tra le chiamate alle funzioni **start_list** ed **end_list**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

strcase

Restituisce la stringa i cui caratteri alfabetici sono stati convertiti in lettere maiuscole o minuscole.

(strcase *stringa* [*maiuscolo o minuscolo*])

Se *maiuscolominuscolo* viene ommesso oppure ha valore nil, tutti i caratteri alfabetici contenuti in *stringa* vengono convertiti in lettere maiuscole. Se *maiuscolominuscolo* viene fornito e non ha valore nil, tutti i caratteri alfabetici contenuti in *stringa* vengono convertiti in lettere minuscole.

```
(strcase "Esempio")           restituisce "ESEMPIO"
(strcase "Esempio" T)        restituisce "esempio"
```

La funzione **strcase** gestirà correttamente la corrispondenza di maiuscole e minuscole della serie di caratteri attualmente configurata (vedere "Supporto per altre lingue").

Capitolo 13 -- AutoLISP: catalogo delle funzioni

strcat

Restituisce una stringa ottenuta dalla concatenazione di più stringhe.

(strcat *stringa1* [*stringa2*]...)

```
(strcat "a" "tipico")         restituisce "atipico"
(strcat "a" "b" "c")         restituisce "abc"
(strcat "a" "" "c")          restituisce "ac"
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

strlen

Restituisce un numero intero che rappresenta il numero di caratteri contenuti in una stringa.

(strlen [*stringa*]...)

Se vengono specificati più argomenti *stringa*, la funzione restituisce la somma delle lunghezze di tutti gli argomenti. L'omissione degli argomenti o l'immissione di una stringa vuota provoca la restituzione del valore 0 (zero).

```
(strlen "abcd")              restituisce 4
(strlen "ab")                restituisce 2
(strlen "uno" "due" "sette" restituisce 11
(strlen)                    restituisce 0
(strlen "")                  restituisce 0
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

subst

Cerca un elemento esistente in una lista e restituisce una copia della lista con un nuovo elemento in sostituzione di quello precedente, per tutte le occorrenze del vecchio elemento.

(subst nuovo_elem vecchio_elem elenco)

Se *vecchio_elem* non viene trovato in *lista*, la funzione **subst** restituisce la *lista* inalterata.

```
(setq esempio '(a b (c d) b))
(subst 'qq 'b esempio)      restituisce (A QQ (C D) QQ)
(subst 'qq 'z sample)      restituisce (A B (C D) B)
(subst 'qq '(c d) esempio)  restituisce (A B QQ B)
(subst '(qq rr) '(c d) esempio) restituisce (A B (QQ RR) B)
(subst '(qq rr) 'z esempio) restituisce (A B (C D) B)
```

Se usato insieme a **assoc**, **subst** fornisce un modo per sostituire il valore associato con un tasto in una lista associata.

```
(setq chi '((prima gianni) (poi q) (ultimo pubblico)))

(setq vecchio (assoc 'first who)      imposta vecchio a (PRIMO GIANNI)
      nuovo '(first j)                imposta nuovo a (PRIMO G)
)
(subst nuovo vecchio chi) restituisce ((PRIMO G) (POI Q) (ULTIMO PUBBLICO))
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

substr

Restituisce una sottostringa di una stringa.

(substr stringa inizio [lunghezza])

La funzione **substr** inizia dalla posizione del carattere di *inizio* della *stringa* e continua per i caratteri della *lunghezza*. Se *lunghezza* non viene specificato, la sottostringa continua fino alla fine della *stringa*. Gli argomenti *inizio* e *lunghezza* devono essere numeri interi positivi.

Il primo carattere della *stringa* è il carattere numero 1. Ciò differisce dalle altre funzioni di elaborazione degli elementi di un elenco (come **nth** e **ssname**) che iniziano a contare il primo elemento con il numero 0.

```
(substr "abcde" 2)      restituisce "bcde"
(substr "abcde" 2 1)   restituisce "b"
(substr "abcde" 3 2)   restituisce "cd"
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

tablet

Richiama ed imposta le graduazioni del digitalizzatore (tavoletta).

(tablet codice [riga1 riga2 riga3 direzione])

In base al numero intero specificato da *codice*, **tablet** richiama la graduazione corrente del digitalizzatore oppure imposta la graduazione. Se *codice* è 0, **tablet** restituisce la graduazione corrente. Se *codice* è 1, deve essere seguito dalle nuove impostazioni della graduazione: *riga1*, *riga2*, *riga3* e *direzione*.

codice

Un numero intero. Se il *codice* passato è uguale a 0, la funzione **tablet** restituisce la graduazione corrente; in questo caso, è necessario omettere i restanti argomenti. Se il *codice* passato è uguale a 1, **tablet** imposta la graduazione in base agli argomenti che seguono; in questo caso, è necessario fornire gli altri argomenti.

riga1, riga2, riga3

Tre punti tridimensionali. Questi tre argomenti indicano le tre righe della matrice di trasformazione della tavoletta.

direzione

Un punto tridimensionale. Si tratta del vettore, espresso nel Sistema di Coordinate Globali o WCS, che è perpendicolare al piano che rappresenta la superficie della tavoletta.

Nota Se la *direzione* specificata non è perpendicolare, **tablet** la corregge in modo tale che la *direzione* restituita al momento dell'impostazione della graduazione possa differire dal valore passato. Allo stesso modo, il terzo elemento in *riga3* (Z) deve sempre essere uguale a 1: **tablet** lo restituisce come 1 anche quando con l'argomento *riga3* incluso nell'elenco viene specificato un valore diverso.

Se l'esito è negativo, **tablet** restituisce nil ed imposta la variabile di sistema ERRNO su un valore che indica la causa dell'esito negativo (vedere il capitolo 16, "Codici e messaggi di errore AutoLISP"). Ciò può verificarsi se il digitalizzatore non è una tavoletta.

Una trasformazione molto semplice che può essere effettuata con **tablet** è la trasformazione di identità:

```
(tablet 1 '(1 0 0) '(0 1 0) '(0 0 1) '(0 0 1))
```

Con questa trasformazione in atto, AutoCAD riceverà dalla tavoletta le coordinate del digitalizzatore in modo efficace. Ad esempio, se il punto viene selezionato con le coordinate del digitalizzatore (5000,15000), AutoCAD lo interpreterà come il punto all'interno del disegno a cui corrispondono quelle stesse coordinate.

La variabile di sistema TABMODE consente alle routine AutoLISP di attivare e disattivare la tavoletta.

Vedere anche

"Graduazione delle tavolette" per ulteriori informazioni sulla matrice di trasformazione della tavoletta.

Capitolo 13 -- AutoLISP: catalogo delle funzioni



Richiama ed imposta le graduazioni del digitalizzatore (tavoletta).

```
(tblnext nome-tabella [primo])
```

L'argomento *nome-tabella* è una stringa che identifica la tabella di simboli. I valori validi per *nome-tabella* sono "LAYER", "LTYPE", "VIEW", "STYLE", "BLOCK", "UCS", "APPID", "DIMSTYLE" e "VPORT". Non è necessario che la stringa sia in lettere maiuscole.

Nota poiché la funzione **vports** restituisce le informazioni correnti relative alla tabella VPORT, potrebbe essere più semplice utilizzare **vports** invece di **tblnext** per richiamare queste informazioni.

Quando **tblnext** viene utilizzata ripetutamente, generalmente restituisce ogni volta la voce successiva contenuta nella tabella specificata. La funzione **tblsearch** può impostare la voce *successiva* da richiamare. Tuttavia, se è presente l'argomento *primo* ed il suo valore non è nil, la tabella di simboli viene esaminata dall'inizio e viene richiamata la prima voce in essa contenuta. Se nella tabella non vi sono altre voci, viene restituito nil. Le voci di tabella cancellate non vengono mai restituite.

Se viene trovata una voce, questa viene restituita come elenco di coppie puntate di codici e valori di tipo DXF. La funzione:

```
(tblnext "layer" T) Richiama il primo layer
```

potrebbe restituire:

```
((0 . "LAYER") Tipo di simbolo
(2 . "0") Nome del simbolo)
```

```
(70 . 0)           Flag
(62 . 7)           Numero del colore, negativo se disattivato
(6 . "CONTINUOUS") Nome del tipo di linea
)
```

Notare che non è presente alcun gruppo -1. AutoCAD considera l'ultima voce restituita da ogni tabella e, ogni volta in cui la funzione **tblnext** viene richiamata per quella tabella, restituisce quella successiva. Quando si inizia ad analizzare una tabella, accertarsi che il secondo argomento fornito non sia nil, in modo tale da poter tornare all'inizio della tabella e, quindi, alla prima voce.

Le voci richiamate dalla tabella di blocco comprendono un gruppo -2 con il nome della prima entità nella definizione del blocco (se esistente). Pertanto, dato un blocco chiamato CASELLA, la funzione:

```
(tblnext "block") Richiama la definizione del blocco
```

potrebbe restituire:

```
((0 . "BLOCK")      Tipo di simbolo
(2 . "BOX")         Nome del simbolo
(70 . 0)           Flag
(10 9.0 2.0 0.0)   Origine X,Y,Z
(-2 . <Nome di entità: 40000126>) Prima entità
)
```

Il nome dell'entità nel gruppo -2 viene accettato da **entget** e **entnext**, ma non dalle altre funzioni di accesso alle entità. Ad esempio, non è possibile utilizzare **ssadd** per inserire il nome in un gruppo di selezione. Specificando a **entnext** il nome dell'entità nel gruppo -2, è possibile analizzare le entità comprese nella definizione di un blocco; **entnext** restituisce nil dopo l'ultima entità contenuta nella definizione del blocco.

Nota Se in un blocco non è contenuta alcuna entità, il gruppo -2 restituito da **tblnext** rappresenta il nome dell'entità che indica la fine del blocco.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

tblobjname

Restituisce il nome dell'entità di una voce specifica della tabella di simboli.

```
(tblobjname nometabella simbolo)
```

La funzione **tblobjname** ricerca il nome del simbolo, *simbolo*, nella tabella dei simboli, *nometabella* e restituisce il nome entità dell'elemento della tabella.

Il nome dell'entità restituito da **tblobjname** può essere utilizzato nelle operazioni **entget** e **entmod**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

tblsearch

Cerca il nome di un simbolo in una tabella di simboli.

```
(tblsearch nome-tabella simb [successivo])
```

La funzione **tblsearch** cerca il nome del simbolo, *simbolo*, nella tabella dei simboli, *nometabella*. Entrambi i nomi vengono automaticamente convertiti in lettere maiuscole. Se la funzione **tblsearch** trova un inserimento per l'elemento dato, restituisce quell'inserimento nel formato descritto per la funzione **tblnext**. Se non viene trovata alcuna voce di quel tipo, la funzione restituisce nil.

```
(tblsearch "style" "standard") Richiama lo stile del testo
```

potrebbe restituire:

((0 . "STYLE")	Nome del simbolo
(70 . 0)	Flag
(40 . 0.0)	Altezza fissa
(41 . 1.0)	Fattore di larghezza
(50 . 0.0)	Inclinazione dell'angolo
(71 . 0)	Flag di generazione
(3 . "txt")	File di font principale
(4 . "")	File big font

Generalmente, **tblsearch** non ha effetti sull'ordine delle voci richiamate da **tblnext**. Tuttavia, se **tblsearch** ha esito positivo e l'argomento *successivo* è presente e non è nil, il contatore delle voci della funzione **tblnext** viene regolato in modo tale che la successiva chiamata di **tblnext** restituisca la voce posizionata subito dopo quella restituita da questa chiamata di **tblsearch**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

term_dialog

Chiude tutte le finestre di dialogo correnti come se ognuna di esse fosse stata annullata dall'utente.

```
(term_dialog)
```

Se un'applicazione viene chiusa mentre vi sono file DCL aperti, AutoCAD richiama automaticamente la funzione **term_dialog** che viene usata principalmente per abbandonare le finestre di dialogo nidificate. La funzione **term_dialog** restituisce sempre nil.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

terpri

Visualizza una riga nuova nella riga di comando.

```
(terpri)
```

La funzione **terpri** non viene utilizzata per operazioni di I/O sui file. Per scrivere una riga nuova in un file, utilizzare **prin1**, **princ** o **print**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

textbox

Misura un determinato oggetto di testo e restituisce le coordinate diagonali della casella in cui è racchiuso il testo.

```
(textbox elenco_ent)
```

L'argomento *elenco_ent* è un elenco di definizione di entità nella forma restituita dalla funzione **entget**. Esso deve definire un oggetto testo. Se da tale argomento vengono omessi i campi che definiscono i parametri di testo diversi dal testo stesso, vengono utilizzate le impostazioni correnti o di default. Se la funzione **textbox** ha esito positivo, questa restituisce un elenco composto da due punti; altrimenti, restituisce nil.

L'elenco minimo accettato dalla funzione **textbox** è quello costituito dal testo stesso.

```
(textbox '((1 . "Ciao a tutti"))) restituisce ad esempio ((0.0 0.0 0.0) (0.8 0.2 0.0))
```

In questo caso, la funzione **textbox** utilizzerebbe per il testo specificato le impostazioni correnti di default per fornire i parametri

restanti.

I punti restituiti da **textbox** descrivono la casella di delimitazione dell'oggetto testo come se il relativo punto di inserimento fosse posizionato nel punto (0,0,0) e l'angolo di rotazione fosse uguale a 0. Di solito, la prima riga restituita è il punto (0.0 0.0 0.0), a meno che l'oggetto testo sia obliquo o verticale oppure contenga lettere con tratti discendenti (come la *g* e la *p*). Il valore del primo elenco di punti indica lo sfalsamento dal punto di inserimento del testo all'angolo inferiore sinistro del più piccolo rettangolo in cui è racchiuso il testo. Il secondo elenco di punti specifica l'angolo superiore destro di questa casella. Indipendentemente dall'orientamento del testo da misurare, l'elenco di punti restituito descrive sempre gli angoli inferiore sinistro e superiore destro della casella di delimitazione.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

textpage

Passa dallo schermo grafico allo schermo di testo.

(**textpage**)

La funzione **textpage** equivale alla funzione **textscr** e restituisce sempre nil.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

textscr

Passa dallo schermo grafico allo schermo di testo, come ad esempio il tasto funzione Grafico/Testo di AutoCAD.

(**textscr**)

La funzione **textscr** restituisce sempre nil.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

trace

È di ausilio nelle operazioni di debug di AutoLISP.

(**trace** *funzione*...)

La funzione **trace** imposta il flag della traccia per le *funzioni* specificate. Ogni volta che una determinata *funzione* viene valutata, appare la traccia che indica l'immissione della funzione, rientrata al livello di profondità della chiamata e viene visualizzato il risultato della funzione.

(**trace** mia-funz) *restituisce* MIA-FUNZ

Viene impostato il flag della traccia per la funzione **MIA-FUNZ**. La funzione **trace** restituisce il nome dell'ultima funzione ad essa passato.

Vedere anche

la funzione **untrace**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

trans

Converte un punto o uno spostamento da un sistema di coordinate ad un altro.

(trans pt da a [spost])

L'argomento *pt* è un elenco composto da tre numeri reali che possono essere interpretati come punto tridimensionale o come spostamento tridimensionale (vettore). L'argomento *da* indica il sistema di coordinate in cui è espresso l'argomento *pt*, mentre l'argomento *a* specifica il sistema di coordinate del punto restituito. L'argomento opzionale *spost*, se presente e se diverso da nil, indica che *pt* deve essere considerato come spostamento tridimensionale e non come punto. Gli argomenti *da* e *a* possono essere un codice intero, come specificato nella tabella riportata di seguito, un nome di entità oppure un vettore di estrusione tridimensionale.

Codici del sistema di coordinate

Codice	Sistema di coordinate
0	Globale (WCS)
1	Utente (UCS corrente)
2	Visualizzazione:

DCS della finestra corrente quando utilizzato con il codice 0 o 1

DCS della finestra spazio modello corrente quando utilizzato con il codice 3.

3	DCS dello Spazio Carta, (utilizzato solo con il codice 2)
---	---

Se si utilizza un nome di entità per gli argomenti *da* o *a*, questo deve essere passato nello stesso modo dei nomi restituiti dalle funzioni **entnext**, **entlast**, **entsel**, **nentsel** e **ssname**. Ciò consente di convertire un punto nel Sistema di Coordinate Oggetto (OCS) di un determinato oggetto. Per alcuni oggetti, l'OCS equivale al WCS; pertanto, la conversione tra OCS e WCS sarà un'operazione nulla. Un vettore di estrusione tridimensionale, vale a dire un elenco di tre numeri reali, rappresenta un altro metodo di conversione nell'OCS di un oggetto. Tuttavia, questa operazione non funziona su quegli oggetti per i quali l'OCS equivale al WCS.

La funzione **trans** restituisce un punto tridimensionale o uno spostamento nel sistema di coordinate richiesto mediante l'argomento *a*. Ad esempio, si supponga che un UCS venga ruotato di 90 gradi in senso antiorario intorno all'asse Z del WCS:

```
(trans '(1.0 2.0 3.0) 0 1)           restituisce (2.0 -1.0 3.0)
(trans '(1.1 2.0 3.0) 1 0)         restituisce (-2.0 1.0 3.0)
```

I sistemi di coordinate vengono descritti in modo più dettagliato in "Trasformazioni del sistema di coordinate."

Ad esempio, per disegnare una linea da un punto di inserimento di un testo senza utilizzare Osnap, è necessario convertire il punto di inserimento dell'oggetto testo dall'OCS all'UCS.

```
(trans punto-inser-testo nome_ent-testo 1)
```

Successivamente è possibile passare il risultato al messaggio di richiesta Dal punto.

Al contrario, prima di passarli alla funzione **entmod**, è necessario convertire i valori di un punto o di uno spostamento nell'OCS di destinazione. Ad esempio, se si desidera spostare un cerchio senza utilizzare il comando SPOSTA in base allo sfalsamento relativo all'UCS (1,2,3), è necessario convertire lo spostamento dall'UCS all'OCS del cerchio:

```
(trans '(1 2 3) 1 nome_ent-cerchio)
```

A questo punto è possibile aggiungere lo spostamento risultante al centro del cerchio.

Ad esempio, se l'utente ha immesso un punto e si desidera conoscere l'estremità di una linea a cui si avvicina maggiormente, occorre convertire il punto immesso dall'utente dall'UCS al DCS.

```
(trans punto-utente 1 2)
```

A questo punto è necessario convertire ogni punto finale della linea dall'OCS al DCS.

```
(trans punto finale nome_ent-linea 2)
```

In base ai punti finali ottenuti è possibile calcolare la distanza tra il punto immesso dall'utente e ciascun punto finale della linea, ignorando le coordinate Z, per stabilire qual è l'estremità più vicina.

La funzione **trans** può anche trasformare i punti bidimensionali, *inserendo* un valore appropriato nella coordinata Z. L'utilizzo di quest'ultima dipende dal sistema di coordinate specificato mediante l'argomento *da* e se il valore deve essere convertito come punto o spostamento. Se il valore deve essere convertito come spostamento, il valore Z è sempre pari a 0.0; se invece il valore deve essere convertito come punto, tale valore viene determinato nel modo riportato nella tabella seguente.

Valori Z convertiti in punti bidimensionali

Da	<i>Valori Z immessi</i>
WCS	0.0
UCS	Elevazione corrente
OCS	0.0
DCS	Proiettato sul piano di costruzione corrente
	(piano XY UCS + elevazione corrente).
PSDCS	Proiettato sul piano di costruzione corrente
	(piano XY UCS + elevazione corrente).

Capitolo 13 -- AutoLISP: catalogo delle funzioni

type

Restituisce il tipo di un determinato elemento.

(type elem)

I tipi vengono restituiti sotto forma di atomi come illustrato nella tabella seguente:

Tipo di simboli

Tipo	<i>Descrizione</i>	Tipo	<i>Descrizione</i>
REAL	<i>Numeri a virgola mobile</i>	SUBR	<i>Funzioni interne</i>
FILE	<i>Descrittori di file</i>	EXSUBR	<i>Funzioni esterne (ARX)</i>
STR	<i>Stringhe</i>	PICKSET	<i>Gruppi di selezione</i>
INT	<i>Numeri interi</i>	ENAME	<i>Nomi di entità</i>
SYM	<i>Simboli</i>	PAGETB	<i>Tabella di paginazione delle funzioni</i>
LIST	<i>Sono elenchi e funzioni utente</i>		

Gli elementi valutati come nil (ad esempio un simbolo non assegnato) restituiscono nil.

Ad esempio, date le assegnazioni

```
(setq a 123 r 3.45 s "Ciao!" x '(a b c))
(setq f (open "nome" "r"))
```

si avrà che

```
(type 'a)           restituisce SYM
(type a)            restituisce INT
(type f)            restituisce FILE
(type r)            restituisce REAL
(type s)            restituisce STR
(type x)            restituisce LIST
(type +)            restituisce SUBR
(type nil)          restituisce nil
```

Nell'esempio seguente viene utilizzata la funzione **type** .

```
(defun isint (a)
  (if (= (type a) 'INT) TYPE è un numero intero?
      T si, restituisce T
      nil no, restituisce nil
  )
)
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

unload_dialog

Scarica un file DCL.

`(unload_dialog id_dcl)`

Scarica il file DCL associato all'argomento `id_dcl`, ottenuto da una precedente chiamata della funzione `new_dialog`.

Questa funzione restituisce sempre nil.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

untrace

Cancella il flag di trace per le funzioni specificate.

`(trace funzione...)`

Restituisce il nome dell'ultima funzione.

Il codice seguente cancella il flag della traccia per la funzione **MIAFUNZ**:

```
(untrace mia-funz)           restituisce MIA-FUNZ
```

Vedere anche

la funzione **trace**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

vector_image

Disegna un vettore nell'immagine della finestra di dialogo attualmente attiva.

`(vector_image x1 y1 x2 y2 colore)`

Questa funzione disegna un vettore nell'immagine della finestra di dialogo attualmente attiva, aperta mediante la funzione **start_image**, passante per i punti $(x1,y1)$ e $(x2,y2)$. Il parametro *colore* è un numero di colore di AutoCAD oppure uno dei numeri di colore logici riportati nella tabella seguente.

Nomi simbolici per l'attributo colore

Numero colore	Mnemonic ADI	Descrizione
-2	BGLCOLOR	È lo sfondo corrente dello schermo grafico di AutoCAD
-15	DBGLCOLOR	È il colore di sfondo corrente della finestra di dialogo.
-16	DFGLCOLOR	È il colore di primo piano corrente della finestra di dialogo (per testo).
-18	LINELCOLOR	È il colore della linea corrente della finestra di dialogo.

L'origine (0,0) è l'angolo superiore sinistro dell'immagine. È possibile ottenere le coordinate dell'angolo inferiore destro richiamando le funzioni di quota `dimx_tile` e `dimy_tile`.

Capitolo 13 -- AutoLISP: catalogo delle funzioni



Restituisce una stringa in cui è contenuto il numero della versione corrente di AutoLISP.

(**ver**)

La funzione **ver** deve essere utilizzata, insieme alla funzione **equal**, per controllare la compatibilità dei programmi. La stringa si presenta nel modo seguente:

```
"AutoLISP Versione X.X (nn) "
```

dove X.X è il numero della versione corrente e nn è la descrizione di due lettere del linguaggio.

```
(ver) restituisce ad esempio "AutoLISP Versione 14.0 (it) "
```

Esempi delle descrizioni dei linguaggi:

en) Amer/Inglese (es) Spagnolo (fr) Francese
(de) Tedesco (it) Italiano

Capitolo 13 -- AutoLISP: catalogo delle funzioni



Restituisce un elenco di descrittori di finestre per la configurazione della finestra corrente.

(**vports**)

Ciascun descrittore di finestre è un elenco costituito dal numero identificativo e dalle coordinate degli angoli inferiore sinistro e superiore destro della finestra.

Se la variabile di sistema TILEMODE di AutoCAD è impostata su 1 (attiva), nell'elenco restituito viene descritta la configurazione della finestra creata utilizzando il comando FINESTRE di AutoCAD. Gli angoli delle finestre sono espressi in valori compresi tra 0.0 e 1.0, con (0.0, 0.0) che rappresenta l'angolo inferiore sinistro dell'area di disegno dello schermo e (1.0, 1.0) l'angolo superiore destro. Se TILEMODE è uguale a 0 (disattiva), l'elenco restituito descrive gli oggetti della finestra creati mediante il comando FINMUL. Gli angoli dell'oggetto della finestra sono espressi nelle coordinate dello spazio carta. La finestra numero 1 è sempre lo spazio carta quando la variabile di sistema TILEMODE non è attiva.

Ad esempio, data una configurazione che prevede una sola finestra con TILEMODE impostata, la funzione **vports** potrebbe restituire quanto segue:

```
((1 (0.0 0.0) (1.0 1.0)))
```

In modo simile, date quattro finestre di dimensioni uguali posizionate nei quattro angoli dello schermo quando TILEMODE è attiva, la funzione **vports** potrebbe restituire quanto segue:

```
( (5 (0.5 0.0) (1.0 0.5))  
  (2 (0.5 0.5) (1.0 1.0))  
  (3 (0.0 0.5) (0.5 1.0))  
  (4 (0.0 0.0) (0.5 0.5)) )
```

Il descrittore della finestra corrente è sempre il primo dell'elenco. Nell'esempio precedente la finestra numero 5 indica la finestra corrente.

Capitolo 13 -- AutoLISP: catalogo delle funzioni



Confronta un modello di caratteri jolly con una stringa.

(wcmatch *stringa modello*)

La funzione **wcmatch** compara l'argomento *stringa* con l'argomento *modello* per vedere se corrispondono. In tal caso, viene restituito T; altrimenti, viene restituito nil. Entrambi gli argomenti possono essere stringhe racchiuse tra virgolette o variabili. L'argomento *modello* può contenere i caratteri corrispondenti al modello di caratteri jolly riportati nella tabella seguente. Vengono confrontati approssimativamente soltanto i primi 500 caratteri della *stringa* e del *modello*; gli altri caratteri vengono ignorati.

Caratteri jolly

Carattere	Definizione
# (cancellato)	Confronta qualsiasi singola cifra.
@ (chiocciola)	Confronta qualsiasi singolo carattere alfabetico.
. (punto)	Confronta qualsiasi carattere non alfanumerico.
* (asterisco)	Confronta qualsiasi sequenza di caratteri, compreso uno spazio e può essere utilizzato in un punto qualsiasi nel modello di ricerca; all'inizio, al centro o alla fine.
? (punto interrogativo)	Confronta qualsiasi singolo carattere.
~ (tilde)	Se è il primo carattere di un modello, esegue il confronto con qualsiasi carattere tranne con il modello stesso.
[...]	Confronta uno qualsiasi dei caratteri racchiusi tra parentesi.
[~...]	Confronta uno qualsiasi dei caratteri racchiusi tra parentesi.
- (trattino)	Utilizzato all'interno delle parentesi quadre per specificare un intervallo per un singolo carattere.
, (virgola)	Separa due modelli.
` (apice inverso)	Ignora caratteri speciali; legge letteralmente il carattere successivo.

```
(wcmatch "Nome" N*) restituisce T
```

Nell'esempio viene verificato se la stringa Nome inizia con il carattere N. È possibile utilizzare le virgole per immettere più condizioni. Questo esempio esegue tre confronti:

```
(wcmatch "Nome" "???,~*m*,N*") restituisce T
```

Se una qualsiasi delle tre condizioni è soddisfatta, la funzione **wcmatch** restituisce T. In tal caso, vengono eseguite tre verifiche: Nome ha tre caratteri (falso); Nome non contiene la lettera m (falso); Nome inizia con la lettera N (vero). Poiché almeno una delle condizioni è stata soddisfatta, la funzione restituisce T.

Il confronto distingue fra maiuscole e minuscole, pertanto i caratteri maiuscoli e minuscoli devono corrispondere. Si consiglia di utilizzare le variabili ed i valori restituiti dalle funzioni AutoLISP per i valori degli argomenti *stringa* e *modello*.

Per verificare la presenza di un carattere jolly in una stringa, è possibile utilizzare il carattere apice inverso (') per *ignorarlo*. Ignorare significa che il carattere successivo all'apice inverso non viene letto come carattere jolly, ma confrontato in base al valore effettivo. Ad esempio, se si desidera ricercare una virgola in un punto qualsiasi della stringa Nome, digitare:

```
(wcmatch "Nome" "*`,*") restituisce nil
```

Nota poiché potrebbero essere aggiunti altri caratteri jolly nei successivi release di AutoLISP, si consiglia di ignorare tutti i caratteri non alfanumerici nel modello in modo da garantire la compatibilità futura.

I linguaggi di programmazione C ed AutoLISP utilizzano la barra inversa (\) come carattere escape; quindi per specificare una barra inversa in una stringa è necessario immetterne due (\\). Per verificare la presenza di una barra inversa in un punto qualsiasi della stringa Nome, digitare:

```
(wcmatch "Nome" "*`\\*") restituisce nil
```

Tutti i caratteri racchiusi tra parentesi quadre ([. . .]) vengono letti letteralmente, pertanto, non è necessario ignorarli, con le seguenti eccezioni: il carattere tilde (~) viene letto letteralmente solo quando *non* è il primo carattere tra parentesi (come in "[A~BC]"); altrimenti viene letto come carattere di negazione, il che significa che la funzione **wcmatch** deve confrontare tutti i caratteri *tranne* quelli che seguono la tilde (come in "[~ABC]"). Il carattere trattino (-) viene letto letteralmente solo quando è il primo o l'ultimo carattere tra parentesi (come in "[-ABC]" o "[ABC-]") oppure quando segue un carattere tilde iniziale (come in "[~ABC]"). Altrimenti, il trattino (-) viene utilizzato all'interno delle parentesi quadre per specificare un intervallo di valori per un determinato carattere. L'intervallo agisce solo per i caratteri singoli; quindi "STR[1-38]" esegue il confronto di STR1, STR2, STR3 e STR8, mentre "[A-Z]" confronta tutte le lettere maiuscole.

Anche il carattere parentesi quadra chiusa ("]") viene letto letteralmente qualora sia il primo carattere tra parentesi oppure segua una tilde iniziale, come ad esempio in "[]ABC]" o "[~]ABC]".

Capitolo 13 -- AutoLISP: catalogo delle funzioni

while

Valuta un'espressione di verifica e, se risulta diversa da nil, ne valuta altre; questo processo viene iterato finché l'espressione valutata non restituisce nil.

```
(while espr_verifica espr...)
```

La funzione **while** continua fino a quando l'argomento *espr_verifica* è uguale a nil. A questo punto, la funzione restituisce il valore più recente dell'ultima *espr*.

Il codice riportato di seguito richiama la funzione utente **SOME-FUNC** dieci volte, utilizzando la variabile test impostata da 1 a 10. In seguito, restituisce 11, che rappresenta il valore dell'ultima espressione valutata.

```
(setq test 1)
(while (<= test 10)
  (some-func test)
  (setq test (1+ test))
)
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

write-char

Scrivere un carattere sullo schermo o in un file aperto.

```
(write-char num [desc_file])
```

L'argomento *num* è un codice ASCII decimale per il carattere da scrivere e rappresenta inoltre il valore restituito dalla funzione **write-char**.

```
(write-char 67) restituisce 67
```

Sullo schermo viene scritta la lettera C. Si supponga che f sia il descrittore per un file aperto:

```
(write-char 67 f) restituisce 67
```

e scrive la C in tale file.

I vari sistemi operativi su cui viene eseguito AutoCAD utilizzano convenzioni diverse per segnalare la fine di una riga in un file di testo ASCII. I sistemi UNIX, ad esempio, utilizzano un unico carattere di riga nuova (LF, codice ASCII 10), mentre i sistemi DOS utilizzano una coppia di caratteri (CR/LF, codici ASCII 13 e 10) per lo stesso scopo. Per facilitare lo sviluppo di programmi AutoLISP, la funzione **write-char** converte un carattere di riga nuova (codice ASCII 10) nel carattere di fine riga (o sequenza di caratteri) utilizzato dal sistema operativo in uso. Pertanto, su un sistema DOS,

```
(write-char 10 f) restituisce 10
```

ma scrive nel file la sequenza di caratteri CR/LF (codici ASCII 13 e 10). La funzione **write-char** non può scrivere un carattere NUL (codice ASCII 0) in un file.

Vedere anche

appendice A, "Codici ASCII", per un elenco dei codici ASCII.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

write-line

Scrivi una stringa sullo schermo o in un file aperto.

```
(write-line stringa [desc_file])
```

La funzione restituisce di solito una *stringa* racchiusa tra virgolette, ma quest'ultime vengono omesse quando tale stringa viene scritta in un file. Ad esempio, supponendo che *f* sia un descrittore valido per un file aperto:

```
(write-line "Verifica" f) scrive Verifica e restituisce "Verifica"
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni **xdroom**

Restituisce la quantità di spazio di dati estesi (Xdata) disponibile per un oggetto (entità).

```
(xdroom nome_ent)
```

Se l'esito è negativo, la funzione **xdroom** restituisce nil. Poiché la quantità massima di dati estesi che può essere assegnata ad una definizione di entità è attualmente di 16 Kb e poiché più applicazioni possono aggiungere dati estesi alla stessa entità, questa funzione serve per verificare se vi è spazio sufficiente per i dati estesi che verranno aggiunti. Può essere richiamata insieme alla funzione **xdsiz**; in tal caso, viene restituita la dimensione dell'elenco dei dati estesi.

Di seguito è riportato un esempio in cui la funzione verifica lo spazio disponibile per i dati estesi di un oggetto finestra. Supponendo che la variabile *vpname* contenga il nome dell'oggetto finestra:

```
(xdroom vpname) restituisce 16162
```

Nell'esempio precedente, sono disponibili 16.162 byte dei 16.383 byte originali dello spazio dei dati estesi, il che significa che vengono utilizzati 221 byte. Per determinare la quantità di spazio di dati disponibile è possibile utilizzare la funzione **xdsiz**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni **xdsiz**

Restituisce la dimensione in byte occupata da un elenco quando è collegato ad un oggetto (entità) sotto forma di dati estesi.

```
(xdsiz elenco)
```

Se l'esito è negativo, la funzione **xdsiz** restituisce nil. L'argomento *elenco* deve essere un elenco di dati estesi valido in cui è contenuto il nome di un'applicazione precedentemente registrata utilizzando la funzione **regapp**. I campi racchiusi tra parentesi graffe (codice di gruppo 1002) devono essere bilanciati. Un *elenco* non valido provoca un errore ed assegna il codice di errore appropriato nella variabile ERRNO. Se i dati estesi contengono il nome di un'applicazione non registrata, verrà visualizzato il messaggio di errore di seguito riportato, supponendo che la variabile di sistema CMDECHO sia attiva:

Nome dell'applicazione non valido nel gruppo 1001

L'*elenco* può iniziare con un codice di gruppo -3 (indicatore di dati estesi), ma non è obbligatorio. Poiché i dati estesi possono contenere informazioni provenienti da più applicazioni, l'elenco deve essere racchiuso tra parentesi.

```
(-3 ("MIAPP" (1000 . "ARMATURA")
          (1002 . "{")
          (1040 . 0.0)
          (1040 . 1.0)
          (1002 . "}")
      )
)
```

Di seguito è riportato un esempio in cui non viene utilizzato il codice di gruppo -3. Questo elenco è semplicemente la funzione **cdr** del primo gruppo, ma è importante che sia racchiuso tra parentesi:

```
( ("MIAPP" (1000 . "ARMATURA")
      (1002 . "{")
      (1040 . 0.0)
      (1040 . 1.0)
      (1002 . "}")
    )
  )
```

Capitolo 13 -- AutoLISP: catalogo delle funzioni

xload

Carica un'applicazione ADS.

(xload applicazione [se_fallisce])

L'argomento *applicazione* viene immesso come stringa racchiusa tra virgolette o come variabile contenente il nome di un file eseguibile. Quando il file viene caricato, viene verificato se è valido per l'applicazione ADS. Inoltre, viene verificato se la versione del programma ADS, l'applicazione ADS stessa e la versione di AutoLISP in esecuzione sono compatibili.

Se la funzione **xload** non riesce, di solito si verifica un errore AutoLISP. Tuttavia, se si fornisce l'argomento *se_fallisce*, la funzione **xload** restituisce il valore di tale argomento non riuscito al posto del messaggio di errore.

Se il caricamento dell'applicazione ha esito positivo, viene restituito il nome dell'applicazione.

```
(xload "/mieapp/ame") se riesce, restituisce "/mieapp/ame"
```

Se si tenta di caricare un'applicazione già caricata, la funzione **xload** visualizza il seguente messaggio:

Applicazione "*applicazione*" già caricata.

e restituisce il nome dell'applicazione. Prima di utilizzare la funzione **xload**, è possibile controllare le applicazioni ADS attualmente caricate mediante la funzione **ads**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

xunload

Scarica un'applicazione ADS.

(xunload applicazione [se_fallisce])

Se l'applicazione viene scaricata con successo, viene restituito il nome dell'applicazione, altrimenti viene visualizzato un messaggio di errore.

Digitare *applicazione* come stringa tra virgolette o come variabile contenente il nome di un'applicazione caricata con la funzione **xload**. Il nome dell'applicazione deve corrispondere esattamente a quello immesso per la funzione **xload**. Se è stato specificato un percorso, ossia il nome della directory, per l'applicazione nella funzione **xload**, tale percorso può essere omesso nella funzione **xunload**.

Nell'esempio seguente la funzione scaricherà l'applicazione precedente che è stata caricata mediante la funzione **xload**.

```
(xunload "ame") se l'esito è positivo, restituisce "ame"
```

Se la funzione **xunload** non riesce, di solito si verifica un errore AutoLISP. Tuttavia, se si fornisce l'argomento *se_fallisce*, la funzione **xunload** restituisce il valore di tale argomento non riuscito invece di inviare un messaggio di errore. Questa caratteristica è simile a quella della funzione **xload**.

Capitolo 13 -- AutoLISP: catalogo delle funzioni

zerop

Verifica che l' argomento sia uguale a zero.

(zerop *numero*)

La funzione restituisce T se la valutazione di *numero* è uguale a zero; altrimenti, restituisce nil.

(zerop 0)	restituisce	T
(zerop 0.0)	restituisce	T
(zerop 0.0001)	restituisce	nil

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

Panoramica

I comandi e le variabili di sistema di AutoCAD definiti dalle applicazioni ARX o AutoLISP vengono detti *definiti esternamente*. È possibile che le applicazioni AutoLISP accedano a questi comandi in modo diverso rispetto ai comandi incorporati di AutoCAD; molti dei comandi definiti esternamente sono provvisti della propria interfaccia di programmazione, che consente alle applicazioni AutoLISP di usufruire della loro funzionalità.

Per ulteriori informazioni sui comandi descritti in questo capitolo, vedere la *Guida di riferimento dei comandi di AutoCAD*.

Argomenti di questo capitolo

```
{button ,JI(','3DSIN_al_u0308')} 3DSIN
{button ,JI(','3DSOUT_al_u0308')} 3DSOUT
{ewc ,JI(','ALIGN_AL_U0308')} ALLINEA
{button ,JI(','ASEADMIN_AL_U0308')} ASEADMIN
{ewc ,JI(','ASEEXPORT_AL_U0308')} ASEEXPORT
{button ,JI(','ASELINKS_AL_U0308')} ASELINKS
{ewc ,JI(','ASEROWS_AL_U0308')} ASEROWS
{button ,JI(','ASESELECT_AL_U0308')} ASESELECT
{button ,JI(','ASESQLED_AL_U0308')} ASESQLED
{ewc ,JI(','BACKGROUND_AL_U0308')} SFONDO
{ewc ,JI(','CAL_AL_U0308')} CAL
{button ,JI(','FOG_AL_U0308')} NEBBIA
{button ,JI(','LIGHT_AL_U0308')} LUCE
{button ,JI(','LSEEDIT_AL_U0308')} MODPAES
{ewc ,JI(','LSLIB_AL_U0308')} LIBPAES
{button ,JI(','LSNEW_AL_U0308')} NPAES
{button ,JI(','MATLIB_AL_U0308')} LIBMAT
{ewc ,JI(','MIRROR3D_AL_U0308')} SPECCHIO3D
{ewc ,JI(','PSDRAG_AL_U0308')} PSTRASCINA
{ewc ,JI(','PSFILL_AL_U0308')} PSMOTIVO
{ewc ,JI(','PSIN_AL_U0308')} PSIN
{ewc ,JI(','RENDER_AL_U0308')} RENDER
```

```
{ewc ,JI(','RENDERUPDATE_AL_U0308')} RENDERUPDATE
{ewc ,JI(','REPLAY_AL_U0308')} REPLAY
{ewc ,JI(','RMAT_AL_U0308')} MATERIALE
{ewc ,JI(','ROTATE3D_AL_U0308')} RUOTA3D
{ewc ,JI(','RPREF_AL_U0308')} RPREF
{button ,JI(','SAVEIMG_AL_U0308')} SALVAIMM
{button ,JI(','SCENE_AL_U0308')} SCENA
{button ,JI(','SETUV_AL_U0308')} MAPPAGGIO
{button ,JI(','SHOWMAT_AL_U0308')} SHOWMAT
{button ,JI(','SOLPROF_AL_U0308')} SOLPROF
{button ,JI(','STATS_AL_U0308')} STATS
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema



Importa un file 3D Studio (.3ds).

(c:3dsin *modalità* [*più_mat creazione*] file)

L'argomento *modalità* è un numero intero che indica se il comando deve essere usato in modo interattivo (*modalità* = 1) o meno (*modalità* = 0). Se *modalità* è impostato a 0, oltre all'argomento *file* è necessario specificare anche *più_mat* e *creazione*. Se invece *modalità* è impostato a 1, è necessario indicare solo l'argomento *file*. L'argomento *file* è una stringa che indica il file .3ds da importare. È necessario specificare l'estensione .3ds del file.

La funzione seguente apre il file 3D Studio *globe.3ds* per importarlo e quindi richiede all'utente le specifiche di importazione.

```
(c:3dsin 1 "globe.3ds")
```

Gli argomenti *più_mat* e *creazione* sono numeri interi che indicano come trattare gli oggetti con più materiali e come creare nuovi oggetti.

Modalità 0 (non interattivo) dispone di alcuni argomenti che indicano come gestire la presenza di più materiali e come organizzare i nuovi oggetti. Questa modalità consente sempre di importare tutti gli oggetti contenuti nel file .3ds.

Valori degli argomenti più_mat e creazione

Argomento	Descrizione
più_mat	Indica come trattare gli oggetti con più materiali: 0-- Crea un nuovo oggetto per ogni materiale. 1 Assegna il primo materiale al nuovo oggetto.
creazione	Indica come creare nuovi oggetti: 0 Crea un layer per ogni oggetto di 3DS. 1 Crea un layer per ogni colore di 3DS. 2 Crea un layer per ogni materiale di 3DS. 3 Posiziona tutti i nuovi oggetti su un singolo layer.

La funzione riportata di seguito importa l'intero contenuto di *globe.3ds* senza nessun input da parte dell'utente, dividendo gli oggetti con più materiali e posizionando tutti i nuovi oggetti sullo stesso layer.

```
(c:3dsin 0 0 3 "globe.3ds")
```

Funzione definita esternamente applicazione ARX render

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

3DSOUT

Esporta un file 3D Studio.

(c:3dsout gsel modalità_o div levigatezza giunzione file)

Per utilizzare la funzione **c:3dsout** è necessario specificare tutti i suoi sei argomenti. L'argomento *ssel* rappresenta un gruppo di selezione nel quale sono contenuti gli oggetti AutoCAD da esportare. L'argomento *modalità_o* è un numero intero che indica la modalità di output per la rappresentazione dei dati di AutoCAD. Attualmente, l'output 3DSOUT è sempre lo stesso indipendentemente dal fatto che *modalità_o* sia impostata su 0 o su 1. L'argomento *div* è un numero intero che indica come dividere gli oggetti di AutoCAD in oggetti di 3D Studio. Se *div* è impostato a 0, viene creato un oggetto per ciascun layer di AutoCAD. Se invece *div* è impostato a 1, viene creato un oggetto per ciascun colore di AutoCAD, e se impostato a 2, viene creato un oggetto per ciascun tipo di oggetto di AutoCAD. L'argomento *levigatezza* è un numero intero che indica l'angolo limite per la levigatezza automatica. Se *levigatezza* è impostato a -1, la levigatezza automatica non viene eseguita; se è impostato ad un valore compreso tra 0 e 360, AutoCAD effettua la levigatezza quando l'angolo compreso tra le linee parallele della faccia è inferiore al valore impostato. L'argomento *giunzione* è un numero reale che specifica la distanza limite per la congiunzione dei vertici adiacenti. Se *giunzione* è impostato ad un valore inferiore a 0, la giunzione è disabilitata; se invece è impostato ad un valore uguale o maggiore di 0, AutoCAD effettua la giunzione dei vertici ancora più ravvicinati rispetto al valore impostato. L'ultimo argomento, *file*, è una stringa che indica il nome del file 3D Studio da creare. È necessario specificare l'estensione .3ds.

La funzione riportata di seguito esporta l'intero contenuto del disegno, creando gli oggetti di 3D Studio in base al layer del disegno, utilizzando il limite di levigatezza di 30 gradi e la distanza di giunzione di 0.1.

```
(c:3dsout (ssel "X") 0 0 30 0.1 "testav.3ds")
```

Funzione definita esternamente applicazione ARX render

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

ALLINEA

Inclina e ruota gli oggetti consentendo di allinearli con altri oggetti.

(allinea arg1 arg2 ...)

L'ordine, il numero ed il tipo di argomenti per la funzione **allinea** sono uguali a quelli che verrebbero immessi alla riga di comando. Per indicare una risposta nulla (RETURN) usare nil o una stringa vuota (""). Se l'esito è positivo, la funzione restituisce il valore T; altrimenti restituisce nil.

Nell'esempio che segue vengono specificate due coppie di punti di origine e destinazione, consentendo di eseguire un movimento bidimensionale.

```
(setq gs (ssel))
(allinea gs o1 d1 o2 d2 "" "2d")
```

Nota il supporto di AutoLISP per la funzione **allinea** viene implementato mediante l'uso della libreria SAGET.

Funzione definita esternamente applicazione ARX geom3d

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

ASEADMIN

Esegue le funzioni di gestione per i comandi di database esterni.

(command "_aseadmin" arg1 arg2 ...)

L'ordine, il numero ed il tipo di argomenti sono esattamente uguali a quelli che vengono digitati alla riga di comando.

Funzione definita esternamente applicazione ARX ase

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

ASEEXPORT

Esporta le informazioni sul collegamento degli oggetti selezionati.

(command "_aseexport" *arg1 arg2* ...)

L'ordine, il numero ed il tipo di argomenti sono esattamente uguali a quelli che vengono digitati alla riga di comando.

Funzione definita esternamente applicazione ARX ase

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

ASELINKS

Manipola i collegamenti tra gli oggetti ed i dati dei database esterni.

(command "_aselinks" *arg1 arg2* ...)

L'ordine, il numero ed il tipo di argomenti sono esattamente uguali a quelli che vengono digitati alla riga di comando.

Funzione definita esternamente applicazione ARX ase

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

ASEROWS

Manipola i dati delle tabelle di database esterni.

(command "_aserows" *arg1 arg2* ...)

L'ordine, il numero ed il tipo di argomenti sono esattamente uguali a quelli che vengono digitati alla riga di comando.

Funzione definita esternamente applicazione ARX ase

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

ASESELECT

Crea un gruppo di selezione dai gruppi di selezione di testo e di grafica.

(command "_aseselect" *arg1 arg2* ...)

L'ordine, il numero ed il tipo di argomenti sono esattamente uguali a quelli che vengono digitati alla riga di comando.

Funzione definita esternamente applicazione ARX ase

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

ASESQLED

Esegue le istruzioni SQL

(command "_asesqled" *arg1 arg2 ...*)

L'ordine, il numero ed il tipo di argomenti sono esattamente uguali a quelli che vengono digitati alla riga di comando.

Funzione definita esternamente applicazione ARX ase

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

SFONDO

Imposta lo sfondo di un'immagine sottoposta a rendering.

(c:sfondo *modalità [opzioni]*)

Il comando SFONDO consente di impostare lo sfondo di un'immagine sottoposta a rendering. La sua funzione presenta quattro modalità, descritte nella tabella riportata di seguito. Ogni modalità viene specificata con un argomento della stringa; gli argomenti *opzione* dipendono dalla *modalità*.

Modalità del comando SFONDO

Modalità	Descrizione
SOLID	Per lo sfondo usa un solo colore.
GRADIENT	Genera uno sfondo con tre colori sfumati.
IMAGE	Per lo sfondo usa un file immagine.
MERGE	Unisce lo schermo corrente.

Funzione definita esternamente applicazione ARX render

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

SFONDO

SOLID

Questa modalità specifica se per lo sfondo viene usato un colore in tinta unita specifico oppure il colore di sfondo definito da AutoCAD.

(c:sfondo "solid" "acad" | *colore*)

Nella tabella riportata di seguito vengono descritti gli argomenti della modalità Solido.

Argomenti della modalità SOLIDO

Argomento	Tipo di dati	Descrizione	Default
acad	STR	Specifica che deve essere usato il colore di sfondo di AutoCAD.	"ACAD"
colore	LIST (di numeri reali)	Colore da impostare	Nessuno

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

SFONDO

GRADIENT

Questa modalità genera uno sfondo a due o tre colori sfumato e ruotato di qualsiasi angolo.

```
(c:sfondo "inclinato" colore1 colore2 colore3 [angolo
[orizzonte [altezza]])
```

Nella tabella riportata di seguito vengono descritti gli argomenti della modalità Inclinato:

Argomenti della modalità GRADIENT

Argomento	Tipo di dati	Descrizione	Default
colore1	LIST (di numeri reali)	Colore della banda superiore della sfumatura	Nessuno
colore2	LIST (di numeri reali)	Colore della banda centrale della sfumatura	Nessuno
colore3	LIST (di numeri reali)	Colore della banda inferiore della sfumatura	Nessuno
angolo	REAL	Rotazione in senso antiorario dell'orizzonte	0.0
orizzonte	REAL	Centro di una sfumatura a due colori o della banda centrale di una sfumatura a tre colori	0.5
altezza	REAL	Percentuale dell'altezza (0.0->1.0) della banda centrale della sfumatura.	0.3

```
(c:sfondo "gradient"
' (1.0 0.0 0.0) ' (0.0 1.0 0.0) ' (0.0 0.0 1.0) (30.0 0.6 0.1))
```

Genera una sfumatura a tre colori di cui il colore superiore è rosso, quello centrale è verde e quello inferiore è blu, e ruotata di 30 gradi con una barra centrale verde stretta (10%) centrata a 60% dal fondo.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

SFONDO

IMAGE

Con questa modalità per lo sfondo viene usato un file immagine che è possibile scalare arbitrariamente o in modo che rientri nello schermo, incasellare o ritagliare.

```
(c:sfondo "image" file [adatta [angolo | [scalax
[scalay [sfalsamentox [sfalsamentoy [casella]]]]]]))
```

Nella tabella riportata di seguito vengono descritti gli argomenti della modalità Immagine:

Argomenti della modalità IMMAGINE

Argomento	Tipo di dati	Descrizione	Default
file	STR	Nome del file	Nessuno
adatta	STR	Modalità Fit FIT: riscalda sia l'asse x che l'asse y per adattare l'output FIT ASPECT: riscalda con scale x e y uguali in modo che la dimensione più piccola sia adatta all'output	FIT
angolo	REAL	Ignorato	0.0
scalax	REAL	Scala x	1.0
scalay	REAL	Scala y	1.0
sfalsamentox	REAL	Sfalsamento x (differenza tra centro dell'immagine e centro dell'output)	0.0
sfalsamentoy	REAL	Sfalsamento y	0.0
casella	INT	Incasellamento:	1

0 = ritaglio
1 = inserimento

```
(c:sfondo "image" "valle_1.tga" "FIT")
```

Usa valle_1.tga come sfondo riscaldando l'immagine in modo da adattarla allo schermo.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

SFONDO

MERGE

Questa modalità viene usata per unire lo sfondo della vista corrente come nuova immagine di sfondo. Non vi è associato alcun parametro.

```
(c:sfondo "merge")
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

SFONDO

ENVIRONMENT

Questa modalità imposta la mappa di riflessione globale sul file specificato o sullo sfondo, immagine, sfumatura o colore in tinta unita corrente.

```
(c:sfondo "environment" blocco | file)
```

Nella tabella riportata di seguito vengono descritti gli argomenti della modalità Environment.

Argomenti della modalità ENVIRONMENT

Argomento	Tipo di dati	Descrizione	Default
blocco	STR	Indica che la mappa di ambiente globale deve essere bloccata sullo sfondo corrente. Il parametro è "LOCK".	LOCK
file	STR	Nome del file di ambiente.	Nessuno

```
(c:background "environment" "lock")
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

CAL

Richiama il calcolatore geometrico in linea e restituisce il valore dell'espressione risolta.

```
(cal espressione)
```

L'argomento *espressione* è una stringa tra virgolette. Vedere "CAL" nella *Guida di riferimento dei comandi di AutoCAD* per una descrizione dei valori validi.

Nell'esempio riportato di seguito, la funzione CAL viene utilizzata in una espressione di AutoLISP, insieme alla funzione **trans**:

```
(trans (cal "[1,2,3]+MID") 1 2)
```

Funzione definita esternamente applicazione ARX geomcal

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

NEBBIA

Consente di aggiungere una distanza dalla macchina fotografica.

(c:nebbia enabled [colore [dist_vicina [dist_lontana [percent_vicina [percent_lontana [sfondo]]]]]])

Il comando NEBBIA consente di fornire informazioni visive sulla distanza degli oggetti dalla macchina fotografica. Per rendere massimo l'effetto del comando, aggiungere bianco all'immagine; per massimizzare l'effetto di profondità, aggiungere nero. A questo comando sono associate sette modalità, descritte nella tabella riportata di seguito.

Argomenti della modalità NEBBIA

Argomento	Tipo di dati	Descrizione	Default
enabled	stringa	Attiva e disattiva la nebbia senza alterare le altre impostazioni.	ON (attiva nebbia)
colore	punto 3D	Le impostazioni del colore corrispondono ai valori AutoCAD standard per i colori.	(111)
dist_vicina	REAL	Definisce l'inizio dell'effetto nebbia.	0.0
dist_lontana	REAL	Definisce il punto finale dell'effetto nebbia.	1.0
percent_vicina	REAL	Definisce la percentuale dell'effetto nebbia all'inizio del banco.	0.0
percent_lontana	REAL	Definisce la percentuale dell'effetto nebbia alla fine del banco.	1.0
sfondo	stringa	Applica l'effetto nebbia sia allo sfondo che alla geometria.	OFF (non applica l'effetto nebbia allo sfondo)

Gli argomenti nil o finali mancanti non vengono modificati.

Funzione definita esternamente applicazione ARX render

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

LUCE

Crea, modifica e cancella luci ed effetti di illuminazione.

(c:luce modalità [opzioni])

Con il comando LUCE, è possibile aggiungere una luce nuova e modificare o cancellare una luce esistente. La sua funzione presenta otto modalità, descritte nella tabella riportata di seguito. Ogni modalità viene specificata con un argomento della stringa; gli argomenti opzione dipendono dalla modalità.

Modalità del comando LUCE

Modalità	Descrizione
A	Imposta o ripristina l'intensità della luce circostante.
D	Cancella le luci esistenti.
L	Elenca tutte le luci contenute nel disegno oppure restituisce una definizione per la luce specificata.
M	Modifica le luci esistenti.
ND	Crea una nuova luce distante.
NP	Crea una nuova luce puntiforme.
NS	Crea un nuovo riflettore.
R	Rinomina una luce esistente.

Nota non è consentito usare questo comando nello spazio carta.

Funzione definita esternamente applicazione ARX render

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

LUCE

A--Ambient (circostante)

Imposta o ripristina l'intensità della luce circostante.

```
(c:luce "A" [intensità [colore ]])
```

L'argomento *intensità* è un numero reale compreso tra 0.0 e 1.0; se *intensità* viene omissso, il valore di default è 1.0. L'argomento *colore* è un elenco che indica i tre valori RGB; se viene omissso, i valori di default sono (1.0 1.0 1.0).

Ad esempio, nel codice riportato di seguito l'intensità della luce circostante viene impostata a 0.6.

```
(c:luce "A" 0.6)
```

Per ripristinare l'intensità della luce circostante corrente, non passare l'argomento *intensità*.

```
(c:luce "A") restituisce (0.3 (1.0 1.0 1.0)), che corrisponde all'intensità 0.3,  
e al colore (1.0 1.0 1.0)
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

LUCE

D--Delete (cancella)

Cancella le luci esistenti.

```
(c:luce "D" nome)
```

L'argomento *nome* è una stringa che indica il nome della luce da cancellare.

```
(c:luce "D" "LCVEC")
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

LUCE

L--List (elenca)

Elenca tutte le luci contenute nel disegno oppure restituisce una definizione per la luce specificata.

```
(c:luce "L" [nome])
```

L'argomento *nome* è una stringa che indica il nome della luce da cancellare. Se l'argomento *nome* viene omissso, **c:luce** restituisce l'elenco di tutte le luci definite nel disegno. Quando invece *nome* viene specificato, **c:luce** restituisce la definizione della luce specificata.

```
(c:luce "L") restituisce ("RIFLETT1")  
(c:luce "L" "RIFLETT1") restituisce:  
("S" <Nome entità: 600039c6> 157.22 (-97.8002 1.21954e-14 70.0222)  
(13.8456 20.103 0.0) (0.944 0.89824 0.736) 0 12.0 16.0 3.0 "ON")
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema



Modifica le luci esistenti.

```
(c:luce "M" nome [intensità [da [a [colore
[dimensmappaombra [brillanza [caduta [attenuazioneombra
[ombra [oggettiombra [mese [giorno [ore [minuti
[luce diurne [latitudine [longitudine
[attenuazione]]]]]]]]]]]]]]]]]]]]))
```

Nella tabella riportata di seguito vengono descritti gli argomenti della modalità Modify.

Argomenti della modalità "M" del comando LUCE

Argomento	Tipo di dati	Descrizione	Default
nome	STR	Rappresenta il nome unico della luce.	Nessuno
intensità	REAL	È un numero reale compreso tra 0.0 ed il valore massimo di default.	In base al valore della attenuazione.
da	LIST	Definisce la posizione della luce.	Vista corrente dal punto.
to	LIST	Definisce la destinazione della luce.	Vista corrente al punto.
colore	LIST	È un gruppo di tre valori RGB qualsiasi.	1.0, 1.0, 1.0
dimensmappaombre	INT	Intero compreso tra 0 e 4096 (le dimensioni in pixel di un lato della mappa ombre)	0
brillanza	REAL	Definisce l'angolo del raggio di luminosità espresso in gradi (deve essere compreso tra 1 e 160).	44.0
caduta	REAL	Definisce l'angolo che include l'area di diminuzione rapida, espresso in gradi (deve essere compreso tra 0 e 160 e deve essere maggiore del valore dell'area di eccessiva brillantezza).	45.0
attenuazioneombra	REAL	Numero reale compreso tra 0.0 e 10.00.0	0.0
ombra	STR	Interruttore per l'applicazione e la rimozione dell'ombra. I valori validi sono: "off" = nessuna ombra, "on" = applica ombra.	0.0
oggettiombra	ENAME	Una selezione di oggetti che racchiudono la mappa ombre.	0.0
mese	INT	Intero compreso tra 1 e 12	9
giorno	INT	Intero compreso tra 1 e 31	21
ore	INT	Intero compreso tra 0 e 24	15
minuti	INT	Intero compreso tra 0 e 59	0
luce diurna	STR	Interruttore per il risparmio di luce diurna. I valori validi sono: "off" = nessun risparmio, "on" = applica il risparmio.	"off"
latitudine	REAL	Numero reale compreso tra 0 e 90	37.62
longitudine	REAL	Numero reale compreso tra 0 e 180	122.37
fuso_orario	INT	Intero compreso tra -12 e 12 che rappresenta l'ora media di Greenwich o GMT (Greenwich Mean Time).	8 (PST)
attenuazione	INT	0 = nessuna attenuazione 1 = attenuazione lineare inversa 2 = attenuazione quadra inversa.	1

Gli argomenti brillantezza e caduta vengono applicati solo ai riflettori. di conseguenza, durante la creazione di una nuova luce puntiforme, è necessario passarli come nil.

Se un argomento è nil oppure è stato ommesso alla fine dell'elenco di argomenti, esso restituisce il suo valore corrente. Se un argomento non può essere applicato al tipo di luce modificato, passarlo come nil oppure ometterlo se si trova alla fine dell'elenco degli argomenti.

Ad esempio, con il codice riportato di seguito il colore della luce distante che si chiama D1 viene modificato in blu.

(c:luce "M" "D1" nil nil nil '(0.0 0.0 1.0))

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema



ND-New Distant Light (nuova luce distante)

Crea una nuova luce distante.

(c:luce "ND" nome [intensità [da [a [colore]]])

Nella tabella riportata di seguito vengono descritti gli argomenti della modalità Nuova luce distante.

Argomenti della modalità "ND" del comando LUCE

Argomento	Tipo di dati	Descrizione	Default
nome	STR	Rappresenta il nome unico della luce.	Nessuno
intensità	REAL	È un numero reale compreso tra 0.0 ed il valore massimo di default.	In base al valore della attenuazione.
da	LIST	Definisce la posizione della luce.	Vista corrente dal punto.
a	LIST	Definizione la destinazione della luce.	Vista corrente al punto.
color	LIST	È un gruppo di tre valori RGB qualsiasi.	1.0, 1.0, 1.0
dimensmappao mbre	INT	Intero compreso tra 0 e 4096 (le dimensioni in pixel di un lato della mappa ombre).	0
brillanza	REAL	Definisce l'angolo del raggio di luminosità espresso in gradi (deve essere compreso tra 1 e 160).	44.0
caduta	REAL	Definisce l'angolo che include l'area di diminuzione rapida, espresso in gradi (deve essere compreso tra 0 e 160 e deve essere maggiore del valore dell'area di eccessiva brillantezza).	45.0
attenuazioneom bra	REAL	Numero reale compreso tra 0.0 e 10.0	0.0
ombra	STR	Interruttore per l'applicazione e la rimozione dell'ombra I valori validi sono: "off" = nessuna ombra, "on" = applica ombra.	0.0
oggettiombra	ENAME	Una selezione di oggetti che racchiudono la mappa ombre.	0.0
mese	INT	Intero compreso tra 1 e 12	9
giorno	INT	Intero compreso tra 1 e 31	21
ore	INT	Intero compreso tra 0 e 24	15
minuti	INT	Intero compreso tra 0 e 59	0
luce diurna	STR	Interruttore per il risparmio di luce diurna. "off" I valori validi sono: "off" = nessun risparmio, "on" = applica il risparmio.	
latitudine	REAL	Numero reale compreso tra 0 e 90	37.62
longitudine	REAL	Numero reale compreso tra 0 e 180	122.37
fuso_orario	INT	Intero compreso tra -12 e 12 che rappresenta l'ora media di Greenwich o GMT (Greenwich Mean Time).	8 (PST)

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema



NP-New Point Light (nuova luce puntiforme)

Crea una nuova luce puntiforme.

```
(c:luce "ND" nome [intensità [da [nil [colore
[dimensmappaombre [nil [nil [attenuazioneombra [ombra
[attenuazione[oggettiombra ]]]]]]]]]))
```

Nella tabella riportata di seguito vengono descritti gli argomenti della modalità Nuova luce puntiforme.

Argomenti della modalità "NP" del comando LUCE

Argomento	Tipo di dati	Descrizione	Default
nome	STR	Rappresenta il nome unico della luce.	Nessuno
intensità	REAL	È un numero reale compreso tra 0.0 ed il valore massimo di default.	In base al valore della attenuazione.
da	LIST	Definisce la posizione della luce.	Vista corrente dal punto.
color	LIST	È un gruppo di tre valori RGB qualsiasi.	1.0, 1.0, 1.0
dimensmappaombre	INT	Intero compreso tra 0 e 4096 (le dimensioni in pixel di un lato della mappa ombre).	0
attenuazioneombra	REAL	Numero reale compreso tra 0.0 e 10.0.	0.0
ombra	STR	Interruttore per l'applicazione e la rimozione dell'ombra. I valori validi sono: "off" = nessuna ombra, "on" = applica ombra.	0.0
attenuazione	INT	0 = nessuna attenuazione 1 = attenuazione lineare inversa 2 = attenuazione quadrata inversa.	1
oggettiombra	ENAME	Una selezione di oggetti che racchiudono la mappa ombre	0.0

I tre argomenti a (dopo da), *brillanza* e *caduta* (dopo *dimensmappaombre*) non vengono applicati alle luci puntiformi. di conseguenza, durante la creazione di una nuova luce puntiforme, è necessario passarli come nil.

Ad esempio, con il codice riportato di seguito viene creata una nuova luce puntiforme con il nome NUOVOPT1.

```
(c:luce "NP" "NUOVOPT1")
```

A NUOVOPT1 verrà assegnato il valore di default dell'intensità, l'impostazione corrente dell'attenuazione, la posizione di default che visualizza la vista corrente ed il colore bianco di default.

Nota Per le luci puntiformi, il massimo valore di default relativo all'intensità dipende dall'impostazione corrente dell'attenuazione del punto e del riflettore. Senza attenuazione il valore dell'intensità è 1.00; con l'attenuazione di lineare inverso sarà la distanza massima tra i limiti del disegno e con l'attenuazione di quadrato inverso sarà il quadrato della distanza massima tra i limiti.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema



Crea un nuovo riflettore.

```
(c:luce "NS" nome [intensità [da [a [colore
[dimensmappaombre [brillanza [caduta [attenuazioneombra
[ombra [attenuazione [oggettiombra]]]]]]]]]]))
```

Nella tabella riportata di seguito vengono descritti gli argomenti della modalità Nuova spotlight.

Argomenti della modalità "NS" del comando LUCE

Argomento	Tipo di dati	Descrizione	Default
nome	STR	Rappresenta il nome unico della luce.	Nessuno
intensità	REAL	È un numero reale compreso tra 0.0 ed il valore massimo di default.	In base al valore della attenuazione.
da	LIST	Definisce la posizione della luce.	Vista corrente dal punto.
a	LIST	Definizione la destinazione della luce.	Vista corrente al punto.
colore	LIST	È un gruppo di tre valori RGB qualsiasi.	1.0, 1.0, 1.0
dimensmappaom	INT	Intero compreso tra 0 e 4096 (le dimensioni	0

<i>bre</i>		<i>in pixel di un lato della mappa ombre).</i>	
<i>brillanza</i>	REAL	<i>Definisce l'angolo del raggio di luminosità espresso in gradi (deve essere compreso tra 1 e 160).</i>	44.0
<i>caduta</i>	REAL	<i>Definisce l'angolo che include l'area di diminuzione rapida, espresso in gradi (deve essere compreso tra 0 e 160 e deve essere maggiore del valore dell'area di eccessiva brillantezza).</i>	45.0
<i>attenuazioneombra</i>	REAL	<i>Numero reale compreso tra 0.0 e 10.0</i>	0.0
<i>ombra</i>	STR	<i>Interruttore per l'applicazione e la rimozione dell'ombra. I valori validi sono: "off" = nessuna ombra, "on" = applica ombra</i>	0.0
<i>attenuazione</i>	INT	<i>0 = nessuna attenuazione 1 = attenuazione lineare inversa 2 = attenuazione quadra inversa.</i>	1
<i>oggettiombra</i>	ENAME	<i>Una selezione di oggetti che racchiudono la mappa ombre</i>	0.0

Ad esempio, con il codice riportato di seguito viene creato un nuovo riflettore con il nome NUOVOSPOT.

```
(c:luce "NS" "NUOVOSPOT" 137.82 '(12.0 6.0 24.0)
 '(78.0 78.0 24.0) nil nil 30.0 32.0)
```

NUOVOSPOT è un riflettore con una intensità di 137.82. Il suo colore è quello bianco di default, la sua posizione è (12,6,24) e la sua destinazione è (78,78,24); il suo cono è ampio 32 gradi, con un'area di eccessiva brillantezza di 30 gradi.

Nota Per i riflettori, il massimo valore di default relativo all'intensità dipende dall'impostazione corrente dell'attenuazione della luce puntiforme/riflettore. Senza attenuazione il valore dell'intensità è 1.00; con l'attenuazione di lineare inverso sarà la distanza massima tra i limiti del disegno e con l'attenuazione di quadrato inverso sarà il quadrato della distanza massima tra i limiti.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

LUCE

R--Rename (rinomina)

Rinomina una luce.

```
(c:luce "R" nome_vecchio nome_nuovo)
```

L'argomento *nome_vecchio* è una stringa che indica il nome della luce da rinominare. L'argomento *nome_nuovo* è una stringa che indica il nome nuovo per la luce.

```
(c:luce "R" "PUNTO5" "SIDEPT")
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

MODPAES

Crea o modifica oggetti di paesaggio.

```
(c:modpaes modalità [opzioni])
```

Il comando MODPAES consente di creare o modificare istanze di oggetti di paesaggio del disegno.

```
(c:modpaes "LIST" oggetto)
```

Elenca gli attributi dell'oggetto di paesaggio specificato.

```
(c:modpaes altezza oggetto [posizione [allineamento]])
```

Modifica un oggetto con orientamento orizzontale; *oggetto* è il nome AutoCAD (entsel) dell'oggetto da modificare. L'altezza, la

posizione e l'allineamento possono essere passati come valori nil; in questo caso il valore rimane inalterato.

Nella tabella riportata di seguito vengono descritti gli argomenti del comando MODPAES.

Argomenti del comando MODPAES

Argomento	Tipo di dati	Descrizione	Default
oggetto	ENAME	Gestore dell'oggetto di paesaggio.	Nessuno
altezza	REAL	Altezza dell'oggetto espressa in unità di disegno.	Nessuno
posizione	LIST (di numeri reali)	La posizione della base dell'oggetto.	Nessuno
allineamento	INT	Specifica la geometria e l'allineamento dell'elemento: 0 = a faccia singola allineato alla macchina fotografica 1 = a faccia singola non allineato alla macchina fotografica 2 = a facce incrociate non allineato alla macchina fotografica 3 = a facce incrociate allineato alla macchina fotografica	Nessuno

```
(c:modpaes oggetto altezza [posizione [allineamento]])
```

Modifica un oggetto di paesaggio; oggetto è il nome AutoCAD (entsel) dell'oggetto da modificare. L'altezza, la posizione e l'allineamento possono essere passati come valori nil; in questo caso il valore rimane inalterato.

```
(c:modpaes <nome_ent> 35.0 '(10.0 23.0) nil)
```

Modifica un oggetto di paesaggio; <nome_ent> è il nome AutoCAD (entsel) dell'oggetto da modificare. L'altezza, la posizione e l'allineamento possono essere passati come valori nil; in questo caso il valore rimane inalterato.

```
(c:modpaes "LIST" <nome_ent>)
```

Restituisce un elenco con il nome, l'altezza, la posizione e l'allineamento alla camera fotografica dell'oggetto specificato.

Funzione definita esternamente applicazione ARX render

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

LIBPAES

Gestisce la libreria di paesaggio.

```
(c:libpaes modalità [opzioni])
```

Nella tabella riportata di seguito vengono descritti gli argomenti del comando LIBPAES.

Argomenti di LIBPAES

Modalità	Descrizione
ADD	Aggiunge un elemento nella libreria di paesaggio.
DELETE	Elimina un elemento dalla libreria di paesaggio.
MODIFY	Modifica un elemento incluso nella libreria di paesaggio.
OPEN	Apri una libreria di paesaggio.
SAVE	Salva la libreria di paesaggio corrente.
LIST	Elenca gli elementi inclusi nella libreria di paesaggio corrente.

Funzione definita esternamente applicazione ARX render.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

 **LIBPAES**

 **ADD**

Aggiunge un elemento nella libreria corrente.

`(c:libpaes "ADD" nome mappa di composizione mappa di opacità allineamento)`

Nella tabella riportata di seguito vengono descritti gli argomenti della modalità ADD.

LSLIB: argomenti di "ADD"

Argomento	Tipo di dati	Descrizione	Default
nome	STR	Nome dell'elemento della libreria di paesaggio.	Nessuno
mappa di composizione	STR	Nome del file immagine dell'elemento.	Nessuno
mappa di opacità	STR	Nome dell'immagine opacità dell'elemento.	Nessuno
allineamento	INT	Specifica la geometria e l'allineamento dell'elemento: 0 = a faccia singola allineato alla macchina fotografica 1 = a faccia singola non allineato alla macchina fotografica 2 = a facce incrociate non allineato alla macchina fotografica 3 = a facce incrociate allineato alla macchina fotografica	Nessuno

`(c:libpaes "ADD" "Acero" "acero.tga" "acero.tga" 0)`

Aggiunge l'elemento "Acero" nella libreria di paesaggio corrente.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

 **LIBPAES**

 **DELETE**

Rimuove un elemento dalla libreria corrente.

`(c:libpaes "DELETE" nome)`

Alla modalità DELETE è associato un solo argomento descritto nella seguente tabella.

LIBPAES: argomenti di "DELETE"

Argomento	Tipo di dati	Descrizione	Default
nome :	STR	Nome dell'elemento della libreria di paesaggio.	Nessuno

`(c:libpaes "delete" "Acero")`

Rimuove l'elemento "Acero" dalla libreria di paesaggio corrente.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

 **LIBPAES**

 **MODIFY**

`(c:libpaes "MODIFY" nome mappa di composizione [mappa di opacità [allineamento]])`

Modifica un elemento nella libreria corrente; mappa di composizione, mappa di opacità e allineamento possono essere passati come valori nil; in questo caso il valore rimane inalterato.

Nella tabella riportata di seguito vengono descritti gli argomenti della modalità MODIFY.

LIBPAES: argomenti di "MODIFY"

Argomento	Tipo di dati	Descrizione	Default
nome	STR	Nome dell'elemento della libreria di paesaggio.	Nessuno
mappa di composizione	STR	Nome del file immagine dell'elemento.	Nessuno
mappa di opacità	STR	Nome dell'immagine opacità dell'elemento.	Nessuno
allineamento	INT	Specifica la geometria e l'allineamento dell'elemento: 0 = a faccia singola allineato alla macchina fotografica 1 = a faccia singola non allineato alla macchina fotografica 2 = a facce incrociate non allineato alla macchina fotografica 3 = a facce incrociate allineato alla macchina fotografica	Nessuno

```
(c:libpaes "MODIFY" "Acero" nil nil 2)
```

Modifica l'elemento "Acero" in modo che sia a facce incrociate e nonallineato alla macchina fotografica.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema



Pre una nuova libreria rendendola corrente.

```
(c:libpaes "OPEN" nome)
```

Alla modalità OPEN è associato un solo argomento descritto nella seguente tabella.

LIBPAES: argomenti di "OPEN"

Argomento	Tipo di dati	Descrizione	Default
nome :	STR	Nome del file della libreria di paesaggio da aprire.	Nessuno

```
(c:libpaes "OPEN" "ALBERI.LLI")
```

Legge il file ALBERI.LLI rendendolo la libreria di paesaggio corrente.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema



Salva la libreria di paesaggio corrente con il nome specificato.

```
(c:libpaes "SAVE" nome)
```

Alla modalità SAVE è associato un solo argomento descritto nella seguente tabella.

LIBPAES: argomenti di "SAVE"

Argomento	Tipo di dati	Descrizione	Default
nome :	STR	Nome del file della libreria di paesaggio da salvare.	Nessuno

```
(c:lslib "SAVE" "ALBERI.LLI")
```

Scriva il file ALBERI.LLI

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema



Elenca tutti gli elementi inclusi nella libreria corrente. A questo comando non è associato alcun argomento. L'elenco include elementi di paesaggio nel seguente formato ('(NOME "MAPPA-COMP" "MAPPA-OPAC" ALLINEAM).

`(c:libpaes "LIST")`

Ad esempio (esempio di output):

```
(("Siepe n1" "8siepe021.tga" "8siepe02o.tga" 0)
("Cactus" "8plnt151.tga" "8plnt15o.tga" 0)
("Betulla" "8albero391.tga" "8albero39o.tga" 0))
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema



Crea oggetti di paesaggio.

`(c:npaes modalità [opzioni])`

Il comando NPAES consente di creare istanze di oggetti di paesaggio del disegno.

`(c:npaes tipo_oggetto altezza posizione allineamento)`

Nella tabella riportata di seguito vengono descritti gli argomenti del comando NPAES.

Argomenti del comando NPAES

Argomento	Tipo di dati	Descrizione	Default
<i>tipo_oggetto</i>	STR	Nome dell'elemento della libreria di paesaggio.	Nessuno
<i>altezza</i>	REAL	Altezza dell'oggetto espressa in unità di disegno.	Nessuno
<i>posizione</i>	LIST (elenco di numeri reali)	La posizione della base dell'oggetto.	Nessuno
<i>allineamento</i>	INT	Specifica la geometria e l'allineamento dell'elemento: 0 = a faccia singola allineato alla macchina fotografica 1 = a faccia singola non allineato alla macchina fotografica 2 = a facce incrociate non allineato alla macchina fotografica 3 = a facce incrociate allineato alla macchina fotografica	Nessuno

`(c:npaes "Acero" 25.0 '(0.0 1.0 3.0) 1)`

Crea una nuova istanza di "Acero" con altezza pari a 25 unità, nella posizione 0, 1, 3, con singola faccia e non allineata alla macchina fotografica.

Funzione definita esternamente applicazione ARX *render*.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

LIBMAT

Gestisce le librerie dei materiali.

(c:matlib *modalità nome [file]*)

L'argomento *modalità*, descritto nella tabella riportata di seguito, è una stringa che indica l'azione eseguita da questa funzione. L'argomento *nome* è una stringa che indica il nome del materiale da importare, esportare o cancellare. L'argomento facoltativo *file* è una stringa che indica il nome del file della libreria dei materiali. L'argomento *file* deve includere l'estensione *.mli*.

Modalità del comando LIBMAT

Modalità	Descrizione
I	Importa un materiale da una libreria.
E	Esporta un materiale in una libreria.
D	Cancella un materiale dal disegno.
C	Cancella i materiali non applicati dal disegno.
L	Elenca i materiali.

Ad esempio:

```
(c:libmat "I" "brass" "render.mli")
```

importa il materiale BRASS dalla libreria dei materiali standard di Render AutoCAD, *render.mli*.

L'argomento *file* non viene utilizzato con la modalità Delete.

```
(c:libmat "D" "steel")
```

Funzione definita esternamente applicazione ARX *render*.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

SPECCHIO3D

Riflette gli oggetti selezionati su un piano definito dall'utente.

(specchio3d *arg1 arg2 ...*)

L'ordine, il numero ed il tipo di argomenti per la funzione **specchio3d** sono uguali a quelli che verrebbero immessi alla riga di comando. Per indicare una risposta nulla (RETURN) usare nil o una stringa vuota (""). Se l'esito è positivo, la funzione restituisce il valore T; altrimenti restituisce nil.

L'esempio seguente riflette gli oggetti selezionati relativi al piano XY che passa attraverso il punto 0,0,5, quindi cancella i vecchi oggetti.

```
(setq gs (ssget))
(specchio3d gs "XY" '(0 0 5) "Y")
```

Nota Il supporto di AutoLISP per la funzione **specchio3d** viene implementato mediante l'uso della libreria SAGET.

Funzione definita esternamente applicazione ARX *geom3d*

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

PSTRASCINA

Controlla l'aspetto di un'immagine PostScript importata mentre il comando PSIN la trascina per effettuarne il posizionamento.

(c:pstrascina *modalità*)

L'argomento *modalità* è un numero intero che dovrebbe essere uguale a 0 o ad 1. Il valore corrente di PSTRASCINA influenza l'uso interattivo del comando PSIN. Se PSTRASCINA è 1, PSIN genera l'immagine PostScript mentre l'utente ne effettua il trascinamento per ridurla in scala. Se PSTRASCINA è 0, PSIN genera e trascina solo la casella di delimitazione dell'immagine. Se l'esito è positivo, la funzione **c:pstrascina** restituisce il vecchio valore di PSTRASCINA, altrimenti restituisce nil.

Nel codice riportato di seguito, PSTRASCINA viene attivato tramite l'impostazione 1. Al successivo richiamo interattivo di PSIN, l'immagine PostScript viene generata mentre l'utente la trascina per effettuarne la riduzione in scala.

```
(c:pstrascina 1)
```

Funzione definita esternamente applicazione ARX *acadps*

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

PSMOTIVO

Riempie una polilinea bidimensionale con un modello di riempimento PostScript.

```
(c:psmotivo ent modello [arg1 [arg2]] ... )
```

L'argomento *ent* è il nome della polilinea. L'argomento *modello* è una stringa che contiene il nome del modello di riempimento. La stringa *modello* deve essere identica al nome di un modello di riempimento definito nel file *acad.psf*. Gli *argomenti* sono argomenti relativi alla procedura di riempimento propria del formato PostScript: il loro nome e tipo corrispondono agli argomenti richiesti da *modello*, come definito in *acad.psf*. Ciascun argomento può essere un numero intero o reale e ad ogni modello è possibile assegnare da 0 a 25 argomenti. Se la funzione indica un numero di argomenti minore rispetto a quelli definiti dal modello, per i rimanenti argomenti vengono utilizzati i valori di default del modello. Se l'esito è positivo, la funzione **c:psmotivo** restituisce T; altrimenti restituisce nil.

Il modello di riempimento scala di grigi ha un solo argomento. La funzione riportata di seguito utilizza al 50 per cento l'argomento di default di scala di grigi.

```
(c:psmotivo nome_elem "Scala di grigi")
```

Questa funzione indica invece un utilizzo del 10 per cento della scala dei grigi.

```
(c:psmotivo nome_elem "Scala di grigi" 10)
```

I modelli di riempimento PostScript sono memorizzati nei file di disegno come dati estesi e vengono identificati dal nome dell'applicazione AUTOCAD_POSTSCRIPT_FIGURE.

Funzione definita esternamente applicazione ARX *acadps*

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

PSIN

Importa un file PostScript.

```
(c:psin nomefile posizione scala)
```

L'argomento *nomefile* è una stringa che contiene il nome dell'immagine PostScript. Non è necessario specificare anche l'estensione *.eps*. L'argomento *posizione* è un punto che indica il punto di inserimento del blocco PostScript (anonimo). L'argomento *scala* è un valore reale che indica il fattore di scala. Se l'esito è positivo, **c:psin** restituisce il nome dell'oggetto appena creato; altrimenti restituisce nil.

Nel codice riportato di seguito, viene importato il file PostScript *esempio.eps*, inserito a (24,19) e riportato in scala con un fattore di 25:

```
(c:psin "esempio" '(24 19) 25)
```

Le immagini PostScript sono memorizzate nei file di disegno come dati estesi e sono identificate dal nome di applicazione AUTOCAD_POSTSCRIPT_FIGURE.

Funzione definita esternamente applicazione ARX *acadps*

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

 **RENDER**

Crea un'immagine, ombreggiata in modo realistico, di un modello wireframe tridimensionale (3D) utilizzando le informazioni relative alla geometria, all'illuminazione ed alla finitura di superficie.

```
(c:render [nomefile|punto1 punto2])
```

Nella tabella riportata di seguito vengono descritti gli argomenti del comando RENDER.

Argomenti di RENDER

Argomento	Tipo di dati	Descrizione	Default
nomefile	STR	Nome del file di rendering.	Nessuno
punto1	LIST (elenco di numeri reali)	Primo punto della finestra di ritaglio.	Nessuno
punto2	LIST (elenco di numeri reali)	Secondo punto della finestra di ritaglio.	Nessuno

Se è stato specificato l'argomento *nomefile*, il rendering viene scritto nel file indicato. Se non è stato configurato un driver per effettuare il rendering in un file, l'argomento *nomefile* sarà ignorato. Il file nel quale effettuare il rendering deve essere stato specificato nella configurazione corrente. Il rendering viene controllato dalle impostazioni attuali; impostare questi valori con la funzione **c:rpref** . Ad esempio:

```
(c:rpref "Toggle" "CropWindow" "On")
```

Nota quando le preferenze correnti di rendering indicano Interrogazione per selezione e la variabile di sistema PICKFIRST è stata attivata, e se **c:render** viene richiamato durante l'utilizzo di un insieme di selezione, il rendering viene eseguito sugli oggetti del gruppo senza ulteriori messaggi di richiesta.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

 **RENDER**

 **Impostazione del render per le opzioni di file**

Imposta il render per le opzioni di file.

```
(c:fileropz formatofile risx risy arapporto  
<opzioni modalità specifiche>)
```

La seguente tabella descrive gli argomenti **c:fileropz**.

Argomenti FILEROPZ

Argomento	Tipo di dati	Descrizione
formatofile	STR	Identificativo per il formato richiesto: TGA = Formato Targa PCX = Formato bitmap Z-Soft BMP = Formato bitmap Microsoft Windows PS = Postscript TIFF = Tagged Image File Format
xris	INT	Risoluzione X del file di output (l'intervallo dei valori validi va da 1 a 4096)
yris	INT	Risoluzione Y del file di output (l'intervallo dei valori validi va da 1 a 4096)
arapporto	REAL	Rapporto prospettico in pixel

La seguente tabella contiene i valori validi per l'argomento *modalità colore*. Ciascun formato di file accetta un sottogruppo di questi valori.

Modalità colori FILEROPZ

Modalità	Descrizione
----------	-------------

MONO	<i>Monocromatico</i>
G8	<i>256 (scala dei grigi)</i>
C8	<i>256 colori</i>
C16	<i>colore a 16 bit</i>
C24	<i>colore a 24 bit</i>
C32	<i>colore a 24 bit con 8 bit di alfa</i>

Funzione definita esternamente applicazione ARX *render*.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema



Specifica il formato Targa.

(c:fileropz "TGA" risx risy arapporto modalitàcolore interlacciamento compressione bassoalto)

Argomenti del formato TGA

Argomento	Tipo di dati	Descrizione
modalitàcolore	STR	Modalitàcolore: G8, C8, C24 o C32
interlacciamento	INT	Modalità interlacciamento: 1 = nessun interlacciamento 2 = interlacciamento 2:1 4 = interlacciamento 4:1
compressione	STR	Compressione (default = "C"): "C" = Compresso nil = Nessuna compressione
bassoalto	STR	Basso Alto (default = "Alto"): Alto = dal basso verso l'alto nil = dall'alto verso il basso

(C:FILEROPZ "TGA" 640 480 1.0 "C32" 1 "COMP" "UP")

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema



Specifica il formato di bitmap Z-Soft.

(c:fileropz "PCX" risx risy arapporto modalitàcolore)

Argomenti del formato PCX

Argomento	Tipo di dati	Descrizione
modalitàcolore	STR	Modalitàcolore: MONO, G8 o C8

(C:FILEROPZ "PCX" 640 480 1.0 "G8")

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

-  **RENDER**
-  **Impostazione del render per le opzioni di file**
-  **BMP**

Specifica il formato bitmap di Microsoft Windows.

(c:fileropz "BMP" risx risy arapporto modalitàcolore)

Argomenti del formato BMP

Argomento	Tipo di dati	Descrizione
modalitàcolore	STR	Modalitàcolore: MONO, G8 o C8

(C:FILEROPZ "BMP" 640 480 1.0 "C8")

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

-  **RENDER**
-  **Impostazione del render per le opzioni di file**
-  **PS**

Specifica il formato PostScript.

(c:fileropz "PS" risx risy arapporto modalitàcolore orientamento dimensioneimmagine [dimensione])

Argomenti del formato PS

Argomento	Tipo di dati	Descrizione
modalitàcolore	STR	Modalitàcolore: MONO, G8, C8 o C24
orientamento	STR	Orizzontale o verticale (default = "O"): V = Verticale O = Orizzontale
dimimmagine	STR	Tipo (default = "U") U = Auto I = Immagine P = Personalizzata
dimensione	INT	Dimensione dell'immagine

(C:FILEROPZ "PS" 640 480 1.0 "C24" "P" "C" 640)

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

-  **RENDER**
-  **Impostazione del render per le opzioni di file**
-  **TIFF**

Specifica il formato Tagged Image File Format.

(c:fileropz "TIFF" risx risy arapporto modalitàcolore compresso)

Argomenti del formato TIFF

Argomento	Tipo di dati	Descrizione
modalitàcolore	STR	Modalitàcolore: MONO, G8, C8 o C24

compressione *STR* *Compressione (default = "C"):*
C--Compresso
nil--Nessuna compressione

```
(C:FILEROPZ "TIFF" 640 480 1.0 "C24" nil)
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

RENDERUPDATE

Rigenera il file ent2face alla successiva operazione di rendering.

`(c:renderupdate [valore_RU])`

Se usato senza argomenti, il comando RENDERUPDATE consente di rigenerare il file *ent2face* alla successiva operazione di rendering. Specificando l'argomento "ALWAYS", il file *ent2face* viene rigenerato ad ogni operazione di rendering. L'argomento "OFF" disattiva l'argomento "ALWAYS".

Modalità di valore_RU

Modalità	<i>Descrizione</i>
ALWAYS	<i>Genera un nuovo file di geometria ad ogni operazione di rendering.</i>
OFF	<i>Ripristina la normale modalità del comando.</i>

Funzione definita esternamente applicazione ARX *render*.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

REPLAY

Visualizza una immagine BMP, TGA o TIFF.

`(c:replay nomefile tipo [sfals_x sfals_y dimen_x dimen_y])`

Con il comando REPLAY, è possibile visualizzare file BMP, TGA o TIFF sul video per il rendering di AutoCAD. Utilizzare la funzione di questo comando per riprodurre il file di immagini con diversi sfalsamenti e dimensioni. Nella tabella riportata di seguito vengono descritti gli argomenti della funzione.

Argomenti del comando REPLAY

Argomento	<i>Tipo di dati</i>	<i>Descrizione</i>	<i>Default</i>
<i>nomefile</i>	<i>STR</i>	<i>Rappresenta il nome del file di immagini.</i>	<i>Nessuno</i>
<i>tipo</i>	<i>STR</i>	<i>I valori validi sono: BMP, TGA e TIFF.</i>	<i>Nessuno</i>
<i>sfals_x</i>	<i>INT</i>	<i>Rappresenta lo sfalsamento X dell'immagine espresso in pixel.</i>	<i>0</i>
<i>sfals_y</i>	<i>INT</i>	<i>Rappresenta lo sfalsamento Y dell'immagine espresso in pixel.</i>	<i>0</i>
<i>dimen_x</i>	<i>INT</i>	<i>Rappresenta la dimensione X dell'immagine espressa in pixel.</i>	<i>Dimensione X attuale.</i>
<i>Dimens_y</i>	<i>INT</i>	<i>Rappresenta la dimensione Y dell'immagine espressa in pixel.</i>	<i>Dimensione Y attuale.</i>

Questa chiamata riproduce l'immagine *test.tga*, visualizzando i pixel a partire dal punto in basso a sinistra dell'immagine (sfalsamento = zero) fino a 500 pixel in larghezza and e 400 pixel in altezza. Ad esempio:

```
(c:replay "TEST" "TGA" 0 0 500 400)
```

Funzione definita esternamente applicazione ARX *render*.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

MATERIALE

Crea, modifica, unisce e distacca i materiali di rendering.

(c:materiale *modalità opzioni*)

Questa funzione dispone di sei modalità, descritte nella tabella riportata di seguito. Ciascuna modalità viene specificata da un argomento della stringa.

Modalità del comando MATERIALE

Modalità	Descrizione
A	Unisce il materiale.
C	Copia il materiale.
D	Distacca il materiale.
L	Elenca tutti i materiali inclusi nel disegno oppure restituisce una definizione per il materiale specificato.
M	Modifica il materiale.
N	Crea un nuovo materiale.

Funzione definita esternamente applicazione ARX *render*.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

MATERIALE

A--Attach (unisci)

La modalità "A" (unisci) consente di unire un materiale agli oggetti selezionati o un valore di indice del colore AutoCAD o ACI (AutoCAD Color Index) a seconda che il terzo argomento (nome-layer) sia un intero o un gruppo di selezione.

(c:materiale "A" *nome [aci | gruppo-selezione | nome-layer]*)

La seguente tabella descrive i vari argomenti di Attacca.

Argomenti di Attacca

Argomento	Tipo di dati	Descrizione
<i>nome</i>	STR	Nome del materiale da unire.
<i>Aci</i>	INT	Numero ACI compreso tra 0 e 255.
<i>Gruppo di selezione</i>	INT	Gruppo di selezione contenente gli elementi da unire.
<i>nome-layer</i>	STR	Nome del layer.

Ad esempio:

```
(c:materiale "A" "ROSSO TIGRATO" 1)
```

unisce il materiale ROSSO TIGRATO ad ACI 1 (rosso).

Se si omette il terzo argomento, la modalità Unisci restituisce un elenco di tre voci.

- Un elenco dei nomi di layer a cui il materiale viene unito.
- Un elenco di valori ACI a cui il materiale viene unito.
- Un gruppo di selezione contenente gli oggetti a cui il materiale viene unito.

Ad esempio:

Comando: **(c:materiale "a" "twood")**

Raccolta oggetti...1 trovato

Nomi dei layer dei valori ACI.

```
(("primo" "secondo")(135) <Gruppo di selezione 12>))
```

Un valore di indice del materiale compreso tra 1 e 255 è un numero ACI; un indice maggiore di 255 indica un materiale di Render di AutoCAD non assegnato da ACI.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

 **MATERIALE**

 **C--Copy (copia)**

Crea un nuovo materiale copiandone uno già presente nel disegno.

```
(c:materiale "C" nome_cor nome_nuovo)
```

L'argomento *nome_cor* è una stringa che indica il nome del materiale da copiare. L'argomento *nome_nuovo* è una stringa che indica il nome per il nuovo materiale. È possibile poi modificare il nuovo (o vecchio) materiale per cambiarne la definizione.

```
(c:materiale "C" "ROSSO" "ROSSO2")
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

 **MATERIALE**

 **D--Detach (distacca)**

La modalità "D" (distacca) consente di staccare un materiale dagli oggetti selezionati, un valore di indice del colore AutoCAD o ACI (AutoCAD Color Index) o layer a seconda che il secondo argomento (gruppo-selezione) sia un intero, un gruppo di selezione o una stringa.

```
(c:materiale "D" nome [aci | gruppo-selezione | nome-layer])
```

La seguente tabella descrive i vari argomenti di Distacca.

Argomenti di Distacca

Argomento	Tipo di dati	Descrizione	Default
<i>nome</i>	STR	Nome del materiale da staccare.	Nessuno
<i>aci</i>	INT	Numero ACI compreso tra 0 e 255.	Nessuno
<i>gruppo-selezione</i>	INT	Gruppo di selezione contenente gli elementi da staccare.	Nessuno
<i>nome-layer</i>	STR	Nome del layer.	Nessuno

Ad esempio:

```
(c:materiale "D" (ssget))
```

richiede all'utente di selezionare gli oggetti, quindi distacca ciascun oggetto dal materiale corrispondente.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

 **MATERIALE**

 **L--List (elenca)**

Elenca le definizioni dei materiali inclusi nel disegno.

```
(c:materiale "L" [nome])
```

L'argomento facoltativo *nome* è una stringa che specifica la definizione del materiale da elencare. Se l'argomento *nome* viene omissso, **c:materiale** elenca tutti i materiali contenuti nel disegno. Ad esempio:

Comando: (c:materiale "L")

("*GLOBALE*" "VETRO BLU" "PLASTICA BIANCA" "TWOOD" "BEIGE OPACO")

La prima stringa dell'elenco indica il materiale globale di default, *GLOBALE*. È possibile passare questa stringa al comando (c:materiale) nello stesso modo in cui vengono passati i nomi dei materiali della libreria o definiti dall'utente:

Comando: (c:materiale "L" "*GLOBALE*")

("*GLOBALE*" (-1.0 -1.0 -1.0) 1.0 (" 0.0 0 (1.0 1.0) (0.0 0.0) 0.0) (0.0 0.0 0.0)
 1.0 (1.0 1.0 1.0) 1.0 (" 0.0 0) 0.5 0.0 (" 0.0 0 (1.0 1.0) (0.0 0.0) 0.0)
 1.0 (" 0.0 0 (1.0 1.0) (0.0 0.0) 0.0))

Le voci dell'elenco nella definizione di un materiale sono uguali agli argomenti relativi alle modalità Modifica o Nuovo.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

MATERIALE

M--Modify (modifica)

Le opzioni della modalità "M" (Modifica) sono le stesse della modalità Nuovo. Se un argomento è nil oppure è stato omissso alla fine dell'elenco di argomenti, esso conserverà il suo valore corrente.

Ad esempio la seguente chiamata modifica MARMO BLU in modo che vi venga applicato un colore blu pietra medio (matrice) e venature nere:

```
(c:materiale "M" "MARMO BLU" "marmo" '(0.5 0.5 1.0) '(0.0 0.0 0.0))
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

MATERIALE

N--New (nuovo)

La modalità "N" (Nuovo) consente di creare un nuovo materiale. Gli argomenti di questa funzione non dipendono solo dalla modalità, ma anche dal tipo di materiale che viene creato. A ciascuno dei materiali procedurali, ovvero marmo, granito e legno, è associato un gruppo di argomenti univoco diverso dal gruppo di argomenti dei materiali standard.

La seguente tabella descrive i nuovi argomenti.

Nuovi argomenti

Argomento	Tipo di dati	Descrizione	Default
nome	STR	Rappresenta il nome del materiale da creare.	Nessuno
tipo materiale	STR	Tipo del nuovo materiale. Le opzioni disponibili sono le seguenti: STANDARD-- materiale standard MARBLE-- materiale marmoreo GRANITE-- materiale in granito WOOD-- materiale in legno	Nessuno
descrizione	(varie)	Gli argomenti variano a seconda del tipo di materiale che si sta creando.	(varie)
gruppo-selezione	INT	Gruppo di selezione contenente gli elementi da staccare.	Nessuno
nome-layer	STR	Nome del layer.	Nessuno

Gli argomenti di ciascun tipo di bitmap sono indicati in un elenco secondario descritto in "Argomenti bitmap."

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema



La stringa del tipo di materiale "STANDARD" indica che si sta creando un nuovo materiale standard.

```
(c:materiale "N" nome "STANDARD" [qtà-colore [modello
[circostante [qtà-circostante [rifl [qtà-rifl [mappa-rifl
[ruvidità [trasparenza[mappaopacità [rifrazione
[bumpmap]]]]]]]]]]))
```

La seguente tabella descrive gli argomenti standard.

Argomenti standard

Argomento	Tipo di dati	Descrizione	Default
color	LIST (elenco di numeri reali)	Colore del materiale come ruppo di tre valori RGB; i tre valori (-1.0 -1.0 -1.0) indicano di prendere il colore dal valore ACI dell'oggetto (colore diffuso).	(-1.0 -1.0 -1.0)-- Da ACI
qtà-colore	REAL	Rappresenta il fattore di quantità (valore del colore), ovvero la quantità di colore diffuso.	0.7
modello	LIST	Argomenti nella mappa composizione/modello.	Nessuno
circostante	LIST (elenco di numeri reali)	Rappresenta il colore circostante (ombra) espresso da un gruppo di tre valori RGB.	(-1.0 -1.0 -1.0)-- Da ACI
qtà-circostante	REAL	Rappresenta il fattore di quantità (valore del colore), ovvero la quantità di colore speculare.	0.1
rifl	LIST (elenco di numeri reali)	Rappresenta il colore del riflesso (speculare) espresso da un gruppo di tre valori RGB.	(-1.0 -1.0 -1.0)-- Da ACI
qtà-rifl	REAL	Rappresenta il fattore di quantità (valore del riflesso), ovvero la quantità di colore speculare.	0.2
mappa-rifl	LIST	Argomenti nella mappa circostante/modello.	Nessuno
ruvidità	REAL	Rappresenta la ruvidità, la dimensione di una evidenziazione speculare.	0.5
trasparenza	REAL	Trasparenza del materiale.	0.0
mappa-opacità	LIST	Argomenti per la mappa opacità.	Nessuno
rifrazione	REAL	Indice di rifrazione.	1.0
bumpmap	LIST	Argomenti nella mappa bump.	Nessuno

Ad esempio, questa chiamata crea un materiale rosso lucido con una mappa podello:

```
(c:comando "N" "LACCATO ROSSO" "STANDARD" ; Nome e tipo
'(1.0 0.0 0.0) (1.0) ; Colore (rosso), quantità e mappa composizione
'("INLAY.TGA 0.75 0 (0.5 0.5) (0.3 0.3) 0.0)
'(1.0 0.0 0.0) 1.0 ; Colore circostante e quantità (uguale al colore diffuso)
'(1.0 0.0 0.0) 1.0 ; Colore del riflesso (bianco) e quantità
nil ; Nessuna mappa riflesso
0.2 ; Ruvidità (bassa)
0.0 ; Trasparenza (nessuna)
nil ; Nessuna mappa opacità
0.0 ; Rifrazione (nessuna)
nil ; Nessuna mappa bump
```

La chiamata seguente crea il materiale MAPPE che utilizza più bitmap:

```
(c:materiale "N" "MAPPE" "STANDARD"
'(1.0 0.0 0.0) (1.0) ("tessuto.tga" 1.0 0)
'(1.0 0.0 0.0) 1.0
'(1.0 0.0 0.0) 1.0 ("stanza.tga" 0.75)
```

```
0.5
0.0
'("foro.tga")
1.0
'("colmi.tga")
```

La seguente chiamata crea un materiale privo di bitmap e senza alcun valore di default, con riflessi generati dalla tracciatura dei raggi quando viene sottoposto a rendering con Photo Raytrace o con mappa circostante tramite Photo Real:

```
(c:materiale "N" "LUCIDO" "STANDARD" nil nil nil nil nil nil nil '(nil nil 1))
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema



La stringa del tipo di materiale "MARBLE" indica che si sta creando un nuovo materiale marmoreo.

```
(c:materiale "N" nome "MARBLE" [colore-pietra [colore-venatura
[rufi [qtà-rifl [mappa-rifl [ruvidità [turbolenza
[chiarezza [scala [bumpmap ]]]]]]]])
```

La seguente tabella descrive gli argomenti di MARBLE.

MATERIALE: argomenti di MARBLE

Argomento	Tipo di dati	Descrizione	Default
colore-pietra	LIST (elenco di numeri reali)	Valore RGB che specifica il colore di matrice principale del marmo.	(-1.0 -1.0 -1.0) bianco
colore-venatura	LIST (elenco di numeri reali)	Valore RGB che specifica il colore della venatura del marmo.	(-1.0 -1.0 -1.0) nero
rifl	LIST (elenco di numeri reali)	Rappresenta il colore del riflesso (speculare) espresso da un valore RGB.	(-1.0 -1.0 -1.0) da ACI
qtà-rifl	REAL	Rappresenta il fattore di quantità (valore del riflesso), ovvero la quantità di colore speculare.	0.2
mappa-rifl	LIST	Argomenti nella mappa circostante/riflesso.	Nessuno
ruvidità	REAL	Rappresenta la ruvidità, la dimensione di una evidenza speculare.	0.5
turbolenza	INT	Rappresenta il fattore di turbolenza, l'ondulatezza delle venature.	3
chiarezza	REAL	Rappresenta il fattore di chiarezza, la quantità di offuscamento.	1.0
scala	REAL	Rappresenta il fattore di scala globale.	0.16
bumpmap	LIST	Argomenti bumpmap	Nessuno

Ad esempio, questa chiamata crea un marmo con matrice rosa e venature nere.

```
(c:materiale"N" "MARMO ROSA" "MARBLE" '(1.0 0.34 0.79))
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema



La stringa del tipo di materiale "GRANITE" indica che si sta creando un nuovo materiale di granito.

```
(c:materiale "N" nome "GRANITE" [primo-colore [quantità
[secondo-colore [quantità2 [terzo-colore [quantità3
[quarto-colore [quantità 4 [rifl [qtà-rifl
[mappa-rifl [ruvidità [chiarezza [scala
[bumpmap ]]]]]]]]]]]))
```

La seguente tabella descrive gli argomenti di GRANITE.

MATERIALE: argomenti di GRANITE

Argomento	Tipo di dati	Descrizione	Default
primo-colore	LIST (elenco di numeri reali)	Valore RGB	(-1.0 -1.0 -1.0) bianco
quantità1	REAL	Rappresenta il fattore di quantità (valore colore) del primo colore.	1.0
secondo-colore	LIST (elenco di numeri reali)	Valore RGB	(0.5 0.5 0.5) grigio scuro
quantità2	REAL	Rappresenta il fattore di quantità (valore colore) del secondo colore.	1.0
terzo-colore	LIST (elenco di numeri reali)	Valore RGB	(0.0 0.0 0.0) nero
quantità3	REAL	Rappresenta il fattore di quantità (valore colore) del terzo colore.	1.0
quarto-colore	LIST (elenco di numeri reali)	Valore RGB	(0.7 0.7 0.7) grigio chiaro
quantità4	REAL	Rappresenta il fattore di quantità (valore colore) del quarto colore.	1.0
rifl	LIST (elenco di numeri reali)	Rappresenta il colore del riflesso (speculare) espresso da un valore RGB.	(-1.0 -1.0 -1.0) da ACI
qtà-rifl	REAL	Rappresenta il fattore di quantità (valore del riflesso), ovvero la quantità di colore speculare.	0.2
mappa-rifl	LIST	Argomenti nella mappa circostante/riflesso.	Nessuno
ruvidità	REAL	Rappresenta la ruvidità, la dimensione 0.5 di una evidenziazione speculare.	0.5
chiarezza	REAL	Rappresenta il fattore di chiarezza, la quantità di offuscamento.	1.0
scala	REAL	Rappresenta il fattore di scala globale.	0.16
bumpmap	LIST	Argomenti bumpmap	Nessuno

Ad esempio, questa chiamata crea un granito senza grigio scuro, con una maggior quantità di nero e con colore giallo anziché grigio chiaro:

```
(c:materiale "N" "GRANITO GIALLO"
nil 0.5 nil 0.0 nil 0.85 '(1.0 1.0 0.0) 0.6)
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema



La stringa del tipo di materiale "WOOD" indica che si sta creando un nuovo materiale di legno.

```
(c:materiale "N" nome "WOOD" [colore-pietra [colore-chiaro
[colore-scuro [rifl [qtà-rifl [mappa-rifl [ruvidità
[rappporto [densità [spessore [forma [bumpmap ]]]]]]]]]))
```

La seguente tabella descrive gli argomenti di WOOD.

MATERIALE: argomenti di WOOD

Argomento	Tipo di dati	Descrizione	Default
colore-chiaro	LIST (elenco di numeri reali)	Valore RGB che specifica il colore degli anelli chiari.	(0.6 0.4 0.3)
colore-scuro	LIST (elenco di numeri reali)	Valore RGB che specifica il colore degli anelli scuri.	(0.3 0.2 0.2) nero
rifl	LIST (elenco di numeri reali)	Rappresenta il colore del riflesso (speculare) espresso da un valore RGB.	(-1.0 -1.0 -1.0) da ACI
qtà-rifl	REAL	Rappresenta il fattore di quantità (valore del riflesso), ovvero la quantità di colore speculare.	0.2
mappa-rifl	LIST	Argomenti nella mappa circostante/riflesso.	Nessuno
ruvidità	REAL	Rappresenta la ruvidità, la dimensione 0.5 di una evidenziazione speculare.	0.5
rapporto	REAL	Rapporto tra anelli chiari ed anelli scuri.	0.5
densità	REAL	Densità degli anelli.	6.0
spessore	REAL	Variazione dello spessore degli anelli.	0.2
forma	REAL	Variazione della forma degli anelli.	0.2
scala	REAL	Rappresenta il fattore di scala globale.	0.16
bumpmap	LIST	Argomenti bumpmap	Nessuno

Ad esempio, questa chiamata crea un tipo dilegno con granatura irregolare:

```
(c:materiale "N" "CRIPTO" "WOOD" nil nil nil nil nil nil nil nil 0.56)
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

MATERIALE

N--New (nuovo)

Argomenti bitmap

Gli argomenti per l'impostazione di una bitmap vengono passati ad un elenco che è possibile includere come elenco secondario nella chiamata "c:materiale" (la forma indicata all'inizio di ciascuna delle sezioni che seguono) o assegnare ad un simbolo prima di eseguire la chiamata "c:materiale".

Modello/composizione

```
' (nome [messaggio [ripetizione [scala [sfalsamento]]]])
```

La seguente tabella descrive gli argomenti di Modello/composizione.

Argomenti Modello/composizione

Argomento	Tipo di dati	Descrizione	Default
nome	STR	Nome del file bitmap.	Nessuno
messaggio	REAL	Quantità di colore mappa da usare.	1.0
ripetizione	INT	INDica se ripetere o meno la bitmap (casella): 0 = nessuna casella (ritaglio) 1 = casella (motivo ripetuto)	0
scala	LIST (elenco di numeri reali)	Fattori di scala U e V.	(1.0 1.0)
sfalsamento	LIST (elenco di numeri reali)	Sfalsamenti U e V.	(0.0 0.0)

Riflesso/circostante

```
' (nome [messaggio [raytrace]])
```

La seguente tabella descrive gli argomenti di Riflesso/circostante.

Argomenti di Riflesso/circostante

Argomento	Tipo di dati	Descrizione	Default
nome	STR	Nome del file bitmap.	Nessuno
missaggio	REAL	Quantità di colore mappa da usare.	1.0
speculare	REAL	Indica se generare o meno riflessi speculari. 0 = nessun riflesso speculare 1 = riflessi speculari durante la riflessione specularare genera riflessi a tracciatura di raggi: durante la scansione di linee usa la mappa ambiente per i riflessi.	0

Opacità

' (nome [missaggio [ripetizione [scala [sfalsamento]]]])

La seguente tabella descrive gli argomenti di Opacità.

Argomenti di Opacità

Argomento	Tipo di dati	Descrizione	Default
nome	STR	Nome del file bitmap.	Nessuno
missaggio	REAL	Quantità di colore mappa da usare.	1.0
ripetizione	INT	Indica se ripetere o meno la bitmap (casella): 0 = nessuna casella (ritaglio) 1 = casella (motivo ripetuto)	0
scala	LIST (elenco di numeri reali)	Fattori di scala U e V.	(1.0 1.0)
sfalsamento	LIST (elenco di numeri reali)	Sfalsamenti U e V.	(0.0 0.0)

Bump Map

' (nome [ampiezza [ripetizione [scala [sfalsamento]]]])

La seguente tabella descrive gli argomenti di bump.

Argomenti di bump

Argomento	Tipo di dati	Descrizione	Default
nome	STR	Nome del file bitmap.	Nessuno
ampiezza	REAL	Grado di bumpiness.	1.0
ripetizione	INT	Indica se ripetere o meno la bitmap (casella): 0 = nessuna casella (ritaglio) 1 = casella (motivo ripetuto)	0
scala	LIST (elenco di numeri reali)	Fattori di scala U e V.	(1.0 1.0)
sfalsamento	LIST (elenco di numeri reali)	Sfalsamenti U e V.	(0.0 0.0)

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

 **RUOTA3D**

Ruota un oggetto su un asse tridimensionale (3D) arbitrario.

(ruota3d arg1 arg2 ...)

L'ordine, il numero ed il tipo di argomenti per la funzione **ruota3d** sono uguali a quelli che verrebbero immessi alla riga di comando. Per indicare una risposta nulla (RETURN) usare nil o una stringa vuota (""). Se l'esito è positivo, la funzione restituisce il valore T; altrimenti restituisce nil.

Nell'esempio seguente, gli oggetti selezionati vengono fatti ruotare di 30 gradi intorno all'asse specificato dai punti p1 e p2.

```
(setq gs (ssget))
(ruota3d gs p1 p2 30)
```

Nota il supporto di AutoLISP per la funzione **ruota3d** viene implementato mediante l'uso della libreria SAGET.

Funzione definita esternamente applicazione ARX *geom3d*

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

RPREF

Imposta le preferenze per il rendering.

(c:rpref *modalità opzione [impostazione]*)

RPREF determina quali parametri di rendering saranno utilizzati e quale tipo di rendering sarà quello di default. Questa funzione dispone di cinque modalità, descritte nella tabella riportata di seguito. Ciascuna modalità viene specificata da un argomento della stringa. Gli altri argomenti dipendono dalla modalità.

Modalità del comando RPREF

Modalità	Descrizione
DEST	Destinazione della finestra, della finestra Render o del file.
ICON	Controlla la dimensione dei blocchi di icone Luce e Materiali.
ROPT	Opzioni di rendering aggiuntive.
SELECT	Indica se richiedere o meno la selezione dell'oggetto.
STYPE	Tipo di rendering di Render, Photo Real o Photo Raytrace.
TOGGLE	Controlla le opzioni di rendering.

Funzione definita esternamente applicazione ARX *render*

Vedere anche "Impostazione del render per le opzioni di file."

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

RPREF

DEST--Destinazione

Seleziona il dispositivo di output utilizzato.

(c:rpref "DEST" *opzione*)

L'argomento *opzione* è una stringa che indica la destinazione del rendering. I valori possibili per *opzione* sono descritti nella tabella riportata di seguito.

Opzioni per "DEST" del comando RPREF

Valore	Descrizione
FRAMEBUFFER	Render da visualizzare.
HARDCOPY	Esegue il rendering nella finestra Render.
FILE	Esegue il rendering in un file.

Ad esempio, questa chiamata specifica il rendering in un file:

```
(c:rpref "DEST" "FILE")
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

RPREF

ICON--Icona

Specifica la dimensione del blocco relativo all'icona della luce o del materiale contenuto nel disegno.

(c:rpref "ICON" *opzione*)

L'argomento *opzione* è un numero reale che indica la dimensione del blocco relativo all'icona (il valore di default è 1.00). Ad esempio, con questa chiamata viene modificata la scala dell'icona, portandola al 50 per cento.

```
(c:rpref "ICON" 0.5)
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

RPREF

STYPE--Tipo di rendering

Specifica il tipo di Render usato.

(c:rpref "STYPE" *opzione*)

L'argomento *opzione* è una stringa che indica il tipo di rendering. I valori possibili per *opzione* sono descritti nella tabella riportata di seguito.

Opzioni per "STYPE" del comando RPREF

Valore	Descrizione
ARENDER	Rendering di base.
ASCAN	Rendering di qualità fotografica.
ARAY	Rendering Raytrace.

Ad esempio, con il seguente codice viene indicato che il rendering successivo sarà generato dal rappresentatore di immagini di AutoCAD.

```
(c:rpref "STYPE" "CRENDER")
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

RPREF

SELECT--Selezione

Specifica se richiedere o meno la selezione dell'oggetto prima di generare un rendering.

(c:rpref "SELECT" *opzione*)

L'argomento *opzione* è una stringa che indica la richiesta. I valori possibili per *opzione* sono descritti nella tabella riportata di seguito.

Opzioni per "SELECT" del comando RPREF

Valore	Descrizione
ALL	Effettua il rendering su tutta la scena.
ASK	Visualizza il messaggio di richiesta per la selezione dell'oggetto.

Ad esempio, con questa chiamata viene impostato il rendering in modo da richiedere la selezione dell'oggetto.

```
(c:rpref "SELECT" "ASK")
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

RPREF

TOGGLE--Interruttore

Controlla le varie opzioni di rendering.

(c:rpref "TOGGLE" *opzione impostazione*)

L'argomento *opzione* è una stringa che indica la richiesta. I valori possibili per *opzione* sono descritti nella tabella riportata di seguito. L'argomento *impostazione* è una stringa che indica lo stato dell'interruttore. I valori possibili per *impostazione* sono "ON" e "OFF".

Opzioni per "TOGGLE" del comando RPREF

Valore	Descrizione
CACHE	Effettua il rendering in un file cache.
SHADOW	Effettua il rendering con ombreggiature.
SMOOTH	Effettua il rendering con la levigatezza.
MERGE	Unisce gli oggetti con lo sfondo.
FINISH	Esegue l'applicazione dei materiali.
SKIPRDLG	La finestra di dialogo Render non viene visualizzata.

Nota L'opzione CACHE indica che le informazioni del rendering devono essere scritte in un file cache nel disco rigido. Fintanto che la geometria o la vista del disegno rimangono inalterate, il file in cache viene usato per le successive operazioni di rendering, senza pertanto dover rieseguire la tassellazione. Ciò consente di risparmiare tempo, soprattutto con il rendering di solidi.

Ad esempio, con le seguenti chiamate viene disattivato il rendering con l'opzione Leviga ed attivata la presenza di ombre.

```
(c:rpref "TOGGLE" "MERGE" "OFF")
(c:rpref "TOGGLE" "SMOOTH" "ON")
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

SALVAIMM

Salva un'immagine sottoposta a rendering in un file di formato BMP, TGA o TIFF.

(c:salvaimm *nomefile tipo [porzione] [sfals_x sfals_y dimen_x dimen_y] [compressione]*)

Quando AutoCAD viene configurato per eseguire il rendering su un video diverso, l'argomento *porzione* non deve essere utilizzato. È possibile specificare la dimensione e lo sfalsamento dell'immagine e per i file TGA e TIFF è possibile indicare anche la combinazione di compressione. Gli argomenti sono descritti nella tabella riportata di seguito.

Argomenti del comando SALVAIMM

Argomento	Tipo di dati	Descrizione	Default
<i>nomefile</i>	STR	Rappresenta il nome del file di immagini.	Nessuno
<i>tipo</i>	STR	Definisce il tipo di file: BMP, TGA o TIFF.	Nessuno
<i>porzione</i>	STR	Definisce la porzione dello schermo da salvare: A: finestra attiva D: area del disegno F: schermo intero	"A"
<i>sfals_x</i>	INT	Rappresenta lo sfalsamento X espresso in pixel.	0
<i>sfals_y</i>	INT	Rappresenta lo sfalsamento Y espresso in pixel.	0
<i>dimen_x</i>	INT	Rappresenta la dimensione X espressa in pixel.	Dimensione X attuale.
<i>Dimens_y</i>	INT	Rappresenta la dimensione Y espressa in pixel.	Dimensione Y attuale.
<i>Compressione</i>	STR	Definisce la combinazione di compressione: NESSUNA	Nessuno

PACK (solo per i file TIFF)
RLE (solo per i file TGA).

Nota Nella nuova versione l'argomento porzione è ignorato, ma viene fornito per compatibilità con gli script.

Nell'esempio riportato di seguito, viene salvata una immagine TIFF a schermo intero con il nome *test.tga* senza alcuna compressione.

```
(c:salvaimm "TEST" "TIF" "NONE")
```

Funzione definita esternamente applicazione ARX render

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

SCENA

Crea nuove scene e modifica o cancella quelle esistenti solo nello spazio carta.

```
(c:scena modalità [opzioni])
```

Questa funzione dispone di sei modalità, descritte nella tabella riportata di seguito. Ciascuna modalità viene specificata da un argomento della stringa. Gli altri argomenti dipendono dalla modalità.

Modalità del comando SCENA

Modalità	Descrizione
D	Cancella una scena esistente.
L	Elenca tutte le scene contenute nel disegno oppure restituisce una definizione per la scena specificata.
M	Modifica una scena esistente.
N	Crea una nuova scena.
R	Rinomina una scena esistente.
S	Imposta la scena corrente.

Funzione definita esternamente applicazione ARX *render*.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

SCENA

D--Delete (cancella)

Cancella una scena esistente.

```
(c:scena "D" nome)
```

L'argomento *nome* è una stringa che indica il nome della scena da cancellare. Se la scena cancellata è quella corrente, *NESSUNA* diventa la scena corrente.

```
(c:scena "D" "VISTPIAN")
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

SCENA

L--List (elenca)

Elenca tutte le scene contenute nel disegno oppure restituisce una definizione per la scena specificata.

(c:scena "L" [nome])

L'argomento *nome* è una stringa che indica il nome della scena da elencare. Se l'argomento *nome* viene omissso, **c:scena** restituisce l'elenco di tutte le scene definite nel disegno. Quando invece *nome* viene specificato, **c:scena** restituisce la definizione della scena specificata.

Con la seguente chiamata viene restituito l'elenco dei nomi delle scene definite nel disegno.

```
(c:scena "L")           restituisce)
(" " "SCENA1" "SCENA2" "SCENA3")
```

La stringa vuota (" ") rappresenta la scena di default, *NESSUNA*, che non può essere modificata. La chiamata:

```
(c:scena "L" "SCENA2")
```

restituisce la definizione della scena specificata. Di seguito vengono riportati alcuni esempi.

```
(T T)                 La vista *corrente* con *tutte* le luci
("VISTA1" nil)        La vista specificata senza luci
("VISTA2" ("LUCE1" "LUCE2")) La vista specificata con due luci
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

SCENA

M--Modify (modifica)

Modifica una scena esistente.

(c:scena "M" nome [vista [luci]])

Le opzioni della modalità Modify sono le stesse della modalità New, tranne per il fatto che è possibile passare l'argomento *vista* come nil per modificare solo le luci.

Nota È necessario passare l'argomento *luce* sotto forma di elenco anche quando viene specificata una sola luce.

Ad esempio, con questa chiamata la scena SCENA1 viene modificata in modo da utilizzare la vista FRONTE e tutte le luci contenute nel disegno.

```
(c:scena "M" "SCENA1" "FRONTE" (C:LUCE "L"))
```

La seguente chiamata modifica SCENA1 in modo da utilizzare la vista SFONDO e solo le luci P1 e P2:

```
(c:scena "M" "SCENA1" "SFONDO" ("P1" "P2"))
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

SCENA

N--New (nuova)

Crea una scena nuova.

(c:scena "N" nome [vista [luci]])

Nella tabella riportata di seguito vengono descritti gli argomenti della modalità New.

Argomenti della modalità "N" del comando SCENA

Argomento	Tipo di dati	Descrizione	Default
nome	STR	Rappresenta il nome della scena.	Nessuno
vista	STR	Rappresenta il nome di una vista AutoCAD contrassegnata da un nome.	Nessuno

	<i>T (SYM)</i>	<i>Definisce la vista *CORRENTE*.</i>	
Luci	<i>LIST (elenco delle stringhe) T (SYM)</i>	<i>Rappresenta l'elenco dei nomi delle luci.</i>	<i>*TUTTE* le luci incluse nel disegno.</i>
	<i>nil</i>	<i>Nel disegno vengono utilizzate *TUTTE* le luci. Nel disegno non viene utilizzata alcuna luce.</i>	<i>Una luce distante, posta -"sopra la spalla".</i>

Nota è necessario passare l'argomento *luce* sotto forma di elenco anche quando viene specificata una sola luce.

Ad esempio, con il codice riportato di seguito viene creata una nuova scena con il nome DEFAULT che utilizza la vista *CORRENTE* e *TUTTE* le luci.

```
(c:scena "N" "DEFAULT")
```

Con il seguente codice viene invece creata una nuova scena con il nome DEBOLE che utilizza la vista *CORRENTE* e l'illuminazione di default "sopra la spalla".

```
(c:scena "N" "DEBOLE" T nil)
```

Con il seguente codice viene creata una nuova scena con il nome SPECIALE che utilizza la vista VISTAMIA con le luci SOLE, LAMPADA e RIFLETTORE.

```
(c:scena "N" "SPECIALE" "VISTAMIA" ("SOLE" "LAMPADA" "RIFLETT"))
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

SCENA

R--Rename (rinomina)

Rinomina una scena.

```
(c:scena "R" nome_vecchio nome_nuovo)
```

L'argomento *nome_vecchio* è una stringa che indica il nome della scena originale, mentre *nome_nuovo* indica il nuovo nome della scena.

Ad esempio:

```
(c:scena "R" "SPECIALE" "LUMINOSA")
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

SCENA

S--Set (imposta)

Imposta la scena corrente.

```
(c:scena "S" [nome])
```

L'argomento *nome* è una stringa che indica il nome della scena da rendere corrente. Se l'argomento *nome* viene omissso, c:scena restituisce il nome della scena attualmente selezionata. Ad esempio:

```
(c:scena "S")
```

```
"PIANO"
```

Se nessuna scena è attualmente corrente, c:scena restituisce una stringa vuota ("").

Ad esempio, con il codice riportato di seguito viene resa corrente la scena SCENA3.

```
(c:scena "S" "SCENA3")
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

 **MAPPAGGIO**

Assegna le coordinate del mappaggio del materiale agli oggetti selezionati. Al comando sono associate due modalità, specificate da un argomento stringa.

Il comando MAPPAGGIO consente di assegnare le coordinate del mappaggio dle material agli oggetti selezionati. Al comando sono associate due modalità, specificate da un argomento stringa.

Modalità di MAPPAGGIO

Modalità	Descrizione
A	Assegna il mappaggio UV al gruppo di selezione.
D	Stacca il mappaggio UV dal gruppo di selezione.

Funzione definita esternamente applicazione ARX render.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

 **MAPPAGGIO**

 **A (Assegna)**

La modalità "A" (Assegna) assegna coordinate di mappaggio. Gli argomenti della modalità variano a seconda che si specifichi il mappaggio di proiezione o di solidi. Nella tabella riportata di seguito vengono descritti gli argomenti della modalità Assegna per il mappaggio di proiezione.

Argomenti della modalità "A" di MAPPAGGIO per il mappaggio di proiezioni

Argomento	Tipo di dati	Descrizione	Default
ssname	PICKSET	Il gruppo di selezione contenente le entità a cui si desidera assegnare coordinate di mappaggio.	Nessuno
tipo_mappaggio	STR	Tipo del mappaggio di proiezione: P = piano D = cilindrico F = sferico	Nessuno
pt1, pt2, pt3	LIST	I tre punti che definiscono la geometria di mappaggio: Piano = angolo inferiore sinistro, angolo inferiore destro, angolo superiore sinistro Cilindrico = parte inferiore centrale, parte superiore centrale, direzione verso la giunzione Sferico = centro della sfera, raggio (nord), direzione verso la giunzione	Nessuno
rip	INT	Incasellamento: 0 = nessuna casella (ritaglio) 1 = casella (motivo ripetuto)	1
scala	LIST (elenco di numeri reali)	Fattori di scala U e V.	(1.0 1.0)
sfalsamento	LIST (elenco di numeri reali)	Sfalsamenti U e V.	(0.0 0.0)

Per il mappaggio di solidi, gli argomenti di opzione specificano solo i punti di mappaggio. Tali punti definiscono in modo implicito la scala nelle dimensioni UVW. Nella tabella riportata di seguito vengono descritti gli argomenti della modalità Assegna per il mappaggio di solidi.

Argomenti della modalità "A" di MAPPAGGIO per il mappaggio di solidi

Argomento	Tipo di dati	Descrizione	Default
<i>ssname</i>	PICKSET	Il gruppo di selezione contenente gli oggetti a cui si desidera assegnare coordinate di mappaggio.	Nessuno
<i>tipo_mappaggio</i>	STR	R = solido	Nessuno
<i>pt1</i>	LIST	Punto per la definizione dell'origine.	Nessuno
<i>pt1</i>	LIST	Punto per la definizione dell'asse U.	Nessuno
<i>pt1</i>	LIST	Punto per la definizione dell'asse V.	Nessuno
<i>pt1</i>	LIST	Punto per la definizione dell'asse W.	Nessuno

Ad esempio, questa chiamata assegna coordinate di mappaggio di tipo cilindrico ad un oggetto scelto dall'utente, con incasellamento e valori di scala e di sfalsamento di default:

```
(c:setuv "A" (ssget) "C" '(5.0 5.0 5.0) '(5.0 5.0 10.0)
' (10.0 0.0 0.0) 1)
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

MAPPAGGIO

D--Detach (distacca)

La modalità "D" (distacca) consente di staccare il mappaggio UV assegnato all'oggetto nel gruppo di selezione. Questi oggetti verranno mappati con le coordinate di mappaggio di default fino a quando non si assegnano coordinate diverse. Nella tabella riportata di seguito vengono descritti gli argomenti della modalità "D".

Argomenti della modalità "D" di SETUV.

Argomento	Tipo di dati	Descrizione	Default
<i>ssname</i>	PICKSET	Il gruppo di selezione contenente gli oggetti di cui si desidera staccare le coordinate di mappaggio.	Nessuno

Ad esempio, questa chiamata richiede all'utente le entità che verranno staccate dalle corrispondenti coordinate di mappaggio.

```
(c:setuv "D" (ssget))
```

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

SHOWMAT

Mostra il tipo di materiale ed il metodo di unione per un oggetto selezionato.

```
(c:showmat arg1)
```

Questa funzione mostra il tipo di materiale ed il metodo di unione basato su *arg1*. L'argomento *arg1* può essere un nome di entità, un numero intero che rappresenta un valore ACI o un nome di layer (una stringa).

Funzione definita esternamente applicazione ARX *render*.

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

SOLPROF

Crea immagini di profilo di solidi tridimensionali.

```
(c:solprof arg1 arg2 ...)
```

L'ordine, il numero ed il tipo di argomenti sono esattamente uguali a quelli che vengono digitati alla riga di comando.

Funzione definita esternamente applicazione ARX *solids*

Capitolo 14 -- Accesso ai comandi definiti esternamente e alle variabili di sistema

STATIST

Visualizza le statistiche relative all'ultimo rendering.

```
(c:statist [nomefile [nil]])
```

Il comando STATIST fornisce le informazioni relative all'ultimo rendering.

Se si specifica un nome di file, (c:statist) salva le informazioni di rendering nel file senza visualizzare la finestra di dialogo Statistiche. Se invece il nome del file viene omissso, la finestra di dialogo Statistiche verrà visualizzata.

Il seguente comando, ad esempio, scrive le statistiche dell'ultima operazione di rendering nel file *figura.txt*:

```
(c:statist "figura.txt")
```

Se il file esiste già, le statistiche vengono aggiunte ai dati esistenti.

Il seguente comando:

```
(c:statist "statist.txt")
```

salva sia le informazioni associate all'ultima operazione di rendering nel file *statist.txt* che le informazioni associate alle operazioni di rendering successive fino a quando non si digita quanto segue:

```
(c:statist nil)
```

Il comando riportato sopra indica di interrompere il salvataggio delle statistiche.

Funzione definita esternamente applicazione ARX *render*.

Capitolo 15 -- Gestione della memoria

Panoramica

AutoCAD è strutturato in modo da gestire autonomamente le risorse di memoria in base alle proprie necessità. Tuttavia, è possibile che sistemi non dotati di grandi quantità di memoria necessitino di una accurata gestione delle risorse. Questo capitolo fornisce informazioni su come gestire questi sistemi.

Tutti i simboli, le funzioni definite dall'utente e quelle standard descritte in questa guida sono presenti in memoria per tutta la durata della sessione di lavoro di AutoCAD. Quando viene avviato, AutoLISP acquisisce due grandi aree di memoria. La prima, chiamata *heap*, è l'area in cui vengono memorizzate le funzioni ed i simboli (chiamati anche *nodi*); maggiore è il numero dei simboli e delle funzioni (e maggiore è la complessità delle funzioni), maggiore è lo spazio di heap utilizzato. La seconda area, chiamata *pila*, contiene gli argomenti ed i risultati parziali delle funzioni; maggiore è la nidificazione o la ricorsività delle funzioni, maggiore è la quantità di spazio di pila che viene utilizzato.

Argomenti di questo capitolo

```
{button ,JI(','Node_Space_al_u0309')} Spazio per i nodi
```

```
{ewc ,JI(','Technical_Notes_al_u0309')} Note tecniche
```

Capitolo 15 -- Gestione della memoria

Spazio per i nodi

Per *nodo* si intende una struttura di memoria in grado di rappresentare tutti i tipi di dati di AutoLISP. Attualmente AutoLISP utilizza nodi a 12 byte. Per evitare la frammentazione della memoria ed il sovraccarico dovuto alla gestione di heap, i nodi vengono allocati nell'heap in gruppi chiamati *segmenti*. Per default, un segmento è composto da 514 nodi (6168 byte).

AutoLISP conserva un elenco dei nodi liberi, ossia i nodi che correntemente non sono legati ad un simbolo. Quando necessita di un nodo in cui memorizzare un simbolo o un valore, AutoLISP esegue una ricerca in questo elenco per trovare un nodo disponibile. Se non ve ne sono, viene eseguita un'operazione automatica di ripulitura dello spazio, in modo da porre in questo elenco i nodi che eventualmente non sono più legati ad un simbolo. Infine, uno di questi nodi viene scelto in modo da soddisfare la richiesta.

Se la ripulitura dello spazio rileva la presenza di un numero troppo esiguo di nodi liberi, AutoLISP richiede un altro segmento dall'heap. Se la richiesta ha esito positivo, i nuovi nodi vengono collocati nell'elenco dei nodi liberi ed uno di essi viene scelto per soddisfare la richiesta originale. Se non è possibile ottenere altri segmenti, AutoLISP ricorre al sistema a paginazione virtuale delle funzioni in modo da liberare i nodi spostando la funzione utilizzata meno recentemente; altrimenti, viene visualizzato il messaggio spazio su nodo insufficiente e la richiesta viene annullata. Si noti che lo spazio per i nodi non viene *mai* restituito all'heap fino a quando non si esce da AutoCAD.

Mediante la funzione **gc** è possibile effettuare una ripulitura forzata dello spazio:

```
(gc)
```

Tuttavia, questo tipo di operazione richiede molto tempo per essere svolta e non dovrebbe essere effettuata se non è necessario; conviene quindi lasciare ad AutoLISP il compito di svolgere automaticamente questa ripulitura dello spazio, in modo tale da eseguirla solo quando necessario.

Capitolo 15 -- Gestione della memoria

Spazio per i nodi

Recupero dello spazio per i nodi

Se si creano funzioni e simboli che vengono utilizzati solo per un breve periodo di tempo, una volta finito di usarli, *annullarne la definizione* assegnando loro *nil*. Ad esempio, se è stata caricata ed utilizzata una funzione chiamata **setup** ed ora questa funzione non viene più usata, è possibile eliminarla nel modo seguente:

```
(setq setup nil)
```

Assegnando *nil* alla funzione, si recupera lo spazio per i nodi impiegato dalla funzione, in modo da poterlo usare per altre funzioni ed altri simboli.

Se si desidera liberare lo spazio di un nodo usato da un simbolo di tipo FILE (come restituito dalla funzione **open**), è necessario chiudere il simbolo prima di impostarlo su *nil*. Se la chiusura di un file non ha esito positivo, la porzione di memoria che esso occupa rimane in uso, riducendo il numero dei nuovi file che possono essere aperti.

Nota Nelle release precedenti alla 12, AutoCAD conservava *atomlist*, un elenco di tutte le funzioni e di tutti i simboli definiti nel simbolo. Questo elenco non viene più gestito in questo modo, quindi il metodo basato sulla "suddivisione dell'elenco degli atomi" non è più valido per cancellare le funzioni ed i simboli. Per richiamare un elenco delle funzioni e dei simboli definiti, usare la funzione **atoms-family**:

```
(atoms-family 0)
```

Questa funzione può essere usata per richiamare l'elenco completo dei simboli o solo quelli selezionati. Per ulteriori informazioni, vedere "**atoms-family**" nel capitolo 13.

Capitolo 15 -- Gestione della memoria

Note tecniche

Le informazioni riportate di seguito riguardano utenti LISP esperti. Tali informazioni descrivono algoritmi interni di AutoLISP che sono soggetti a modifica senza preavviso.

Capitolo 15 -- Gestione della memoria

Note tecniche

Spazio per le stringhe

Lo spazio di memoria per le stringhe proviene dallo stesso heap dei segmenti dei nodi. Se il programma richiama le funzioni di allocazione manuale (descritte di seguito) per assegnare tutta la memoria disponibile come nodi, è probabile che venga visualizzato il messaggio spazio su nodo insufficiente. Si raccomanda di lasciare la gestione dell'allocazione della memoria all'allocazione automatica dei nodi. Lo spazio per le stringhe viene utilizzato per qualsiasi stringa, dai nomi dei simboli alle stringhe dei messaggi di richiesta ed alle stringhe dei menu trasmesse ad AutoLISP per la valutazione. Se si usano voci di menu lunghe con le espressioni di AutoLISP per la valutazione, si tenga presente che richiedono all'heap una grande quantità di spazio contiguo per le stringhe.

Capitolo 15 -- Gestione della memoria

Note tecniche

Memoria per i simboli

La rappresentazione della memoria di AutoLISP è basata sull'uso intensivo dei puntatori. L'uso dei nodi è molto esteso e tutto viene rappresentato mediante una struttura basata su di essi. Il semplice gesto di impostare un simbolo con un nome lungo uguale ad un valore richiede due nodi: uno per memorizzare il nome del simbolo e l'altro per memorizzarne il valore.

```
(setq longsym 3.1415)
```

Se il nome del simbolo è lungo fino ad un massimo di sei caratteri, esso viene memorizzato direttamente nel nodo del nome del simbolo; in caso contrario, lo spazio per la stringa viene allocato sull'heap per memorizzare il nome e il nodo del nome del simbolo punterà quindi a questa stringa. In breve, l'uso di nomi brevi per i simboli (un massimo di sei caratteri) consente di ridurre la quantità dello spazio dedicato alle stringhe e la frammentazione dell'heap.

Capitolo 15 -- Gestione della memoria

Note tecniche

Allocazione manuale

Per controllare manualmente l'allocazione dei nodi e dello spazio per le stringhe, è possibile usare le funzioni **alloc** ed **expand**. Usando queste espressioni all'inizio del proprio file *acad.lsp*, è possibile preassegnare i nodi e riservare lo spazio per le stringhe. In questo modo si riduce il numero delle operazioni di ripulitura dello spazio, migliorando i tempi di esecuzione dell'applicazione.

Per alterare la dimensione delle future richieste di segmenti in modo che non siano di 514 nodi ognuno, è possibile usare la funzione **alloc**.

```
(alloc numero)
```

La funzione **alloc** imposta la dimensione dei segmenti su *numero* nodi e restituisce l'impostazione precedente.

Usando la funzione **expand**, è possibile allocare manualmente lo spazio per i nodi richiedendo un certo numero di segmenti:

```
(expand numero)
```

dove *numero* indica il numero dei segmenti che si desidera allocare.

La funzione **expand** restituisce il numero di segmenti che ha acquisito dall'heap; tale numero potrebbe essere più piccolo del numero richiesto a causa della esigua quantità restante di spazio di heap.

Esempio

```
(alloc 1024)
```

imposta la dimensione dei segmenti su 1024 nodi, richiedendo in tal modo 12.288 byte di spazio di heap per ogni segmento.

```
(alloc 100)
```

imposta la dimensione dei segmenti su 100, richiedendo in tal modo solo 1200 byte per ogni segmento.

Se per la dimensione dei segmenti è impostato il valore di default, ossia 514 nodi, una chiamata al seguente codice richiede 10 segmenti (10x12x514=61.480 byte).

```
(expand 10)
```

È possibile richiedere più segmenti di quanti ne siano disponibili.

Di seguito è riportato un esempio pratico:

```
(alloc 3000)           Imposta la dimensione dei segmenti dei nodi a 3000
(expand 1)             Ottiene 3000 nodi liberi (un segmento)
(alloc 10000)         Imposta un valore elevato per la dimensione dei segmenti per evitare di aggiungere altri
segmenti
```

Questo schema usa 36.000 byte per lo spazio per i nodi, lasciando il resto dell'heap per lo spazio per le stringhe. I nodi sono preassegnati e vengono posti nell'elenco dei nodi liberi. Non è necessario effettuare alcuna operazione di ripulitura dello spazio fino a quando non sono stati usati tutti i 3000 nodi. Una volta usati questi nodi, le ripuliture dello spazio vengono richiamate in modo da soddisfare richieste future di nodi. La funzione (alloc 10000) impedisce l'allocazione di altri segmenti dall'heap, consentendo quindi di riservare questo spazio per le stringhe.

Se si applica la strategia contraria riducendo sequenzialmente la dimensione dei segmenti fino a quando una richiesta per un nodo non ha esito negativo, (ossia, "(alloc 1) (expand 1)" restituisce 0), *tutto* l'heap viene usato per lo spazio per i nodi, provocando in tal modo la visualizzazione del messaggio di errore Spazio stringa insufficiente. Si consiglia di evitare che ciò avvenga, in quanto AutoLISP risulta del tutto inutile senza spazio disponibile per le stringhe.

Capitolo 15 -- Gestione della memoria

Note tecniche

Sistema a paginazione virtuale delle funzioni

Il sistema a paginazione virtuale delle funzioni può entrare in funzione solo dopo che sono stati esauriti tutti gli altri tipi di memoria virtuale; nella maggior parte delle piattaforme ciò avviene solo di rado. La funzione **vmon** viene fornita per risolvere i problemi di compatibilità con le versioni precedenti di AutoLISP.

Se l'applicazione AutoLISP diventa troppo grande perché lo spazio disponibile per i nodi sia in grado di contenerla, è possibile attivare il *paginatore virtuale di funzioni* di AutoLISP per consentire ulteriori sviluppi del programma. A tal fine, eseguire la funzione **vmon** prima della prima funzione **defun** del programma.

```
(vmon)
```

Questa funzione attiva il sistema a paginazione virtuale delle funzioni per il resto della sessione di disegno di AutoCAD. Una volta attivato, il sistema a paginazione delle funzioni non può essere più disattivato. La paginazione può essere effettuata solo per le funzioni che, create con **defun**, seguono la chiamata **vmon**; quindi, se si definiscono delle funzioni prima di tale chiamata, queste funzioni non possono essere paginate e potrebbero terminare il programma con un messaggio di errore Spazio nodo insufficiente.

Una volta eseguita la funzione **vmon**, AutoLISP pagina le funzioni usate meno di frequente ogni volta che lo spazio per i nodi risulta esaurito e le rilegge automaticamente quando necessario. Le funzioni vengono trasferite in un file temporaneo gestito mediante il paginatore di file di AutoCAD.

Il sistema a memoria virtuale pagina solo le *funzioni*, quindi è necessario ancora avere a disposizione spazio per i nodi sufficiente per collocarvi tutti gli elenchi di dati, i nomi delle funzioni e quelli delle variabili usati dal programma.

Quando **vmon** è attivata, la funzione **mem** visualizza altri due campi. *Richiamate* indica il numero delle funzioni *richiamate* dal file di paginazione creato per il sistema a memoria virtuale. Si tratta delle funzioni per cui è stata richiesta la paginazione e che verranno richiamate dal file di paginazione quando richieste in una semplice chiamata di funzione. Il valore del *file di paginazione* indica la dimensione del file di paginazione creato per contenere le funzioni paginate.

Una volta eseguita **vmon**, tutte le funzioni **defun** posizionano un nuovo nodo chiamato *tabella di paginazione* all'inizio di ogni elenco di funzioni. Questo nodo viene aggiunto prima dell'elenco contenente gli argomenti formali. I nodi tabelle di pagina vengono usati esclusivamente dal paginatore e non dovrebbero essere utilizzati in alcun modo da programmi dell'utente. La funzione **type** restituisce PAGETB per questi nodi.

Quando vengono esauriti i nodi, la funzione utilizzata meno di frequente viene spostata nel momento in cui AutoLISP la scrive nel file di paginazione, salvando l'indirizzo del file di paginazione nella tabella di paginazione e rilasciando tutti i nodi della funzione che seguono tale tabella. La tabella di paginazione viene contrassegnata per indicare che la funzione è stata spostata. Quando una funzione spostata viene valutata, viene riletta dal file di paginazione (possibilmente spostando altre funzioni) prima dell'esecuzione. Una volta che una funzione è stata scritta nel file di paginazione, eventuali spostamenti successivi ne rilasciano semplicemente i

nodi; non è necessario spostare ulteriormente la funzione, in quanto è già presente nel file.

In AutoLISP, le funzioni create con **defun** consistono in elenchi che possono essere manipolati. I programmi che svolgono questi tipi di operazioni devono tener presente l'azione svolta dal paginatore oppure non devono mai utilizzare la funzione **vmon**. Le funzioni create con **defun** possiedono un nodo tabella di paginazione nella parte iniziale, per cui, se si effettua l'analisi della funzione, si dovrebbe ignorare questo nodo. Se si crea una funzione come un elenco (ignorando la funzione **defun**), tale funzione verrà eseguita correttamente, ma non risulterà idonea per il trasferimento, quindi è possibile che venga esaurita tutta la memoria. È invece possibile bloccare una funzione in memoria ridefinendola senza la relativa tabella di paginazione. Ad esempio, per bloccare una funzione con il nome **zorp** in memoria, è possibile usare:

```
(setq zorp (cdr zorp))
```

per cancellare la tabella di paginazione iniziale. Quando si stampa la funzione, una tabella di paginazione viene stampata sotto forma di uno spazio. È possibile indicare se una funzione può essere trasferita, controllando se dopo la prima parentesi aperta è presente uno spazio: in tal caso, è possibile trasferirla.

Se si cerca di effettuare l'analisi di una funzione come dati e questa risulta spostata, nell'elenco delle funzioni compare solo la tabella di paginazione. Accedere alla funzione non significa richiamarla, in quanto ciò avviene solo quando la si *valuta*. Se si costruiscono delle funzioni e le si modifica temporaneamente, costruirle come elenchi invece di usare **defun**, oppure usare il metodo precedente per bloccarle in memoria.

Capitolo 16 -- Codici e messaggi di errore AutoLISP

Panoramica

In questo capitolo, vengono descritti i codici ed i messaggi di errore AutoLISP.

Argomenti di questo capitolo

{button „JI(‘,‘Error_Codes_al_u0310’)} Codici di errore

{button „JI(‘,‘Error_Messages_al_u0310’)} Messaggi di errore

Capitolo 16 -- Codici e messaggi di errore AutoLISP

Codici di errore

Nella tabella riportata di seguito sono elencati i valori dei codici di errore generati da AutoLISP. La variabile di sistema **ERRNO** viene impostata su uno di questi valori quando la chiamata di una funzione AutoLISP provoca un errore che viene rilevato da AutoCAD. Le applicazioni AutoLISP possono verificare il valore corrente di **ERRNO** con (**getvar "errno"**).

Non sempre la variabile di sistema **ERRNO** viene azzerata; se non viene controllata immediatamente dopo la comparsa di un errore generato da una funzione AutoLISP, l'errore indicato dal suo valore potrebbe creare confusione. Questa variabile viene sempre azzerata quando si avvia o si apre un disegno.

Nota I possibili valori di **ERRNO** ed il loro significato potrebbero subire delle modifiche nelle release successive di AutoCAD

Codici di errore del programma in linea

Valore	Significato
0	Nessun errore.
1	Il nome della tabella di simboli non è valido.
2	Il nome dell'entità o del gruppo di selezione non è valido.
3	Il numero massimo di gruppi di selezione è stato superato.
4	Il gruppo di selezione non è valido.
5	L'uso della definizione del blocco è improprio.
6	L'uso del comando xrif è improprio.
7	Selezione dell'oggetto: la selezione non è riuscita.
8	La fine del file di entità è stata raggiunta.
9	La fine del file di definizione del blocco è stata raggiunta.
10	L'ultima entità non è stata trovata.
11	Il tentativo di eliminare l'oggetto delle finestre non è valido.
12	L'operazione non è consentita durante PLINEA .
13	Il gestore non è valido.
14	I gestori non sono attivati.

- 15 *Gli argomenti nella richiesta di trasformazione delle coordinate non sono validi.*
- 16 *Lo spazio nella richiesta di trasformazione delle coordinate non è valido.*
- 17 *L'uso dell'entità cancellata non è valido.*
- 18 *Il nome della tabella non è valido.*
- 19 *L'argomento della funzione della tabella non è valido.*
- 20 *Si è tentato di impostare una variabile di sola lettura.*
- 21 *Il valore zero non è consentito.*
- 22 *Il valore non è compreso nell'intervallo.*
- 23 *RIGEN complesso in corso.*
- 24 *Si è tentato di modificare il tipo di entità.*
- 25 *Il nome del layer è errato.*
- 26 *Il nome del tipo di linea è errato.*
- 27 *Il nome del colore è errato.*
- 28 *Il nome dello stile del testo è errato.*
- 29 *Il nome della forma è errato.*
- 30 *Il campo per il tipo di entità è errato.*
- 31 *Si è tentato di modificare un'entità cancellata.*
- 32 *Si è tentato di modificare la sottoentità seqend.*
- 33 *Si è tentato di modificare il gestore.*
- 34 *Si è tentato di modificare l'aspetto della finestra.*
- 35 *Sul layer bloccato è presente un'entità.*
- 36 *Il tipo di entità è errato.*
- 37 *L'entità di polilinea è errata.*
- 38 *L'entità complessa non è completa nel blocco.*
- 39 *Il campo del nome di blocco non è valido.*
- 40 *Campi del flag del blocco duplicati.*
- 41 *Campi del flag del blocco duplicati.*
- 42 *Il vettore perpendicolare è errato.*
- 43 *Il nome del blocco è mancante.*
- 44 *I flag del blocco sono mancanti.*
- 45 *Il blocco anonimo non è valido.*
- 46 *La definizione di blocco non è valida.*
- 47 *Campo obbligatorio mancante.*
- 48 *Il tipo di dati estesi (XDATA) non è riconosciuto.*
- 49 *La nidificazione dell'elenco in XDATA è impropria.*
- 50 *La posizione del campo APPID è impropria.*
- 51 *La dimensione massima di XDATA è stata superata.*
- 52 *Selezione dell'entità: la risposta è nulla.*
- 53 *APPID duplicato.*
- 54 *Si è tentato di creare o modificare l'entità della finestra.*
- 55 *Si è tentato di modificare o creare un xrif, xdef o xdep.*
- 56 *Filtro ssgset : la fine dell'elenco non è prevista.*
- 57 *Filtro ssgset : l'operando di verifica è mancante.*
- 58 *Filtro ssgset : la stringa opcode (-4) non è valida.*
- 59 *Filtro ssgset: la nidificazione è impropria o la clausola condizionale è vuota.*
- 60 *Filtro ssgset: non c'è corrispondenza iniziale e finale della clausola condizionale.*
- 61 *Filtro ssgset: il numero di argomenti nella clausola condizionale è errato (per NOT o XOR).*
- 62 *Filtro ssgset: il limite massimo di nidificazione è stato superato.*
- 63 *Filtro ssgset: il codice di gruppo non è valido.*
- 64 *Filtro ssgset: la verifica della stringa non è valida.*
- 65 *Filtro ssgset: la verifica del vettore non è valida.*
- 66 *Filtro ssgset: la verifica del tipo di dati "real" non è valida.*
- 67 *Filtro ssgset: la verifica del numero intero non è valida.*
- 68 *Il digitalizzatore non è una tavoletta.*
- 69 *La tavoletta non è calibrata.*
- 70 *Gli argomenti non sono validi.*
- 71 *Errore ADS: è impossibile assegnare il nuovo buffer dei risultati.*
- 72 *Errore ADS: il puntatore ha rilevato un valore nullo.*
- 73 *È impossibile aprire il file eseguibile.*
- 74 *L'applicazione è già stata caricata.*
- 75 *Il numero massimo di applicazioni è già stato caricato.*
- 76 *È impossibile eseguire l'applicazione.*
- 77 *Il numero di versione non è compatibile.*
- 78 *È impossibile scaricare l'applicazione nidificata.*
- 79 *È impossibile scaricare l'applicazione.*
- 80 *L'applicazione non è attualmente caricata.*
- 81 *La memoria per caricare l'applicazione è insufficiente.*
- 82 *Errore ADS: la matrice di conversione non è valida.*

- 83 *Errore ADS: il nome di simbolo non è valido.*
 84 *Errore ADS: il valore di simbolo non è valido.*
 85 *L'azione di AutoLISP/ADS non è consentita durante la visualizzazione di una finestra di dialogo.*

Capitolo 16 -- Codici e messaggi di errore AutoLISP

Messaggi di errore

Quando AutoLISP rileva una condizione di errore, annulla la funzione in corso e richiama la funzione ***error*** dell'utente con un messaggio che indica il tipo di errore. Se non viene definita alcuna funzione ***error*** dell'utente (o se ***error*** è nil), l'azione di errore standard prevede la visualizzazione di un messaggio con questo formato:

error: messaggio

Il messaggio è seguito dalla parte di codice in cui è contenuto l'errore. Se esiste una funzione ***error*** definita dall'utente, la parte di codice in cui è contenuta non viene visualizzata, ma la funzione definita dall'utente viene richiamata passando il *messaggio* come suo unico argomento.

Capitolo 16 -- Codici e messaggi di errore AutoLISP

Messaggi di errore

Errori del programma dell'utente

In questa sottosezione vengono descritti i messaggi di errore che possono essere visualizzati durante la scrittura e le operazioni di debugging delle funzioni AutoLISP. La maggior parte di questi messaggi indica errori di programmazione AutoLISP, ad esempio:

- Nomi di funzioni o simboli errati
- Tipo o numero di argomenti di una funzione errato
- Mancanza di corrispondenza nelle parentesi
- Mancanza di corrispondenza nelle virgolette (stringhe non terminate)
- Errore nella verifica dell'esatto completamento di una funzione prima di tentare l'utilizzo del relativo risultato.

Sebbene questi messaggi di solito indichino gli errori di programmazione dell'utente, essi possono essere causati anche da errori di programmazione all'interno dello stesso AutoLISP. Se non viene rilevato alcun errore all'interno del proprio programma, compilare il modulo della relazione sugli errori di programmazione interni (Bug Report) ed inviarlo ad Autodesk insieme ad una copia del programma (preferibilmente su un dischetto).

gli argomenti di defun non possono avere lo stesso nome

Una funzione definita da più argomenti con lo stesso nome non avrà esito positivo e causerà la visualizzazione di questo messaggio.

la funzione è stata respinta da AutoCAD

Gli argomenti passati ad una funzione di AutoCAD non erano validi (come nell'esecuzione del comando **modivar** con una variabile di sistema di sola lettura o di **tblnext** con un nome di tabella non valido) oppure la funzione stessa non è valida nel contesto corrente. Ad esempio, non è possibile usare una funzione di input utente **getxxx** all'interno della funzione **command**.

overflow della pila di AutoLISP

È stata superata la quantità di memoria dello stack di AutoLISP. Tale condizione di errore può essere causata dalla ripetizione eccessiva di una funzione o da elenchi di argomenti della funzione troppo lunghi.

il tipo di argomento è errato

Ad una funzione è stato passato un tipo di argomento non corretto. Ad esempio, non è possibile prendere il valore di **strlen** di un

valore intero.

la lista associativa è errata

La lista fornita alla funzione **assoc** non è composta da elenchi di valori chiave.

il codice di conversione è errato

Un identificatore di spazio non valido è stato passato alla funzione **trans**.

la lista ENTMOD è errata

L'argomento passato a **entmod** non è una lista appropriata di dati dell'entità (come restituito da **entget**).

il valore della lista ENTMOD è errato

Una delle liste secondarie nella lista associativa passata a **entmod** contiene un valore non valido.

la lista di argomenti formali è errata

Durante la valutazione di questa funzione, AutoLISP ha individuato una lista di argomenti formali non valida. Forse non si tratta affatto di una funzione, ma piuttosto di una lista di dati.

la funzione è errata

Il primo elemento della lista non è un nome di funzione valido, ma potrebbe trattarsi di un nome di variabile o un numero. Questo messaggio può indicare anche che la funzione nominata è stata definita in modo non appropriato. Tenere sempre in considerazione la lista di argomenti formali richiesta.

il codice di funzione è errato

Un identificatore di funzione errato è stato passato al comando **tavolet**.

il valore della lista grvecs è errato

All'elenco **grvecs** non è stato passato un punto bidimensionale o tridimensionale.

il valore matrice di grvecs è errato

Una matrice passata a **grvecs** non è stata formata correttamente oppure contiene un tipo di dati non valido (ad esempio STR, SYM e così via).

la lista è errata

Una lista formata in modo non appropriato è stata passata ad una funzione. Ciò può verificarsi se un numero reale inizia con un punto decimale. In questi casi, è necessario inserire uno zero iniziale.

la lista dei punti è errata

Una lista nulla o una lista contenente voci diverse da punti è stata inviata insieme ad una richiesta IN, IP o FP. Usato da **ssget** e **grvecs**.

il nodo è errato

La funzione **type** ha rilevato un tipo di elemento non valido.

il tipo di nodo nell'elenco è errato

La funzione **foreach** ha rilevato un tipo di elemento non valido.

l'argomento del punto è errato il valore del punto è errato

Un punto definito in modo incompleto (un elenco di due valori) è stato passato ad una funzione che prevede il passaggio di un punto. Fare attenzione a non iniziare un numero reale con un punto decimale; in questo caso è necessario inserire uno zero iniziale.

bad real number detected (trovato numero reale errato)

È stato effettuato un tentativo per passare un numero reale non valido da AutoLISP ad AutoCAD.

la lista ssget è errata

L'argomento passato a (**ssget "X"**) non è una lista appropriata di dati dell'entità (come restituito da **entget**).

il valore della lista ssget è errato

Una delle liste secondarie nella lista associativa passata a (**ssget "X"**) contiene un valore non valido.

la stringa di modalità ssget è errata

Errore causato quando a **ssget** viene passata una stringa non valida nell'argomento *modalità*.

la lista xdata è errata

Errore causato quando a **xdsite**, **ssget**, **entmod**, **entmake** o **txtbox** viene passato un elenco di dati estesi dell'entità formato in modo non valido.

è richiesto il punto base

La funzione **getcorner** è stata richiamata senza il necessario argomento del punto base.

Boole arg1 0 o 15

Il primo argomento passato alla funzione Boole deve essere un numero intero compreso tra 0 e 15.

impossibile valutare l'espressione

Un punto decimale è stato posizionato in modo non appropriato oppure qualche altra espressione è stata formata in modo non corretto.

impossibile aprire (file) per input -- LOAD non è riuscito

Il file nominato nella funzione **load** non è stato trovato oppure l'utente non ha accesso alla lettura di quel file.

impossibile tornare in AutoLISP

Il buffer di comunicazione di AutoCAD/AutoLISP è utilizzato da una funzione attiva. Nessuna nuova funzione può essere richiamata finché quella attiva non è completata.

interruzione da tastiera

L'utente ha digitato CTRL+C durante l'elaborazione di una funzione.

dividere per zero

La divisione per zero non è consentita.

overflow di divisione

La divisione per un valore molto piccolo ha avuto come risultato un quoziente non valido.

è stata superata la lunghezza massima della stringa

Una stringa passata ad una funzione è composta da più di 132 caratteri.

una parentesi destra è di troppo

Sono state rilevate una o più parentesi destra superflue.

il file non è aperto

Il descrittore del file per l'operazione di I/O non è quello di un file aperto.

lettura del file-lo spazio di stringa è insufficiente

Lo spazio della stringa è stato esaurito mentre AutoLISP stava leggendo un file. Vedere il capitolo 15, "Gestione della memoria"..

file size limit exceeded (limite del file superato)

Un file ha superato la dimensione massima consentita per i file del sistema operativo.

floating-point exception (eccezione a virgola mobile)

(Solo per i sistemi basati su UNIX). In questi casi è necessario inserire uno zero.

la funzione è stata annullata

L'utente ha digitato CTRL+C o ESC (annulla) in risposta ad una richiesta di input.

la funzione per l'argomento non è definita

L'argomento passato a **log** o **sqrt** non è compreso nell'intervallo.

la funzione non è definita per real

Un numero reale è stato passato come argomento ad una funzione che richiedeva un numero intero, ad esempio (**lsh val 1.2**).

il punto finale è mancato a grvecs

L'elenco dei vettori passato a **grvecs** non contiene un punto finale.

tipo illecito in LEFT

Il file *.lsp* non è ASCII puro, ma è stato salvato da un programma di elaborazione di testi e nel file sono inclusi i codici di formattazione.

l'argomento non è corretto

L'argomento passato a **gcd** è negativo o uguale a zero.

l'oggetto nella funzione è inappropriato

Una funzione costruita in modo non appropriato è stata individuata dal paginatore di funzioni **vmon**.

il numero di argomenti è incorretto

Per la funzione **quote** è previsto un solo argomento, ma è stato fornito un altro numero di argomenti.

il numero di argomenti per una funzione è incorretto

Il numero di argomenti passati alla funzione definita dall'utente non corrisponde al numero di argomenti formati specificati in *defun*.

la richiesta per i dati della lista di comandi non è valida

È stata rilevata una funzione di comando che non può essere eseguita a causa di un'altra funzione attiva oppure l'interprete dei comandi non è stato inizializzato completamente. Ciò può verificarsi a causa di una chiamata alla funzione **command** nel file *acad.lsp*, *acadr13.lsp* o *.mnl*.

l'input è stato annullato

È stato rilevato un errore o una condizione di fine file prematura, che provoca il termine dell'input in un file.

lo spazio nodale è insufficiente

La quantità di memoria heap non è sufficiente per l'esecuzione dell'azione richiesta. Vedere il capitolo 15, "Gestione della memoria".

lo spazio di stringa è insufficiente

La quantità di memoria heap non è sufficiente per la stringa di testo specificata. Vedere il capitolo 15, "Gestione della memoria".

l'argomento non è valido

Il tipo di argomento non è appropriato oppure l'argomento non è incluso nell'intervallo.

l'elenco di argomenti non è valido

Un elenco di argomenti danneggiato è stato passato ad una funzione.

il carattere non è valido

Un'espressione contiene un carattere non appropriato.

il paio punteggiato non è valido

Le coppie punteate sono elenchi contenenti due elementi separati dalla costruzione *spazio-punto-spazio*. Questo messaggio di errore può essere visualizzato se un numero reale inizia con un punto decimale; in questo caso, è necessario inserire uno zero iniziale.

il valore intero non è valido

È stato rilevato un numero inferiore o superiore al numero intero minore o maggiore.

overflow di LISPSTACK

È stata superata la quantità di memoria dello stack di AutoLISP. Tale condizione di errore può essere causata dalla ripetizione eccessiva di una funzione o da elenchi di argomenti della funzione troppo lunghi.

l'elenco è errato

Un elenco in lettura da un file è terminato in modo prematuro. La causa più comune è la mancata corrispondenza negli accoppiamenti di parentesi e virgolette aperte e chiuse.

la stringa è errata

Una stringa in lettura da un file è terminata prematuramente.

il punto è spostato

Un numero reale inizia con un punto decimale. In questi casi, è necessario inserire uno zero iniziale.

funzione null

È stato effettuato un tentativo di valutazione per una funzione che ha una definizione nulla.

esci./continua

È stata richiamata la funzione **quit** o **exit**.

la stringa è troppo lunga

Una stringa passata a **setvar** è troppo lunga.

gli argomenti sono insufficienti

Un numero di argomenti insufficiente è stato passato ad una funzione.

gli argomenti per grvecs sono insufficienti

Un numero di argomenti insufficiente è stato passato a **grvecs**.

gli argomenti sono troppi

Un numero di argomenti eccessivo è stato passato ad una funzione incorporata.

Capitolo 16 -- Codici e messaggi di errore AutoLISP

Messaggi di errore

Errori interni

I messaggi di errore riportati di seguito indicano errori interni al programma AutoLISP. Questi messaggi devono essere annotati sul modulo della relazione sugli errori di programmazione (Bug Report) ed inviare tale modulo ad Autodesk.

bad argument to system call (argomento sbagliato alla chiamata di sistema)

(Solo per i sistemi basati su UNIX). Il sistema operativo ha rilevato una chiamata di sistema errata che è stata generata da AutoLISP.

bus error (errore bus)

(Solo per i sistemi basati su UNIX). Il sistema operativo ha rilevato un errore del bus.

hangup (hangup)

(Solo per i sistemi basati su UNIX). Il sistema operativo ha rilevato un segnale di *hangup*.

illegal instruction (istruzione illecita)

(Solo per i sistemi basati su UNIX). Il sistema operativo ha rilevato un'istruzione macchina non valida.

segmentation violation (violazione di segmentazione)

(Solo per i sistemi basati su UNIX). Il sistema operativo ha rilevato un tentativo di indirizzamento all'esterno dell'area di memoria di

questo processo.

unexpected signal nnn (segnale inatteso nnn)

(Solo per i sistemi basati su UNIX). Il sistema operativo ha ricevuto un segnale non previsto.

Parte III -- Finestre di dialogo programmabili

Panoramica

La parte *Finestre di dialogo programmabili* descrive come realizzare e implementare finestre di dialogo simili a quelle usate da AutoCAD. Le funzioni per la visualizzazione di finestre di dialogo, per l'elaborazione di input da parte dell'utente e per altre attività sono fornite sia per AutoLISP che per ADSRX.

Poiché le finestre di dialogo sono controllate da applicazioni AutoLISP e ADSRX, è necessario avere familiarità con le funzioni di gestione delle finestre di dialogo descritte nel capitolo 10, "Gestione delle finestre di dialogo", e nella documentazione relativa alle applicazioni ARX. Nel capitolo 17, "Finestre di dialogo programmabili", sono utilizzate come esempio le funzioni AutoLISP. Il corso interattivo AutoLISP nel capitolo 11, "Corso interattivo di AutoLISP", mostra come aggiungere un'interfaccia per una finestra di dialogo ad un'applicazione.

Il supporto AutoCAD per finestre di dialogo incorporate e per finestre definite dall'utente è indipendente dalla piattaforma. I componenti e la funzionalità di una finestra di dialogo restano invariati tra le piattaforme, ma l'aspetto dipende dall'interfaccia grafica utente della piattaforma (GUI). La funzionalità di finestre di dialogo programmabili fornisce layout e dimensionamento automatico, limitando il lavoro del programmatore e facilitando l'uniformità dell'intero progetto.

Capitolo 17 -- Finestre di dialogo programmabili

Panoramica

Le finestre di dialogo sono definite da file ASCII scritti con il linguaggio DCL (Dialog Control Language). La disposizione degli elementi (caselle) in una finestra di dialogo, come pulsanti e caselle di modifica, viene determinata dal loro ordine nel file DCL. La dimensione e la funzionalità di ogni elemento è controllata dagli attributi della casella. Il disegno delle finestre di dialogo ha alcune limitazioni: la dimensione della casella ed il layout delle sue parti vengono impostati automaticamente, con una informazione minima sul posizionamento. Le parti di una finestra di dialogo definiscono il suo comportamento; tuttavia, l'uso ed il comportamento dipendono dall'applicazione che la utilizza. AutoLISP ed ADSRX forniscono le funzioni necessarie per il controllo delle finestre di dialogo.

In questo capitolo vengono presentati gli elementi che costituiscono le finestre di dialogo; viene spiegata la struttura e la sintassi dei file DCL e viene mostrato un esempio di codice AutoLISP e DCL per una finestra di dialogo. In questo capitolo vengono descritte anche alcune tecniche di codifica DCL per la gestione dei problemi relativi al layout.

Argomenti di questo capitolo

{button ,JI('`Dialog_Box_Components_al_u0401')} Componenti della finestra di dialogo

{button ,JI('`DCL_File_Structure_al_u0401')} Struttura dei file DCL

{ewc ,JI('`DCL_Techniques_al_u0401')} Tecniche DCL

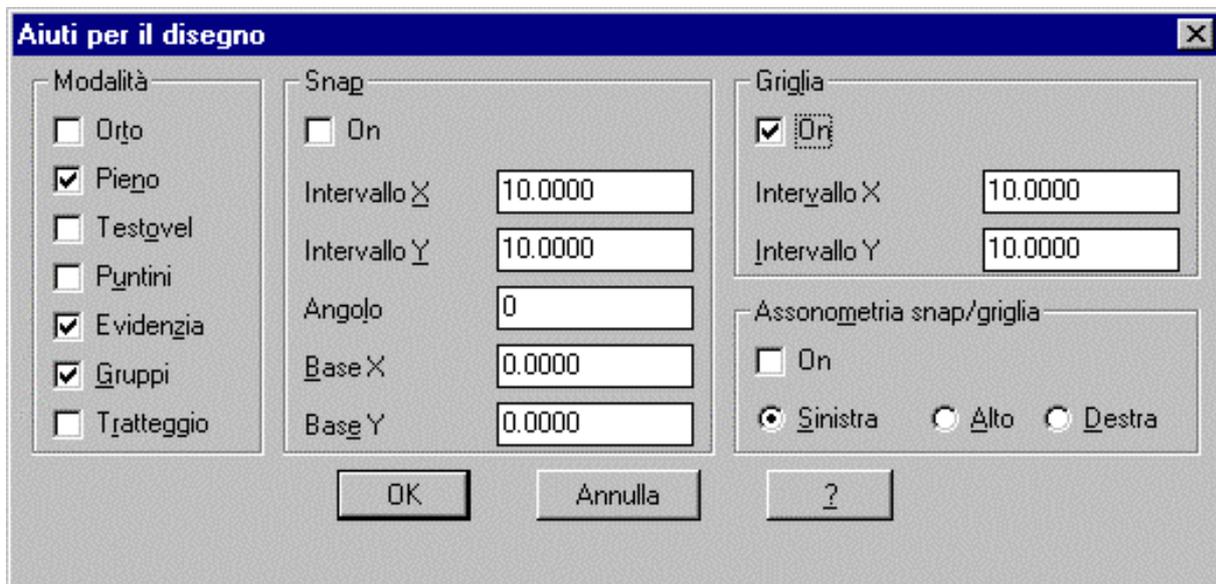
{ewc ,JI('`Design_Guidelines_al_u0401')} Direttive di progettazione

{button ,JI('`Related_Documentation_al_u0401')} Documentazione correlata

Capitolo 17 -- Finestre di dialogo programmabili

Componenti della finestra di dialogo

Nella figura successiva viene mostrata una delle finestre di dialogo standard di AutoCAD, con alcuni dei suoi componenti forniti di etichetta. Nella creazione e personalizzazione delle finestre di dialogo, questi componenti sono conosciuti come *caselle*.



Caselle tipiche di una finestra di dialogo

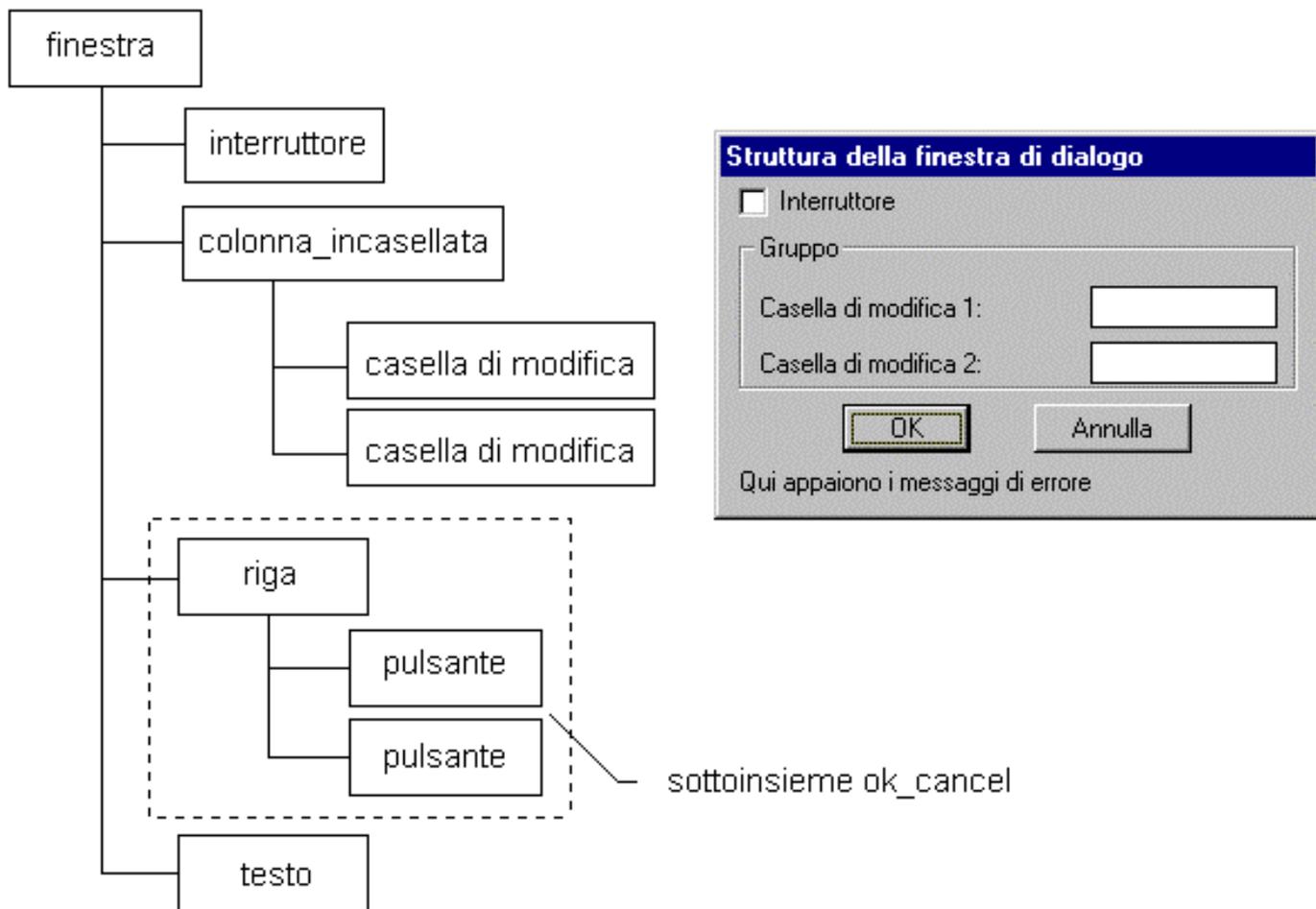
Una finestra di dialogo è composta dal riquadro stesso e dagli elementi o caselle in esso contenute. I tipi di base vengono predefiniti dalla funzione PDB. È possibile creare caselle complesse, chiamate sottoinsiemi, raggruppando le caselle in righe e colonne, con o senza un bordo o un riquadro intorno ad esse. I sottoinsiemi consentono di definire gruppi di caselle che possono essere usati in più finestre di dialogo. Ad esempio, i pulsanti OK, Annulla e ? sono raggruppati in un sottoinsieme. I sottoinsiemi vengono considerati come singole caselle, mentre le caselle che li compongono vengono chiamate *sottocaselle*. I file DCL sono organizzati secondo una struttura ad albero. La casella che si trova all'inizio della struttura (dialog) definisce la finestra di dialogo.

Il layout, l'aspetto ed il comportamento di una casella o di un sottoinsieme sono specificati in DCL dagli attributi della casella. Ad esempio, la finestra di dialogo e la maggior parte dei tipi di caselle predefinite dispongono di un attributo etichetta (label) che specifica il testo associato alla casella. L'etichetta di una finestra di dialogo definisce l'intestazione che verrà posizionata nella parte superiore di una finestra di dialogo, mentre l'etichetta di un pulsante specifica il testo all'interno del pulsante, e così via.

DCL consente, inoltre, di definire nuove caselle, chiamate *prototipi*, che non sono necessariamente associate ad una specifica finestra di dialogo. Ciò è particolarmente utile quando si desidera utilizzare lo stesso componente in più finestre di dialogo. È possibile fare riferimento alle caselle prototipo da altri file DCL e modificare i loro attributi nello stesso modo in cui vengono modificate le caselle predefinite.

Una finestra di dialogo è essenzialmente un albero di caselle ed un file DCL è la versione leggibile di questi alberi, sia per gli sviluppatori che per la funzione PDB di AutoCAD. La gerarchia di un file DCL è indicata dalla sua sintassi. È possibile dividere il file DCL per esaminarne la gerarchia; si consiglia di utilizzare il criterio di divisione mostrato negli esempi contenuti in questo capitolo.

Utilizzando DCL, è possibile creare descrizioni per finestre di dialogo in base alla struttura ad albero mostrata nella figura seguente. La finestra di dialogo a destra indica la posizione di ogni casella nella struttura ad albero.



Struttura di una finestra di dialogo

Le descrizioni DCL delle finestre di dialogo sono file ASCII, molto simili alle descrizioni del linguaggio sorgente di programmazione o del sorgente del menu (.mnu) di AutoCAD. Tali file hanno come estensione .dcl. Un singolo file .dcl può contenere la descrizione di una o più finestre di dialogo oppure può contenere solo caselle prototipo e sottoinsiemi che possono essere utilizzati da altri file .dcl.

Prima di programmare una finestra di dialogo, progettare in tutti i particolari sia la finestra di dialogo che l'applicazione, quindi eseguire la codifica e la successiva individuazione degli errori. Poiché ogni utente effettuerà la digitazione dei dati con una sequenza diversa, è necessario che la struttura del programma sia meno lineare di quella imposta dagli standard di programmazione convenzionali, ma che rifletta il modo di lavorare dell'utente.

Capitolo 17 -- Finestre di dialogo programmabili

Struttura dei file DCL

Oltre alla definizione delle finestre di dialogo, i file DCL possono contenere la definizione delle caselle prototipo ed i sottoinsiemi che verranno utilizzati da altri file DCL e possono fare riferimento a prototipi e sottoinsiemi definiti in altri file DCL. Un file DCL è formato da tre parti, descritte di seguito, disposte in qualsiasi ordine. In base al tipo di applicazione utilizzata, potrebbe non essere necessario usarle tutte.

- *Riferimenti ad altri file DCL.* Sono costituiti da direttive include come descritto in "Riferimenti ai file DCL."
- *Definizioni di caselle prototipo e sottoinsiemi.* Sono le definizioni delle caselle alle quali si può fare riferimento in successive definizioni di caselle, comprese le definizioni delle finestre di dialogo.
- *Definizioni delle finestre di dialogo.* Sono molto simili a quelle dell'esempio mostrato in "Esempio di finestra di dialogo."

Capitolo 17 -- Finestre di dialogo programmabili

Struttura dei file DCL

File *base.dcl* e *acad.dcl*

I file *base.dcl* e *acad.dcl* sono inclusi in AutoCAD e si trovano nella directory *support*.

Nel file *base.dcl* si trovano le definizioni DCL per le caselle ed i tipi di caselle di base predefinite. In esso sono contenute inoltre le definizioni dei prototipi usati più frequentemente. La funzione PDB non consente di ridefinire le caselle predefinite. Nel file *acad.dcl* sono contenute le definizioni standard di tutte le finestre di dialogo utilizzate da AutoCAD.

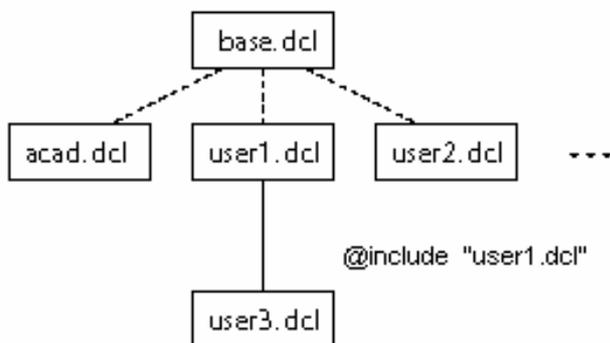
Attenzione Qualsiasi tipo di errore nel file *base.dcl* potrebbe modificare l'aspetto delle finestre di dialogo standard di AutoCAD ed anche quello delle finestre di dialogo personalizzate nella propria o in altre applicazioni.

Capitolo 17 -- Finestre di dialogo programmabili

Struttura dei file DCL

Riferimenti ai file DCL

Durante la creazione delle finestre di dialogo, è necessario creare un nuovo file DCL, specifico per l'applicazione. Tutti i file DCL definiti possono utilizzare le caselle definite nel file *base.dcl*. Un file DCL può utilizzare anche le caselle definite in un altro file DCL, nominando questo file nella direttiva *include*. È possibile creare la propria gerarchia di file DCL, come mostrato nella figura riportata di seguito.



Gerarchia di un file DCL

Nella figura precedente, i file *user1.dcl* e *user2.dcl* sono indipendenti l'uno dall'altro, ma *user3.dcl* utilizza le caselle definite in *user1.dcl*. La direttiva include ha il seguente formato:

```
@include nomefile
```

dove *nomefile* è una stringa tra virgolette che contiene il nome completo dell'altro file DCL, con estensione *.dcl*. Ad esempio, la seguente direttiva include il file che si chiama *usercore.dcl*:

```
@include "usercore.dcl"
```

Se viene specificato solo il nome del file, la funzione PDB ricerca il file nella directory corrente e, quindi, nella stessa directory del file DCL contenente l'istruzione di inclusione. Se viene specificato un percorso completo, la funzione PDB effettua la ricerca soltanto nella directory specificata nel percorso.

Nota Nei file DCL creati dall'utente non è possibile usare le finestre di dialogo definite in *acad.dcl*. Non è quindi possibile specificare `@include "acad.dcl"`. Tuttavia, se si intende creare finestre di dialogo simili, è possibile tagliare ed incollare le definizioni nel proprio file DCL.

Capitolo 17 -- Finestre di dialogo programmabili

Struttura dei file DCL

Sintassi DCL

In questa sezione viene descritta la sintassi DCL che consente di specificare caselle, attributi delle caselle e valori degli attributi. Le nuove caselle vengono create utilizzando *definizioni* di caselle. Se la definizione di una casella non si trova all'interno della definizione della finestra di dialogo, allora si tratta di un prototipo o di un sottoinsieme; tali componenti possono essere usati nelle definizioni delle finestre di dialogo tramite i *riferimenti* alle caselle. Ogni riferimento ad una definizione eredita gli attributi della casella originaria. Nei riferimenti a prototipi, è possibile modificare i valori degli attributi ereditati oppure aggiungere nuovi attributi. Nei riferimenti ai sottoinsiemi, invece, non è possibile modificare o aggiungere attributi.

Se sono necessarie più istanze di una casella con alcuni attributi in comune, è più semplice definire e nominare un prototipo che contenga soltanto gli attributi in comune. Successivamente, in ogni riferimento al prototipo, è possibile cambiare gli attributi o aggiungerne di nuovi, ma non è necessario elencare tutti gli attributi in comune ogni volta che si fa riferimento alla casella. Poiché gli attributi vengono ereditati, sarà spesso necessario creare riferimenti alle caselle, soprattutto quelli a caselle predefinite, piuttosto che definire nuove caselle.

Capitolo 17 -- Finestre di dialogo programmabili

Struttura dei file DCL

Sintassi DCL

Definizioni di caselle

Le definizioni delle caselle hanno il seguente formato:

```
nome : elemento1 [ : elemento2 : elemento3 ... ] {
    attributo = valore;
    ...
}
```

dove ogni *elemento* rappresenta una casella definita precedentemente. La nuova casella (*nome*) eredita gli attributi di tutte le caselle specificate (*elemento1*, *elemento2*, *elemento3*, ...), e le definizioni di attributi all'interno della parentesi graffa ({}), integrano o sostituiscono (nel caso in cui il nome dell'attributo sia identico) le definizioni ereditate. Se lo stesso attributo è stato specificato in più caselle, il primo incontrato sarà quello usato, in base alla regola di precedenza per gli attributi da sinistra a destra.

Se la nuova definizione non contiene sottocaselle, si tratta di un prototipo ed eventuali riferimenti ad essa possono modificare o aumentare i suoi attributi. Se invece si tratta di un sottoinsieme con sottocaselle, i suoi attributi non possono subire modifiche.

Il *nome* di una casella o un prototipo di casella può contenere solo lettere, numeri o il carattere di sottolineatura () e deve iniziare con una lettera. I nomi delle caselle sono sensibili al maiuscolo/minuscolo, ad esempio, pulsante non è la stessa cosa di Pulsante o di PULSANTE. Prestare, quindi, particolare attenzione all'uso delle lettere maiuscole.

Questa è la definizione (interna) di un pulsante:

```
button : tile {
    fixed_height = true;
    is_tab_stop = true;
}
```

Nel file *base.dcl* viene definito `default_button` come indicato di seguito:

```
default_button : button {
    is_default = true;
}
```

Il `default_button` eredita i valori della casella `button` per gli attributi `fixed_height` e `is_tab_stop`; aggiunge un nuovo attributo, `is_default` e lo imposta su `true`.

Capitolo 17 -- Finestre di dialogo programmabili

Struttura dei file DCL

Sintassi DCL

Riferimenti alle caselle

I riferimenti alle caselle hanno uno dei seguenti formati:

```
nome;
```

oppure

```
: nome {
    attributo = valore;
    . . .
}
```

dove *nome* rappresenta il nome di una casella precedentemente definita. I nomi delle caselle sono sensibili al maiuscolo/minuscolo. Nella prima istanza, tutti gli attributi definiti in *nome* vengono incorporati nel riferimento. Nella seconda istanza, le definizioni degli attributi all'interno delle parentesi graffe integrano o sostituiscono le definizioni ereditate da *nome*. Poiché si tratta di un riferimento ad una casella, e non di una definizione, le modifiche relative agli attributi vengono applicate soltanto a questa istanza della casella.

Nota Il formato della seconda istanza può fare riferimento soltanto a prototipi, non a sottoinsiemi.

Nella definizione di una finestra di dialogo, la casella `spacer` viene usata per il layout. I suoi attributi non sono univoci, quindi i riferimenti ad esso relativi possono usare il formato della prima istanza:

```
spacer;
```

La casella `ok_cancel` definita nel file *base.dcl* è un sottoinsieme, quindi ad essa si può fare riferimento solo con il formato della prima istanza:

```
ok_cancel;
```

È possibile, invece, ridefinire gli attributi di una singola casella. Ad esempio, per creare un pulsante con le stesse caratteristiche ma testo diverso, usare il seguente formato:

```
: retirement_button {
    label = "Arrivederci";
}
```

Per ulteriori informazioni, vedere "Personalizzazione del testo dei pulsanti di uscita."

Capitolo 17 -- Finestre di dialogo programmabili

Struttura dei file DCL

Sintassi DCL

Attributi e valori degli attributi

All'interno delle parentesi graffe della definizione o del riferimento di una casella, è possibile specificare gli attributi ed assegnare loro dei valori utilizzando il seguente formato:

```
attributo = valore;
```

dove *attributo* rappresenta una parola chiave valida e *valore* il valore assegnato all'attributo. Un segno di uguale (=) separa l'attributo dal valore ed il punto e virgola (;) conclude l'assegnazione.

Come i nomi delle caselle, anche i nomi ed i valori degli attributi sono sensibili al maiuscolo/minuscolo. Larghezza e larghezza non sono la stessa cosa, così come Vero e vero non producono lo stesso effetto.

Capitolo 17 -- Finestre di dialogo programmabili

Struttura dei file DCL

Sintassi DCL

Commenti

Nel codice di programmazione vengono di solito inclusi alcuni commenti per fornire informazioni utili, come il nome del programma o i file ad esso correlati, il nome dell'autore e tutte le informazioni esplicative e relative all'utilizzo. Un commento in un file DCL è preceduto da due barre (*//*). Tutto ciò che è compreso tra *//* e la fine della riga viene ignorato. DCL consente di introdurre commenti anche con lo stile del linguaggio C, nel formato */* testo del commento */*. Il contrassegno iniziale */** e quello finale **/* possono trovarsi su righe diverse.

Capitolo 17 -- Finestre di dialogo programmabili

Struttura dei file DCL

Gestione degli errori DCL

Quando un file DCL viene caricato per la prima volta, la funzione PDB ne esegue la verifica. Se AutoCAD rileva un errore di sintassi, un uso improprio degli attributi o un altro tipo di errore, come la specifica errata di un attributo chiave per una casella attiva, non eseguirà il caricamento del file DCL, ma riporterà gli errori in un elenco all'interno di un file di testo chiamato *acad.dce*. Per individuare il problema è possibile esaminare il contenuto di *acad.dce*. Quando la lettura di un file DCL ha esito positivo, il file *acad.dce* viene cancellato. AutoCAD colloca il file *acad.dce* nella directory di lavoro corrente.

Se nell'applicazione vengono utilizzati più file DCL, il file *acad.dce* viene ricoperto (o cancellato se non si verificano errori) al caricamento del file successivo. Quando viene effettuata la verifica del programma, *acad.dce* mostra gli errori, se presenti, relativi solo all'ultimo file DCL che è stato letto. Di conseguenza, si consiglia di caricare manualmente ogni file con la funzione **load_dialog** per eseguire il debug degli errori. La funzione **load_dialog** mostrata di seguito carica il file DCL *fileciaio.dcl*:

Comando: (**load_dialog "fileciaio"**)

3

La funzione **load_dialog** restituisce il numero di identificazione DCL. Prendere nota di questo numero, poiché viene passato in qualità di argomento alle successive chiamate delle funzioni **new_dialog** e **unload_dialog**.

Se la presenza di errori provoca la creazione del file *acad.dce*, viene visualizzata una casella di avviso. Altrimenti, la funzione **load_dialog** restituisce un numero intero positivo che identifica il file DCL. Questo valore viene passato alla funzione **new_dialog** per inizializzare le singole finestre di dialogo contenute nel file.

Alcuni errori segnalati da una casella di avviso in seguito alla chiamata della funzione **new_dialog** non vengono riportati in *acad.dce*. La funzione **new_dialog** restituisce T se ha avuto esito positivo, altrimenti restituisce nil. Se il valore di ritorno è T, utilizzare la funzione **start_dialog** per visualizzare il dialogo.

Dopo aver eseguito l'individuazione degli errori su ogni file DCL, è possibile caricare il programma e verificare le finestre di dialogo in combinazione.

Capitolo 17 -- Finestre di dialogo programmabili

Struttura dei file DCL

Gestione degli errori DCL

Verifica semantica per i file DCL

AutoCAD fornisce la scelta di quattro livelli (0-3) di verifica semantica per i file DCL (consultare la tabella riportata di seguito). In questo modo si tenta di individuare il codice contenuto nel file DCL che potrebbe generare problemi oppure che sembra non essere necessario. Queste verifiche vengono eseguite al momento del caricamento DCL. Per impostare il livello di verifica, includere una riga simile alla seguente all'interno del file DCL, in qualsiasi punto ma non nelle definizioni delle caselle:

```
dcl_settings : default_dcl_settings { audit_level = 3; }
```

Se il file DCL fa riferimento ad altri file DCL con l'istruzione di inclusione @include, è necessario utilizzare un solo dcl_settings in un unico file. Il livello di verifica definito viene utilizzato in tutti i file inclusi.

Livelli di verifica semantica

Livello	Descrizione
0	Nessun controllo. Usare solo se i file DCL sono stati verificati e in seguito non più modificati.
1	Errori. Individua gli errori DCL che possono causare l'interruzione di AutoCAD. Questo livello di controllo è quello di default e non provoca quasi nessun ritardo. Gli errori possono comprendere l'uso di caselle non definite e definizioni circolari di prototipi.
2	Avvertimenti. Individua gli errori DCL che si trovano in dialoghi con un layout o un comportamento non desiderato. Un file DCL modificato dovrebbe essere verificato a questo livello almeno una volta. Il livello di attenzione rileva errori quali la mancanza degli attributi richiesti ed i valori di attributi di tipi non appropriati.
3	Suggerimenti. Individua le definizioni di attributi ridondanti.

Per sfruttare al massimo le capacità della funzione di verifica, gli sviluppatori DCL dovrebbero impostare audit_level su 3 durante lo sviluppo e cancellare la riga dcl_settings prima di inviare i file agli utenti.

Capitolo 17 -- Finestre di dialogo programmabili

Struttura dei file DCL

Esempio di finestra di dialogo

Per visualizzare una finestra di dialogo e controllare le caselle al suo interno, è necessario usare AutoLISP o ADSRX. La seguente applicazione AutoLISP, sviluppata e spiegata nel capitolo 10, "Gestione delle finestre di dialogo", può essere usata per richiamare le finestre di dialogo contenute in questa sezione. È possibile digitare il seguente codice AutoLISP alla riga di comando oppure salvarlo in un file .lsp e caricarlo con la funzione **load**.

```
(defun c:ciao ( / dcl_id )
  (setq dcl_id (load_dialog "ciao.dcl"))
  (if (not (new_dialog "ciao" dcl_id) ) (exit))
  (action_tile "accept" "(done_dialog)")
  (start_dialog)
  (unload_dialog dcl_id)
  (princ)
)
```

Creare un file di testo ASCII chiamato *ciao.dcl* che contenga la seguente descrizione di finestra di dialogo:

```
ciao : dialog {
  label = "Esempio di finestra di dialogo";
  : text {
    label = "Ciao a tutti";
  }
  : button {
    key = "accept";
    label = "OK";
    is_default = true;
  }
}
```

Supponendo che la funzione **C:CAIO** sia stata definita, questo frammento di DCL produce la seguente finestra di dialogo dopo aver digitato **ciao** alla riga di comando.



Finestra di dialogo di esempio

La finestra di dialogo ha come titolo "Esempio di finestra di dialogo" e contiene solo il messaggio Ciao a tutti ed il pulsante OK. La dimensione del riquadro e la disposizione del contenuto vengono determinati in base ai valori di default. L'unico problema di questa finestra di dialogo è il pulsante OK la cui visualizzazione occupa tutta la sua lunghezza.

Per migliorare l'aspetto di questa finestra di dialogo, è possibile modificare il file DCL ed aggiungere nuovi attributi alla casella del pulsante (button). Per evitare che il pulsante occupi tutto lo spazio disponibile, aggiungere l'attributo `fixed_width` ed impostarlo su `true`. In tal modo il bordo del pulsante viene ristretto e la nuova larghezza sarà appena più ampia del testo al suo interno. Per centrare il pulsante, aggiungere l'attributo `alignment` con la specifica `centered`. Le caselle di una colonna vengono allineate a sinistra per default. La descrizione DCL è ora la seguente:

```
ciao : dialog {
  label = "Esempio di finestra di dialogo";
  : text {
    label = "Ciao a tutti";
  }
  : button {
    key = "accept";
    label = "OK";
    is_default = true;
    fixed_width = true;
    alignment = centered;
  }
}
```

La finestra di dialogo ora è la seguente:



Finestra di dialogo di esempio con il layout modificato

Sebbene i pulsanti rappresentino un buon metodo per la dimostrazione degli attributi, nelle finestre di dialogo devono essere usati i sottoinsieme standard dei pulsanti di uscita. Vedere "Pulsanti di uscita dalla finestra di dialogo e caselle di errore." Per creare una finestra di dialogo che, sostanzialmente, è analoga a quella mostrata nella figura precedente, usare il sottoinsieme `ok_only` come mostrato nell'esempio riportato di seguito:

```
ciao : dialog {
  label = "Esempio di finestra di dialogo";
  : text {
    label = "Ciao a tutti";
  }
  ok_only;
}
```

Capitolo 17 -- Finestre di dialogo programmabili

Tecniche DCL

Molti dei problemi più comuni possono essere risolti utilizzando le tecniche descritte nelle sottosezioni successive.

Verificare se il layout di default è adatto per la finestra di dialogo che si sta creando. Se viene rilevato un problema, regolare il layout

a livello del cluster, modificando i valori di default. Modificare le singole caselle solo in base alle necessità.

Capitolo 17 -- Finestre di dialogo programmabili

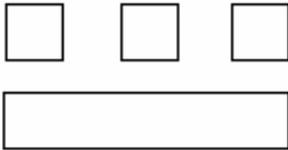
Tecniche DCL

Distribuzione delle caselle in un cluster

Quando le caselle vengono posizionate in una finestra di dialogo, è necessario disporle in righe e colonne in base alla relativa dimensione di ogni casella. Nel codice DCL riportato di seguito viene definita una riga con tre caselle che viene eseguita insieme alla parte superiore di un'altra casella.

```
: column {
  : row {
    : compact_tile {
    }
    : compact_tile {
    }
    : compact_tile {
    }
  }
  : large_tile {
  }
}
```

Se i componenti `compact_tile` hanno l'attributo `fixed_width` e `large_tile` è più grande dello spazio minimo necessario per la riga di `compact_tiles`, per default questo sottoinsieme risulta simile al seguente:



Allineamento orizzontale di default

Il bordo iniziale del primo `compact_tile` incluso nella riga viene allineato con il bordo iniziale di `large_tile` ed il bordo finale dell'ultimo `compact_tile` viene allineato con il bordo finale di `large_tile`. Le caselle intermedie vengono equamente distribuite. La situazione è analoga nel caso di colonne contigue.

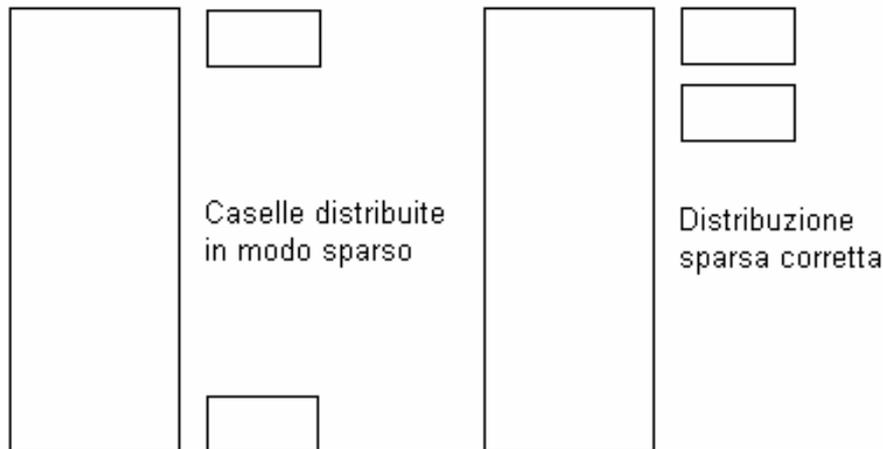
È possibile controllare la distribuzione di default utilizzando le caselle `spacer_0` e `spacer_1`, che sono delle varianti alla casella `spacer` definita in `base.dcl`.

Capitolo 17 -- Finestre di dialogo programmabili

Tecniche DCL

Spazio tra le caselle

Se due colonne contigue occupano quantità di spazio notevolmente diverse, le caselle contenute nella colonna che necessita di una quantità di spazio minore non saranno posizionate una accanto all'altra. Questa situazione può essere migliorata, impostando su `true` l'attributo `fixed_height` relativo alla colonna visualizzata in modo incongruente. Il risultato è illustrato nel seguente diagramma.



Distribuzione verticale delle caselle

Capitolo 17 -- Finestre di dialogo programmabili

Tecniche DCL

Spazio superfluo a destra o in basso

Una finestra di dialogo può presentare parte di spazio inutilizzato lungo il lato destro. Ciò può essere a volte causato da una casella text la cui larghezza specificata in modo esplicito è maggiore di quella richiesta dal suo valore corrente. Spesso la quantità di spazio in esubero è accettabile, ma se rappresenta un problema è possibile forzare a casella di testo ad avere un bordo utilizzando `boxed_row`.

Ad esempio, viene di seguito riportato un frammento della finestra di dialogo Creazione oggetti di AutoCAD:

```
: text {
  key = "l_text";
  width = 18;
  fixed_width = true;
}
```

La casella text non visualizzerà nulla, dal momento che il suo value è nullo, fino a quando l'applicazione non vi immetterà qualcosa con una istruzione di questo tipo:

```
(set_tile "l_text" "Da layer")
```

Poiché "Da layer" non è lungo 18 caratteri, la finestra di dialogo presenta spazio superfluo lungo il lato destro.

Una situazione simile si verifica quando viene utilizzato `errtile` (vedere "Pulsanti di uscita dalla finestra di dialogo e caselle di errore") per visualizzare i messaggi di errore. Se non è attualmente visualizzato un messaggio di errore, sembrerà esserci spazio in esubero nella parte inferiore della finestra di dialogo. In questo caso, una ulteriore casella spacer all'inizio della finestra di dialogo può facilitare il bilanciamento del layout verticale.

Capitolo 17 -- Finestre di dialogo programmabili

Tecniche DCL

Spazio superfluo intorno ad una riga o colonna incasellata

Se l'attributo etichetta ((label) di una riga o colonna incasellata è vuoto (" ") o nullo (""), il riquadro racchiude il cluster ma non verrà visualizzato alcun testo e quindi un singolo spazio vuoto non verrà visualizzato nella casella come uno spazio. Tuttavia, esiste una differenza nel modo in cui le etichette vuote e nulla vengono visualizzate:

- Se label è un singolo spazio vuoto, lo spazio verticale che il testo occupava *dentro* la casella viene annullato, mentre lo spazio

verticale che l'etichetta occupava *sopra* la casella sarà conservato.

- Se l'etichetta è una stringa nulla, tutto lo spazio verticale sarà annullato, sia che si tratti di spazio sopra che dentro la casella.

Nel codice DCL riportato di seguito, le righe superiori delle caselle attorno alle prime due colonne verranno sicuramente allineate (con la stessa coordinata Y), mentre la riga superiore della casella attorno alla terza colonna sicuramente non presenterà alcuno spazio sopra o sotto di essa, tranne che per i normali margini di default.

```
: row {
  : boxed_column {
    label = "Testo";
  }
  : boxed_column {
    label = " ";           // singolo spazio vuoto: il valore di default
  }
  : boxed_column {
    label = "";           // stringa nulla
  }
}
```

Capitolo 17 -- Finestre di dialogo programmabili

Tecniche DCL

Personalizzazione del testo dei pulsanti di uscita

Può verificarsi il caso in cui si desideri modificare il testo di uno dei pulsanti di uscita di alcune finestre di dialogo. Ad esempio, se viene creata una finestra di dialogo in grado di distruggere dati, il testo più appropriato per il pulsante sarà Distruggi e non OK . Per eseguire questa modifica, usare il prototipo `retirement_button` come indicato di seguito:

```
destroy_button : retirement_button {
  label= "&Distruggi";
  key= "destroy";
}
```

Fare attenzione all'uso di & (ampersand) nell'attributo etichetta (label): tale carattere assegna un mnemonico alla casella. In questo caso la lettera D è sottolineata nel testo del pulsante e diventa il mnemonico.

Nota Durante la personalizzazione dei sottoinsiemi dei pulsanti di uscita, potrebbe essere utilizzato il codice DCL corretto del file `base.dcl` invece di inserirlo manualmente.

Dopo aver definito un pulsante di uscita personalizzato, è necessario incorporarlo in un sottoinsieme che corrisponda per aspetto grafico e funzionalità ai cluster standard. Nell'esempio riportato di seguito viene mostrata la definizione corrente di `ok_cancel_help`.

```
ok_cancel_help : column {
  : row {
    fixed_width = true;
    alignment = centered;
    ok_button;
    : spacer { width = 2; }
    cancel_button;
    : spacer { width = 2; }
    help_button;
  }
}
```

Creare un nuovo sottoinsieme che sostituisca `ok_button` con il nuovo pulsante, come mostrato di seguito:

```
destroy_cancel_help : column {
  : row {
    fixed_width = true;
    alignment = centered;
    destroy_button;
    : spacer { width = 2; }
    cancel_button;
    : spacer { width = 2; }
  }
}
```

```

        help_button;
    }
}

```

Nel sottinsieme, il pulsante OK è quello di default, ma questo attributo non è stato aggiunto a `destroy_button`. Dove l'azione della finestra di dialogo può essere distruttiva (o molto lenta nell'elaborazione) si raccomanda di attribuire al pulsante Annulla il valore di default. In tal modo, esso avrà la funzionalità sia del valore di default che del pulsante di annullamento:

```

destroy_cancel_help : column {
  : row {
    fixed_width = true;
    alignment = centered;
    destroy_button;
    : spacer { width = 2; }
    : cancel_button { is_default = true; }
    : spacer { width = 2; }
    help_button;
  }
}

```

Poiché è stato modificato un attributo, il comando Annulla originario viene usato come prototipo. I due punti ora si trovano davanti a `cancel_button`.

Attenzione Quando il pulsante di annullamento è anche quello di default (sia `is_default` che `is_cancel` sono true) e si omette di assegnare una azione che richiama `done_dialog` ad un altro pulsante, nessun altro pulsante consentirà di uscire dalla finestra di dialogo e verrà *sempre* annullato.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Per progettare una finestra di dialogo funzionale, è necessario considerare non solo il suo fine pratico, ma anche il suo aspetto estetico ed ergonomico, nonché gli standard GUI per l'ambiente Windows. Nelle sottosezioni successive vengono fornite alcune direttive relative alla progettazione GUI, alla creazione delle finestre di dialogo, alla portabilità, ai cluster ed alle caselle predefinite.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Estetica ed ergonomia

L'aspetto di una finestra di dialogo è molto importante. Se la casella è visivamente troppo disordinata, diventa difficile utilizzarla e non ha più alcuna efficacia. Inoltre, le caselle devono essere posizionate in maniera ergonomica in modo da facilitarne l'utilizzo. Considerare quali sono le caselle usate più frequentemente, metterle in evidenza nella progettazione e disporle in modo tale da facilitare il passaggio da una all'altra, soprattutto quando vengono utilizzate in associazione.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Coerenza di progettazione e chiarezza di linguaggio

L'interfaccia utente di un'applicazione deve essere internamente coerente ed allineata con le applicazioni correlate. Una finestra di

dialogo non familiare viene appresa più facilmente se il suo disegno è coerente con quello delle altre finestre di dialogo dell'applicazione, delle applicazioni correlate o del sistema host. Un esempio di ciò è la coerenza nel posizionamento di pulsanti come OK ed Annulla. È necessario che presenti un certo allineamento anche la tecnica associata ad ogni tipo di casella, vale a dire il modo in cui il testo viene digitato nella casella di testo e quello in cui viene selezionata una voce da una casella di riepilogo. Il modo migliore per ottenere tale allineamento è quello di riutilizzare le caselle ed il codice che le controlla. La standardizzazione contribuisce ad ottenere un prodotto coerente ed allineato.

L'uso di definizioni standard per il controllo delle finestre di dialogo riduce la quantità di lavoro del programmatore, contribuisce a rendere l'applicazione coerente e rende più semplice per l'utente l'apprendimento e l'uso delle finestre di dialogo.

Utilizzare un linguaggio chiaro. Anche se le finestre di dialogo sono considerate parte di un'interfaccia grafica, la maggior parte delle caselle e le informazioni in esse contenute sono composte da testo. Le etichette delle finestre di dialogo, i nomi dei pulsanti e la fraseologia dei messaggi devono essere diretti ed espressi chiaramente. Evitare, quindi, di usare termini gergali o tecnici che gli utenti potrebbero non comprendere.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Controllo dell'utente

Consentire agli utenti di controllare almeno in parte il modo con cui utilizzare le finestre di dialogo per immettere l'input. Uno dei vantaggi rappresentato dall'uso delle finestre di dialogo piuttosto che dell'interfaccia con una riga di comando è che le prime non limitano gli utenti ad una ristretta sequenza di messaggi di richiesta. In una finestra di dialogo gli utenti devono essere in grado di immettere l'input in qualsiasi sequenza. Alcune restrizioni sono sicuramente necessarie, ad esempio quando la selezione di un'opzione provoca la disabilitazione di un'altra, ma vanno incluse soltanto se sono veramente necessarie ai fini del funzionamento dell'applicazione.

Ad esempio, nella figura riportata di seguito viene mostrata la finestra di dialogo *Inserisci* nella quale gli utenti possono prima digitare il punto di base e poi selezionare gli oggetti, digitare il nome di un blocco, cambiare idea e selezionare oggetti differenti ed infine premere OK. Oppure essi possono selezionare gli oggetti prima di definire il punto di base. La restrizione principale è di tipo logico e prevede che gli utenti non possano digitare il nome di un blocco se l'interruttore *Senza nome* è selezionato.

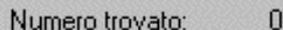


Finestra di dialogo Inserisci

Nel caso di più finestre di dialogo, è necessario visualizzarle una sopra l'altra, in modo da non dover uscire dalla casella corrente prima di poterne aprire un'altra. Inoltre, l'utente dovrebbe essere sempre in grado di tornare alla prima finestra di dialogo visualizzata. Per lo stesso motivo, non si devono obbligare gli utenti ad effettuare una scelta prima che siano pronti ad uscire dalla finestra di dialogo. Poiché la finestra di dialogo corrente viene visualizzata sopra quella precedente, ciò consente agli utenti di ricordare il contesto: da dove hanno iniziato e dove torneranno.

Ogni volta che gli utenti eseguono azioni per modificare lo stato o le opzioni correnti, è necessario provvedere alla visualizzazione immediata di un feedback. Se gli utenti selezionano qualcosa, questo va visualizzato o descritto immediatamente. Se una scelta ne esclude altre, assicurarsi che vengano disabilitate immediatamente le scelte non valide.

Nella finestra di dialogo Selezione colore di AutoCAD, ad esempio, un gruppo di immagini visualizza il colore immediatamente dopo la selezione del relativo numero da parte dell'utente. Nella finestra di dialogo Definizione blocchi il numero di oggetti selezionato viene sempre visualizzato in un messaggio sotto il pulsante Seleziona oggetti.



Numero trovato: 0

Casella di testo usata per visualizzare un messaggio di stato

Capitolo 17 -- Finestre di dialogo programmabili

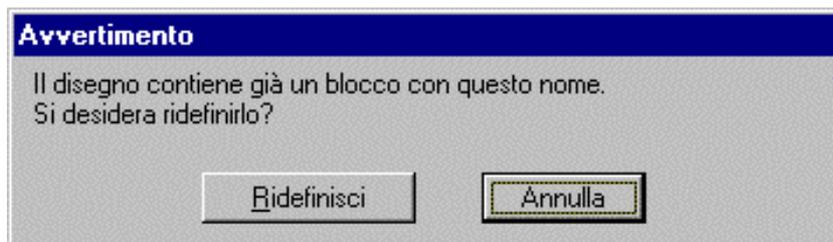
Direttive di progettazione

Annullamento degli errori

È necessario provvedere all'annullamento di eventuali errori nelle finestre di dialogo, in modo tale che gli utenti si sentano liberi di esaminare il prodotto senza il timore di provocare errori riversibili. Segnalare gli errori minori con messaggi, introducendo una casella di errore nella parte inferiore della finestra di dialogo; per gli errori più gravi, invece, visualizzare una casella di avviso. A tal fine, utilizzare la funzione **alert** che consente di visualizzare una casella di avviso semplice, solo con il pulsante OK. Vedere "**alert**" nel capitolo 13.

Se l'utente seleziona un'azione di carattere distruttivo o che richiede molto tempo, è necessario che nella finestra di dialogo venga visualizzata una casella di avviso tramite la quale l'utente può scegliere se proseguire con l'opzione selezionata oppure annullarla.

Ad esempio, nella finestra di dialogo Inserisci viene visualizzata una casella di avviso quando gli utenti tentano di creare un blocco già esistente. Gli utenti possono scegliere di proseguire e ricoprire il blocco originario oppure di annullare l'operazione senza effettuare alcuna modifica.



Casella di avviso con le opzioni per proseguire o annullare

Le finestre di dialogo nidificate che visualizzano messaggi di avvertimento, devono tornare alla finestra di dialogo precedente. Provocare la chiusura di tutte le finestre di dialogo attualmente nidificate solo in caso di errori gravi o che potrebbero essere irreversibili.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Funzione di Guida in linea

La funzione che sicuramente deve essere fornita è la guida in linea. La quantità di informazioni alle quali è possibile accedere con questa funzione dipende dalla complessità dell'applicazione e dalla capacità auto-esplicativa di ogni finestra di dialogo. Si consiglia, comunque, che la finestra di dialogo principale includa sempre almeno un pulsante Guida che visualizzi un'unica finestra di dialogo contenente informazioni importanti. Nella maggior parte dei casi, il pulsante Guida dovrebbe visualizzare la finestra di dialogo standard Guida di AutoCAD, tramite la chiamata alla funzione AutoLISP **help**. Vedere "Personalizzazione della documentazione in linea."

Se la propria applicazione è molto sofisticata, sarà necessario sviluppare una funzione di Guida sensibile al contesto con più finestre di dialogo, ognuna delle quali sarà associata ad una particolare finestra di dialogo.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Utenti disabili

Le considerazioni relative ad utenti disabili possono rendere più semplice il programma per chiunque intenda utilizzarlo. Durante la progettazione delle finestre di dialogo, quindi, considerare gli argomenti descritti di seguito.

Colore

Molte persone non sono in grado di distinguere alcuni colori. Se viene utilizzata una codifica di colori per presentare le informazioni, è necessario integrarla presentando le stesse informazioni in altri modi, di solito con del testo.

Ad esempio, la finestra di dialogo dei colori standard di AutoCAD visualizza un messaggio di testo che riporta il numero o i nomi dei colori insieme ad un gruppo di immagini che visualizza il colore.

Guida in linea

Molti utenti presentano alcune difficoltà nella lettura di manuali stampati con caratteri molto piccoli oppure nella gestione pratica delle pubblicazioni. Anche un unico pulsante Guida nella finestra di dialogo principale può, quindi, essere utile.

Uso della tastiera

Per alcuni utenti potrebbe essere difficile usare un dispositivo di puntamento, mentre altri potrebbero non essere in grado di farlo. Per questa ragione è necessario specificare le lettere mnemoniche in modo che le finestre di dialogo possano essere utilizzate anche con la tastiera come unico dispositivo di input.

Chiarezza e semplicità

Finestre di dialogo progettate con particolare attenzione e soprattutto con un linguaggio chiaro e semplice facilitano gli utenti ostacolati da menomazioni verbali o conoscitive. Non forzare gli utenti a ricordare molte cose diverse. Usare, invece, terminologia coerente e, quando possibile, presentare varie possibilità di scelta.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Uso delle lettere maiuscole

Di seguito vengono descritte alcune direttive di carattere generale per l'inserimento di testo con lettere maiuscole nelle finestre di dialogo.

Intestazioni delle finestre di dialogo, aree e colonne

Usare nei titoli i caratteri maiuscoli per la prima lettera di tutte le parole tranne che per gli articoli, le preposizioni e le congiunzioni. Tuttavia, se una finestra di dialogo viene richiamata da un menu e non dalla riga di comando, il suo titolo deve corrispondere alla voce del menu.

Etichette di controllo

Usare i caratteri maiuscoli nelle etichette delle caselle di controllo come, ad esempio, i pulsanti. *Non* concludere le etichette con un punto; se si tratta di etichette di una casella di testo o di un elenco a comparsa, farle seguire da due punti (:). Se l'etichetta è lunga oppure è espressa come una domanda, è possibile usare i caratteri maiuscoli come vengono di solito usati nella scrittura normale, cioè con la prima lettera e con le parole normalmente scritte in maiuscolo.

Messaggi di richiesta e messaggi

Usare i caratteri maiuscoli come vengono di solito usati nella scrittura normale.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Abbreviazioni

Evitare di usare le abbreviazioni, poiché possono essere ambigue e difficili da interpretare; ma se la quantità di spazio disponibile costringe a farlo, usare le abbreviazioni in modo coerente all'interno di un gruppo come, ad esempio, le colonne all'interno di una casella. *Non* scrivere alcuni termini per intero quando altri sono stati abbreviati, ma sforzarsi di essere coerenti.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Layout

Disporre le sezioni della finestra di dialogo in modo logico all'interno di righe o colonne, cosicché gli utenti possano esaminarle da sinistra verso destra oppure dall'alto verso il basso. Allineare i relativi campi di immissione, come le caselle di modifica o di riepilogo, sia in senso verticale che orizzontale, in modo che gli utenti possano passare da un campo all'altro con il tasto TAB ed il cursore si sposti lungo una linea ortogonale dritta.

Se esiste un ordine naturale per la digitazione dei dati, come per quelli delle coordinate X, Y e Z, ordinare i campi nello stesso modo. Allineare le aree provviste di caselle sia in senso verticale che in senso orizzontale. Non lasciare spazio non necessario tra questo tipo di aree o intorno ad esse, ma estenderne la larghezza verso destra, se necessario.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Dimensione e posizionamento

Per visualizzare le informazioni in modo chiaro, la finestra di dialogo non deve essere più grande del necessario.

Nota La risoluzione di schermo più bassa utilizzabile è 640x480. Nel caso in cui si sviluppino applicazioni su monitor con risoluzione maggiore, è necessario verificare che la visualizzazione delle finestre di dialogo sia corretta anche ad una risoluzione inferiore.

Per default, AutoCAD visualizza inizialmente tutte le finestre di dialogo al centro dello schermo del disegno. Alcune piattaforme consentono di visualizzare le finestre di dialogo nell'ultima posizione specificata dall'utente. Le funzioni che provvedono al posizionamento delle finestre di dialogo sono **new_dialog** e **done_dialog** in AutoLISP e **ads_new_positioned_dialog()** e **ads_done_positioned_dialog** in ADSRX.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Caselle disabilitate

Se, in base alle impostazioni delle opzioni correnti, una casella o un'area non è valida o rilevante, disabilarla immediatamente in modo che venga visualizzata in grigio e non possa essere selezionata dall'utente. Non eccedere però nell'utilizzo della funzione di disabilitazione delle caselle, poiché troppe caselle visualizzate in grigio potrebbero distrarre l'utente.

Se viene disabilitata una casella che visualizza un valore, quest'ultimo non subirà alcuna modifica, quindi la casella dovrà visualizzare lo stesso valore quando sarà abilitata di nuovo. I valori che vengono modificati in modo inspiegabile provocano un maggiore carico di lavoro per gli utenti.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Finestre di dialogo nidificate

AutoCAD limita ad otto il numero delle finestre di dialogo nidificate. Per informazioni sulla loro gestione, vedere "Finestre di dialogo nidificate." Inserire le ellissi (...) nell'etichetta di un pulsante che visualizza una finestra di dialogo nidificata, a meno che non si tratti di una casella di avviso. *Non* creare nidificazioni superiori a tre livelli ; la quarta finestra è significativa solo se si tratta di unacasella di avviso. Poiché le finestre di dialogo vengono inizialmente visualizzate per default al centro dello schermo, quelle nidificate devono essere più piccole della finestra di dialogo principale.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Come nascondere le finestre di dialogo

Se un utente deve effettuare una selezione dallo schermo di disegno prima che venga chiusa la finestra di dialogo, è necessario chiudere momentaneamente la finestra di dialogo in modo che l'utente possa vedere lo schermo ed effettuare la selezione. In seguito, la finestra di dialogo viene visualizzata di nuovo. Tale azione è quella di *nascondere* una finestra di dialogo.

L'etichetta di un pulsante che provoca la scomparsa della finestra di dialogo *non* deve contenere i puntini di sospensione, ma uno spazio seguito dal simbolo minore di (<). Quando la finestra di dialogo viene nascosta, deve essere visualizzato un messaggio di richiesta che descrive le azioni che l'utente deve intraprendere.

Nella maggior parte dei casi, è possibile ottenere l'input con una delle funzioni **getxxx** che dispongono di un argomento con il quale è possibile specificare un messaggio di richiesta.

Quando la finestra di dialogo viene visualizzata di nuovo, essa deve contenere il feedback relativo all'elaborazione della selezione come, ad esempio, alcune nuove informazioni nei campi delle caselle di modifica, una casella di riepilogo aggiornata, un messaggio di testo che indica lo stato oppure una loro combinazione.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Valori di default

Fornire valori di default significativi per tutte le immissioni e le opzioni. Valori di default scelti in modo appropriato possono aiutare gli utenti a completare velocemente e facilmente una finestra di dialogo.

Si consiglia di provvedere sempre all'aggiornamento dei valori di default, in altre parole di salvare le impostazioni precedenti dell'utente e di utilizzarle come nuovi valori di default, ogni volta che la finestra di dialogo viene utilizzata. Anche se l'utente dovrà comunque modificarne alcuni, è sicuramente più comodo che ripartire ogni volta dall'inizio.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Input della tastiera

Durante la creazione di una finestra di dialogo personalizzata, è possibile specificare il modo in cui saranno gestiti i tasti di scelta rapida. Alcuni tasti di scelta rapida sono comuni a tutte le finestre di dialogo. Ad esempio, il tasto TAB generalmente consente all'utente di spostarsi da una casella all'altra ed il tasto SPACEBAR di attivare o disattivare gli interruttori. Come valore di default, ad

ogni casella attiva deve corrispondere una tabulazione.

Esistono due tasti conosciuti come tasti *di scelta*: il tasto *accept* (di solito il tasto RETURN) che accetta la finestra di dialogo ed i valori digitati, ed il tasto *Annulla* (ESC), che annulla la finestra di dialogo ed i suoi valori. Quando AutoCAD visualizza una finestra di dialogo per la prima volta, una delle sue caselle appare evidenziata. La digitazione dell'utente riguarderà questa casella fino a quando l'utente non sposterà l'evidenziazione su un'altra.

Per passare da una casella ad un'altra, l'utente può premere il tasto TAB, scegliere un'altra casella o digitare uno dei tasti di scelta rapida conosciuti come tasti *mnemonici*. Ad esempio, nella finestra di dialogo Aiuti per il disegno è possibile selezionare uno dei pulsanti di scelta Assonometria snap/griglia premendo L, T o R. Passando da una casella ad un'altra viene spostata l'evidenziazione, ma non viene eseguita alcuna selezione; infatti, per selezionare la casella evidenziata, è necessario premere il tasto accept. Per alcuni tipi di caselle, un doppio clic equivale alla pressione di questo tasto.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Direttive per caselle e cluster predefiniti

Nella traduzione in determinate lingue è possibile che certe parole diventino più lunghe. Se le finestre di dialogo sono destinate alla traduzione, è consigliabile lasciare lo spazio massimo possibile. Nella tabella che segue sono riportati alcuni termini comuni di AutoCAD in inglese, francese e tedesco.

Termini AutoCAD in inglese, francese e tedesco

Inglese	Francese	Tedesco
Line	<i>Ligne</i>	<i>Linie</i>
Arc	<i>Arc</i>	<i>Bogen</i>
Circle	<i>Cercle</i>	<i>Kreis</i>
3D Polylines	<i>Polylignes 3D</i>	<i>3D-Polylinien</i>
Diameter dimensioning	<i>Cotation de diamètre</i>	<i>Durchmesserbemaßung</i>
Layers	<i>Calque</i>	<i>Layer</i>
Linetypes	<i>Types de ligne</i>	<i>Linientypen</i>
Entity creation modes	<i>Modes de création des objets</i>	<i>Modus für Objekterzeugung</i>
Select objects	<i>Choix des objets</i>	<i>Objekte wählen</i>
OK	<i>OK</i>	<i>OK</i>
Cancel	<i>Annuler</i>	<i>Abbruch</i>
Help	<i>Aide</i>	<i>Hilfe</i>

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Direttive per caselle e cluster predefiniti

In questa sezione vengono elencate le regole convenzionali consigliate e le direttive di progettazione associate a particolari tipi di caselle e cluster di caselle predefiniti.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Direttive per caselle e cluster predefiniti

Pulsanti

L'azione associata ad un pulsante deve essere visibile per l'utente e deve avere luogo immediatamente. L'etichetta di un pulsante non deve essere ambigua; di solito è un verbo che descrive l'effetto della pressione del pulsante, anche se può essere ritenuta valida

un'altra etichetta, come OK o Opzioni, se il significato è evidente. Per pulsanti che richiamano altre finestre di dialogo o che nascondono quella corrente, vedere le sezioni "Finestre di dialogo nidificate" e "Come nascondere le finestre di dialogo."

I pulsanti all'interno di una colonna devono avere tutti la stessa larghezza. In altri casi, i pulsanti devono avere una larghezza fissa (con `fixed_width = true;` o `children_fixed_width = true;`) nel cluster di origine comune.

Capitolo 17 -- Finestre di dialogo programmabili

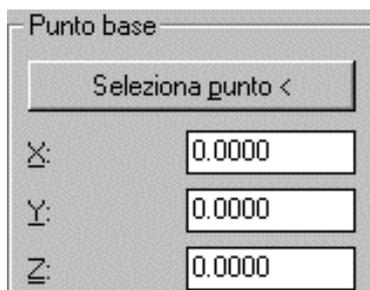
Direttive di progettazione

Direttive per caselle e cluster predefiniti

Cluster

Un cluster provvisto di casella (una riga o una colonna) viene detto *casella di gruppo* o *area*. Anche lo spazio bianco è un metodo valido per raggruppare le caselle. L'area può contenere tante caselle, righe e colonne (sprovviste di casella) quante ne sono necessarie. L'etichetta di un'area deve indicare il suo scopo.

Se i controlli sono tra loro relativi, inserirli in un'area. Il cluster Punto di base nella finestra di dialogo di esempio Inserisci mostra questa tecnica.



Area formata da un cluster con un'etichetta ed un bordo

Fare attenzione, però, a non utilizzare troppo spesso le aree. Anche lo spazio bianco è un metodo valido per raggruppare le caselle. Non aggiungere quindi una casella attorno ad una casella di ripiegolo: ne risentirebbe l'estetica dell'intera finestra di dialogo.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Direttive per caselle e cluster predefiniti

Caselle di modifica

In una casella di modifica, la lunghezza della parte in cui viene digitato il testo deve approssimativamente eguagliare una immissione di media lunghezza. In caso di dubbio, usare la larghezza di carattere 10 per i campi di numeri reali e 20 per i campi di testo.

L'etichetta di una casella di modifica deve terminare con due punti (:).

Se esistono limitazioni relative al testo che l'utente può digitare, è necessario inserire una casella di testo a destra della casella di modifica per descrivere brevemente queste restrizioni. Se, ad esempio, l'utente deve digitare un nome di file non è necessario spiegare cos'è, ma se la stringa è un numero che non deve essere superiore a 100, è opportuno ricordare questo limite.

Per dati, come i punti, fornire due o tre caselle di modifica invece di richiedere all'utente di ricordare la sintassi della riga di comando relativa all'immissione del punto. Una eccezione a questo è una casella di modifica creata per la digitazione di sintassi avanzata, come la casella di modifica con il carattere jolly nella finestra di dialogo di AutoCAD relativo alla ricerca di file.

Nome file:	<input type="text"/>
Tipo file:	Disegno (*.dwg) 

Casella di modifica per sintassi particolare

Capitolo 17 -- Finestre di dialogo programmabili

-  **Direttive di progettazione**
-  **Direttive per caselle e cluster predefiniti**
-  **Pulsanti immagine e gruppi di immagini**

Se un pulsante immagine o un gruppo di immagini viene usato come icona per avvisare l'utente, ad esempio, un segnale di avvertimento quale un segnale di arresto, usarlo in modo coerente in tutte le finestre di dialogo.

Nel caso vengano utilizzati dei pulsanti immagine per rappresentare le selezioni, corredare l'immagine con testo esplicativo, soprattutto se il colore dell'immagine, o di parte di essa, è fattore determinante ai fini della selezione.

Capitolo 17 -- Finestre di dialogo programmabili

-  **Direttive di progettazione**
-  **Direttive per caselle e cluster predefiniti**
-  **Caselle di riepilogo**

Poiché lo scorrimento degli elenchi DCL non può essere eseguito in senso orizzontale, la larghezza della casella di riepilogo dipende dalla voce più lunga compresa nell'elenco. È necessario inserire una etichetta (o una casella di testo) per spiegare il contenuto della casella di riepilogo, a meno che non si tratti della casella principale all'interno della finestra di dialogo; in questo caso, infatti, sarà sufficiente l'etichetta della finestra di dialogo, sebbene sia necessario etichettare la casella di riepilogo se si desidera che l'utente sia in grado di spostarsi in essa utilizzando un tasto mnemonico.

Elencare le voci in ordine alfabetico, a meno che non esista un'altra ragione di ordine logico per disporle in modo diverso. Se la lunghezza dell'elenco è fissa e insufficiente, è opportuno utilizzare una colonna di scelta invece di una casella di riepilogo.

Capitolo 17 -- Finestre di dialogo programmabili

-  **Direttive di progettazione**
-  **Direttive per caselle e cluster predefiniti**
-  **Pulsanti, righe e colonne di scelta**

Le colonne di scelta hanno un aspetto estetico migliore e sono più semplici da usare rispetto alle righe di scelta, le quali sono appropriate solo quando contengono un numero limitato di pulsanti (di solito da due a quattro) oppure se le etichette sono brevi.

Se un'opzione selezionata in qualche altra finestra di dialogo rende non valide o irrilevanti le selezioni contenute nella riga o nella colonna di scelta, è necessario disattivare l'intera riga o colonna. In certi casi, un'opzione selezionata in un'altra finestra di dialogo può rendere alcuni pulsanti di scelta non validi o irrilevanti; in tal caso, è possibile disattivare i pulsanti individualmente.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Direttive per caselle e cluster predefiniti

Dispositivi di scorrimento:

L'intervallo di spostamento di un dispositivo di scorrimento non deve essere molto ampio. Ad esempio, se al dispositivo di scorrimento sono stati assegnati solo quattro valori incrementali, ma sono stati disposti in una sezione di 5 cm della finestra di dialogo, l'utente dovrà spostarsi di 1,2 cm per vedere il cambiamento. Mettendo in scala la dimensione del dispositivo di scorrimento si facilita il lavoro all'utente.

Se è necessario che l'utente conosca il valore controllato dal dispositivo di scorrimento, è possibile visualizzarlo nella finestra di dialogo. È opportuno aggiornare questo valore ogni volta che il dispositivo di scorrimento viene spostato. Si raccomanda di visualizzare anche una casella di modifica che consenta agli utenti di digitare il valore invece di usare il dispositivo di scorrimento. Se la casella di modifica viene utilizzata in tal senso, sarà necessario aggiornarne il valore; in caso contrario, è sufficiente visualizzarlo in una casella di testo. Nella figura riportata di seguito viene mostrata una combinazione tipica di dispositivo di scorrimento e casella di modifica.



Dispositivo di scorrimento con casella di modifica

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Direttive per caselle e cluster predefiniti

Testo

Quando le etichette non sono sufficienti, utilizzare le caselle di testo per descrivere lo scopo delle singole caselle o delle diverse aree della finestra di dialogo. Le caselle di testo possono essere usate anche per visualizzare messaggi di stato o promemoria, compresi i messaggi di errore e gli avvertimenti.

Il testo deve essere diretto e chiaro. Descrivere, quindi, le opzioni ed i campi di immissione con i termini che avrebbe utilizzato l'utente. Ad esempio, il messaggio di errore Voce invalida nella casella di elenco comunica poche informazioni, mentre un messaggio come Layer inesistente è sicuramente più utile.

È necessario allineare i messaggi con le caselle di controllo che essi descrivono.

Inserire il testo che identifica un gruppo di caselle di controllo o una sezione della finestra di dialogo *sopra* le caselle descritte dal titolo.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Direttive per caselle e cluster predefiniti

Interruttori

Quando le opzioni controllate dagli interruttori sono relative allo stesso argomento, è necessario includerle in un gruppo.

Usare un singolo interruttore che controlli se altre caselle, spesso in una riga o in una colonna, sono o meno attive. In questo caso, la posizione dell'interruttore deve essere messa in evidenza. Quando l'interruttore controlla solo un'altra casella, è possibile posizionarlo anche a destra di quella casella. L'interruttore Senza nome nella finestra di dialogo Definizione blocchi rappresenta un esempio di tale situazione.

Scala generale:	<input type="text" value="1.00000"/>	<input type="checkbox"/> Scala allo spazio carta
Scala generale:	<input type="text" value="0.00000"/>	<input checked="" type="checkbox"/> Scala allo spazio carta

Interruttore che attiva o disattiva un'altra casella

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Gestione degli errori

Nelle finestre di dialogo possono essere visualizzati messaggi di errore ed avvertimenti tramite una casella di testo conosciuta come *casella di errore* (errtile) oppure con una casella di avvertimento nidificata. In entrambi i casi, considerare le seguenti direttive:

- I messaggi di errore devono essere delle frasi complete, provviste di punteggiatura, con la lettera iniziale maiuscola ed il punto finale.
- I messaggi di errore devono descrivere il problema o il potenziale problema e devono essere espressi con parole chiare e comprensibili.
- Dopo aver segnalato l'errore, portare l'evidenziazione della finestra di dialogo sulla casella che ha causato l'errore, se possibile.

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Gestione degli errori

Caselle di errore

Usare una casella di errore per gli errori secondari o gli avvertimenti, soprattutto per quelli causati da errori di digitazione o altri errori di immissione.

Non visualizzare gli errori in caselle di testo usate per i messaggi di stato, poiché quest'ultimi possono essere facilmente ignorati.

Se viene utilizzata una casella di errore, è necessario visualizzarla nella parte inferiore della finestra di dialogo. Utilizzare la casella errtile standard descritta in "Pulsanti di uscita dalla finestra di dialogo e caselle di errore."

Capitolo 17 -- Finestre di dialogo programmabili

Direttive di progettazione

Gestione degli errori

Caselle di allarme

Richiamando la funzione AutoLISP **alert** è possibile visualizzare una casella di avviso standard con un singolo pulsante OK. Usare le caselle di avviso per errori seri o potenzialmente fatali, ma non farne un uso eccessivo, poiché nelle caselle di avviso è necessario l'input dell'utente e ciò potrebbe essere noioso, soprattutto quando nelle caselle vengono segnalati errori minori o viene celata l'immissione che è necessario correggere.

Usare le caselle di avviso per avvertire l'utente che l'azione che si vuole intraprendere può distruggere i dati oppure essere molto lenta nell'elaborazione. Nelle caselle di avviso di questo tipo è necessario consentire all'utente di scegliere se proseguire o annullare l'azione. Se la casella dispone di tale selezione, come Procedi o Annulla, è necessario costruirla.

Se nella casella di avviso è prevista la possibilità di scelta, nel testo deve essere prima descritto il problema e poi formulata l'azione successiva sotto forma di domanda. In questi casi è importante che il pulsante che consente di proseguire sia etichettato con un verbo che descriva la conseguenza della sua selezione. In questo contesto, Sovrascrivi, ad esempio, è meno ambiguo di OK e rappresenta anche un aiuto per gli utenti più esperti che dissimuleranno il testo poiché essi hanno già visto molte volte questa casella di avviso.

A meno che l'errore non sia veramente fatale, è necessario a far tornare l'utente al passo precedente oppure uscire dall'operazione che ha causato la visualizzazione della casella di avviso.

Di solito il pulsante di default di una finestra di dialogo è OK o il suo equivalente, ma quando la situazione descritta dalla casella di avviso presenta conseguenze molto gravi, rendere come default il pulsante Annulla o il suo equivalente.

Capitolo 17 -- Finestre di dialogo programmabili

Documentazione correlata

Le direttive GUI e la filosofia di progettazione vengono trattate in maggior dettaglio nelle seguenti pubblicazioni:

- Apple Computer, Inc., *Apple Human Interface Guidelines: The Apple Desktop Interface*. Menlo Park: Addison-Wesley, 1987.
- International Business Machines Corp., *IBM System Application Architecture. Common User Access Guide to User Interface Design*. Boca Raton: IBM, 1991. (Numero di pubblicazione IBM SC34-4289-00).
- International Business Machines Corp., *IBM System Application Architecture. Common User Access Advanced Interface Design Reference*. Boca Raton: IBM, 1991. (Numero di pubblicazione IBM SC34-4290-00.)

Capitolo 18 -- Linguaggio DCL

Panoramica

Questo capitolo descrive tutte le caselle del linguaggio DCL (Dialog Control Language) ed i relativi attributi associati. Poiché la maggior parte degli attributi sono comuni a più caselle, verranno descritti prima delle caselle. Successivamente verranno descritte le caselle DCL, a seconda della loro categoria o funzione. Alla fine del capitolo si trova un elenco di tutte le caselle DCL predefinite.

Argomenti di questo capitolo

{button ,JI(','Attributes_al_u0402')} Attributi

{ewc ,JI(','Functional_Synopsis_of_DCL_Tiles_al_u0402')} Sinopsi delle funzioni delle caselle DCL

{ewc ,JI(','DCL_Tile_Catalog_al_u0402')} Elenco delle caselle DCL

Capitolo 18 -- Linguaggio DCL

Attributi

Gli attributi di una casella definiscono il suo layout e la sua funzione. Un attributo è come una variabile di un linguaggio di programmazione: è composto da un nome e da un valore.

Capitolo 18 -- Linguaggio DCL

Attributi

Tipi di attributi

Il valore deve essere di uno dei seguenti tipi:

- *Numero intero.* I valori numerici (sia numeri interi che reali) con i quali si esprime, ad esempio, la larghezza o l'altezza di una casella tra virgolette, vengono misurati nell'unità *larghezza-carattere* o *altezza-carattere*.
- *Numero reale.* Un numero reale frazionario deve avere una cifra iniziale: ad esempio, 0.1, *non* .1.
- *Stringa tra virgolette.* Una stringa tra virgolette consiste in un testo racchiuso tra virgolette (""). I valori dell'attributo sono sensibili al maiuscolo/minuscolo: N1 non è lo stesso di n1. Se la stringa deve contenere delle virgolette, inserire prima della virgoletta una barra rovesciata (\"). Una stringa tra virgolette può contenere anche altri caratteri di controllo. I caratteri riconosciuti da DCL sono elencati nella tabella seguente.

Caratteri di controllo consentiti nelle stringhe DCL

Carattere di controllo

Carattere di controllo	Significato
\"	virgolette (incluse)
\\	barra rovesciata
\n	nuova linea
\t	tab orizzontale

- *Parola destinata.* Una parola destinata è un identificatore composto da caratteri alfanumerici ed inizia con una lettera. Molti attributi, ad esempio, richiedono il valore true o false. Le parole destinate sono sensibili al maiuscolo/minuscolo: True non equivale a true.

Nota Come le parole e le stringhe destinate, anche i nome degli attributi sono sensibili al maiuscolo/minuscolo: non è possibile assegnare la larghezza chiamandola Larghezza.

I programmi dell'applicazione ricercano gli attributi sempre sotto forma di stringa. Se la propria applicazione utilizza valori numerici, deve convertirli in e da valori di stringa. Per ulteriori informazioni su come gestire i valori delle caselle all'interno di un programma AutoLISP, vedere "Gestione delle caselle."

Alcuni attributi, come width ed height sono comuni a tutte le caselle. Le specifiche degli attributi sono opzionali. Molti attributi hanno valori di default che vengono utilizzati nel caso in cui l'attributo non sia specificato. Altri attributi vengono menzionati appositamente per determinati tipi di caselle, ad esempio il colore dello sfondo di una immagine. Se questo attributo viene assegnato ad un diverso tipo di casella, è possibile che AutoCAD segnali un errore. Normalmente, però, AutoCAD ignora l'attributo.

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi limitati

Non utilizzare i seguenti attributi di casella nei file DCL:

horizontal_margin
vertical_margin
type

Capitolo 18 -- Linguaggio DCL

Attributi

Indice degli attributi predefiniti

Questa sezione descrive gli attributi definiti dalla funzione PDB. La tabella seguente riporta, in ordine alfabetico, l'elenco degli

attributi predefiniti. Gli attributi verranno descritti dettagliatamente dopo la tabella.

Attributi predefiniti

Nome attributo	Associato a	Significato (se specificato o vero)
action	tutte le caselle attive	espressione operativa di AutoLISP
alignment	tutte le caselle	posizione orizzontale o verticale rispetto ad un cluster
allow_accept	edit_box, image_button e list_box	attiva il pulsante is_default quando questa casella è selezionata
aspect_ratio	image, image_button	proporzioni di una immagine
big_increment	slider	distanza incrementale per lo spostamento
children_alignment	row, column, radio_row, radio_column, boxed_row, boxed_column, boxed_radio_row e boxed_radio_column	allineamento delle sottocaselle di un cluster
children_fixed_height	row, column, radio_row, radio_column, boxed_row, boxed_column, boxed_radio_row e boxed_radio_column	altezza delle sottocaselle di un cluster, che non aumenta in fase di layout
children_fixed_larghezza	row, column, radio_row, radio_column, boxed_row, boxed_column, boxed_radio_row e boxed_radio_column	larghezza di sottocaselle di un cluster, che non aumenta in fase di layout
color	image, image_button	colore (riempimento) dello sfondo di un'immagine
edit_limit	edit_box	numero massimo di caratteri che l'utente può immettere
edit_width	edit_box, popup_list	larghezza della parte di modifica (input) della casella
fixed_height	tutte le caselle	altezza che non aumenta in fase di layout
fixed_width	tutte le caselle	larghezza che non aumenta in fase di layout
fixed_width_font	list_box, popup_list	visualizza testo con un font di dimensioni definite
height	tutte le caselle	altezza della casella
initial_focus	dialog	chiave della casella con evidenziazione iniziale
is_bold	text	visualizzato in grassetto
is_cancel	button	pulsante attivato quando si preme il tasto di annullamento, in genere ESC
is_default	button	pulsante attivato quando si preme il tasto di accettazione, in genere
is_enabled	tutte le caselle attive	casella inizialmente abilitata
is_tab_stop	tutte le caselle attive	la casella è un punto di tabulazione
key	tutte le caselle attive	nome casella utilizzato dall'applicazione
label	boxed_row, boxed_column, boxed_radio_row, boxed_radio_column, button, dialog, edit_box, list_box, popup_list, radio_button, text e toggle	etichetta visualizzata della casella
layout	slider	se il dispositivo di scorrimento è verticale o orizzontale
list	list_box, popup_list	valori iniziali da visualizzare nell'elenco
max_value	slider	valore massimo di un dispositivo di scorrimento
min_value	slider	valore minimo di un dispositivo di scorrimento
mnemonic	tutte le caselle attive	caratteri mnemonici per la casella
multiple_select	list_box	la casella di riepilogo permette la selezione di più caselle
password_char	edit_box	nasconde i caratteri immessi nella casella edit_box in cui è specificato
small_increment	slider	distanza incrementale per lo spostamento
tabs	list_box, popup_list	punti di tabulazione per la visualizzazione dell'elenco
tab_truncate	list_box, popup_list	tronca il testo che è più grande del punto di tabulazione associato
valore	testo, caselle attive (eccetto pulsanti e pulsanti di immagine)	valore iniziale della casella
larghezza	tutte le caselle	larghezza della casella

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

Nel catalogo degli attributi riportato di seguito sono elencati gli attributi utilizzati dalle caselle DCL predefinite.

Capitolo 18 -- Linguaggio DCL



```
action = "(funzione)";
```

Specifica un'espressione AutoLISP per effettuare un'azione quando viene selezionata questa casella. Questa procedura è chiamata anche *funzione di richiamo*. Per alcuni tipi di casella può verificarsi un'azione anche quando l'utente pone l'evidenziazione su una diversa casella.

Il valore possibile è una stringa tra virgolette, che costituisce un'espressione AutoLISP valida. Non è possibile specificare un attributo di un'azione in ADSRX, sebbene sia possibile scrivere l'applicazione che supporta la propria finestra di dialogo in entrambi i linguaggi. Una casella può definire una sola azione. Se l'applicazione assegna un'azione ad una casella (con **action_tile**), l'attributo **action** verrà sovrascritto.

Nota Non è possibile richiamare la funzione AutoLISP **command** dall'attributo **action**.

Capitolo 18 -- Linguaggio DCL



```
alignment = posizione;
```

Specifica il posizionamento orizzontale o verticale (giustificazione) di una casella all'interno del suo cluster.

Per una casella che è una sottocasella di una colonna, i valori possibili sono left, right o centered (default: left).

Per una casella che è una sottocasella di una riga, i possibili valori sono top, bottom o centered (default: centered).

Non è possibile specificare l'allineamento rispetto all'asse di un cluster. La prima e l'ultima casella nel cluster si allineano da sole alle estremità della colonna o della riga. Le altre caselle nel cluster vengono distribuite equamente a meno che non si modifichi questa distribuzione, utilizzando come riempimento i punti di inserimento (vedere "spacer_0").

Capitolo 18 -- Linguaggio DCL



```
allow_accept = true-false;
```

Specifica se la casella è attivata quando l'utente preme il tasto di accettazione (normalmente RETURN). Se la casella ha il valore true e l'utente preme il tasto di accettazione, l'eventuale pulsante di default è "premutato". Il pulsante di default è la casella button, il cui attributo is_default è impostato su true. Questo attributo per default è false.

Capitolo 18 -- Linguaggio DCL

-  **Attributi**
-  **Attributi predefiniti**
-  **aspect_ratio**

`aspect_ratio = reale;`

Specifica il rapporto tra larghezza ed altezza di un'immagine (larghezza divisa per altezza). Se la proporzione è uguale a zero (0.0), la casella viene adattata alla dimensione dell'immagine.

I valori possibili sono i valori a virgola mobile (default: nessuno).

Capitolo 18 -- Linguaggio DCL

-  **Attributi**
-  **Attributi predefiniti**
-  **big_increment**

`big_increment = intero;`

Specifica il valore utilizzato dai controlli incrementali di un dispositivo di scorrimento. Il valore di default di `big_increment` è un decimo dell'intervallo totale. Il valore deve essere compreso nell'intervallo tra `min_value` e `max_value`.

Capitolo 18 -- Linguaggio DCL

-  **Attributi**
-  **Attributi predefiniti**
-  **children_alignment**

`children_alignment = posizione;`

Specifica l'allineamento di default (simile ad `alignment`) per tutte le caselle all'interno di un cluster. *Non* sovrascrive un attributo di allineamento di una sottofinestra, quando questo viene esplicitamente specificato.

I valori possibili per le colonne sono `left`, `right` o `centered` (default: `left`).

I valori possibili per le righe sono `top`, `bottom` o `centered` (default: `centered`).

Capitolo 18 -- Linguaggio DCL

-  **Attributi**
-  **Attributi predefiniti**
-  **children_fixed_height**

`children_fixed_height = true-false;`

Specifica l'altezza di default (simile a `height`) per tutte le caselle all'interno di un cluster. *Non* sovrascrive un attributo di altezza di una

sottofinestra, quando questo viene esplicitamente specificato.

I valori possibili sono true o false (default: false).

Nota Utilizzare gli attributi fixed_ con discrezione. Sovrascritture inconsistenti generano dei layout inconsistenti.

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

children_fixed_width

```
children_fixed_width = true-false;
```

Specifica la larghezza di default (simile a width) per tutte le caselle all'interno di un cluster. *Non* sovrascrive un attributo di larghezza di una sottofinestra, quando questo viene esplicitamente specificato.

I valori possibili sono true o false (default: false).

Nota Utilizzare gli attributi fixed_ con discrezione. Sovrascritture inconsistenti generano dei layout inconsistenti.

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

color

```
color = nomecolore;
```

Specifica il colore dello sfondo (riempimento) di un'immagine. Il valore possibile è un numero intero oppure una parola riservata (default: 7) specificato come numero di colore di AutoCAD oppure come uno dei nomi simbolici riportati nella seguente tabella.

Nomi simbolici

Nome simbolico	Significato
dialog_line	È il colore della linea corrente della finestra di dialogo.
dialog_foreground	È il colore di primo piano corrente della finestra di dialogo (per testo).
dialog_background	È il colore di sfondo corrente della finestra di dialogo.
graphics_background	Sfondo corrente dello schermo di disegno AutoCAD (in genere equivale a 0)
black	Colore AutoCAD = 0 (nero) (appare più chiaro su sfondo nero)
red	Colore AutoCAD = 1 (rosso)
yellow	Colore AutoCAD = 2 (giallo)
green	Colore AutoCAD = 3 (verde)
cyan	Colore AutoCAD = 4 (ciano)
blue	Colore AutoCAD = 5 (blu)
magenta	Colore AutoCAD = 6 (magenta)
white	Colore AutoCAD = 7 (bianco)
graphics_foreground	(appare nero su sfondo bianco)

I nomi simbolici graphics_background e graphics_foreground sono previsti come alternative per black e white. Utilizzare un colore specifico può creare confusione, poichè il colore visualizzato può variare a seconda della configurazione corrente di AutoCAD. Inoltre, i vettori visualizzati nelle diapositive di un'immagine sono spesso disegnati in bianco e nero. Se la casella visualizzata la prima volta risulta vuota, è possibile tentare di modificare il suo colore in graphics_background o graphics_foreground.

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

edit_limit

`edit_limit = intero;`

Specifica il massimo numero di caratteri che un utente può inserire in una casella di modifica. Il valore possibile è un numero intero (default: 132). Ogni volta che un utente raggiunge tale limite, AutoCAD rifiuta ulteriori caratteri (eccetto per BACKSPACE or DEL). Il limite massimo consentito è di 256 caratteri.

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

edit_width

`edit_width = numero;`

Specifica la larghezza nell'unità di larghezza carattere della parte modificabile (input) della casella, l'attuale parte incasellata della casella edit_box. I valori possibili sono numeri interi o reali. Se edit_width non è specificato o è uguale a zero e la larghezza della casella non è fissa, la casella si espande fino a riempire lo spazio disponibile. Se edit_width è diverso da zero, la casella appare allineata sulla destra entro il limite di spazio occupato dalla casella. Se è necessario "stirare" la casella per esigenze di layout, la funzione PDB inserisce uno spazio bianco tra l'etichetta e la parte modificabile della casella.

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

fixed_height

`fixed_height = true-false;`

Specifica se è consentito che l'altezza di una casella occupi tutto lo spazio disponibile. Se questo attributo è di valore true, la casella non riempirà lo spazio supplementare, reso disponibile durante la procedura di layout/allineamento.

I valori possibili sono true o false (default: false).

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

fixed_width

`fixed_width = true-false;`

Specifica se è consentito che la larghezza di una casella occupi tutto lo spazio disponibile. Se questo attributo è di valore true, la

casella non riempirà lo spazio supplementare, reso disponibile durante la procedura di layout/allineamento.

I valori possibili sono true o false (default: false).

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

fixed_width_font

```
fixed_width_font = true-false;
```

Specifica se una casella list_box o popup_list visualizzerà il testo con un font di dimensioni definite. Ciò consente una maggiore facilità di spaziatura e di allineamento delle colonne con la tabulazione.

I valori possibili sono true o false (default: false).

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

height

```
height = numero;
```

Specifica l'altezza di una casella. I valori possibili sono numeri interi o reali, che rappresentano la distanza in unità di altezza carattere. Non specificare questo valore, a meno che i valori di default assegnati non risultino inaccettabili. È necessario, tuttavia, specificare l'altezza delle caselle e dei pulsanti di immagine.

L'attributo height specifica l'altezza *minima* di una casella. Questa dimensione può essere espansa quando la casella verrà rappresentata nel disegno, a meno che l'altezza non sia prefissata da uno degli attributi fixed_. I valori di default vengono assegnati dinamicamente entro i limiti del layout.

L'*unità di altezza carattere* è definita come altezza massima dei caratteri video (incluso spaziatura linea).

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

initial_focus

```
initial_focus = "stringa";
```

Specifica la chiave della casella all'interno della finestra di dialogo, sul quale è posta inizialmente l'evidenziazione. Il valore possibile è una stringa tra virgolette (nessun default).

Capitolo 18 -- Linguaggio DCL



```
is_bold = true-false;
```

Specifica se i caratteri di un testo verranno visualizzati in grassetto. I valori possibili sono true o false (default: false). Se il valore è true, il testo verrà visualizzato in grassetto.

Capitolo 18 -- Linguaggio DCL



```
is_cancel = true-false;
```

Specifica se il pulsante è attivato quando l'utente preme il tasto di annullamento (ESC). I valori possibili sono true o false (default: false).

Se l'espressione di azione per tutti i pulsanti aventi l'attributo is_cancel impostato su true non determina l'uscita dalla funzione **done_dialog**, la finestra di dialogo verrà chiusa automaticamente dopo l'esecuzione dell'espressione di azione e la variabile di sistema DIASTAT viene impostata su 0.

Solo uno dei pulsanti all'interno di una finestra di dialogo può avere l'attributo is_cancel impostato su true.

Capitolo 18 -- Linguaggio DCL



```
is_default = true-false;
```

Specifica se il pulsante è il pulsante di default selezionato ("premuta") quando l'utente preme il tasto di accettazione. I valori possibili sono true o false (default: false). Se l'utente è all'interno della casella edit_box, list_box o image_button che ha l'attributo allow_accept impostato su true, il pulsante di default è selezionato anche se l'utente preme il tasto di accettazione oppure effettua un doppio clic (per caselle di elenchi e pulsanti di immagine). Il pulsante di default *non* viene selezionato qualora l'evidenziazione sia su un altro pulsante. In questo caso, il pulsante selezionato è quello su cui è l'evidenziazione.

Solo uno dei pulsanti all'interno di una finestra di dialogo può avere l'attributo is_default impostato su true.

Capitolo 18 -- Linguaggio DCL



```
is_enabled = true-false;
```

Specifica se una casella è comunque inizialmente disabilitata (non disponibile). I valori possibili sono true o false (default: true). Se il valore è false, la casella è inizialmente disabilitata.

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

is_tab_stop

```
is_tab_stop = true-false;
```

Specifica se è possibile impostare l'evidenziazione sulla casella dalla tastiera quando l'utente si muove tra le diverse caselle premendo il tasto TAB. I valori possibili sono true o false (default: true). Se la casella è disabilitata, non è un arresto di tabulazione, neanche quando ha il valore true. Se il valore è false, la casella non è un punto di tabulazione.

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

key

```
key = "stringa";
```

Specifica un nome ASCII utilizzato dal programma per far riferimento a questa specifica casella. Il valore possibile è una stringa tra virgolette (nessun default). All'interno della stessa finestra di dialogo, ogni valore chiave deve essere univoco. Questa stringa è sensibile al maiuscolo/minuscolo: se la chiave specificata è BigTile, non è possibile fare riferimento ad essa come bigtile.

Poiché il valore di una chiave non è visibile all'utente, il nome può essere scelto liberamente (entro il limite di compatibilità con la finestra di dialogo). Per la stessa ragione, non è necessario convertire gli attributi della chiave per applicazioni in più lingue.

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

label

```
label = "stringa";
```

Specifica il testo visualizzato all'interno di una casella. Il valore possibile è una stringa tra virgolette (default: una stringa nulla, " "). La posizione del testo label è una specifica della casella.

L'attributo etichetta può specificare un carattere mnemonico per la casella. Il carattere mnemonico è sottolineato nell'etichetta della casella.

Se un carattere della stringa dell'etichetta è preceduto dalla ampersand (&), questo carattere diventerà mnemonico. Il carattere non deve essere necessariamente univoco all'interno della finestra di dialogo: se più caselle hanno lo stesso carattere mnemonico, l'utente premerà quel tasto per passare in sequenza tutte le caselle.

I caratteri mnemonici spostano l'evidenziazione; ovvero, non selezionano mai un casella. Se l'utente specifica un tasto mnemonico per una casella che contiene un gruppo di elementi, come ad esempio un cluster o una casella di riepilogo, l'evidenziazione si sposta sul primo elemento della casella che è un punto di tabulazione. Ogni casella attiva è un punto di tabulazione, a meno che il suo attributo is_tab_stop non sia impostato su false.

Nota l'attributo mnemonic, inoltre, specifica un carattere mnemonico.

Capitolo 18 -- Linguaggio DCL

-  **Attributi**
-  **Attributi predefiniti**
-  **layout**

```
layout = posizione;
```

Specifica l'orientamento di un dispositivo di scorrimento. I valori possibili sono horizontal o vertical (default: hor-izontal). Per dispositivi di scorrimento orizzontali, il valore aumenta da sinistra verso destra; per quelli verticali aumenta dal basso verso l'alto.

Capitolo 18 -- Linguaggio DCL

-  **Attributi**
-  **Attributi predefiniti**
-  **list**

```
list = "stringa";
```

Specifica il gruppo iniziale di righe (scelte) da posizionare nelle caselle popup_list o list_box. Il valore possibile è una stringa tra virgolette (nessun default). Le linee vengono separate dal simbolo nuova riga (\n). All'interno di ogni riga possono trovarsi caratteri di tabulazione (\t).

Capitolo 18 -- Linguaggio DCL

-  **Attributi**
-  **Attributi predefiniti**
-  **max_value**

```
max_value = intero;
```

Specifica il valore superiore che un dispositivo di scorrimento riporta. Il valore massimo di default è 10000. Questo valore deve avere il segno e deve essere un numero intero a 16 bit non maggiore di 32767.

Capitolo 18 -- Linguaggio DCL

-  **Attributi**
-  **Attributi predefiniti**
-  **min_value**

```
min_value = intero;
```

Specifica il valore inferiore che un dispositivo di scorrimento riporta. Il valore minimo di default è 0. Questo valore deve avere il segno e deve essere un numero intero a 16 bit non minore di -32768. È possibile che il valore min_value sia maggiore di max_value.

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

mnemonic

```
mnemonic = "char";
```

Specifica un carattere mnemonico di tastiera per una casella. Il carattere mnemonico è sottolineato nell'etichetta della casella. Il valore possibile è una stringa tra virgolette o un carattere *singolo* (nessun default). Il carattere deve essere una lettera contenuta nell'etichetta della casella. Il carattere non deve essere necessariamente univoco all'interno della finestra di dialogo: se più caselle hanno lo stesso carattere mnemonico, l'utente premerà quel tasto per passare in sequenza tutte le caselle.

I caratteri mnemonici non sono sensibili al maiuscolo/minuscolo; ad esempio, se il carattere mnemonico è la lettera A, inserendo **a** o **A** l'evidenziazione viene spostata su quel pulsante. Tuttavia, nei file DCL il carattere mnemonico deve essere uno di quelli contenuti nell'etichetta della casella e deve apparire nello stesso modo in cui appare nella stringa label.

I caratteri mnemonici spostano l'evidenziazione; ovvero, non selezionano mai un casella. Se l'utente specifica un tasto mnemonico per una casella che contiene un gruppo di elementi, come ad esempio un cluster o una casella di riepilogo, l'evidenziazione si sposta sul primo elemento della casella che è un punto di tabulazione. Ogni casella attiva è un punto di tabulazione, a meno che il suo attributo `is_tab_stop` non sia impostato su `false`.

Nota Anche l'attributo `label` può specificare un carattere mnemonico.

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

multiple_select

```
multiple_select = true-false;
```

Specifica se più di un elemento della casella `list_box` può essere selezionato (ed evidenziato) allo stesso tempo. I valori possibili sono `true` o `false` (default: `false`). Se il valore è `true` si possono selezionare più elementi contemporaneamente.

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

password_char

```
password_char= "char";
```

Specifica il carattere da usare come carattere della password. Se `password_char` è specificato e non è nullo, quel carattere viene visualizzato nella casella di modifica al posto dei caratteri immessi dall'utente. L'utilizzo di questo attributo non ha effetti sul recupero del valore immesso dall'utente; esso altera soltanto la visualizzazione dei caratteri nella casella di modifica.

Per un esempio sull'uso dell'attributo `password_char` in una applicazione, vedere "Richiesta della parola d'ordine."

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

small_increment

```
small_increment = intero;
```

Specifica il valore utilizzato dai controlli incrementali di un dispositivo di scorrimento. Il valore di default di `small_increment` è un centesimo dell'intervallo totale. Il valore deve essere compreso nell'intervallo tra `min_value` e `max_value`. Questo attributo è opzionale.

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

tabs

```
tabs = "stringa";
```

Specifica il posizionamento dei tabulatori nell'unità di larghezza carattere. Il valore possibile è una stringa tra virgolette che contiene numeri interi o numeri a virgola mobile, separati da spazi (nessun default). Questi valori sono utilizzati per l'allineamento verticale delle colonne di testo nelle caselle `popup_list` o `list_box`.

Ad esempio, nel codice riportato di seguito viene specificato un arresto di tabulazione dopo ogni 8 caratteri.

```
tabs = "8 16 24 32";
```

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

tab_truncate

```
tab_truncate = true-false;
```

Specifica se il testo nelle caselle `list_box` o `popup_list` viene troncato nel caso in cui sia più largo del punto di tabulazione associato. Ciò evita che un testo con molti spazi *M* seguito da una tabulazione, ad esempio, spinga il testo che segue oltre il suo punto di tabulazione.

I valori possibili sono `true` o `false` (default: `false`).

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

value

```
value = "stringa";
```

Specifica il valore iniziale di una casella. Il valore possibile è una stringa tra virgolette. Il significato del valore di una casella varia a seconda del tipo di casella. Il valore di una casella può essere modificato durante l'esecuzione da un input dell'utente o chiamando la funzione **set_tile**.

L'attributo value della casella non viene considerato quando la finestra di dialogo è in fase di layout. Quando la fase di layout è terminata e la finestra di dialogo è stata visualizzata, il comando **new_dialog** utilizza gli attributi di valore per inizializzare ogni casella nella finestra di dialogo. L'attributo di valore di una casella non ha effetto sulla dimensione o la spaziatura della stessa all'interno della finestra di dialogo.

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi predefiniti

width

```
width = number;
```

Specifica la larghezza di una casella. I valori possibili sono numeri interi o reali, che rappresentano la distanza in unità di larghezza carattere. Non specificare questo valore, a meno che i valori di default non diano risultati inaccettabili. È necessario comunque specificare la larghezza della casella immagine e dei pulsanti.

L'attributo width di una casella specifica la larghezza *minima*. Questa dimensione può essere espansa quando la casella verrà rappresentata nel disegno, a meno che la larghezza non sia prefissata da uno degli attributi fixed_. I valori di default vengono assegnati dinamicamente entro i limiti del layout.

L'*unità di larghezza carattere* è definita come la larghezza media tra i caratteri maiuscoli e minuscoli delle lettere dell'alfabeto, oppure la larghezza dello schermo diviso per 80, che è sempre minore (larghezza media è $(\text{width}(A \dots Z) + \text{width}(a \dots z))/52$).

Capitolo 18 -- Linguaggio DCL

Attributi

Attributi definiti dall'utente

Se l'utente lo desidera, può assegnare degli attributi personalizzati alle caselle, quando questi vengono definiti. Il nome di questi attributi può essere un nome valido qualsiasi, purché non sia in conflitto con i nomi degli attributi standard e predefiniti descritti nella sottosezione precedente ed elencati nella tabella "Indice degli attributi predefiniti." Come una parola chiave, anche il nome dell'attributo può contenere lettere, numeri o il segno di sottolineatura (_), e deve essere preceduto da una lettera.

Nota Se un attributo definito dall'utente entra in conflitto con uno predefinito, la funzione PDB non riconosce questo attributo come uno nuovo e tenterà di utilizzare il valore assegnatogli dall'utente con l'attributo *standard*. In questo caso, il debug può risultare difficile.

I valori assegnati agli attributi ed il loro significato sono definiti dalla propria applicazione. I valori per gli attributi definiti dall'utente devono essere conformi ai tipi descritti in "Attributi."

La definizione degli attributi è paragonabile alla definizione dei dati client delle specifiche dell'applicazione. Entrambe le tecniche abilitano la funzione PDB a gestire i dati forniti dall'utente; ovvero, questi dati risultano statici quando la finestra di dialogo è attiva. Se è necessario modificare i dati dinamicamente, occorre utilizzare i dati client al momento dell'esecuzione. L'utente finale, inoltre, può prendere visione dei valori degli attributi definiti dall'utente nel file DCL dell'applicazione, ma i dati client rimarranno invisibili.

La definizione della finestra di dialogo di AutoCAD Aiuti per il disegno specifica il proprio attributo, errmsg, che dispone di una stringa univoca per ogni casella. Un comune gestore di errori utilizza il valore dell'attributo errmsg quando viene visualizzato un messaggio di avvertimento. Ad esempio, la casella potrebbe assegnare il seguente valore all'attributo errmsg:

```
errmsg = "Grid Y Spacing";
```

Se l'utente immette un valore inutilizzabile, come ad esempio un numero negativo, AutoCAD visualizza il seguente messaggio di avviso:

Spaziatura griglia Y invalida.

La parolaInvalida ed il punto finale (.) vengono forniti dal gestore di errori.

Gli attributi definiti dall'utente possono essere utilizzati anche per delimitare i valori di una casella ed il nome di una sottofinestra di dialogo che viene attivata dalla stessa casella. Vedere "Finestre di dialogo nidificate."

Capitolo 18 -- Linguaggio DCL

Sinopsi delle funzioni delle caselle DCL

Questa sezione presenta le caselle DCL suddivise in gruppi di funzioni.

Capitolo 18 -- Linguaggio DCL

Sinopsi delle funzioni delle caselle DCL

Caselle attive predefinite

La funzione PDB di AutoCAD dispone di un gruppo di caselle incorporate o predefinite che possono essere utilizzate così come sono o come base per caselle più complesse. Le loro definizioni appaiono nei commenti al file *base.dcl*. Vedere "File base.dcl e acad.dcl."

Quando l'utente sceglie una casella attiva, ad esempio un pulsante, la finestra di dialogo risponde notificando l'applicazione che controlla la finestra di dialogo. Ogni casella attiva e predefinita può avere un'azione associata. Il risultato di questa azione può essere visibile all'utente o può essere completamente interna (ad esempio, un aggiornamento di stato). Ogni azione è seguita da un codice che indica il motivo per il quale è stato originata l'azione. Questo motivo dipende dal tipo di casella che ha originato l'azione stessa. Le caselle seguenti sono selezionabili ed attive:

button	popup_list
edit_box	radio_button
image_button	slider
list_box	toggle

Capitolo 18 -- Linguaggio DCL

Sinopsi delle funzioni delle caselle DCL

Cluster di caselle

È possibile raggruppare le caselle in righe o colonne composte (dette comunemente *cluster* o *disegno d'insieme secondario*). Per esigenze di layout, una riga o una colonna viene trattata come una singola casella. Una riga o una colonna può essere incasellata e quindi avere un'etichetta opzionale (un cluster privo di casella non può essere etichettato).

L'utente non può selezionare un cluster, ma solo singole caselle (selezionabili, attive) contenute nel cluster. I cluster non possono avere azioni ad essi associate, eccetto le righe e le colonne di scelta. Le caselle seguenti definiscono un cluster:

boxed_column	dialog
boxed_radio_column	radio_column
boxed_radio_row	radio_row
boxed_row	row
column	

Capitolo 18 -- Linguaggio DCL

Sinopsi delle funzioni delle caselle DCL

Caselle informative e decorative

Le caselle elencate qui di seguito *non* provocano azioni e *non possono* essere selezionate. Visualizzano informazioni o effetti particolari oppure assistono l'utente durante la fase di layout della finestra di dialogo.

image	spacer_0
text	spacer_1
spacer	

Capitolo 18 -- Linguaggio DCL

Sinopsi delle funzioni delle caselle DCL

Cluster di testo

Una casella di testo è racchiusa tra margini (come ogni altro tipo di caselle) che presentano problemi quando occorre combinare più parti di testo. Ad esempio, nel caso si voglia visualizzare il messaggio:

Sono esattamente le ore 0800 e 37 secondi.

I valori attuali (0800 e 37) sono forniti dal programma. È possibile visualizzare questa frase concatenando delle singole righe di testo costituite da singole caselle `text_part`. È possibile utilizzare parti del testo anche verticalmente per creare un paragrafo che non abbia troppo spazio tra le singole righe.

I cluster di testo seguenti sono prototipi definiti nel file *base.dcl*.

concatenation	text_part
paragraph	

Capitolo 18 -- Linguaggio DCL

Sinopsi delle funzioni delle caselle DCL

Pulsanti di uscita dalla finestra di dialogo e caselle di errore

Il file *base.dcl* fornisce un disegno d'insieme secondario del pulsante standard per l'uscita (o "abbandono") da una finestra di dialogo. Utilizzare questa versione standard per mantenere l'uniformità tra tutte le applicazioni.

È possibile personalizzare il testo su questi pulsanti utilizzando il prototipo `retirement_button`, come descritto in "Personalizzazione del testo dei pulsanti di uscita."

errtile	ok_cancel_help
ok_only	ok_cancel_help_errtile
ok_cancel	ok_cancel_help_info

Capitolo 18 -- Linguaggio DCL

Sinopsi delle funzioni delle caselle DCL

Caselle limitate

I file DCL non dovrebbero utilizzare le caselle cluster o tile. Non utilizzare, inoltre, i tipi di base dei pulsanti di uscita (cancel_button, help_button, info_button e ok_button) a meno che l'utente non stia ridefinendo il disegno di insieme secondario dei pulsanti di uscita standard secondo le istruzioni contenute in questa sezione in "Pulsanti di uscita dalla finestra di dialogo e caselle di errore."

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

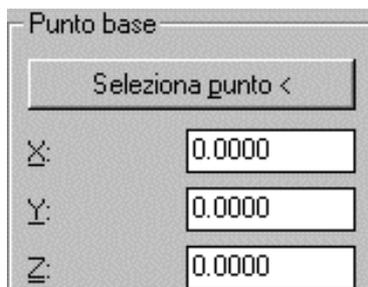
Questa sezione descrive tutte le caselle DCL predefinite. La frase di sintassi che segue il nome della casella elenca tutti gli attributi associati con questa casella. Ogni funzionalità specifica dell'attributo è riportata dopo la descrizione della casella.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

boxed_column

```
: boxed_column {
  alignment children_alignment
  children_fixed_height children_fixed_width
  fixed_height fixed_width height label width
}
```



Una colonna incasellata ha una cornice. Una finestra di dialogo è disposta come una colonna incasellata. Se una colonna incasellata ha un'etichetta, essa appare sopra il bordo della colonna o inclusa nello stesso, a seconda dell'interfaccia GUI. Se non c'è alcuna etichetta, oppure è vuota (" ") o nulla (""), verrà visualizzata solo la casella.

label

Appare come un titolo. La spaziatura tra un'etichetta vuota e nulla può essere differente (vedere "Spazio superfluo intorno ad una riga o colonna incasellata").

Capitolo 18 -- Linguaggio DCL

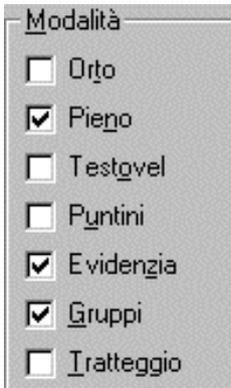
Elenco delle caselle DCL

boxed_radio_column

```

: boxed_radio_column {
  alignment children_alignment
  children_fixed_height children_fixed_width
  fixed_height fixed_width height label width
}

```



Una colonna di scelta incasellata ha una cornice. L'etichetta della colonna di scelta verrà trattata come quella di una colonna incasellata.

label

Appare come un titolo. Se l'etichetta risulta vuota (valore di default) o nulla (""), verrà visualizzata solo la casella. La spaziatura tra un'etichetta vuota e nulla può essere differente (vedere "Spazio superfluo intorno ad una riga o colonna incasellata").

valore

Specifica la chiave del pulsante di scelta correntemente selezionato (l'attributo con il valore è "1").

Capitolo 18 -- Linguaggio DCL

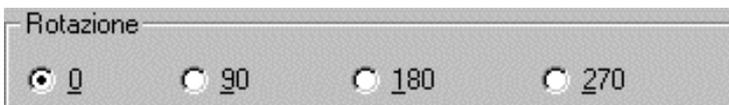
Elenco delle caselle DCL

boxed_radio_row

```

: boxed_radio_row {
  alignment children_alignment
  children_fixed_height children_fixed_width
  fixed_height fixed_width height label width
}

```



Una riga di scelta incasellata ha una cornice. L'etichetta della riga di scelta verrà trattata come l'etichetta di una colonna incasellata.

label

Appare come un titolo. Se l'etichetta risulta vuota (valore di default) o nulla (""), verrà visualizzata solo la casella. La spaziatura tra un'etichetta vuota e nulla può essere differente (vedere "Spazio superfluo intorno ad una riga o colonna incasellata").

valore

Specifica la chiave del pulsante di scelta correntemente selezionato (l'attributo con il valore è "1").

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

boxed_row

```
: boxed_row {
  alignment children_alignment
  children_fixed_height children_fixed_width
  fixed_height fixed_width height label width
}
```



Una riga incasellata ha una cornice. Se una riga incasellata ha un'etichetta, essa verrà inclusa nella cornice.

label

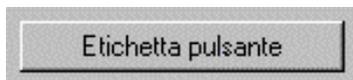
Appare come un titolo. Se l'etichetta risulta vuota (valore di default) o nulla (""), verrà visualizzata solo la casella. La spaziatura tra un'etichetta vuota e nulla può essere differente (vedere "Spazio superfluo intorno ad una riga o colonna incasellata").

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

button

```
: button {
  action alignment fixed_height fixed_width
  height is_cancel is_default is_enabled
  is_tab_stop key label mnemonic width
}
```



Una casella pulsante somiglia al pulsante da premere. L'attributo label specifica il testo che appare all'interno del pulsante. I pulsanti sono ideati ad azioni immediatamente visibili all'utente: abbandonare la finestra di dialogo, passare ad una sottofinestra di dialogo e così via.

Le finestre di dialogo devono includere un pulsante OK (o l'equivalente) che l'utente deve premere dopo aver utilizzato (o letto) la casella. Diverse finestre di dialogo includono un pulsante di annullamento che permette all'utente di abbandonare la finestra di dialogo senza eseguire modifiche.

Le finestre di dialogo devono utilizzare il disegno d'insieme secondario del pulsante di uscita standard, descritto in "Pulsanti di uscita dalla finestra di dialogo e caselle di errore." Il disegno d'insieme secondario garantisce la corretta assegnazione degli attributi descritti in questa sezione.

Nota se viene stabilito che il pulsante di default coincide con il pulsante di annullamento, è necessario assicurarsi che almeno un ulteriore pulsante di uscita sia associato ad un'azione che chiama la funzione **done_dialog**. Altrimenti, la finestra di dialogo verrà sempre cancellata.

label

Specifica il testo che appare sul pulsante.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

column

```
: column {
  alignment children_alignment
  children_fixed_height children_fixed_width
  fixed_height fixed_width height label width
}
```

On

Intervallo X

Intervallo Y

Le caselle all'interno di una colonna vengono disposte verticalmente, nell'ordine in cui appaiono nel file DCL. Una colonna può contenere qualsiasi tipo di casella (eccetto i pulsanti di chiamata solitari), *includere* righe ed altre colonne.

Una colonna senza una casella non ha attributi supplementari oltre quelli standard.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

concatenation

```
: concatenation {
}
```

Una concatenazione è una riga di testo composta da diverse caselle `text_part` concatenate. Questo può risultare utile quando si desidera inserire del testo che, durante l'esecuzione, può diventare un messaggio standard. La casella `concatenation` è completamente circondata da un margine.

La casella `concatenation` viene definita nel file `base.dcl`. Vedere l'esempio in "paragraph" dove viene utilizzato `concatenation`.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

dialog

```
: dialog {
  initial_focus label value
}
```

Dialog è la casella che definisce la finestra di dialogo. *Non* è necessario specificare sia `label` che `value` in quanto l'attributo `value` sostituisce quello di `label`.

label

Specifica il titolo opzionale visualizzato sulla barra del titolo della finestra di dialogo.

valore

Specifica una stringa da visualizzare come titolo opzionale di una finestra di dialogo. È impossibile, comunque, prendere visione del valore al momento della procedura di layout; quando si effettua un'assegnazione del valore in questo modo, occorre accertarsi quindi, che la finestra di dialogo sia abbastanza larga o che il testo possa essere troncato.

Per una casella dialog, i valori di label e value sono equivalenti, eccetto che per considerazioni di layout. Per modificare il titolo durante l'esecuzione, utilizzare la funzione `set_tile` (vedere "`set_tile`" nel capitolo 13).

initial_ focus

Specifica il pulsante della casella all'interno della finestra di dialogo, sul quale è posta inizialmente l'evidenziazione.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

`edit_box`

```
: edit_box {
  action alignment allow_accept edit_limit
  edit_width fixed_height fixed_width height
  is_enabled is_tab_stop key label mnemonic
  value width password_char
}
```



Una casella di modifica è un campo che abilita l'utente ad inserire o modificare una singola riga di testo. Una casella label opzionale può apparire sulla sinistra della casella. Se il testo inserito è più lungo della casella di modifica, scorrerà orizzontalmente.

Giustificando label a sinistra e la casella di modifica a destra, si facilita l'allineamento verticale della caselle `edit_box`.

label

Appare come un titolo. Se viene specificato, l'etichetta è giustificata a sinistra rispetto a tutta la larghezza della casella `edit_box`.

valore

È il valore ASCII iniziale inserito nella casella. Verrà visualizzato con la giustificazione a sinistra, nella parte di modifica (input) della casella. Il valore di una casella di modifica termina con un carattere nullo (\0, oppure con EOS in ADSRX). Se l'utente inserisce un numero di caratteri maggiore della casella `edit_limit` e la stringa viene troncata, verrà aggiunto il carattere nullo.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

`errtile`

```
errtile;
```



Una casella di errore è una casella di testo disposta alla base della finestra di dialogo. Per default è vuota, ma i programmi possono visualizzare messaggi al suo interno impostando il valore della casella il cui valore chiave è "error".

La casella `errtile` è definita nel file `base.dcl`.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

image

```
: image {
  action alignment aspect_ratio color
  fixed_height fixed_width height is_enabled
  is_tab_stop key label mnemonic value width
}
```



Un'immagine è un rettangolo all'interno del quale viene visualizzato un disegno grafico vettoriale. Le immagini sono utilizzate per visualizzare icone, tipi di linea, font di testo frammenti di superficie a colori nelle finestre di dialogo di AutoCAD. Per le istruzioni relative alla creazione di immagini per le caselle immagine, vedere "Creazione di immagini."

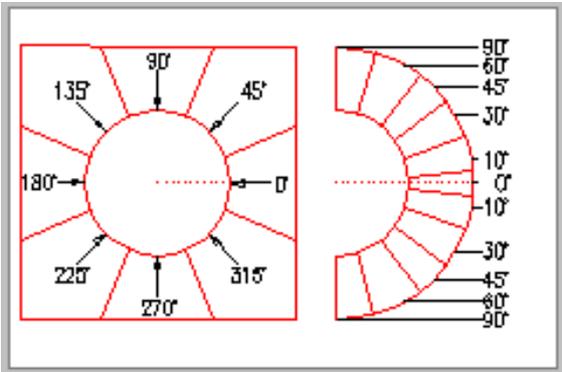
È necessario assegnare un attributo width ed height esplicito all'immagine, oppure uno dei due più l'attributo aspect_ratio.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

image_button

```
: image_button {
  action alignment allow_accept aspect_ratio
  color fixed_height fixed_width height
  is_enabled is_tab_stop key label mnemonic
  width
}
```



La casella pulsante immagine è un pulsante che visualizza un'immagine grafica al posto di un'etichetta.

Quando l'utente seleziona un pulsante immagine, il programma individua le coordinate del punto selezionato. Ciò risulta utile quando si desidera visualizzare un disegno in miniatura ed assegnare significati diversi a diverse regioni all'interno di questo disegno.

Per le istruzioni relative alla creazione di immagini per le caselle immagine, vedere "Creazione di immagini."

È necessario assegnare un attributo width ed height esplicito all'immagine, oppure uno dei due più l'attributo aspect_ratio.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

list_box

```

: list_box {
  action alignment allow_accept fixed_height
  fixed_width height is_enabled is_tab_stop
  key_label list mnemonic multiple_select tabs
  value width
}

```



Un casella di riepilogo contiene un elenco di stringhe di testo, disposte su righe. Questo elenco è, in genere, di lunghezza variabile, ma le caselle di riepilogo possono essere utilizzate per elenchi di lunghezza fissa, quando un diverso tipo di casella, come ad esempio un gruppo di pulsanti di scelta, prende troppo spazio nella casella di riepilogo. Quando l'utente seleziona una riga, quest'ultima viene evidenziata. Una casella di riepilogo può contenere un numero di righe maggiore di quante siano visualizzabili nella casella stessa, in tal caso viene sempre visualizzata una barra di scorrimento sulla destra della casella di riepilogo. La barra di scorrimento è abilitata solo nel caso in cui l'elenco contenga più elementi di quanti non siano visualizzabili contemporaneamente. Trascinando il cursore della barra di scorrimento o facendo clic sulle frecce, l'utente può scorrere tra gli elementi della casella di riepilogo. Alcune applicazioni consentono all'utente di selezionare più righe.

Per le istruzioni relative alla gestione degli elenchi di una casella di riepilogo o di un elenco a comparsa, vedere "Caselle di riepilogo ed elenchi a comparsa."

label

Testo visualizzato sopra la casella di riepilogo.

valore

Il valore è una stringa tra virgolette che contiene zero (" ") o più numeri interi, separati da spazi (nessun default). Ogni numero intero è un indice che parte da zero ed indica una casella dell'elenco, inizialmente selezionata. Se `multiple_select` è false, `value` non può contenere più di un numero intero.

Se la stringa `valore` risulta vuota (" "), nessuna casella nell'elenco è stata inizialmente selezionata. In tal caso, non è necessario specificare l'attributo `value`.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

ok_only

```
ok_only;
```



La casella `ok_only` è un pulsante OK unico, come quelli utilizzati dalle finestre di avviso. Il tasto del pulsante OK è "accept".

La casella `ok_only` è definita nel file `base.dcl`.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

ok_cancel

```
ok_cancel;
```



La casella `ok_cancel` è una combinazione dei pulsanti OK ed Annulla, ed è la combinazione standard per le finestre di dialogo che possono modificare i dati. Il tasto del pulsante Annulla è "cancel".

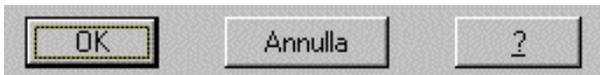
La casella `ok_cancel` è definita nel file `base.dcl`.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

ok_cancel_help

```
ok_cancel_help;
```



Questa casella è il cluster `ok_cancel` combinato con il pulsante ?. Il tasto del pulsante ? è "help". I pulsanti della guida sono raccomandati per le finestre di dialogo principali di un'applicazione e quelle particolarmente complesse. La funzione che gestisce il pulsante ? può visualizzare la finestra di dialogo della Guida di AutoCAD chiamando la funzione AutoLISP `help`.

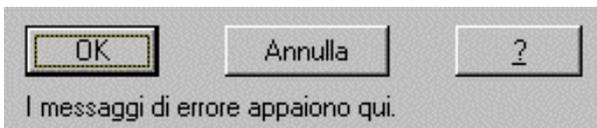
La casella `ok_cancel_help` è definita nel file `base.dcl`.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

ok_cancel_help_errtile

```
ok_cancel_help_errtile;
```



La casella `ok_cancel_help_errtile` fornisce un modo comodo per specificare in una sola volta i pulsanti di uscita e la casella di errore.

La casella `ok_cancel_help_errtile` è definita nel file `base.dcl`.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

ok_cancel_help_info

```
ok_cancel_help_info;
```



La casella `ok_cancel_help_info` ha le stesse caratteristiche della casella `ok_cancel_help`, ma include anche un pulsante di informazione per visualizzare indicazioni supplementari. Può visualizzare il nome della propria applicazione, il logo della propria ditta, il numero della versione dell'applicazione, come ottenere supporto e così via. Il tasto del pulsante di informazione è "info".

La casella `ok_cancel_help_info` è definita nel file *base.dcl*.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

paragraph

```
: paragraph {  
}
```

```
Batti il ferro  
finché è caldo
```

Un paragrafo è un cluster di caselle `text_part` o concatenation disposte verticalmente. È possibile costruire i paragrafi di testo sia a priori che in fase di esecuzione. La casella `paragraph` è completamente circondata da un margine.

La casella `paragraph` è definita nel file *base.dcl*.

Le indicazioni del margine sono generate dal DCL seguente:

```
: paragraph {  
  : concatenation {  
    : text_part {  
      label = "Batti";  
    }  
    : text_part {  
      label = "il ferro";  
    }  
  }  
  : text_part {  
    label = "finché è caldo";  
  }  
}
```

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

popup_list

```

: popup_list {
  action alignment edit_width fixed_height
  fixed_width height is_enabled is_tab_stop
  key_label list mnemonic tabs value width
}

```



Un elenco a comparsa ha la stessa funzione di una casella di riepilogo. Quando una casella di riepilogo è visualizzata la prima volta, la casella a comparsa si presenta in uno stato "compresso" ed appare come un pulsante eccetto che per una freccia che punta in basso, disposta sulla destra della casella. Quando l'utente attiva il testo o la freccia, l'elenco compare visualizzando ulteriori possibilità di selezione. L'elenco a comparsa visualizzato si presenta con una barra di scorrimento sulla destra, che funziona come la barra di scorrimento di una casella di riepilogo. Quando un elenco a comparsa è compresso, la selezione corrente viene visualizzata nel campo di visualizzazione dell'elenco stesso. Gli elenchi a comparsa non consentono selezioni multiple.

Per le istruzioni relative alla gestione degli elenchi di una casella di riepilogo o di un elenco a comparsa, vedere "Caselle di riepilogo ed elenchi a comparsa."

label

Appare come titolo sulla sinistra dell'elenco a comparsa. Se viene specificato, la casella label è giustificata a sinistra rispetto a tutta la larghezza della casella popup_list.

edit_width

Specifica la larghezza della parte del testo di un elenco in unità di larghezza carattere. Non include l'etichetta supplementare sulla sinistra o la freccia a comparsa (o la barra di scorrimento) sulla destra. Se edit_width non è specificato o è uguale a zero e la larghezza della casella non è fissa, la casella si espande fino a riempire lo spazio disponibile. Il valore possibile è un numero intero o reale. Se edit_width è diverso da zero, la casella appare allineata sulla destra entro il limite di spazio occupato dalla casella. Se fosse necessario "stirare" la casella per necessità di layout, la funzione PDB inserisce dello spazio bianco tra l'etichetta e la parte destinata alle modifica della casella.

valore

Una stringa tra virgolette contenente un numero intero (default: "0"). Il numero intero è un indice che parte da zero e indica la casella selezionata correntemente nell'elenco (ovvero, quello che viene visualizzato quando l'elenco intero non compare).

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

radio_button

```

: radio_button {
  action alignment fixed_height fixed_width
  height is_enabled is_tab_stop key_label
  mnemonic value width
}

```

 **Principale**

Il pulsante di scelta è parte di un gruppo di pulsanti che insieme costituiscono una *colonna di scelta* o *riga di scelta*. Questi pulsanti funzionano come quelli di un'autoradio: sono selezionabili uno alla volta e quando uno è premuto, tutti gli altri pulsanti della colonna o della riga sono disattivati. Un'etichetta supplementare viene visualizzata sulla destra del pulsante di scelta. La funzione PDB segnala un errore qualora si tentasse di posizionare un pulsante di scelta al di fuori di una colonna o riga.

label

Il testo visualizzato sulla destra del pulsante di scelta.

valore

Una stringa tra virgolette (nessun default). Se value è "1", radio_button è attivato; se è "0", radio_button è disattivato; tutti gli altri valori sono pari a "0".

Se casualmente più di un radio_button nel cluster radio è value = "1", solo l'ultimo di questi è attivato. Tale condizione può verificarsi in un file DCL. Una volta che una finestra di dialogo è attivata, sarà la funzione PDB a gestire i pulsanti di scelta ed assicurerà che sia attivato un solo pulsante alla volta per cluster.

Capitolo 18 -- Linguaggio DCL
 **Elenco delle caselle DCL**
 **radio_column**

```
: radio_column {
  alignment children_alignment
  children_fixed_height children_fixed_width
  fixed_height fixed_width height label width
}
```

 Principale
 Lineare
 Radiale
 Angolare

Una colonna di scelta contiene pulsanti di scelta selezionabili singolarmente uno alla volta. Le colonne di scelta contengono un gruppo fisso di scelte che si escludono reciprocamente. A differenza delle colonne ordinarie è possibile assegnare delle azioni alle colonne di scelta.

valore

Una stringa tra virgolette contenente la chiave del pulsante di scelta correntemente selezionato (il cui valore è "1").

Capitolo 18 -- Linguaggio DCL
 **Elenco delle caselle DCL**
 **radio_row**

```
: radio_row {
  alignment children_alignment
  children_fixed_height children_fixed_width
  fixed_height fixed_width height label width
}
```



Una riga di scelta, come una colonna di scelta, contiene pulsanti di scelta, selezionabili uno alla volta. È possibile assegnare un'azione alle righe di scelta.

valore

Una stringa tra virgolette contenente la chiave del pulsante di scelta correntemente selezionato (il cui valore è "1").

Nota Poiché il mouse deve percorrere un tratto più ampio, è più difficile utilizzare le righe di scelta che non le colonne di scelta. Usare le righe di scelta solo per specificare da due a quattro opzioni, o quando le etichette sono brevi.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

row

```
: row {
  alignment children_alignment
  children_fixed_height children_fixed_width
  fixed_height fixed_width height label width
}
```

Una riga è come una colonna, ma le sue caselle sono disposte orizzontalmente e non verticalmente, nell'ordine con cui appaiono nel file DCL.

Una riga senza casella non ha attributi supplementari oltre a quelli standard di layout.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

slider

```
: slider {
  action alignment big_increment fixed_height
  fixed_width height key label layout
  max_value min_value mnemonic small_increment
  value width
}
```



Un dispositivo di scorrimento ha un valore numerico. L'utente può trascinare l'indicatore del dispositivo di scorrimento a sinistra o a destra, in alto o in basso, per ottenere un valore il cui significato dipende dall'applicazione. Il valore viene riportato come stringa contenente un numero intero con segno, entro un intervallo specificato (il numero intero ha un valore a 16-bit, quindi l'intervallo maggiore è tra -32,768 e 32,767). A richiesta, l'applicazione può scalare questo valore.

valore

Una stringa tra virgolette contenente il valore corrente (numero intero) del dispositivo di scorrimento (default: min_value).

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

text

```
: text {
  alignment fixed_height fixed_width height
  is_bold key label value width
}
```

Una casella di testo visualizza una stringa di testo per formare titoli o informazioni.

Non è necessario utilizzare sempre le caselle di testo, poiché molte caselle hanno attributi propri per le etichette (label). Ma è utile usare una casella di testo, che in genere rimane vuota, per visualizzare il riscontro delle azioni dell'utente, i messaggi di errore o gli avvertimenti.

Caselle di avviso e di errore sono illustrate in "Pulsanti di uscita dalla finestra di dialogo e caselle di errore" e "Gestione degli errori."

Occorre specificare nell'attributo label se si desidera creare un messaggio a priori, ed in questo caso non viene specificato né width, né value. Se si specifica un messaggio modificabile in fase di esecuzione, occorre assegnare un valore di width sufficiente per contenere qualsiasi stringa si voglia assegnare come value. Una volta che la finestra di dialogo è impostata, è impossibile modificare la dimensione delle sue caselle, e quindi, qualora si usasse **set_tile** per assegnare una stringa più lunga della larghezza della casella, il testo in eccedenza verrà troncato.

label

Il testo visualizzato. Quando una casella di testo è impostata, la sua larghezza complessiva è il valore del suo attributo width, se quest'ultimo è specificato nel file DCL, oppure, se specificato, il valore della larghezza richiesta dal suo attributo label. Almeno uno di questi attributi deve essere specificato.

valore

Come label, l'attributo value specifica una stringa da visualizzare nella casella di testo. Non influenza comunque il layout della casella.

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

text_part

```
: text_part {
  label
}
```

Una porzione di testo è una casella di testo che fa parte di una combinazione di testo più ampia. I margini di una porzione text_part sono soppressi in modo tale da permettere alle singole porzioni di essere combinate tra di loro in una casella di concatenazione o di paragrafo.

La casella text_part viene definita nel file *base.dcl*. Per un esempio relativo all'uso di text_part, vedere "paragraph."

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

toggle

```
: toggle {
  action alignment fixed_height fixed_width
  height is_enabled is_tab_stop label width
}
```

Attiva grip

Un interruttore controlla un valore booleano ("0" o "1"). Appare come una piccola casella con una label supplementare sulla destra della casella. Quando l'utente seleziona la casella appare (o scompare) un segno di spunta o una X. Gli interruttori consentono all'utente di prendere visione o di modificare lo stato di on/off delle opzioni. Sono noti anche come *caselle di spunta*.

label

Il testo visualizzato sulla destra nella casella di interruttori.

valore

Una stringa tra virgolette contenente un numero intero (default: "0") che specifica lo stato iniziale del toggle. Se la stringa è uguale a "0", la casella dell'interruttore è vuota (senza segno di spunta). Se il valore è "1", la casella contiene un segno di spunta (o una X).

Capitolo 18 -- Linguaggio DCL

Elenco delle caselle DCL

spacer

```
: spacer {
  alignment fixed_height fixed_width
  height width
}
```

Uno spaziatore è una casella vuota. È utilizzato solo per esigenze di layout per modificare la dimensione ed il layout di caselle adiacenti. Utilizzare lo spaziatore di caselle per assicurare la compattezza con altre finestre di dialogo solo in casi particolari, poiché la funzione PDB gestisce automaticamente la spaziatura. Vedere "Tecniche DCL."

La casella spacer non ha attributi supplementari oltre a quelli standard di layout.

Capitolo 18 -- Linguaggio DCL

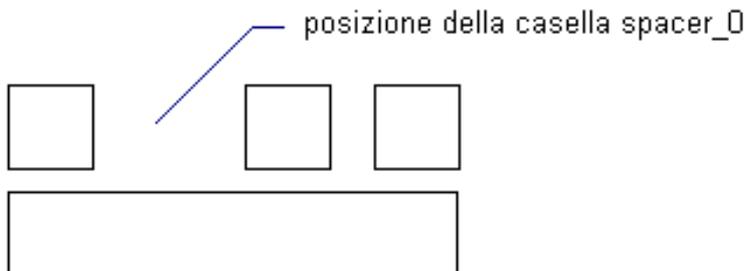
Elenco delle caselle DCL

spacer_0

```
spacer_0;
```

Una casella spacer_0, mostrata nella figura seguente, è uno spaziatore che in genere non ha il valore di larghezza. Indica comunque un punto in un gruppo di caselle dove si desidera maggior spazio se l'intero gruppo deve essere stirato in fase di layout. Se viene assegnato un valore positivo a delle caselle spacer_0 appartenenti allo stesso gruppo, queste verranno disposte a distanze eguali.

La casella spacer_0 è definita nel file *base.dcl*.



Utilizzo della casella spacer_0

Capitolo 18 -- Linguaggio DCL

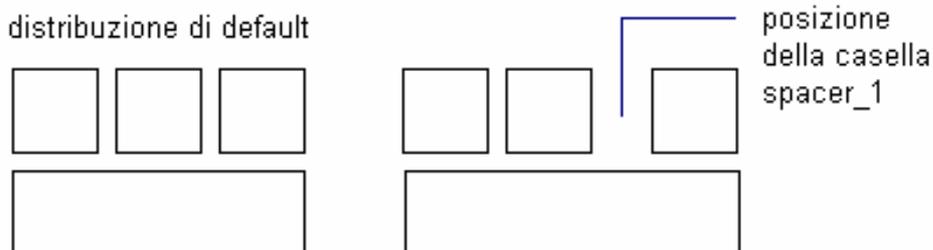
Elenco delle caselle DCL

spacer_1

spacer_1;

La casella spacer_1, mostrata nella figura seguente, è uno spaziatore con valori di larghezza ed altezza pari ad uno. Viene utilizzata per la spaziatura minima ancora riconoscibile dall'utente.

La casella spacer_1 è definita nel file *base.dcl*.



Utilizzo della casella spacer_1

Appendice A -- Codici ASCII

Panoramica

Questa appendice descrive i codici ASCII. Il valore ottale, nella forma $\backslash nnn$, è utile per le costanti di carattere o di stringa. A seconda del sistema utilizzato è possibile che siano definiti i codici addizionali (con valore superiore a 127) del set esteso da 256 caratteri. Alcuni sistemi ridefiniscono anche i codici ASCII meno utilizzati, come quelli dall'1 al 6 e dal 14 al 26. Eseguendo la funzione ASCII di AutoLISP, è possibile visualizzare e stampare su un file denominato *ascii.txt* il set di caratteri utilizzati dal proprio sistema, con i relativi codici in formato decimale ed ottale (come descritto in "Conversione di codici ASCII").

Grafico di conversione dei codici ASCII

Dec.	Ott.	Esa.	Carattere	Dec.	Ott.	Esa.	Carattere
0	000	00	NUL	23	027	17	ETB
1	001	01	SOH	24	030	18	CAN
2	002	02	STX	25	031	19	EM
3	003	03	ETX	26	032	1A	SUB
4	004	04	EOT	27	033	1B	ESC (escape)
5	005	05	ENQ	28	034	1C	FS
6	006	06	ACK	29	035	1D	GS
7	007	07	BEL (bell)	30	036	1E	RS
8	010	08	BS (backspace)	31	037	1F	US
9	011	09	HT	32	040	20	(spazio)
10	012	0A	LF (line-feed)	33	041	21	!
11	013	0B	VT	34	042	22	"
12	014	0C	FF	35	043	23	#
13	015	0D	CR (ritorno a capo)	36	044	24	\$
14	016	0E	SO	37	045	25	%
15	017	0F	SI	38	046	26	&
16	020	10	DLE	39	047	27	'
17	021	11	DC1	40	050	28	(
18	022	12	DC2	41	051	29)
19	023	13	DC3	42	052	2A	*
20	024	14	DC4	43	053	2B	+
21	025	15	NAK	44	054	2C	,
22	026	16	SYN	45	055	2D	-

46	056	2E	.	69	105	45	E
47	057	2F	/	70	106	46	F
48	060	30	0	71	107	47	G
49	061	31	1	72	110	48	H
50	062	32	2	73	111	49	I
51	063	33	3	74	112	4A	J
52	064	34	4	75	113	4B	K
53	065	35	5	76	114	4C	L
54	066	36	6	77	115	4D	M
55	067	37	7	78	116	4E	N
56	070	38	8	79	117	4F	O
57	071	39	9	80	120	50	P
58	072	3A	:	81	121	51	Q
59	073	3B	;	82	122	52	R
60	074	3C	<	83	123	53	S
61	075	3D	=	84	124	54	T
62	076	3E	>	85	125	55	U
63	077	3F	?	86	126	56	V
64	100	40	@	87	127	57	W
65	101	41	A	88	130	58	X
66	102	42	B	89	131	59	Y
67	103	43	C	90	132	5A	Z
68	104	44	D	91	133	5B	[
92	134	5C	\	115	163	73	s
93	135	5D]	116	164	74	t
94	136	5E	^	117	165	75	u
95	137	5F	~	118	166	76	v
96	140	60	¯	119	167	77	w
97	141	61	a	120	170	78	x
98	142	62	b	121	171	79	y
99	143	63	c	122	172	7A	z
100	144	64	d	123	173	7B	{
101	145	65	e	124	174	7C	
102	146	66	f	125	175	7D	}
103	147	67	g	126	176	7E	~
104	150	68	h	127	177	7F	CANC (cancella)
105	151	69	i				
106	152	6A	j				
107	153	6B	k				
108	154	6C	l				
109	155	6D	m				
110	156	6E	n				
111	157	6F	o				
112	160	70	p				
113	161	71	q				
114	162	72	r				

Appendice B -- Formato dei file di interscambio dei disegni

Panoramica

AutoCAD può essere utilizzato come un editor di disegno completo. Tuttavia, in alcune applicazioni è necessario che altri programmi esaminino i disegni AutoCAD o generino i disegni che devono essere visualizzati, modificati o stampati con AutoCAD. In questa appendice vengono descritti cinque tipi di formati di file per l'importazione e l'esportazione dei dati dei disegni.

I file DXF (formato di interscambio dei disegni) consentono l'interscambio dei disegni tra AutoCAD ed altri programmi. I file DXF possono essere in formato ASCII o binario. Poiché i file DXF ASCII sono più comuni dei file in formato binario, in questa appendice il termine *file DXF* viene utilizzato per indicare i file DXF ASCII e il termine *file DXF binari* per indicare il formato binario.

Sia i file DXF ASCII che i file DXF binari contengono una descrizione completa del disegno AutoCAD. Poiché la maggior parte dei dati di un disegno AutoCAD non ha un tipo di oggetto equivalente in altri programmi, viene fornito il formato dei file DXB (Drawing Interchange Binary), che crea una descrizione geometrica più semplice del disegno.

Questa appendice descrive anche il formato di file di diapositiva ed il formato di file di libreria di diapositive.

{button ,JI(','ASCII_DXF_File_Format_al_u05_b')} Formato di file DXF ASCII
 {button ,JI(','Binary_DXF_File_Format_al_u05_b')} Formato dei file DXF binari
 {ewc ,JI(','DXB_Files_al_u05_b')} File DXB
 {button ,JI(','Slide_File_Format_al_u05_b')} Formato dei file di diapositiva
 {button ,JI(','Slide_Library_File_Format_al_u05_b')} Formato dei file di libreria di diapositiva

Appendice B -- Formato dei file di interscambio dei disegni

Formato di file DXF ASCII

Questa sezione descrive il formato dei file DXF ASCII. Le informazioni in essa contenute sono necessarie solo se si scrivono programmi per l'elaborazione dei file DXF o per gestire le informazioni relative alle entità ottenute da applicazioni AutoLISP e ARX.

Appendice B -- Formato dei file di interscambio dei disegni

Formato di file DXF ASCII

Struttura generale dei file

Essenzialmente, un file DXF è costituito da coppie di codici e valori associati. Tali codici, detti *codici di gruppo*, indicano il tipo di valore che segue. Utilizzando queste coppie formate da codice di gruppo e valore, un file DXF è organizzato in sezioni costituite da record, che a loro volta sono costituiti da un codice di gruppo e da una voce di dati. Ciascun codice di gruppo e ciascun valore occupano una riga a parte nel file DXF.

Ciascuna sezione inizia con un codice di gruppo 0 seguito dalla stringa, SECTION. Tale stringa è seguita da un codice di gruppo 2 e da una stringa che indica il nome della sezione (ad esempio, HEADER). Ciascuna sezione è costituita da codici di gruppo e da valori che ne definiscono gli elementi. Una sezione termina con uno 0 seguito dalla stringa ENDSEC.

Potrebbe risultare utile creare un file DXF da un piccolo disegno, stamparlo e tenerlo presente durante la lettura delle informazioni seguenti.

Appendice B -- Formato dei file di interscambio dei disegni

Formato di file DXF ASCII

Struttura generale dei file

Struttura dei file DXF

Segue l'organizzazione generale di un file DXF:

- *Sezione HEADER.* Questa sezione include informazioni generali sul disegno. È costituita dal numero di versione del database AutoCAD e da alcune variabili di sistema. Ciascun parametro contiene un nome di variabile ed il valore ad esso associato.
- *Sezione CLASSES.* Contiene informazioni relative alle classi definite dall'applicazione, che compaiono nelle sezioni BLOCKS, ENTITIES, e OBJECTS del database. Una definizione di classe è stabilita in modo permanente nella gerarchia di classe.
- *Sezione TABLES.* Questa sezione contiene definizioni per le seguenti tabelle di simboli.

APPID (tabella di identificazione delle applicazioni)

BLOCK_RECORD (tabella dei riferimenti ai blocchi)

DIMSTYLE (tabella degli stili di quota)

LAYER (tabella dei layer)

LTYPE (tabella dei tipi di linea)

STYLE (tabella degli stili di testo)

UCS (tabella degli UCS)

VIEW (tabella delle viste)

VPORT (tabella di configurazione delle finestre).

- *Sezione BLOCKS* . Questa sezione contiene la definizione dei blocchi e le entità del disegno che costituiscono ciascun riferimento di un blocco nel disegno.
- *Sezione ENTITIES* . Questa sezione contiene gli oggetti grafici (entità) del disegno, inclusi i riferimenti di un blocco (entità di inserimento).
- *Sezione OBJECTS* . Questa sezione contiene gli oggetti non grafici del disegno. In questa sezione sono contenuti tutti gli oggetti che non sono entità o record della tabella di simboli o tabelle di simboli. Un esempio di voci della sezione OBJECTS sono i dizionari che contengono stili multilinea e gruppi.

Se si usa l'opzione Entità del comando DXFOUT, il file DXF risultante contiene solo la sezione ENTITIES ed il contrassegno EOF. La sezione ENTITIES contiene solo gli oggetti selezionati per l'output. Se si seleziona un'entità di inserimento, la definizione del blocco corrispondente non viene inclusa nel file di output.

Appendice B -- Formato dei file di interscambio dei disegni

Formato di file DXF ASCII

Struttura generale dei file

Codici di gruppo

I codici di gruppo ed i valori ad essi associati definiscono un aspetto specifico di un oggetto o di un'entità. La riga successiva a quella contenente il codice di gruppo contiene il valore ad esso associata. Tale valore può essere una stringa, un numero intero o un valore a virgola mobile, come la coordinata X di un punto. Le righe successive alla seconda riga del gruppo, se sono presenti, sono determinate dalla definizione del gruppo e dai dati ad esso associati.

I codici di gruppo speciali vengono usati come separatori dei file, ad esempio i contrassegni per l'inizio e la fine di sezioni e tabelle e per la fine del file stesso.

Le entità, gli oggetti, le classi, le tabelle, le voci delle tabelle ed i separatori dei file sono introdotti da un codice di gruppo 0 seguito da un nome che descrive il gruppo.

La lunghezza massima della stringa in un file DXF è di 256 caratteri. Se il disegno AutoCAD contiene stringhe di lunghezza superiore, tali stringhe verranno troncate durante l'esecuzione del comando DXFOUT. DXFIN ha esito negativo se il file DXF contiene stringhe che superano questa lunghezza.

Caratteri di controllo ASCII

DXFOUT gestisce i caratteri di controllo ASCII in stringhe di testo aggiungendo al carattere un segno di omissione (^) seguito dalla lettera appropriata. Ad esempio, un ASCII Control-G (BEL, codice decimale 7) viene visualizzato come ^G. Se il testo contiene un segno di omissione, ad esso viene aggiunto uno spazio (^). DXFIN effettua la conversione complementare.

Appendice B -- Formato dei file di interscambio dei disegni

Formato di file DXF ASCII

Esempio di tabella di simboli

Questa sequenza DXF rappresenta tre oggetti completi: la tabella di simboli più due voci.

TABLE *È una tabella, ma di che tipo?*
2
STYLE *Di questo tipo. Eccezione alla regola che il codice 0 dà un'undefinizione completa del tipo.*
5
1C *Gestore di tabella STYLE; uguale per entità ed altri oggetti.*
70
3 *Numero massimo di record di tabella STYLE che seguono (campo precedente alla Release 13).*
1001
APP_X *APP_X ha inserito xdata in una tabella di simboli.*
1040
42.0 *Numero singolo a virgola mobile.*
0
STYLE *Inizio del primo elemento nella tabella di simboli STYLE.*
5
3A *Gestore della prima voce. (Le voci DIMSTYLE visualizzano 105).*
2
ENTRY_1 *Nome di testo della prima voce.*
70
64 *Valori di flag standard.*
40
.4 *Altezza del testo.*
41
1.0 *Fattore di scala della larghezza.*
50
0.0 *Angolo obliquo.*
71
0 *Flag di generazione di testo.*
42
0.4 *Ultima altezza usata.*
3
BUFONTS.TXT *Nome del file del font primario.*
0
STYLE *Inizio della seconda voce. Nessun xdata o reattore costante nella prima voce.*
5
C2 *Gestore della seconda voce.*
2
ENTRY_2 *Nome di testo della seconda voce.*
...
... *Altri campi fino al codice di gruppo 3.*
3
BUFONTS.TXT *Nome del file del font primario ed ultimo tipo di oggetto-gruppo specifico.*
102
{ACAD_REACTORS *Questa voce ha un paio di reattori costanti.*
330
3C2 *ID soft per il primo oggetto reattore.*
330
41B *ID soft per il primo oggetto reattore.*
102
} *Indica la fine della serie di reattori.*
1001
APP_1 *Dati estesi allegati a questa voce.*
1070
45
1001
APP_2
1004
18A5B3EF2C199A
0
UCS *Inizio della tabella degli UCS (e fine dei record e della tabella precedenti).*

Appendice B -- Formato dei file di interscambio dei disegni

Formato di file DXF ASCII

Scrittura di programmi interfaccia DXF

La scrittura di un programma che comunichi con AutoCAD tramite il meccanismo DXF sembra più complessa di quanto non sia in realtà. Con il formato DXF è più semplice ignorare le informazioni non necessarie e leggere quelle importanti.

Il seguente esempio è un programma Microsoft BASIC™ che legge un file DXF ed estrae tutte le entità di linea dal disegno (ignorando le linee che compaiono all'interno dei blocchi). Sullo schermo vengono visualizzati i punti finali di tali linee. Questo programma è un esempio di quanto sia semplice un programma di lettura di DXF.

```

1000 REM
1010 REM Estrazione linee da file DXF
1020 REM
1030 G1% = 0
1040 LINE INPUT "DXF file name: "; A$
1050 OPEN "i", 1, A$ + ".dxf"
1060 REM
1070 REM Ignorare fino a inizio sezione
1080 REM
1090 GOSUB 2000
1100 IF G% <> 0 THEN 1090
1110 IF S$ <> "SECTION" THEN 1090
1120 GOSUB 2000
1130 REM
1140 REM Saltare a meno che non sia sezione ENTITIES
1150 REM
1160 IF S$ <> "ENTITIES" THEN 1090
1170 REM
1180 REM Scansionare fino alla fine della sezione, elaborando LINES
1190 REM
1200 GOSUB 2000
1210 IF G% = 0 AND S$ = "ENDSEC" THEN 2200
1220 IF G% = 0 AND S$ = "LINE" THEN GOSUB 1400 : GOTO 1210
1230 GOTO 1200
1400 REM
1410 REM Accumulare gruppi di entità LINE
1420 REM
1430 GOSUB 2000
1440 IF G% = 10 THEN X1 = X : Y1 = Y : Z1 = Z
1450 IF G% = 11 THEN X2 = X : Y2 = Y : Z2 = Z
1460 IF G% = 0 THEN PRINT "Line from
      (";X1;",";Y1;",";Z1;") to (";X2;",";Y2;",";Z2;")":RETURN
1470 GOTO 1430
2000 REM
2010 REM Lettura codice di gruppo e valore seguente
2020 REM Per le coordinate X, leggere Y e se possibile Z
2030 REM
2040 IF G1% < 0 THEN G% = -G1% : G1% = 0 ELSE INPUT #1, G%
2050 IF G% < 10 OR G% = 999 THEN LINE INPUT #1, S$ : RETURN
2060 IF G% >= 38 AND G% <= 49 THEN INPUT #1, V : RETURN
2080 IF G% >= 50 AND G% <= 59 THEN INPUT #1, A : RETURN
2090 IF G% >= 60 AND G% <= 69 THEN INPUT #1, P% : RETURN
2100 IF G% >= 70 AND G% <= 79 THEN INPUT #1, F% : RETURN
2110 IF G% >= 210 AND G% <= 219 THEN 2130
2115 IF G% >= 1000 THEN LINE INPUT #1, T$ : RETURN
2120 IF G% >= 20 THEN PRINT "Invalid group code";G% : STOP
2130 INPUT #1, X
2140 INPUT #1, G1%
2150 IF G1% <> (G%+10) THEN PRINT "Invalid Y coord code";
      G1% : STOP
2160 INPUT #1, Y
2170 INPUT #1, G1%
2180 IF G1% <> (G%+20) THEN G1% = -G1% ELSE INPUT #1, Z

```

```
2190 RETURN
2200 CLOSE 1
```

La scrittura di un programma che costruisce un file DXF pone problemi diversi. È necessario mantenere la coerenza all'interno del disegno, sebbene con AutoCAD sia possibile omettere molte voci di un file DXF ed ottenere comunque un disegno utilizzabile. È possibile omettere completamente la sezione HEADER, se non si impostano le variabili di intestazione. È possibile omettere le tabelle della sezione TABLES, se non è necessario effettuare delle immissioni e l'intera sezione TABLES può essere eliminata se contiene dei dati che non sono richiesti. Se si definiscono dei tipi di linea nella tabella LTYPE, essa deve comparire prima della tabella LAYER. Se nel disegno non viene usata alcuna definizione di blocco, la sezione BLOCKS può essere omessa. Nel caso in cui la sezione BLOCKS è presente, deve comparire prima della sezione ENTITIES. All'interno della sezione ENTITIES, è possibile fare riferimento ai nomi dei layer anche se non sono stati definiti nella tabella LAYER. Tali layer vengono creati automaticamente con il colore 7 ed il tipo di linea CONTINUOUS. La voce EOF deve essere inserita alla fine del file.

Il seguente programma Microsoft BASIC costruisce un file DXF che rappresenta un poligono con il numero di lati, il punto di origine a sinistra e la lunghezza dei lati specificati. Questo programma fornisce solo la sezione ENTITIES del file DXF ed inserisce tutte le voci generate nel layer di default 0. Poiché il programma non crea l'intestazione del disegno, i limiti, l'estensione e la vista corrente del disegno non saranno validi dopo l'esecuzione di un comando DXFIN SUL DISEGNO GENERATO. È POSSIBILE ESEGUIRE UNO ZOOM dell'estensione per fare in modo che il disegno generato riempi lo schermo e quindi regolare i limiti manualmente.

```
1000 REM
1010 REM Generatore di poligoni
1020 REM
1030 LINE INPUT "Nome del file (DXF) disegno: "; A$
1040 OPEN "o", 1, A$ + ".dxf"
1050 PRINT #1, 0
1060 PRINT #1, "SECTION"
1070 PRINT #1, 2
1080 PRINT #1, "ENTITIES"
1090 PI = ATN(1) * 4
1100 INPUT "Numero di lati per poligono: "; S%
1110 INPUT "Punto iniziale (X,Y): "; X, Y
1120 INPUT "Lato poligono: "; D
1130 A1 = (2 * PI) / S%
1140 A = PI / 2
1150 FOR I% = 1 TO S%
1160 PRINT #1, 0
1170 PRINT #1, "LINE"
1180 PRINT #1, 8
1190 PRINT #1, "0"
1200 PRINT #1, 10
1210 PRINT #1, X
1220 PRINT #1, 20
1230 PRINT #1, Y
1240 PRINT #1, 30
1250 PRINT #1, 0.0
1260 NX = D * COS(A) + X
1270 NY = D * SIN(A) + Y
1280 PRINT #1, 11
1290 PRINT #1, NX
1300 PRINT #1, 21
1310 PRINT #1, NY
1320 PRINT #1, 31
1330 PRINT #1, 0.0
1340 X = NX
1350 Y = NY
1360 A = A + A1
1370 NEXT I%
1380 PRINT #1, 0
1390 PRINT #1, "ENDSEC"
1400 PRINT #1, 0
1410 PRINT #1, "EOF"
1420 CLOSE 1
```

Se sulla linea in cui sono previsti i dati compare una voce formattata in modo appropriato, DXFIN la accetta. Ovviamente, le voci di stringa non devono essere precedute da spazi a meno che essi non facciano parte della stringa stessa. Il programma sfrutta questa flessibilità nel formato di input e non genera un file esattamente uguale ad un file generato da AutoCAD.

In caso di errore nell'uso di DXFIN TO per il caricamento, AutoCAD notifica l'errore con un messaggio che indica sia la natura dell'errore che l'ultima linea del file *dxf* elaborata prima che fosse rilevato l'errore. Tale linea potrebbe anche non essere quella su cui

si è verificato l'errore, specialmente nel caso di errori quali l'omissione di gruppi richiesti.

Appendice B -- Formato dei file di interscambio dei disegni

Formato dei file DXF binari

Il formato dei file DXFASCII è una rappresentazione completa di un disegno AutoCAD in un formato di testo ASCII e può essere facilmente elaborato da altri programmi. Inoltre, AutoCAD può produrre o leggere un formato binario del file DXF completo ed accettare immissioni limitate in un altro formato di file binario.

Il comando DXFOUT fornisce un'opzione Binario che scrive file *dxf* binari. Tale file contiene tutte le informazioni presenti in un file DXF ASCII, ma in un formato più compatto che solitamente occupa il 25 per cento di spazio in meno e può essere letto e scritto più velocemente (normalmente, cinque volte più velocemente) da AutoCAD. A differenza dei file *dxf* ASCII, che comportano un compromesso tra dimensioni e accuratezza alla virgola mobile, i file *dxf* binari conservano tutta l'accuratezza nel database del disegno. La Release 10 di AutoCAD è stata la prima versione a supportare questo formato di file DXF; questo formato non può essere letto da versioni precedenti.

Un file DXF binario inizia con una sentinella di 22 byte costituita da quanto segue:

```
AutoCAD Binary DXF<CR><LF><SUB><NULL>
```

La sentinella è seguita da coppie (gruppo, valore) come quelle di un file DXF ASCII ma rappresentate in formato binario. Il codice di gruppo è un valore binario a byte singolo ed il valore che lo segue è uno dei seguenti:

- Un numero intero a 2 byte con il byte meno significativo per primo e il byte più significativo per ultimo
- Un numero a virgola mobile a doppia precisione IEEE di 8 byte memorizzato con il byte meno significativo per primo ed il byte più significativo per ultimo
- Una stringa ASCII chiusa da un byte 0 (NULL).

Il tipo di dati che segue un gruppo è determinato dal codice di gruppo in base alle stesse regole usate nella decodifica dei file DXF ASCII. La conversione degli angoli in gradi e delle date nella rappresentazione frazionale della data giuliana viene eseguita sia per i file binari che per i file DXF ASCII. Il gruppo dei commenti, 999, non viene usato nei file DXF binari.

I codici di gruppo dei dati estesi vengono rappresentati nei DXF binari come un byte singolo con valore 255, seguito da un valore intero a 2 byte contenente il codice di gruppo effettivo, seguito dal valore effettivo.

I valori lunghi di dati estesi (codice di gruppo 1071) occupano 4 byte di dati. I frammenti binari di dati estesi (codice di gruppo 1004) sono rappresentati come una lunghezza di un numero intero senza segno a byte singolo, seguito dal numero di byte di dati frammentari specificato. Ad esempio, per trasferire un gruppo lungo di dati estesi comparirebbero i seguenti valori, che occupano rispettivamente 1, 2 e 4 byte.

255	<i>Codice di gruppo Escape</i>
1071	<i>Codice di gruppo True</i>
999999	<i>Valore per il codice di gruppo 1071</i>

DXFOUT scrive i file DXF binari con lo stesso tipo di file (*.dxf*) dei file DXF ASCII. Il comando DXFIN riconosce automaticamente un file binario tramite la stringa sentinella. Non è necessario identificarlo manualmente come file binario.

Nel caso in cui DXFIN rilevi un errore in un file DXF binario, ne notifica l'indirizzo di byte all'interno del file in cui è stato rilevato l'errore.

Appendice B -- Formato dei file di interscambio dei disegni

File DXB

I formati di file DXF sono rappresentazioni complete di un disegno AutoCAD che può essere scritto e letto da AutoCAD e da altri programmi. Tuttavia, i programmi eseguiti tramite la funzione dei comandi esterni spesso devono fornire un semplice input geometrico ad AutoCAD. Per questo motivo, AutoCAD supporta un altro formato di file denominato DXB (Drawing Interchange Binary), che ha dei limiti per quanto riguarda le entità che è in grado di rappresentare.

Per caricare un file DXB, immettere il comando DXBIN:

```
dxbin
```

Quando AutoCAD lo richiede, indicare il nome del file che si desidera caricare. Non è necessario includere un file *.dxb* perché è il tipo sottinteso.

Non esiste un comando diretto di AutoCAD per scrivere un file DXB, ma il driver del plotter ADI "Formati di output di file AutoCAD" è in grado di scrivere tale file. Se si desidera creare un file DXB da un disegno AutoCAD, configurare il plotter dei "formati di output dei file" e selezionarne l'opzione di output "File DXB AutoCAD".

Appendice B -- Formato dei file di interscambio dei disegni

File DXB

Formato di file DXB

Queste informazioni sono destinate a programmatori esperti e possono essere modificate senza preavviso.

Il formato di un file DXB è il seguente:

```
Header: "AutoCAD DXB 1.0" CR LF ^Z NUL      (19 byte)
Data:      ...Zero o più record di dati...
Terminator: NUL      (1 byte)
```

Ciascun record di dati inizia con un byte singolo di identificazione del tipo di record, seguito dagli elementi dei dati. Gli elementi dei dati hanno vari formati di rappresentazione e codifica. Nelle seguenti descrizioni dei codici lettera, ciascun elemento dei dati è preceduto da una lettera e da un trattino.

w-

Numero intero a 16 bit, con inversione di byte nello stile standard 80x86 (il byte meno significativo per primo e il byte più significativo per secondo).

f-

Valore a virgola mobile a 64 bit IEEE memorizzato con il byte meno significativo per primo e il byte più significativo per ultimo (come in 80x87).

l-

Numero intero a 32 bit con inversione di byte nello stile 80x86.

n-

Un numero intero a 16 bit o un numero a virgola mobile, a seconda dell'impostazione più recente dell'elemento dei dati relativo al modo numerico. Il modo numerico come default è impostato su 0, che sta per numeri interi. Se è impostato su 1, tutte le voci n- saranno lette come voci a virgola mobile.

u-

Un numero intero a 32 bit o un numero a virgola mobile, a seconda dell'impostazione del modo numerico più recente. Se è un numero intero a 32 bit, il valore viene scalato moltiplicandolo per 65,536 (2¹⁶). Se è un valore a virgola mobile, non viene effettuata alcuna scalatura.

a-

Un angolo. Se il modo numerico è numero intero, questo è un numero intero a 32 bit che rappresenta un angolo in unità di un milionesimo di grado (intervallo da 0 a 360,000,000). Se è un numero a virgola mobile, rappresenta i gradi.

Nella seguente tabella, le lunghezze includono il byte relativo al tipo di elemento ed è sottinteso che il modo numerico sia impostato su 0 (modo numero intero). Se il modo numerico è a virgola mobile, aggiungere 6 byte alla lunghezza di ciascun elemento n-presente e 4 byte a ciascun elemento a- o u-.

Lunghezza in byte per tipo di elemento

Tipo di elemento	Codice (decimale)	Elementi dei dati	Lunghezza (in byte)
Linea	1	n-fromx n-fromy n-tox n-toy n-fromx n-fromy n-fromz	13

Punto	2	<i>n-tox n-toy n-toz</i>	5
Cerchio	3	<i>n-x n-y</i>	7
Arco	8	<i>n-ctrx n-ctry n-rad</i>	19
Traccia	9	<i>a-starta a-enda</i> <i>n-x1 n-y1 n-x2 n-y2</i>	17
Solido	11	<i>n-x3 n-y3 n-x4 n-y4</i> <i>n-x1 n-y1 n-x2 n-y2</i>	17
Fine seq	17	<i>n-x3 n-y3 n-x4 n-y4</i> (nessuno)	1
Polilinea	19	<i>w-closureflag</i>	3
Vertice	20	<i>n-x n-y</i>	5
3Dface	22	<i>n-x1 n-y1 n-z1</i> <i>n-x2 n-y2 n-z2</i> <i>n-x3 n-y3 n-z3</i> <i>n-x4 n-y4 n-z4</i>	25
Fattore di scala	128	<i>f-scalefac</i>	9
Nuovo layer	129	"nomelayer" NUL	lunghezza nomelayer + 2
Estensione di linea	130	<i>n-tox n-toy</i>	5
Estensione di traccia	131	<i>n-x3 n-y3 n-x4 n-y4</i>	9
Base blocco	132	<i>n-bx n-by</i>	5
Curvatura	133	<i>u-2h/d</i>	5
Larghezza	134	<i>n-startw n-endw</i>	5

L'elemento estensione di linea estende l'ultima linea o estensione della linea dal proprio *its* Al punto ad un nuovo Al punto.

L'elemento estensione di traccia estende l'ultimo solido di traccia o estensione di traccia dalla propria linea finale *x3,y3-x4,y4* ad una nuova linea *x3,y3-x4,y4*.

Il fattore di scala è un valore a virgola mobile per il quale vengono moltiplicate tutte le coordinate intere in modo da ottenere le coordinate a virgola mobile usate dalle entità. Il fattore di scala iniziale durante la lettura di un file è 1.0. L'elemento nuovo layer crea un layer se non ne esiste nessuno, assegna al nuovo layer gli stessi valori di default del comando LAYER Nuovo ed imposta tale layer come quello corrente per le entità successive. Alla fine del caricamento del file DXB, viene ripristinato il layer attivo prima che venisse eseguito il comando.

L'elemento base del blocco specifica il punto base (di origine) di un blocco creato. La base del blocco deve essere definita prima che venga rilevato il primo record di entità. Se il file DXB non definisce un blocco, questa specifica viene ignorata.

Una polilinea è costituita da segmenti retti di larghezza costante che collegano i vertici, con le eccezioni costituite dagli elementi curvatura e larghezza. Il flag di chiusura dovrebbe essere 0 o 1; se è 1, esiste un segmento implicito dall'ultimo vertice (subito prima di Fine seq) al primo vertice.

Un elemento curvatura, presente tra due elementi vertice (oppure dopo l'ultimo vertice di una polilinea chiusa), indica che i due vertici sono connessi da un arco piuttosto che da un segmento retto. Se la distanza tra i vertici è di lunghezza *d* e la distanza perpendicolare dal punto intermedio di tale segmento all'arco è *h*, la grandezza della curvatura è $(2 * h / d)$. Il segno è negativo se l'arco si estende dal primo vertice al secondo in senso orario. Quindi un semicerchio ha una curvatura di 1 (o -1). Se il modo numerico è 0 (intero), gli elementi curvatura sono scalati di 2¹⁶. Se il modo numerico è stato impostato su virgola mobile, il valore a virgola mobile fornito è $2 * h / d$ (non scalato).

L'elemento larghezza indica la larghezza iniziale e finale del segmento (retto o curvo) che connette i due vertici. Questa larghezza resta attiva fino al successivo elemento larghezza o fine seq. Se esiste un elemento larghezza tra l'elemento polilinea ed il primo vertice, esso viene memorizzato come larghezza di default della polilinea. Questo consente di risparmiare una considerevole quantità di spazio di database, se la polilinea ha diversi segmenti di questa larghezza.

L'elemento modo numerico controlla il modo degli elementi con i tipi descritti nella precedente tabella come *n-*, *a-* o *u-*. Se il valore fornito è 0, tali valori saranno numeri interi, altrimenti sono a virgola mobile.

Per memorizzare l'ultimo Al punto, le linee condividono le stesse celle, quindi è importante *non* mescolare i gruppi di estensioni per le due entità senza un gruppo iniziale prima dell'estensione. Non esiste alcun gruppo di estensioni per Facce 3D, perché non esiste un limite ovvio da cui effettuare l'estensione.

Il gruppo nuovo colore specifica il colore per le entità successive nel file DXB. L'argomento *w-colornum* è nell'intervallo compreso tra 0 e 256. 0 significa colore per blocco, 1-255 sono i colori standard AutoCAD e 256 significa colore per layer. Un colore al di fuori dell'intervallo 0-256 reimposta il colore su quello dell'entità corrente (è possibile eseguire questa operazione deliberatamente e talvolta può risultare piuttosto utile). Il colore dell'entità iniziale del materiale aggiunto da DXBIN è quello dell'entità corrente.

Tutti i punti specificati nel file DXB sono interpretati in base all'UCS corrente al momento dell'esecuzione del comando DXBIN.

Appendice B -- Formato dei file di interscambio dei disegni

Formato dei file di diapositiva

Nota Queste informazioni sono destinate a programmatori esperti e possono essere modificate senza preavviso.

I file di diapositiva AutoCAD sono immagini video scritte tramite il comando GENDIA e lette tramite il comando VISDIA. Questa sezione descrive il formato dei file di diapositiva per i programmatori che desiderano incorporare un supporto per le diapositive AutoCAD nei propri programmi.

Un file di diapositiva è costituito da una intestazione (31 byte) e da uno o più record di dati di lunghezza variabile. Tutte le coordinate e le dimensioni scritte nel file di diapositiva riflettono l'area grafica del dispositivo video da cui la diapositiva è stata creata, con il punto (0,0) posizionato nell'angolo inferiore sinistro dell'area grafica. Per la Release 9 e successive di AutoCAD l'intestazione del file di diapositiva è costituita dai seguenti campi:

Intestazione del file di diapositiva

Campo	Byte	Descrizione
ID stringa	17	"AutoCAD Slide" CR LF ^Z NUL
Type indicator	1	Attualmente impostato su 56 (decimale)
Level indicator	1	Attualmente impostato su 2
High X dot	2	Larghezza dell'area grafica: 1, in pixel
High Y dot	2	Altezza dell'area grafica: 1, in pixel
Aspect ratio	4	Rapporto prospettico dell'area grafica (dimensioni orizzontali/dimensioni verticali in pollici), scalato per 10,000,000. Questo valore è sempre scritto con il byte meno significativo per primo.
Hardware fill	2	0 o 2 (il valore non è importante)
Test number	2	Un numero (1234 esa) usato per determinare se tutti i valori a 2 byte nella diapositiva sono stati scritti con il byte di ordine superiore per primo (CPU Intel 8086) o con il byte di ordine inferiore per primo (CPU Motorola 68000).

I record di dati seguono l'intestazione. Ciascun record di dati inizia con un campo a 2 byte il cui byte di ordine superiore è il tipo di record. La parte restante del record può essere costituita da campi di 1 o 2 byte, come descritto nella seguente tabella. Per determinare se i campi di 2 byte sono scritti con il byte di ordine superiore per primo o con il byte di ordine inferiore per primo, esaminare il campo Test number dell'intestazione descritto in precedenza.

Record di dati del file di diapositiva

Tipo di record (hex)	Byte	Significato	Descrizione
00-7F	8	Vettore	La coordinata da X per un vettore ordinario. Da Y, a X e a Y seguono in questo ordine come valori a 2 byte. Il punto da viene salvato come ultimo punto.
80-FA	--	Non definito	Destinato ad uso futuro.
FB	5	Vettore di sfalsamento	Il byte di ordine inferiore ed i seguenti tre byte specificano i punti finali (da X, da Y, a X, a Y) di un vettore, in termini di sfalsamento (da -128 a +127) dall'ultimo punto salvato. Il punto da regolato viene salvato come ultimo punto per essere usato da vettori seguenti.
FC	2	Fine file	Il byte di ordine inferiore è 00.
FD	6	Riempimento solido	Il byte di ordine inferiore è sempre zero. I seguenti due valori a 2 byte specificano le coordinate X e Y di uno dei vertici di un poligono di cui effettuare il riempimento solido. In una sequenza sono rilevabili da tre a dieci di questi record. Un record Riempimento solido con una coordinata Y negativa indica l'inizio o la fine di una sequenza di riempimento. Nel record iniziale, la coordinata X indica il numero di record vertice che seguono.
FE	3	Vettore di punto finale comune	Questo è un vettore che inizia all'ultimo punto. Il byte di ordine inferiore ed il byte seguente specificano a X e a Y in termini di sfalsamento (da -128 a +127) dall'ultimo punto salvato. L'Al punto regolato viene salvato come ultimo punto per l'uso da parte di vettori seguenti.
FF	2	Nuovo colore	I vettori seguenti devono essere disegnati usando

il numero del colore indicato dal byte di ordine inferiore.

Se una diapositiva contiene vettori, il primo record di dati sarà il record Nuovo colore. L'ordine dei vettori in una diapositiva e l'ordine dei punti finali di tali vettori può variare.

Ad esempio, segue un dump esadecimale annotato di un file di diapositiva semplice creato su un PC/AT IBM con un Adattatore grafico avanzato IBM. La diapositiva è costituita da una linea diagonale bianca dall'angolo inferiore sinistro all'angolo superiore destro dell'area grafica e da un piccolo rettangolo rosso posto nell'angolo inferiore sinistro.

41 75 74 6F 43 41	<i>ID stringa ("Diapositiva AutoCAD" CR LF ^Z NUL)</i>
44 20 53 6C 69 64	
65 0D 0A 1A 00	
56	<i>Type indicator (56)</i>
02	<i>Level indicator (2)</i>
3C 02	<i>High X dot (572)</i>
24 01	<i>High Y dot (292)</i>
0B 80 DF 00	<i>Aspect ratio (14,647,307 / 10,000,000 = 1.46)</i>
02 00	<i>Hardware fill (2)</i>
34 12	<i>Test number (1234 hex)</i>
07 FF	<i>Nuovo colore (7 = bianco)</i>
3C 02 24 01 00 00 00 00	<i>Vettore da 572,292 a 0,0. 572,292 diventa l'"ultimo" punto</i>
3 FF	<i>Nuovo colore (3 = bianco)</i>
0F 00 32 00 0F 00 13 00	<i>Vettore da 15,50 a 15,19. \x1115,50 diventa l'"ultimo" punto</i>
01 FF	<i>Nuovo colore (1 = rosso)</i>
12 FB E7 12 CE	<i>Vettore di sfalsamento da 15+18,50-25 (33,25) a 5+18,50-50 (33,0). 33,25 diventa l'"ultimo" punto</i>
DF FE 00	<i>Vettore di punto finale comune da 33,25 a 33-33,25+0 (0,25). 0,25 diventa l'"ultimo" punto</i>
00 FE E7	<i>Vettore di punto finale comune da (0,25) a 0+0,25-25 (0,0). 0,0 diventa l'"ultimo" punto</i>
21 FE 00	<i>Vettore di punto finale comune da (0,0) a 0+33,0+0 (33,0). 33,0 diventa l'"ultimo" punto</i>
00 FC	<i>Fine del file</i>

Appendice B -- Formato dei file di interscambio dei disegni

Formato dei file di diapositiva

Intestazione diapositive per versioni precedenti

Il formato di diapositiva descritto nella sezione precedente viene prodotto dalla Release 9 e successive di AutoCAD ed è trasportabile in tutti i computer in cui è in esecuzione la Release 9 o successiva di AutoCAD. Le versioni precedenti di AutoCAD, oltre a AutoShade 1.0 e AutoSketch 1.02, producono diapositive con un'intestazione leggermente diversa, come illustrato nella seguente tabella.

Formato delle intestazioni di file di diapositiva precedenti

Campo	Byte	Descrizione
ID string	17	"AutoCAD Slide" CR LF ^Z NUL
Type indicator	1	56 (decimale)
Level indicator	1	1 (vecchio formato)
High X dot	2	Larghezza dell'area grafica: 1, in pixel
High Y dot	2	Altezza dell'area grafica: 1, in pixel
Aspect ratio	8	Rapporto prospettico dell'area grafica (dimensioni orizzontale/dimensioni verticali in pollici), scritto come un numero a virgola mobile
Hardware fill	2	0 o 2 (il valore non è importante)
Filler byte	1	Non utilizzato

Notare che l'intestazione del formato precedente non contiene un campo Test number. Il valore del rapporto prospettico a virgola mobile e tutti i numeri interi a 2 byte sono scritti in formato conforme alla CPU utilizzata per creare il file (per CPU 8086, doppia precisione IEEE e byte di ordine inferiore per primo). I file di diapositiva di formato precedente non sono trasportabili tra i diversi tipi di apparecchi, ma possono essere letti da qualsiasi versione di AutoCAD in esecuzione sullo stesso tipo di CPU di quella con cui è stata creata la diapositiva.

Appendice B -- Formato dei file di interscambio dei disegni

Formato dei file di libreria di diapositiva

Questa sezione illustra il formato delle librerie di diapositive di AutoCAD (Release 9 e successive) per i programmatori che desiderano incorporare il supporto per le librerie di diapositive nei propri programmi.

Il formato generale di una libreria di diapositive è il seguente:

```
"AutoCAD Slide Library 1.0" CR LF ^Z NUL NUL NUL NUL Intestazione (32 byte)
Una o più voci di directory di diapositive (36 byte ciascuna)
Una o più diapositive (lunghezza variabile)
```

Le voci delle directory di diapositive hanno il seguente formato:

```
Nome diapositiva (chiusa da NUL) (32 byte)
Indirizzo diapositiva all'interno del file di libreria (4 byte)
```

L'indirizzo della diapositiva è sempre scritto con il byte di ordine inferiore per primo. Ciascuna diapositiva a cui punta la directory è un file di diapositiva completo, come descritto nella sezione precedente. La fine della directory di diapositive è indicata da una voce con un nome di diapositiva nullo (il primo byte è NUL). Una libreria di diapositive può contenere un misto di diapositive di formato precedente e di formato attuale.

Appendice C -- Codici di gruppo DXF

Panoramica

Questo capitolo descrive i codici di gruppo DXF rilevati da un'applicazione AutoLISP, ADS o ARX. La prima sezione contiene informazioni generali sui codici di gruppo DXF; tali codici sono elencati in ordine numerico ed ordinati per tipo di oggetto. Per le descrizioni delle funzioni AutoLISP che utilizzano i codici di gruppo, vedere il capitolo 9, "Funzioni dei gruppi di selezione, oggetti e tabelle dei simboli". Le altre sezioni contengono la descrizione dei codici di gruppo nell'ordine in cui si trovano nei file DXF.

Argomenti di questo capitolo

```
{ewc ,JI(','General_DXF_Conventions_al_u05_c')} Convenzioni DXF generali
{button ,JI(','HEADER_Section_al_u05_c')} Sezione HEADER
{ewc ,JI(','CLASSES_Section_al_u05_c')} Sezione CLASSES
{button ,JI(','TABLES_Section_al_u05_c')} Sezione TABLES
{ewc ,JI(','BLOCKS_Section_al_u05_c')} Sezione BLOCKS
{ewc ,JI(','ENTITIES_Section_al_u05_c')} Sezione ENTITIES
{ewc ,JI(','OBJECTS_Section_al_u05_c')} Sezione OBJECTS
{button ,JI(','Advanced_DXF_Issues_al_u05_c')} Problemi DXF avanzati
```

Appendice C -- Codici di gruppo DXF

Convenzioni DXF generali

Il formato DXF è una rappresentazione con dati contrassegnati di tutte le informazioni contenute in un file di disegno di AutoCAD di una versione specifica. Il termine *dati contrassegnati* indica che ciascun elemento costituito da dati di un file è preceduto da un numero intero chiamato *codice di gruppo*. Il valore di un codice di gruppo indica il tipo di elemento seguente e il significato di un elemento di dati di un determinato tipo di oggetto o record. Praticamente tutte le informazioni di un file di disegno specificate dall'utente possono essere rappresentate in formato DXF. Il formato DXF è essenzialmente lo stesso quando viene utilizzato con le applicazioni (AutoLISP, ADS e ARX). Tuttavia per alcuni gruppi di dati vi sono delle piccole differenze.

Per informazioni specifiche sul formato dei file DXF, vedere l'appendice B "Formato dei file di interscambio dei disegni".

I codici di gruppo riportati in questa sezione possono essere validi per i file DXF, per le applicazioni (AutoLISP, ADS o ARX) o per entrambi. Quando la descrizione di un codice è diversa per le applicazioni e per i file DXF (oppure è valida solo in uno dei due casi), essa è preceduta dai seguenti indicatori:

APP

Descrizione specifica per l'applicazione

DXF

Descrizione specifica per i file DXF

Se la descrizione è comune ai file DXF ed alle applicazioni, non è presente alcun indicatore. In caso contrario, è presente l'indicatore appropriato.

Appendice C -- Codici di gruppo DXF

Convenzioni DXF generali

Intervalli dei codici di gruppo

I codici di gruppo definiscono il tipo di valore associato come un valore intero, un numero a virgola mobile o una stringa, in base alla seguente tabella degli intervalli dei codici di gruppo.

Intervalli dei codici di gruppo

Intervallo dei codici	Tipo di valore di gruppo
0 - 9	Stringa (massimo 255 caratteri; un numero inferiore per le stringhe Unicode)
10 - 59	Punto 3D a doppia precisione
60 - 79	Valore intero a 16 bit
90-99	Valore intero a 32 bit.
100	Stringa (massimo 255 caratteri; un numero inferiore per le stringhe Unicode).
102	Stringa (massimo 255 car.atteri; un numero inferiore per le stringhe Unicode)
105	Stringa che rappresenta il valore di un gestore esadecimale (esa).
140 - 147	Valore a virgola mobile scalare a doppia precisione.
170 - 175	Valore intero a 16 bit.
280 - 289	Valore intero a 8 bit.
300 - 309	Stringa di testo libero.
310-319	Stringa che rappresenta il valore esadecimale di un frammento binario.
320-329	Stringa che rappresenta il valore di un gestore esadecimale
330-369	Stringa che rappresenta ID oggetto esadecimale.
999	Commento (stringa).
1000 - 1009	Stringa (massimo 255 caratteri; un numero inferiore per le stringhe Unicode).
1010-1059	Valore a virgola mobile.
1060 - 1069	Valore intero a 16 bit.
1071	Valore intero a 32 bit.

Appendice C -- Codici di gruppo DXF

Convenzioni DXF generali

Codici di gruppo in ordine numerico

La tabella seguente riporta i codici di gruppo o l'intervallo dei codici di gruppo e la spiegazione dei valori dei codici di gruppo. La voce "fisso" nella tabella indica che il codice di gruppo ha sempre la stessa funzione. La funzione dei codici di gruppo che non sono fissi può variare a seconda del contesto.

Codici di gruppo delle entità in ordine numerico

Codice di gruppo	Descrizione
-5	APP: sequenza di reattori costanti.
-4	APP: operatore condizionale (utilizzato solo con ssget).
-3	APP: indicatore di dati estesi (XDATA) (fisso).
-2	APP: riferimento al nome dell'entità (fisso).
-1	APP: nome dell'entità. Viene modificato ogni qualvolta si apre un disegno. Non viene mai salvato (fisso).
0	Stringa di testo che indica il tipo di entità (fisso).
1	Valore di testo primario per un'entità.
2	Nome: etichetta dell'attributo, nome blocco e così via.
3-4	Altri valori di testo o nomi.
5	Gestore di entità. Stringa di testo contenente fino a 16 cifre esadecimali (fisso).
6	Nome del tipo di linea (fisso).
7	Nome dello stile di testo (fisso).
8	Nome del layer (fisso).
9	DXF: identificativo del nome di variabile (utilizzato solo nella sezione HEADER del file DXF).
10	Punto primario. Punto iniziale di una linea o un'entità di testo, centro di un cerchio e così via. DXF: valore X del punto primario (seguito dai codici dei valori Y e Z, 20 e 30)
11-18	APP: punto 3D (elenco di tre numeri reali). Altri punti. DXF: valore X di altri punti (seguito dai codici del valore Y, 21-28 e dai codici del valore Z, 31-38)
20, 30	APP: punto 3D (elenco di tre numeri reali). DXF: valore Y e Z del punto primario.
21-28, 31-37	DXF: valore Y e Z di altri punti.
38	DXF: elevazione dell'entità se il valore è diverso da zero. Presente solo nell'output delle versioni precedenti alla Release 11 di AutoCAD.
39	Spessore dell'entità se il valore è diverso da zero (fisso).
40-48	Valori a virgola mobile (altezza testo, fattori di scala e così via).
48	Scala tipo di linea. Valore scalare a virgola mobile. Il valore di default è definito per tutti i tipi di entità.
49	Valore a virgola mobile ripetuto. Un'unica entità può contenere diversi gruppi 49 per tabelle di lunghezza variabile (come la lunghezza dei trattini nella tabella LTYPE). Un gruppo 7x precede sempre il primo gruppo 49 per specificare la lunghezza della tabella.
50-58	Angoli (output in gradi nei file DXF e in radianti nelle applicazioni AutoLISP e ARX).
60	Visibilità entità. Valore intero. L'assenza o 0 indicano la visibilità; 1 indica la non visibilità.
62	Numero colore (fisso).
66	Flag "seguono entità" (fisso).
67	Spazio (ovvero, spazio modello o spazio carta) (fisso).
68	APP: specifica se la finestra è attiva, ma completamente fuori dallo schermo, se non è attiva o se è chiusa.
69	APP: numero di identificazione della finestra.
70-78	Valori interi, ad esempio calcoli ripetitivi, bit dei flag o modalità
90-99	Valori interi ad 32 bit.
100	Contrassegni di dati di sottoclassi (con nomi di classi derivate, ad esempio una stringa). Necessario per tutti gli oggetti e le classi di entità derivate da un'altra classe concreta per separare i dati definiti da classi diverse nella catena di adozione per lo stesso oggetto. Questo si aggiunge alla richiesta di assegnare un nome DXF per

	<i>ogni diversa classe concreta derivata da ARX ("Contrassegni di sottoclasse").</i>
102	<i>Stringa di controllo seguita da "{<nome libero>" o "}". Simile al codice di gruppo 1002 per dati estesi, tranne che quando la stringa inizia con "{" può essere seguita da una stringa libera la cui interpretazione dipende dall'applicazione. L'unica altra stringa di controllo consentita è "}" come terminatore di un gruppo. Come detto in precedenza, AutoCAD interpreta queste stringhe solo durante le operazioni di controllo; esse vengono utilizzate esclusivamente dall'applicazione.</i>
105	<i>Gestore di oggetto della voce di tabella di simboli DIMVAR.</i>
210	<i>Direzione di estrusione (fisso). DXF: valore X della direzione di estrusione APP: vettore della direzione di estrusione 3D</i>
220, 230	<i>DXF: valore Y e Z della direzione di estrusione.</i>
280-289	<i>Valori interi a 8 bit.</i>
300-309	<i>Stringhe di testo libere.</i>
310-319	<i>Frammenti binari liberi con la stessa rappresentazione e gli stessi limiti dei codici di gruppo 1004: stringhe esadecimali di un massimo di 254 caratteri rappresentano frammenti di dati di un massimo di 127 byte.</i>
320-329	<i>Gestori di oggetti liberi. I valori dei gestori rimangono invariati e non vengono convertiti durante le operazioni INSER e XRIF.</i>
330-339	<i>Gestore di puntatore soft. Puntatori soft liberi che puntano ad altri oggetti nello stesso disegno o file DXF. Convertito durante le operazioni INSER e XRIF.</i>
340-349	<i>Gestore di puntatore hard. Puntatori hard liberi che puntano ad altri oggetti nello stesso disegno o file DXF. Convertito durante le operazioni INSER e XRIF.</i>
350-359	<i>Gestore di proprietà soft. Collegamenti di proprietà soft liberi ad altri oggetti nello stesso disegno o file DXF. Convertito durante le operazioni INSER e XRIF.</i>
360-369	<i>Gestore di proprietà hard. Collegamenti di proprietà hard liberi ad altri oggetti nello stesso disegno o file DXF. Convertito durante le operazioni INSER e XRIF.</i>
999	<i>DXF: Il codice di gruppo 999 indica che la riga che segue è una stringa di commento. Questi gruppi non vengono inclusi da DXFOUT in un file di output DXF; vengono invece osservati da DXFIN, che ignora i commenti. È possibile utilizzare il gruppo 999 per includere commenti in un file DXF che è stato modificato.</i>
1000	<i>Stringa ASCII (lunga fino a 255 byte) in XDATA.</i>
1001	<i>Nome registrato dell'applicazione (stringa ASCII lunga fino a 31 byte) per XDATA (fisso).</i>
1002	<i>Stringa di controllo di dati estesi ("{" o "}").</i>
1003	<i>Nome dei layer dei dati estesi.</i>
1004	<i>Frammento di byte (lungo fino a 127 byte) nei dati estesi.</i>
1005	<i>Gestore di entità nei dati estesi (stringa di testo composta al massimo da 16 cifre esadecimali).</i>
1010	<i>Un punto nei dati estesi DXF: valore X (seguito dai gruppi 1020 e 1030) APP: punto 3D.</i>
1020, 1030	<i>DXF: valori Y e Z di un punto.</i>
1011	<i>Posizione nello spazio globale tridimensionale nei dati estesi DXF: valore valore X (seguito dai gruppi 1021 e 1031) APP: punto 3D.</i>
1021, 1031	<i>DXF: valore Y e Z della posizione nello spazio globale.</i>
1012	<i>Spostamento nello spazio globale tridimensionale nei dati estesi. DXF: valore X (seguito dai gruppi 1022 e 1032). APP: vettore 3D.</i>
1022, 1032	<i>DXF: valore Y e Z dello spostamento nello spazio globale.</i>
1013	<i>Direzione nello spazio globale tridimensionale nei dati estesi. DXF: valore X (seguito dai gruppi 1022 e 1032). APP: vettore 3D.</i>
1023, 1033	<i>DXF: valore Y e Z della direzione nello spazio globale.</i>
1040	<i>Valore a virgola mobile nei dati estesi.</i>
1041	<i>Valore della distanza nei dati estesi.</i>
1042	<i>Fattore di scala nei dati estesi.</i>
1070	<i>Numero intero con segno a 16 bit nei dati estesi.</i>
1071	<i>Numero intero lungo con segno a 32 bit nei dati estesi.</i>

Appendice C -- Codici di gruppo DXF

Convenzioni DXF generali

Codici degli oggetti e delle entità

Nel formato DXF, la definizione degli oggetti differisce da quella delle entità: al contrario delle entità, gli oggetti non hanno una rappresentazione grafica. Ad esempio, i dizionari sono oggetti, non entità. Per indicare le entità viene utilizzata anche l'espressione *oggetti grafici* mentre per gli oggetti viene utilizzata l'espressione *oggetti non grafici*.

Le entità sono presenti sia nella sezione BLOCK che nella sezione ENTITIES del file DXF. L'aspetto delle entità nelle due sezioni è uguale.

Alcuni codici di gruppo che definiscono un'entità sono sempre presenti; altri sono opzionali e sono presenti solo se i loro valori sono diversi dai valori di default.

Nei programmi che leggono i file DXF non si deve presupporre che i gruppi che descrivono un'entità siano nell'ordine qui indicato. La fine di un'entità è indicata dal successivo gruppo 0 che segna l'inizio dell'entità successiva o indica la fine della sezione.

Nota La gestione dei file DXF nelle release future di AutoCAD sarà più semplice se si scrivono i programmi di elaborazione DXF sotto forma di tabella, ignorando i codici di gruppo non definiti e senza fare supposizioni sull'ordine dei codici di gruppo di un'entità. Ogni volta che AutoCAD viene aggiornato, si aggiungono nuovi codici di gruppo alle entità per gestire le nuove funzioni.

Appendice C -- Codici di gruppo DXF

Sezione HEADER

I codici di gruppo descritti in questa sezione sono validi solo per i file DXF.

La sezione HEADER del file DXF contiene le impostazioni delle variabili associate al disegno. Ciascuna variabile è specificata nella sezione di intestazione da un gruppo 9 che specifica il nome della variabile, seguito dai gruppi che ne specificano il valore. Si noti che alcune variabili indicate nell'appendice A, "Variabili di sistema," della *Guida di riferimento dei comandi di AutoCAD* non compaiono nel file DXF.

Le applicazioni possono richiamare i valori di queste variabili con la funzione **getvar**.

Viene di seguito riportato un esempio della sezione HEADER di un file DXF:

```

0                Inizio della sezione HEADER
SECTION
2
HEADER
9                Si ripete per ogni variabile di intestazione
$<variabile>
<codice di gruppo>
<valore>
0                Fine della sezione HEADER
ENDSEC

```

La tabella seguente elenca le variabili che vengono salvate in un file DXF.

Variabili di sistema DXF

Variabile	Codice di gruppo	Descrizione
\$ACADMAINTVER	70	
\$ACADVER	1	Il numero di versione del database dei disegni di AutoCAD: AC1006 = R10, AC1009 = R11 e R12, AC1012 = R13, AC1013 = R14.
\$ANGBASE	50	Direzione dell'angolo 0.
\$ANGDIR	70	1 = angoli in senso orario, 0 = senso antiorario.^p
\$ATTDIA	70	Finestre di dialogo delle voci degli attributi: 1 = on, 0 = off
\$ATTMODE	70	Visibilità degli attributi: 0 = nessuno, 1 = normale, 2 = tutti

\$ATTREQ	70	Richiesta di attributi durante il comando INSER: 1 = on, 0 = off
\$AUNITS	70	Formato delle unità per gli angoli
\$AUPREC	70	Precisione delle unità per gli angoli
\$BLIPMODE	70	Attiva la modalità contrassegno se il valore è diverso da zero
\$CECOLOR	62	Numero colore dell'entità corrente: 0 = DABLOCCO, 256 = DALAYER
\$CELTSCALE	40	Scala del tipo di linea dell'entità corrente
\$CELTYPE	6	Nome del tipo di linea dell'entità, DABLOCCO o DALAYER
\$CHAMFERA	40	Prima distanza di cimatura
\$CHAMFERB	40	Seconda distanza di cimatura
\$CHAMFERC	40	Lunghezza di cimatura
\$CHAMFERD	40	Angolo di cimatura
\$CLAYER	8	Nome del layer corrente
\$CMLJUST	70	Giustificazione multilinea corrente 0=Superiore, 1=Media, 2=Inferiore
\$CMLSCALE	40	Scala multilinea corrente
\$CMLSTYLE	2	Nome dello stile multilinea corrente
\$COORDS	70	Visualizzazione delle coordinate: 0 = statica, 1 = aggiornamento continuo, 2 = formato "d<a"
\$DELOBJ	70	Controlla la cancellazione degli oggetti: 0=eliminati, 1=memorizzati
\$DIMALT	70	Alterna la quotatura delle unità eseguita se il valore è diverso da zero
\$DIMALTD	70	Alterna le posizioni decimali delle unità
\$DIMALTF	40	Alterna il fattore scalare delle unità
\$DIMALTTD	70	Il numero delle posizioni decimali per i valori di tolleranza di una quota delle unità alternative
\$DIMALTTZ	70	Controlla la soppressione degli zeri per i valori di tolleranza alternativi: 0 = Sopprime zero piedi e precisamente zero pollici 1 = Include zero piedi e precisamente zero pollici 2 = Include zero piedi e sopprime zero pollici 3 = Include zero pollici e sopprime zero piedi
\$DIMALTU	70	Formato delle unità per le unità alternative di tutti i membri dei gruppi di stile di quota, ad eccezione di quello angolare: 1 = Scientifico; 2 = Decimale; 3 = Ingegneristico; 4 = Architettonico (impilato); 5 = Frazionario (impilato); 6 = Architettonico; 7 = Frazionario
\$DIMALTZ	70	Controlla la soppressione degli zeri per i valori delle quote relative alle unità alternative: 0 = Sopprime zero piedi e precisamente zero pollici 1 = Include zero piedi e precisamente zero pollici 2 = Include zero piedi e sopprime zero pollici 3 = Include zero pollici e sopprime zero piedi
\$DIMAPOST	1	Suffisso di quotatura alternativa
\$DIMASO	70	1 = crea una quotatura associativa, 0 = disegna singole entità
\$DIMASZ	40	Dimensione delle quotature delle frecce
\$DIMAUNIT	70	Formato dell'angolo per le quote angolari: 0=Gradi decimali, 1=Gradi/minuti/secondi, 2=Gradi centesimali, 3=Radiani, 4=Unità topografiche
\$DIMBLK	1	Nome del blocco della freccia
\$DIMBLK1	1	Nome del blocco della prima freccia
\$DIMBLK2	1	Nome del blocco della seconda freccia
\$DIMCEN	40	Dimensione del centro o delle linee del centro
\$DIMCLRDR	70	Colore della linea di quota: l'intervallo è 0 = DABLOCCO, 256 = DALAYER
\$DIMCLRRE	70	Colore della linea di estensione di quota: l'intervallo è 0 = DABLOCCO, 256 = DALAYER
\$DIMCLRRT	70	Colore del testo di quota: l'intervallo è 0 = DABLOCCO, 256 = DALAYER
\$DIMDEC	70	Numero di posizioni decimali per i valori di tolleranza di una quota delle unità primarie
\$DIMDLE	40	Estensione della linea di quota
\$DIMDLI	40	Incremento della linea di quota
\$DIMEXE	40	Estensione della linea di estensione

\$DIMEXO	40	Sfalsamento della linea d'estensione
\$DIMFIT	70	Posizionamento del testo e delle punte di freccia; i valori validi sono compresi tra 0 e 3 (vedere l'appendice A, "Variabili di sistema," nella Guida di riferimento dei comandi di AutoCAD)
\$DIMGAP	40	Spaziatura della linea di quota
\$DIMJUST	70	Posizione orizzontale del testo di quota: 0=sopra la linea di quota e giustificato al centro tra le linee d'estensione, 1=sopra la linea di quota ed accanto alla prima linea d'estensione, 2=sopra la linea di quota ed accanto alla seconda linea d'estensione, 3=sopra la prima linea di estensione e giustificato al centro rispetto ad essa, 4=sopra la seconda linea di estensione e giustificato al centro rispetto ad essa
\$DIMLFAC	40	Fattore di scala per le misurazioni lineari
\$DIMLIM	70	Genera i limiti di quota se il valore è diverso da zero
\$DIMPOST	1	Suffisso di quotatura generale
\$DIMRND	40	Valore di arrotondamento per le distanze di quotatura
\$DIMSAH	70	Utilizza blocchi di freccia separati se il valore è diverso da zero
\$DIMSCALE	40	Fattore di scala globale per la quotatura
\$DIMSD1	70	Soppressione della prima linea d'estensione: 0=non soppressa, 1=soppressa
\$DIMSD2	70	Soppressione della seconda linea d'estensione: 0=non soppressa, 1=soppressa
\$DIMSE1	70	Sopprime la prima linea d'estensione se il valore è diverso da zero
\$DIMSE2	70	Sopprime la seconda linea d'estensione se il valore è diverso da zero
\$DIMSHO	70	1 = Ridefinisce le quote durante il trascinamento, 0 = trascina l'immagine originale
\$DIMSOXD	70	Elimina le linee di quota esterne alle linee d'estensione se il valore è diverso da zero
\$DIMSTYLE	2	Nome dello stile di quota
\$DIMTAD	70	Testo sopra la linea di quota se il valore è diverso da zero
\$DIMTDEC	70	Numero di posizioni decimali per la visualizzazione dei valori di tolleranza
\$DIMTFAC	40	Fattore di scala per la visualizzazione della tolleranza della quota
\$DIMTIH	70	Il testo viene inserito all'interno orizzontalmente se il valore è diverso da zero
\$DIMTIX	70	Il testo viene forzato all'interno delle linee d'estensione se il valore è diverso da zero
\$DIMTM	40	Tolleranza meno
\$DIMTOFL	70	Se il testo si trova all'esterno delle linee d'estensione, le estensioni della linea vengono forzate tra le estensioni se il valore è diverso da zero
\$DIMTOH	70	Il testo viene inserito all'esterno orizzontalmente se il valore è diverso da zero
\$DIMTOL	70	Vengono generate le tolleranze delle quotature se il valore è diverso da zero
\$DIMTOLJ	70	Giustificazione verticale per i valori di tolleranza: 0=Superiore, 1=Medio, 2=Inferiore
\$DIMTP	40	Tolleranza più
\$DIMTSZ	40	Dimensione dei trattini della quotatura 0 = nessun trattino
\$DIMTVP	40	Posizione verticale del testo
\$DIMTXSTY	7	Stile del testo di quota
\$DIMTXT	40	Altezza del testo di quota
\$DIMTZIN	70	Controlla la soppressione degli zeri per i valori di tolleranza: 0 = Sopprime zero piedi e precisamente zero pollici 1 = Include zero piedi e precisamente zero pollici 2 = Include zero piedi e sopprime zero pollici 3 = Include zero pollici e sopprime zero piedi
\$DIMUNIT	70	Formato delle unità per tutti i membri dei gruppi di stile di quota ad eccezione di quello angolare: 1 = Scientifico; 2 = Decimale; 3 = Ingegneristico; 4 = Architettonico (impilato); 5 = Frazionario (impilato); 6 = Architettonico; 7 = Frazionario

\$DIMUPT	70	Funzionalità del cursore per il testo posizionato dall'utente: 0=controlla solo la posizione della linea di quota, 1=controlla la posizione del testo e della linea di quota
\$DIMZIN	70	Controlla la soppressione degli zeri per i valori delle unità primarie: 0 = Sopprime zero piedi e precisamente zero pollici 1 = Include zero piedi e precisamente zero pollici 2 = Include zero piedi e sopprime zero pollici 3 = Include zero pollici e sopprime zero piedi
\$DISPSILH	70	Controlla la visualizzazione di curve di sagome relative ad oggetti corpo in modalità wireframe: 0=Off, 1=On
\$DRAGMODE	70	0 = off, 1 = on, 2 = automatica
\$DWGCODEPAGE	3	Tabella dei codici del disegno; impostata sulla tabella dei codici del sistema quando viene creato un nuovo disegno, ma altrimenti non memorizzata da AutoCAD
\$ELEVATION	40	Elevazione corrente impostata dal comando ELEV
\$EXTMAX	10, 20, 30	Angolo superiore destro delle estensioni del disegno X, Y e Z (in WCS)
\$EXTMIN	10, 20, 30	Angolo inferiore sinistro delle estensioni del disegno X, Y e Z (in WCS)
\$FILLETRAD	40	Raggio di raccordo
\$FILLMODE	70	Modalità riempimento attivata se il valore è diverso da zero
\$HANDLING	70	Gestore disponibile successivo
\$HANDSEED	5	Gestore disponibile successivo
\$INSBASE	10, 20, 30	Punto base di inserimento impostato dal comando BASE (in WCS)
\$LIMCHECK	70	Diversa da zero se la verifica dei limiti è attiva
\$LIMMAX	10, 20	Limiti del disegno XY nell'angolo superiore destro (in WCS)
\$LIMMIN	10, 20	Limiti del disegno XY nell'angolo inferiore sinistro (in WCS)
\$LTSCALE	40	Scala globale per il tipo di linea
\$LUNITS	70	Formato delle unità per le coordinate e le distanze
\$LUPREC	70	Precisione delle unità per le coordinate e le distanze
\$MAXACTVP	70	Imposta il numero massimo di finestre da rigenerare
\$MENU	1	Nome del file di menu
\$MIRRTEXT	70	Testo speculare se il valore è diverso da zero
\$ORTHOMODE	70	Modalità orto attivata se il valore è diverso da zero
\$OSMODE	70	Modalità correnti di snap ad oggetti
\$PDMODE	70	Modalità di visualizzazione del punto
\$PDSIZE	40	Dimensione di visualizzazione del punto
\$PELEVATION	40	Elevazione dello spazio carta corrente
\$PEXTMAX	10, 20, 30	Estensioni X, Y e Z massime per lo spazio carta
\$PEXTMIN	10, 20, 30	Estensioni X, Y e Z minime per lo spazio carta
\$PICKSTYLE	70	Controlla la selezione del gruppo e la selezione del tratteggio associativo: 0=Nessuna selezione di gruppo o di tratteggio associativo, 1=Selezione di gruppo, 2=Selezione di tratteggio associativo, 3=Selezione di gruppo e di tratteggio associativo
\$PINSBASE	10, 20, 30	
\$PLIMCHECK	70	Verifica dei limiti nello spazio carta quando il valore è diverso da zero
\$PLIMMAX	10, 20	Limiti X ed Y massimi per lo spazio carta
\$PLIMMIN	10, 20	Limiti X ed Y minimi per lo spazio carta
\$PLINEGEN	70	Imposta la generazione dei modelli del tipo di linea intorno ai vertici di una polilinea 2D: 1 = il tipo di linea viene generato in modo continuo intorno ai vertici della polilinea, 0 = ciascun segmento della polilinea inizia e finisce con un trattino
\$PLINEWID	40	Spessore di default della polilinea
\$PROXYGRAPHICS	70	Controlla il salvataggio delle immagini oggetto proxy.
\$PSLTSCALE	70	Controlla la scala del tipo di linea per lo spazio carta: 1 = nessuna scala particolare per il tipo di linea 0 = la scala della finestra determina la scala del tipo di linea
\$PUCSNAME	2	Nome dell'UCS per lo spazio carta corrente

\$PUCSORG	10, 20, 30	Origine dell'UCS per lo spazio carta corrente
\$PUCSXHR	10, 20, 30	Asse X dell'UCS per lo spazio carta corrente
\$PUCSYDIR	10, 20, 30	Asse Y dell'UCS per lo spazio carta corrente
\$QTEXTMODE	70	Modalità testo veloce attivata se il valore è diverso da zero
\$REGENMODE	70	Modalità RIGENAUTO attivata se il valore è diverso da zero
\$SHADEDGE	70	0 = facce ombreggiate, spigoli non evidenziati 1 = facce ombreggiate, spigoli evidenziati in nero 2 = facce non riempite, spigoli del colore dell'entità 3 = facce del colore dell'entità, spigoli in nero
\$SHADEDIF	70	Percentuale luce d'ambiente/luce diffusa, intervallo 1-100, valore di default 70
\$SKETCHINC	40	Incremento del record del comando Schizzo
\$SKPOLY	70	0 = schizzo di linee, 1 = schizzo di polilinee
\$SPLFRAME	70	Visualizzazione del poligono di controllo per spline: 1 = on, 0 = off
\$SPLINESEGS	70	Numero di segmenti di linea per frammento di spline
\$SPLINETYPE	70	Tipo di curva spline per EDITPL Spline
\$SURFTAB1	70	Numero di tabulazioni mesh nella prima direzione
\$SURFTAB2	70	Numero di tabulazioni mesh nella seconda direzione
\$SURFTYPE	70	Tipo di superficie per EDITPL Leviga
\$SURFU	70	Densità della superficie (per EDITPL Leviga) nella direzione M
\$SURFV	70	Densità della superficie (per EDITPL Leviga) nella direzione N
\$TDCREATE	40	Data/ora di creazione del disegno
\$TDINDWG	40	Tempo di modifica cumulativo per questo disegno
\$TDUPDATE	40	Data/ora dell'ultimo aggiornamento del disegno
\$TDUSRTIMER	40	Timer utente del tempo trascorso
\$TEXTSIZE	40	Altezza del testo di default
\$TEXTSTYLE	7	Nome dello stile del testo corrente
\$THICKNESS	40	Spessore corrente impostato dal comando ELEV
\$TILEMODE	70	1 per la compatibilità con le release precedenti, altrimenti 0
\$TRACEWID	40	Larghezza di traccia di default
\$TREEDEPTH	70	Specifica la profondità massima dell'indice per lo spazio.
\$UCSNAME	2	Nome dell'UCS corrente
\$PUCSORG	10, 20, 30	Origine dell'UCS corrente (in WCS)
\$PUCSXHR	10, 20, 30	Direzione dell'asse X dell'UCS corrente (in WCS)
\$PUCSYDIR	10, 20, 30	Direzione dell'asse Y dell'UCS corrente (in WCS)
\$SUNITMODE	70	Impostazione bit inferiore = visualizzazione delle frazioni, di piedi e pollici e di angoli topografici in formato di input
\$USER11 - 5	70	Cinque variabili intere per l'uso di altri sviluppatori
\$USERR1 - 5	40	Cinque variabili reali per l'uso di altri sviluppatori
\$USRTIMER	70	0 = timer off, 1 = timer on
\$VISRETAIN	70	0 = non mantiene le impostazioni della visibilità dipendenti da riferimenti esterni, 1 = mantiene le impostazioni della visibilità dipendenti da riferimenti esterni
\$WORLDVIEW	70	1 = imposta l'UCS su WCS durante VISTAD/PVISTA, 0 = non modifica l'UCS

Le seguenti variabili di intestazione esistevano prima della Release 11 di AutoCAD, ma attualmente hanno impostazioni indipendenti per ciascuna finestra attiva. DXFIN tiene conto di queste variabili quando legge i file DXF, ma se è presente una tabella dei simboli VPORT con voci *ACTIVE (come accade per tutti i file DXF prodotti dalla Release 11 o successiva), i valori della tabella VPORT sostituiscono i valori di queste variabili di intestazione.

Variabili di intestazione VPORT modificate

Variable	Codice di gruppo	Descrizione
\$FASTZOOM	70	Zoom veloce abilitato se il valore è diverso da zero
\$GRIDMODE	70	Modalità griglia attiva se il valore è diverso da zero
\$GRIDUNIT	10, 20	Spaziatura della griglia X ed Y
\$SNAPANG	50	Angolo di rotazione per snap/griglia
\$SNAPBASE	10, 20	Punto base per snap/griglia (in UCS)
\$SNAPISOPAIR	70	Piano assonometrico: 0 = sinistra, 1 = superiore, 2 = destro

\$SNAPMODE	70	Modalità snap attiva se il valore è diverso da zero
\$SNAPSTYLE	70	Stile snap: 0 = standard, 1 = assonometrico
\$SNAPUNIT	10, 20	Spaziatura dello snap X ed Y
\$VIEWCTR	10, 20	Centro XY della vista corrente sullo schermo
\$VIEWDIR	10, 20, 30	Direzione di vista (direzione dalla destinazione in WCS)
\$VIEWSIZE	40	Altezza della vista

L'output delle variabili data/ora (\$TDCREATE e \$TDUPDATE) è espressa in numeri reali nel seguente formato:

<data giuliana>.<Frazione>

Il formato delle variabili del tempo trascorso (\$TDINDWG e \$TDUSRTIMER) è simile a quello seguente:

<numero di giorni>.<Frazione>

Le variabili della data e dell'ora sono descritte nella *Guida di riferimento dei comandi di AutoCAD*.

Appendice C -- Codici di gruppo DXF

Sezione CLASSES

I codici di gruppo descritti in questa sezione si trovano solo nei file DXF.

La sezione CLASSES contiene le informazioni per le classi definite dalle applicazioni che ricorrono nelle sezioni BLOCKS, ENTITIES e OBJECTS del database. Si presuppone che la definizione della classe sia fissata in modo permanente nella gerarchia delle classi. È necessario specificare tutti i campi.

Viene di seguito riportato un esempio di sezione CLASSES di un file DXF:

```

0                Inizio della sezione CLASSES
SECTION
2
CLASSES
9                Si ripete per ciascuna voce
<record dxf classe>
1
<nome classe>
2
<nome app.>
90
<numero ver.>
280
<flag>
281
<flag>
0                Fine della sezione CLASSES
ENDSEC

```

Ciascuna voce della sezione CLASSES contiene i gruppi descritti nella seguente tabella.

Codici di gruppo della sezione CLASSES

Codice di gruppo	Descrizione
0	<i><nome record classe DXF></i> : identifica l'inizio dei record delle sezioni BLOCKS, ENTITIES e OBJECTS. Se due tipi di oggetto usano lo stesso nome DXF, AutoCAD aggiunge a questi dei suffissi numerici univoci finché non vengono distinti nel disegno.
1	<i><nome classe C++></i> : utilizzato per il collegamento al software che definisce il comportamento della classe di oggetti. Sono sempre univoci. Si noti che la stessa classe C++ può avere nomi DXF leggermente diversi in disegni diversi.
2	<i><nome applicazione></i> : riportato nella casella Alert

90	quando una definizione di classe presente in questa sezione non è correntemente caricata. <numero versione classe>: impostato sulla versione della classe caricata l'ultima volta che sono state memorizzate delle voci di questa classe.
280	<flag era-proxy>: impostato su 1 se la classe non era caricata quando è stato creato questo file DXF, altrimenti su 0.
281	<flag entità>: impostato su 1 se la classe deriva dalla classe AcDbEntity e può risiedere nella sezione BLOCKS o ENTITIES. Se il valore è impostato su 0, è possibile che le voci compaiano solo nella sezione OBJECTS.

Appendice C -- Codici di gruppo DXF

Sezione TABLES

I codici di gruppo descritti in questa sezione si trovano nei file DXF e vengono utilizzati dalle applicazioni. La sezione TABLES contiene diverse tabelle, ciascuna delle quali può contenere un numero variabile di voci. Questi codici vengono utilizzati anche dalle applicazioni AutoLISP e ARX negli elenchi di definizione delle entità.

I codici di gruppo riportati in questa sezione possono essere validi per i file DXF, per le applicazioni (AutoLISP, ADS o ARX) o per entrambi. Quando la descrizione di un codice è diversa per le applicazioni e per i file DXF (oppure è valida solo in uno dei due casi), è preceduta dai seguenti indicatori:

APP

Descrizione specifica per l'applicazione

DXF

Descrizione specifica per i file DXF

Se la descrizione è comune ai file DXF ed alle applicazioni, non è presente alcun indicatore. In caso contrario, è presente l'indicatore appropriato. I codici opzionali sono riportati in grigio.

Appendice C -- Codici di gruppo DXF

Sezione TABLES

Tabelle dei simboli nei file DXF

L'ordine delle tabelle può cambiare, ma la tabella LTYPE precede sempre la tabella LAYER. Ciascuna tabella inizia con un gruppo 0 con l'etichetta TABLE. Questo è seguito da un gruppo 2 che identifica una tabella specifica (APPID, DIMSTYLE, LAYER, LTYPE, STYLE, UCS, VIEW, VPORT o BLOCK_RECORD), da un gruppo 5 (un gestore), da un gruppo 100 (contrassegno di sottoclasse AcDbSymbolTable) e da un gruppo 70 che specifica il numero massimo di voci successive nella tabella. I nomi delle tabelle sono in lettere maiuscole. Il gestore DIMSTYLE è un gruppo 105, non un gruppo 5.

Le tabelle di un disegno possono contenere voci cancellate, ma queste non vengono scritte nel file DXF. Di conseguenza, è possibile che le voci di tabella che seguono l'intestazione siano in numero inferiore rispetto a quanto indicato dal gruppo 70, quindi non si deve utilizzare il conteggio del gruppo 70 come indice per leggere la tabella. Questo gruppo viene fornito in modo che un programma che legge i file DXF possa allocare una serie sufficientemente lunga per contenere tutte le voci di tabella che seguono.

Dopo questa intestazione, ripetuta per ciascuna tabella, sono riportate le voci della tabella. Ciascuna voce della tabella è composta da un gruppo 0 che identifica il tipo di voce (uguale al nome della tabella, ad esempio LTYPE o LAYER), un gruppo 2 che indica il nome della voce della tabella, un gruppo 70 che specifica i flag per la voce della tabella (definiti per ciascuna delle tabelle che seguono) e da altri gruppi che indicano il valore della voce della tabella. La fine di ciascuna tabella è indicata da un gruppo 0 con il valore ENDTAB.

Viene di seguito riportato un esempio di sezione TABLES di un file DXF:

```

0                Inizio della sezione TABLES
SECTION
2
TABLES
0                Codici di gruppo comuni della tabella,
TABLE           ripetuti per ciascuna voce della
2                tabella
<tipo tabella>
5
<database>
100
AcDbSymbolTable
70
<n. max voci>

0                Dati della voce della tabella, ripetuti
<tipo tabella>  per ciascuna voce della tabella
.
. <dati>
.
0                Fine della tabella
ENDTAB
0                Fine della sezione TABLES
ENDSEC

```

Sia i record della tabella dei simboli che le tabelle dei simboli sono oggetti del database. Questo implica che, nell'uso prevalente all'interno di AutoCAD, è presente come minimo un gestore, posizionato dopo i codici di gruppo 2 sia per gli oggetti record della tabella dei simboli che per gli oggetti tabella dei simboli.

La tabella DIMSTYLE è il solo tipo di record nel sistema ad avere un codice del gestore 105, poiché utilizzava in precedenza il codice di gruppo 5. Come regola generale, i programmatori non devono preoccuparsi di questa eccezione, a meno che non sia nel contesto della sezione della tabella DIMSTYLE. Questo è il solo contesto in cui si può verificare questa eccezione.

Appendice C -- Codici di gruppo DXF

Sezione TABLES

Codici di gruppo della tabella dei simboli

La tabella seguente mostra i codici di gruppo validi per tutte le tabelle dei simboli.

Codici di gruppo validi per tutte le tabelle dei simboli

Codice di gruppo	Descrizione
-1	APP: nome dell'entità (cambia ogni qualvolta si apre un disegno)
0	Tipo di oggetto (TABLE)
2	Nome della tabella
5	Gestore
100	Contrassegno di sottoclasse (AcDbSymbolTable)
70	Numero massimo di voci della tabella.

Appendice C -- Codici di gruppo DXF

Sezione TABLES

Codici di gruppo comuni per le voci della tabella dei simboli

La tabella seguente mostra i codici di gruppo validi per tutte le voci della tabella dei simboli. I codici opzionali sono riportati in grigio. Quando si fa riferimento alla tabella dei codici di gruppo per tipo di entità, che elenca i codici associati a entità specifiche, tenere

presente che possono esservi contenuti anche i codici riportati di seguito.

Codici di gruppo validi per tutte le voci della tabella dei simboli

Codice di gruppo	<i>Descrizione</i>
-1	<i>APP: nome dell'entità (cambia ogni qualvolta si apre un disegno)</i>
0	<i>Tipo di entità (nome della tabella)</i>
5	<i>Gestore (tutte tranne DIMSTYLE)</i>
105	<i>Gestore (solo tabella DIMSTYLE)</i>
102	<i>Inizio del gruppo definito dall'applicazione "{nome_applicazione". Ad esempio, "{ACAD_REACTORS" indica l'inizio del gruppo di reattori costanti di AutoCAD.</i>
Codici definiti dalle applicazioni	<i>I codici ed i valori dei gruppi 102 sono definiti dalle applicazioni.</i>
102	<i>Fine del gruppo, "}"</i>
100	<i>Contrassegno di sottoclasse (AcDbSymbolTableRecord)</i>

La seguente tabella mostra i codici di gruppo risultanti se sono stati collegati ad un oggetto dei reattori costanti.

Record ACAD_REACTORS

Codice di gruppo	<i>Descrizione</i>
102	<i>"{ACAD_REACTORS" indica l'inizio del gruppo di reattori costanti di AutoCAD</i>
330	<i>ID/gestore puntatore soft per il dizionario di proprietà</i>
102	<i>Fine del gruppo, "}"</i>

La tabella seguente mostra i codici di gruppo risultanti se è stato collegato ad un oggetto un dizionario di estensione.

Record ACAD_XDICTIONARY

Codice di gruppo	<i>Descrizione</i>
102	<i>"{ACAD_XDICTIONARY" indica l'inizio di un gruppo di dizionari estesi.</i>
360	<i>ID/gestore proprietario hard del dizionario di proprietà.</i>
102	<i>Fine del gruppo, "}"</i>

Nella tabella seguente sono descritti i valori codificati in bit dei flag comuni del gruppo 70. Altri valori del gruppo 70 validi per le voci delle tabelle LAYER, STYLE e VIEW sono descritti in tali tabelle.

Valori codificati in bit del gruppo 70 validi per tutte le voci delle tabelle

Valori codificati in bit	<i>Descrizione</i>
16	<i>Se impostato, la voce della tabella dipende esternamente da un xrif.</i>
32	<i>Se sono impostati sia questo bit che il bit 16, l'xrif esternamente dipendente è stato risolto.</i>
64	<i>Se impostato, l'ultima volta che il disegno è stato modificato, almeno un'entità del disegno ha fatto riferimento alla voce della tabella. Questo flag viene utilizzato dai comandi di AutoCAD. Può essere ignorato dalla maggior parte dei programmi che leggono i file DXF e non deve essere impostato dai programmi che scrivono i file DXF.</i>

Appendice C -- Codici di gruppo DXF

Sezione TABLES

APPID

I seguenti codici di gruppo sono validi per le voci della tabella dei simboli APPID.

Codici di gruppo APPID

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbRegAppTableRecord)</i>
2	<i>Nome dell'applicazione specificato dall'utente (per i dati estesi). Queste voci della tabella contengono una serie di nomi per tutte le</i>

- 70 *applicazioni registrate.*
Valori flag standard (vedere "Codici di gruppo comuni per le voci della tabella dei simboli"):
1 = Se impostato, i dati estesi associati a questa APPID non vengono scritti quando viene eseguito il comando SALVACOMER12.

Nota Il formato dei dati ASE non è cambiato tra le versioni R13 e R14 di AutoCAD, quindi il nome dell'applicazione ASE (ACADASER13) non è cambiato.

Appendice C -- Codici di gruppo DXF

Sezione TABLES

BLOCK_RECORD

I seguenti codici di gruppo sono validi per le voci della tabella dei simboli BLOCK_RECORD.

Codici di gruppo BLOCK_RECORD

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbBlockTableRecord)</i>
2	Nome blocco

Appendice C -- Codici di gruppo DXF

Sezione TABLES

DIMSTYLE

I seguenti codici di gruppo sono validi per le voci della tabella dei simboli DIMSTYLE. Le variabili di sistema DIMSTYLE sono descritte nell'appendice A "Variabili di sistema," nella *Guida di riferimento dei comandi di AutoCAD*.

Codici di gruppo DIMSTYLE

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbDimStyleTableRecord)</i>
2	<i>Nome dello stile di quota</i>
70	<i>Valori flag standard (vedere "Codici di gruppo comuni per le voci della tabella dei simboli"):</i>
3	<i>DIMPOST</i>
4	<i>DIMAPOST</i>
5	<i>DIMBLK</i>
6	<i>DIMBLK1</i>
7	<i>DIMBLK2</i>
40	<i>DIMSCALE</i>
41	<i>DIMASZ</i>
42	<i>DIMEXO</i>
43	<i>DIMDLI</i>
44	<i>DIMEXE</i>
45	<i>DIMRND</i>
46	<i>DIMDLE</i>
47	<i>DIMTP</i>
48	<i>DIMTM</i>
140	<i>DIMTXT</i>
141	<i>DIMCEN</i>
142	<i>DIMTSZ</i>
143	<i>DIMALTF</i>
144	<i>DIMLFAC</i>
145	<i>DIMTVP</i>
146	<i>DIMTFAC</i>
147	<i>DIMGAP</i>

71	<i>DIMTOL</i>
72	<i>DIMLIM</i>
73	<i>DIMTIH</i>
74	<i>DIMTOH</i>
75	<i>DIMSE1</i>
76	<i>DIMSE2</i>
77	<i>DIMTAD</i>
78	<i>DIMZIN</i>
170	<i>DIMALT</i>
171	<i>DIMALTD</i>
172	<i>DIMTOFL</i>
173	<i>DIMSAH</i>
174	<i>DIMTIX</i>
175	<i>DIMSOXD</i>
176	<i>DIMDLRD</i>
177	<i>DIMCLRE</i>
178	<i>DIMCLRT</i>
270	<i>DIMUNIT</i>
271	<i>DIMDEC</i>
272	<i>DIMTDEC</i>
273	<i>DIMALTU</i>
274	<i>DIMALTTD</i>
340	<i>Gestore dell'oggetto STYLE di riferimento (utilizzato invece di memorizzare il valore DIMTXSTY)</i>
275	<i>DIMAUNIT</i>
280	<i>DIMJUST</i>
281	<i>DIMSD1</i>
282	<i>DIMSD2</i>
283	<i>DIMTOLJ</i>
284	<i>DIMTZIN</i>
285	<i>DIMALTZ</i>
286	<i>DIMALTTZ</i>
287	<i>DIMFIT</i>
288	<i>DIMUPT</i>

Appendice C -- Codici di gruppo DXF

Sezione TABLES

LAYER

I seguenti codici di gruppo sono validi per le voci della tabella dei simboli LAYER.

Codici di gruppo LAYER

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbSymbolTableRecord)</i>
2	<i>Nome layer</i>
70	<i>Flag standard. (vedere "Codici di gruppo comuni per le voci della tabella dei simboli"). Oltre ai flag standard, anche i seguenti valori sono validi per i layer (codificati in bit)</i> <i>1 = Il layer è congelato, altrimenti il layer è scongelato</i> <i>2 = Per default, nelle nuove finestre il layer è congelato</i> <i>4 = Il layer è bloccato</i>
62	<i>Numero di colore (se negativo, il layer è Off)</i>
6	<i>Nome del tipo di linea</i>

Vengono prodotti dei layer dipendenti da xrif durante il comando DXFOUT. Per questi layer, il nome del tipo di linea associato nel file DXF è sempre CONTINUOUS.

Appendice C -- Codici di gruppo DXF

Sezione TABLES

LTYPE

I seguenti codici di gruppo sono validi per le voci della tabella dei simboli LTYPE.

Codici di gruppo LTYPE

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (AcDbLinetypeTableRecord)
2	Nome del tipo di linea
70	Flag standard. (vedere "Codici di gruppo comuni per le voci della tabella dei simboli").
3	Testo descrittivo per il tipo di linea.
72	Codice di allineamento; il valore è sempre 65, il codice ASCII per A.
73	Il numero di elementi del tipo di linea.
40	Lunghezza totale del modello.
49	Lunghezza del trattino, del punto o dello spazio (una voce per elemento).
74	Tipo di elemento per tipo di linea complesso (uno per elemento): 0 = non complesso, 2 = stringa di testo incorporata, 4 = forma incorporata
75	Codice di forma complessa (uno per elemento se il codice 74 > 0) 1 se il codice 74 = 2
340	Puntatore all'oggetto STYLE (uno per elemento se il codice 74 > 0)
46	S= valore di scala (opzionale). Possono esistere più voci.
50	R= valore di rotazione (opzionale). Possono esistere più voci.
44	X= valore di sfalsamento x (opzionale). Possono esistere più voci.
45	Y= valore di sfalsamento y (opzionale). Possono esistere più voci.
9	Stringa di testo (una per elemento se il codice 74 = 2)

I codici di gruppo 74, 75, 340, 46, 50, 44, 45 e 9 non vengono prodotti dalla funzione **tblsearch** o **tblnext**. Per richiamare questi valori in un'applicazione, è necessario utilizzare **tblobjname**.

Appendice C -- Codici di gruppo DXF

Sezione TABLES

STYLE

I codici di gruppo seguenti sono validi per le voci della tabella dei simboli STYLE.

Codici di gruppo STYLE

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (AcDbTextStyleTableRecord)
2	Nome dello stile
70	Valori di flag standard, ovvero valori codificati in bit (vedere "Codici di gruppo comuni per le voci della tabella dei simboli"): 1=se impostata, questa voce descrive una forma 4=Testo verticale
40	Altezza del testo fissa; 0 se non è fissa
41	Fattore di larghezza
50	Angolo obliquo
71	Indicatori di generazione testo 2=Testo rovesciato (speculare rispetto ad X) 4=Testo capovolto (speculare rispetto ad Y)
42	Ultima altezza utilizzata
3	Nome file del font primario
4	Nome file dei big font (se non sono disponibili, la stringa è vuota).

Un elemento della tabella STYLE viene utilizzato anche per registrare le richieste del comando CARICA del file di forma. In questo

caso, il primo bit (1) è impostato nei flag del gruppo 70 e solo il gruppo 3 (nome del file di forma) è significativo (tuttavia, si ha comunque l'output di tutti gli altri gruppi).

Appendice C -- Codici di gruppo DXF

Sezione TABLES

UCS

I seguenti codici di gruppo sono validi per le voci della tabella dei simboli UCS.

Codici di gruppo UCS

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (<i>AcDbUCSTableRecord</i>)
2	Nome UCS
70	Valori flag standard (vedere "Codici di gruppo comuni per le voci della tabella dei simboli"):
10	Origine (in WCS). DXF: valori X; APP: punto 3D
20, 30	DXF: valori Y e Z dell'origine (in WCS)
11	Vettore di direzione dell'asse X (in WCS). DXF: valori X; APP: vettore 3D
21, 31	DXF: valori Y e Z della direzione dell'asse X (in WCS)
12	Direzione dell'asse Y (in WCS). DXF: valori X; APP: vettore 3D
22, 32	DXF: valori Y e Z della direzione dell'asse Y (in WCS)

Appendice C -- Codici di gruppo DXF

Sezione TABLES

VIEW

I seguenti codici di gruppo sono validi per le voci della tabella dei simboli VIEW.

Codici di gruppo VIEW

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (<i>AcDbViewTableRecord</i>)
2	Nome vista
70	Valori di flag standard, ovvero valori codificati in bit (vedere "Codici di gruppo comuni per le voci della tabella dei simboli"):
	1 = Se impostato, attiva la visualizzazione in spazio carta
40	Altezza vista (in DCS)
10	Punto centrale della vista (in DCS). DXF: valori X; APP: punto 2D
20	DXF: valori Y del punto centrale della vista (in DCS)
41	Larghezza vista (in DCS)
11	Direzione di vista dalla destinazione (in WCS)
	DXF: valore X; APP: vettore 3D
21, 31	DXF: valori Y e Z della direzione della vista dalla destinazione (in WCS)
12	Punto di destinazione (in WCS). DXF: valori X; APP: punto 3D
22, 32	DXF: valori Y e Z del punto di destinazione (in WCS)
42	Lunghezza obiettivo
43	Piano di ritaglio anteriore (sfalsamento dal punto di destinazione)
44	Piano di ritaglio posteriore (sfalsamento dal punto di destinazione)
50	Angolo di inclinazione
71	Modalità di visualizzazione (vedere la variabile di sistema)

VIEWMODE)

Appendice C -- Codici di gruppo DXF

Sezione TABLES

VPORT

I seguenti codici di gruppo sono validi per le voci della tabella dei simboli VPORT.

Codici di gruppo VPORT

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (<i>AcDbViewportTableRecord</i>)
2	Nome finestra
70	Valori flag standard (vedere "Codici di gruppo comuni per le voci della tabella dei simboli")
10	Angolo inferiore sinistro della finestra. DXF: valore X; APP: punto 2D
20	DXF: valori Y dell'angolo inferiore sinistro della finestra
11	Angolo superiore destro della finestra DXF: valore X; APP: punto 2D
21	DXF: valori Y dell'angolo superiore destro della finestra
12	Punto centrale della vista (in DCS). DXF: valori X; APP: punto 2D
22	DXF: valori Y del punto centrale della vista (in DCS)
13	Punto base dello snap. DXF: valori X; APP: punto 2D
23	DXF: valori Y del punto base dello snap
14	Intervallo dello snap X ed Y. DXF: valore X; APP: punto 2D
24	DXF: valori Y dell'intervallo dello snap X ed Y
15	Spaziatura della griglia X ed Y. DXF: valore X; APP: punto 2D
25	DXF: valori Y della spaziatura della griglia X ed Y
16	Direzione della vista dal punto di destinazione (in WCS) DXF: valore X; APP: punto 3D
26, 36	DXF: valori Y e Z della direzione della vista dal punto di destinazione (in WCS)
17	Punto di destinazione della vista (in WCS). DXF: valori X; APP: punto 3D
27, 37	DXF: valori Y e Z del punto di destinazione della vista (in WCS)
40	Altezza della vista
41	Aspetto prospettico della finestra
42	Lunghezza obiettivo
43	Piano di ritaglio anteriore (sfalsamento dal punto di destinazione)
44	Piano di ritaglio posteriore (sfalsamento dal punto di destinazione)
50	Angolo di rotazione dello snap
51	Angolo di inclinazione della vista
68	APP: Campo di stato (mai salvato in DXF)
69	APP: ID (mai salvato in DXF)
71	Modalità di visualizzazione (vedere la variabile di sistema VIEWMODE)
72	Percentuale zoom cerchio
73	Impostazione zoom veloce
74	Impostazione ICONAUCS
75	Snap on/off
76	Griglia on/off
77	Stile snap
78	Snap assonometrico

La tabella VPORT è univoca: può contenere diverse voci con lo stesso nome (che indicano una configurazione a più finestre). Le voci che corrispondono alla configurazione della finestra attiva hanno tutte il nome *ACTIVE. La prima di queste voci descrive la finestra corrente.

Appendice C -- Codici di gruppo DXF

Sezione BLOCKS

I codici di gruppo descritti in questa sezione si trovano nei file DXF e vengono utilizzati dalle applicazioni. La sezione BLOCKS contiene una voce per ciascun riferimento ad un blocco del disegno.

I codici di gruppo riportati in questa sezione possono essere validi per i file DXF, per le applicazioni (AutoLISP, ADS o ARX) o per entrambi. Quando la descrizione di un codice è diversa per le applicazioni e per i file DXF (oppure è valida solo in uno dei due casi), essa è preceduta dai seguenti indicatori:

APP

Descrizione specifica per l'applicazione

DXF

Descrizione specifica per i file DXF

Se la descrizione è comune ai file DXF ed alle applicazioni, non è presente alcun indicatore. In caso contrario, è presente l'indicatore appropriato. I codici opzionali sono riportati in grigio.

Appendice C -- Codici di gruppo DXF

Sezione BLOCKS

Blocchi nei file DXF

La sezione BLOCKS del file DXF contiene tutte le definizioni dei blocchi. Contiene le entità che costituiscono i blocchi utilizzati nel disegno, compresi i blocchi anonimi generati dal comando RETINO e dalla quotatura associativa. Il formato delle entità in questa sezione è identico a quello della sezione ENTITIES. Tutte le entità della sezione BLOCKS sono comprese tra un'entità block e un'entità endblk. Le entità block e endblk sono presenti solo nella sezione BLOCKS. Le definizioni dei blocchi non sono mai nidificate, ossia non si trova mai un'altra entità block o endblk tra una coppia block-endblk, anche se una definizione di blocco può contenere un'entità insert.

I riferimenti esterni vengono scritti nel file DXF come definizioni di blocchi, ma comprendono anche una stringa (codice di gruppo 1) che specifica il percorso ed il nome del file del riferimento esterno.

Il gestore della tabella dei blocchi, come gli eventuali dati estesi ed i reattori costanti, è presente in ogni definizione di blocco immediatamente dopo il record BLOCK, che contiene tutte le informazioni specifiche memorizzate in un record della tabella dei blocchi. Di conseguenza, ciascuna definizione di blocco ha una sequenza di record simile alla seguente.

Viene di seguito riportato un esempio di sezione BLOCKS di un file DXF:

```

0                               Inizio della sezione BLOCKS
SECTION
2
BLOCKS
0                               Inizia ciascuna voce di blocco
BLOCK                           (definizione di entità block)
5
<database>
100
AcDbEntity
8
<layer>
100
AcDbBlockBegin
2
<nome blocco>
70
<flag>
10
```

```

<valore X>
20
<valore Y>
30
<valore Z>
3
<nome blocco>
1
<percorso xrif>
0          Una voce per ogni definizione di
<tipo entità>          entità block
.
. <dati>
.
0          Fine di ogni voce del blocco
ENDBLK          (definizione di entità endblk)
5
<database>
100
AcDbBlockEnd
0          Fine della sezione BLOCKS
ENDSEC
    
```

Appendice C -- Codici di gruppo DXF

Sezione BLOCKS

BLOCK

I seguenti codici di gruppo sono validi per le entità block (blocco).

Codici di gruppo block

Codici di gruppo	<i>Descrizione</i>
0	<i>Tipo entità (BLOCK)</i>
5	<i>Gestore</i>
102	<i>Inizio del gruppo definito dall'applicazione "{nome_applicazione}". Ad esempio, "ACAD_REACTORS" indica l'inizio del gruppo di reattori costanti di AutoCAD.</i>
codici definiti dalle applicazioni	<i>I codici ed i valori dei gruppi 102 sono definiti dalle applicazioni.</i>
102	<i>Fine del gruppo, "}"</i>
100	<i>Contrassegno di sottoclasse (AcDbEntity)</i>
8	<i>Nome layer</i>
100	<i>Contrassegno di sottoclasse (AcDbBlockBegin)</i>
2	<i>Nome blocco</i>
70	<i>Flag di tipo blocco (valori codificati in bit che possono essere combinati):</i> 1 = Questo è un blocco anonimo, generato da un tratteggio, da una quota associativa, da altre operazioni interne o da un'applicazione 2 = Questo blocco contiene definizioni di attributo 4 = Questo blocco è un riferimento esterno (Xrif) 8 = Questo blocco è una sovrapposizione di xrif 16 = Questo blocco è esternamente dipendente 32 = Questo è un riferimento esterno risolto oppure dipende da un riferimento esterno (ignorato nell'input) 64 = Questa definizione è un riferimento con riferimenti esterni (ignorato nell'input)
10	<i>Punto base. DXF: valore X; APP: punto 3D</i>
20, 30	<i>DXF: valore Y e Z del punto base</i>
3	<i>Nome blocco</i>
1	<i>Nome percorso Xrif (opzionale; presente solo se il blocco è un xrif)</i>

L'UCS attivo quando viene creata una definizione di blocco diventa il WCS per tutte le entità della definizione del blocco. La nuova

origine per queste entità viene spostata in modo che corrisponda al punto base definito per la definizione del blocco. Tutti i dati dell'entità vengono convertiti in modo da essere adattati al nuovo WCS.

Definizione di blocco *MODEL_SPACE e *PAPER_SPACE

Nella sezione BLOCKS, sono sempre presenti altre due definizioni vuote: *MODEL_SPACE e *PAPER_SPACE. Queste definizioni indicano la nuova rappresentazione dello spazio modello e dello spazio carta come definizioni di blocco interne. Le entità contenute in queste definizioni sono ancora presenti nella sezione ENTITIES per ragioni di compatibilità.

Isolamento delle entità spazio modello e spazio carta

Per ragioni di organizzazione interna, l'interlacciamento tra lo spazio modello e lo spazio carta non si verificherà più. Ora, vengono prodotte prima tutte le entità dello spazio carta, seguite da tutte le entità dello spazio modello. Il flag che le distingue è il codice di gruppo 67.

Appendice C -- Codici di gruppo DXF

Sezione BLOCKS

ENDBLK

I seguenti codici di gruppo sono validi per gli oggetti endblk.

Codici di gruppo Endblk

Codici di gruppo	<i>Descrizione</i>
0	<i>Tipo entità (ENDBLK)</i>
5	<i>Gestore</i>
102	<i>Inizio del gruppo definito dall'applicazione "{nome_applicazione}". Ad esempio, "{ACAD_REACTORS}" indica l'inizio del gruppo di reattori costanti di AutoCAD.</i>
<i>codici definiti dalle applicazioni</i>	<i>I codici ed i valori dei gruppi 102 sono definiti dalle applicazioni.</i>
102	<i>Fine del gruppo, "}"</i>
100	<i>Contrassegno di sottoclasse (AcDbBlockEnd)</i>

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

Questa sezione descrive i codici di gruppo validi per gli oggetti grafici. Questi codici si trovano nella sezione ENTITIES di un file DXF e vengono utilizzati dalle applicazioni AutoLISP e ARX negli elenchi di definizioni delle entità.

I codici di gruppo riportati in questa sezione possono essere validi per i file DXF, per le applicazioni (AutoLISP, ADS o ARX) o per entrambi. Quando la descrizione di un codice è diversa per le applicazioni e per i file DXF (oppure è valida solo in uno dei due casi), essa è preceduta dai seguenti indicatori:

APP

Descrizione specifica per l'applicazione

DXF

Descrizione specifica per i file DXF

Se la descrizione è comune ai file DXF ed alle applicazioni, non è presente alcun indicatore. In caso contrario, è presente l'indicatore appropriato. I codici opzionali sono riportati in grigio.

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

Codici di gruppo comuni per gli oggetti grafici

La seguente tabella mostra i codici di gruppo validi praticamente per tutti gli oggetti grafici. Alcuni dei codici di gruppo mostrati qui sono inclusi nella definizione di entità solo se l'entità contiene dei valori diversi da quelli di default per la proprietà. Quando si fa riferimento ai codici di gruppo per tipo di entità, che elencano i codici associati a entità specifiche, tenere presente che sono inclusi anche i codici qui riportati.

Quando un gruppo viene omissso, il relativo valore di default al momento dell'input (quando si utilizza DXFIN) viene indicato nella terza colonna. Se il valore di un codice di gruppo è uguale a quello di default, viene omissso al momento dell'output (quando si utilizza DXFOUT).

Codici di gruppo validi per tutti gli oggetti grafici

Codice di gruppo	Descrizione	Se omissso, il valore di default è...
-1	APP: nome dell'entità (cambia ogni qualvolta si apre un disegno)	Non omissso
0	Tipo di entità	Non omissso
5	Gestore	Non omissso
102	Inizio del gruppo definito dall'applicazione "{nome_applicazione". Ad esempio, "{ACAD_REACTORS" indica l'inizio del gruppo di reattori costanti di AutoCAD.	nessun valore di default
codici definiti dalle applicazioni	I codici ed i valori dei gruppi 102 sono definiti dalle applicazioni.	nessun valore di default
102	Fine del gruppo, "}"	nessun valore di default
100	Contrassegno di sottoclasse (AcDbEntity)	Non omissso
67	Un valore mancante o uguale a zero indica che l'entità è in modalità spazio modello. Uno indica che l'entità è in modalità spazio carta (opzionale).	0
8	Nome layer	Non omissso
6	Nome tipo di linea (presente in assenza di DALAYER). Il nome speciale DABLOCCO indica un tipo di linea mobile (opzionale).	DALAYER
62	Numero colore (presente se non è DALAYER). Il valore zero indica il colore DABLOCCO (mobile). 256 indica DALAYER. Un valore negativo indica che il layer è disattivato (opzionale).	DALAYER
48	Scala tipo di linea (opzionale)	1.0
60	Visibilità oggetto (opzionale): 0 = visibile, 1 = invisibile.	0

La seguente tabella mostra i codici di gruppo che risultano se sono collegati ad un'entità dei reattori costanti.

Record ACAD_REACTORS

Codice di gruppo	Descrizione
102	"{ACAD_REACTORS" indica l'inizio del gruppo di reattori costanti di AutoCAD
330	ID/gestore puntatore soft per il dizionario di proprietà
102	Fine del gruppo, "}"

La seguente tabella mostra i codici di gruppo che risultano se un dizionario esteso è stato collegato ad un'entità.

Record ACAD_XDICTIONARY

Codice di gruppo	Descrizione
102	"{ACAD_XDICTIONARY" indica l'inizio di un gruppo di dizionari estesi.
360	ID/gestore puntatore hard per il dizionario di proprietà
102	Fine del gruppo, "}"

Nota Non scrivere programmi che si basano sull'ordine riportato in queste tabelle dei codici DXF. Benché queste tabelle mostrino l'ordine in cui sono normalmente riportati codici di gruppo, in determinate condizioni o nelle release future di AutoCAD l'ordine

potrebbe cambiare. Il codice che controlla un'entità deve essere basato su una condizione (parametro) o una tabella, in modo che sia possibile elaborare correttamente ciascun gruppo anche se l'ordine non è quello previsto.

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

3DFACE

I seguenti codici di gruppo sono validi per le entità 3dface (3dfaccia).

Codici di gruppo 3dface

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (AcDbFace)
10	Primo angolo (in WCS). DXF: valore X; APP: punto 3D
20, 30	DXF: valore Y e Z del primo angolo (in WCS)
11	Secondo angolo (in WCS). DXF: valore X; APP: punto 3D
21, 31	DXF: valore Y e Z del secondo angolo (in WCS)
12	Terzo angolo (in WCS). DXF: valore X; APP: punto 3D
22, 32	DXF: valore Y e Z del terzo angolo (in WCS)
13	Quarto angolo (in WCS). Se vengono inseriti solo tre angoli, questo angolo equivale al terzo angolo
	DXF: X; APP: punto 3D
23, 33	DXF: valore Y e Z del quarto angolo (in WCS)
70	Flag degli spigoli invisibili (opzionale; default = 0):
	1 = Il primo spigolo è invisibile.
	2 = Il secondo spigolo è invisibile.
	4 = Il terzo spigolo è invisibile.
	8 = Il quarto spigolo è invisibile.

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

3DSOLID

I seguenti codici di gruppo sono validi per le entità 3dsolid (solido 3d).

Codici di gruppo 3dsolid

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (AcDbModelerGeometry)
70	Numero di versione del formato del modellatore (correntemente = 1)
1	Dati di proprietà (più righe < 255 caratteri ciascuna)
3	Righe supplementari dei dati di proprietà (se una stringa del gruppo 1 era maggiore di 255 caratteri)

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

ARC

I seguenti codici di gruppo sono validi per le entità arc (arco).

Codici di gruppo arc

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbCircle)</i>
39	<i>Spessore (opzionale; default = 0)</i>
10	<i>Punto centrale (in OCS). DXF: valore X; APP: punto 3D</i>
20, 30	<i>DXF: valore Y e Z del punto centrale (in OCS)</i>
40	<i>Raggio</i>
100	<i>Contrassegno di sottoclasse (AcDbArc)</i>
50	<i>Angolo iniziale</i>
51	<i>Angolo finale</i>
210	<i>Direzione di estrusione (opzionale; default = 0, 0, 1). DXF: X; APP: vettore 3D</i>
220, 230	<i>DXF: valore Y e Z della direzione di estrusione</i>

Appendice C -- Codici di gruppo DXF **Sezione ENTITIES** **ATTDEF**

I seguenti codici di gruppo sono validi per le entità attdef (entità di definizione degli attributi).

Codici di gruppo Attdef

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbText)</i>
39	<i>Spessore (opzionale; default = 0)</i>
10	<i>Primo punto di allineamento (in OCS). DXF: X; APP: punto 3D</i>
20, 30	<i>DXF: valore Y e Z del punto iniziale del testo (in OCS)</i>
40	<i>Altezza testo</i>
1	<i>Valore di default (stringa)</i>
100	<i>Contrassegno di sottoclasse (AcDbAttributeDefinition)</i>
50	<i>Rotazione testo (opzionale, default = 0):</i>
41	<i>Fattore di scala X relativo (larghezza) (opzionale; default = 1) Questo valore viene regolato anche quando si utilizza il testo di tipo Adatta.</i>
51	<i>Angolo obliquo (opzionale, default = 0):</i>
7	<i>Nome dello stile di testo (opzionale, default = STANDARD)</i>
71	<i>Flag di generazione del testo (opzionale, default = 0). Vedere il comando TESTO.</i>
72	<i>Tipo di giustificazione del testo orizzontale (opzionale, default = 0). Vedere il comando TESTO .</i>
11	<i>Secondo punto di allineamento (in OCS). DXF: X; APP: punto 3D Presente solo se il gruppo 72 o 74 è diverso da zero</i>
21, 31	<i>DXF: valore Y e Z del secondo punto di allineamento (in OCS)</i>
210	<i>Direzione di estrusione (opzionale; default = 0, 0, 1). DXF: X; APP: vettore 3D</i>
220, 230	<i>DXF: valore Y e Z della direzione di estrusione</i>
100	<i>Contrassegno di sottoclasse (AcDbAttributeDefinition)</i>
3	<i>Stringa del messaggio di richiesta</i>
2	<i>Stringa di etichetta</i>
70	<i>Flag dell'attributo 1 = L'attributo è invisibile (non appare). 2 = Questo è un attributo costante. 4 = Richiesta di verifica dell'input di questo attributo. 8 = L'attributo è preimpostato (nessun messaggio di richiesta durante l'inserimento).</i>
73	<i>Lunghezza campo (opzionale; default: 0) (correntemente non utilizzato)</i>
74	<i>Tipo di giustificazione del testo verticale (opzionale; default: 0). Vedere il codice di gruppo 73 del comando TESTO</i>

Se i valori del gruppo 72 o 74 (o di entrambi) sono diversi da zero, i valori del primo punto di allineamento vengono ignorati ed AutoCAD calcola i nuovi valori in base al secondo punto di allineamento ed alla lunghezza ed all'altezza della stringa di testo stessa

(dopo l'applicazione dello stile di testo). Se i valori 72 e 74 sono uguali a zero o mancanti, il secondo punto di allineamento non è significativo.

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

ATTRIB

I seguenti codici di gruppo sono validi per le entità attrib (attributo).

Codici di gruppo Attrib

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (AcDbText)
39	Spessore (opzionale; default = 0)
10	Punto iniziale del testo (in OCS). DXF: valore X; APP: punto 3D
20, 30	DXF: valore Y e Z del punto iniziale del testo (in OCS)
40	Altezza testo
1	Valore di default (stringa)
100	Contrassegno di sottoclasse (AcDbAttribute)
2	Etichetta dell'attributo (stringa)
70	Flag dell'attributo 1 = L'attributo è invisibile (non appare). 2 = Questo è un attributo costante. 4 = Richiesta di verifica dell'input di questo attributo. 8 = L'attributo è preimpostato (nessun messaggio di richiesta durante l'inserimento).
73	Lunghezza campo (opzionale; default: 0) (correntemente non utilizzato)
50	Rotazione testo (opzionale, default = 0):
41	Fattore di scala X relativo (larghezza) (opzionale; default = 1) Questo valore viene regolato anche quando si utilizza il testo di tipo Adatta.
51	Angolo obliquo (opzionale, default = 0):
7	Nome dello stile di testo (opzionale, default = STANDARD)
71	Flag di generazione del testo (opzionale, default = 0). Vedere il comando TESTO.
72	Tipo di giustificazione del testo orizzontale (opzionale, default = 0). Vedere il comando TESTO.
74	Tipo di giustificazione del testo verticale (opzionale; default: 0). Vedere il codice di gruppo 73 del comando TESTO.
11	Punto di allineamento (in OCS). DXF: valore X; APP: punto 3D Presente solo in presenza del gruppo 72 o 74 con valore diverso da zero.
21, 31	DXF: valore Y e Z del punto di allineamento (in OCS)
210	Direzione di estrusione Presente solo se la direzione dell'estrusione dell'entità non è parallela all'asse Z WCS (opzionale; default = 0, 0, 1). DXF: valore X; APP: vettore 3D
220, 230	DXF: valore Y e Z della direzione di estrusione

Se il valore del gruppo 72 o 74 (o di entrambi) è diverso da zero, i valori del punto di inserimento del testo vengono ignorati ed AutoCAD calcola i nuovi valori in base al punto di allineamento del testo ed alla lunghezza della stringa di testo stessa (dopo l'applicazione dello stile di testo). Se i valori dei gruppi 72 e 74 sono uguali a zero o mancanti, il punto di allineamento del testo viene ignorato e ricalcolato in base al punto di inserimento del testo e alla lunghezza della stringa di testo stessa (dopo l'applicazione dello stile di testo).

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

BODY

I seguenti codici di gruppo sono validi per le entità body (corpo).

Codici di gruppo Body

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbModelerGeometry)</i>
70	<i>Numero di versione del formato del modellatore (correntemente = 1)</i>
1	<i>Dati di proprietà (più righe < 255 caratteri ciascuna)</i>
3	<i>Righe supplementari dei dati di proprietà (se una stringa del gruppo 1 era maggiore di 255 caratteri)</i>

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

CIRCLE

I seguenti codici di gruppo sono validi per le entità circle (cerchio).

Codici di gruppo circle

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbCircle)</i>
39	<i>Spessore (opzionale; default = 0)</i>
10	<i>Punto centrale (in OCS). DXF: valore X; APP: punto 3D</i>
20, 30	<i>DXF: valore Y e Z del punto centrale (in OCS)</i>
40	<i>Raggio</i>
210	<i>Direzione di estrusione (opzionale; default = 0, 0, 1). DXF: X; APP: vettore 3D</i>
220, 230	<i>DXF: valore Y e Z della direzione di estrusione</i>

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

DIMENSION

Le definizioni delle entità dimension sono costituite da codici di gruppo comuni a tutti i tipi di quota, seguiti dai codici specifici di ogni tipo.

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

DIMENSION

Codici di gruppo comuni per le quote

I seguenti codici di gruppo sono validi per tutti i tipi di entità dimension (quota).

Codici di gruppo comuni per le quote

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (<i>AcDbDimension</i>)
2	Nome del blocco contenente le entità dimension
10	Punto di definizione (in WCS). DXF: valore X; APP: punto 3D
20, 30	DXF: valore Y e Z del punto di definizione (in WCS)
11	Punto medio del testo di quota (in OCS). DXF: X; APP: punto 3D
21, 31	DXF: valore Y e Z del punto medio del testo di quota (in OCS).
70	Tipo di quota. I valori 0 - 6 sono valori interi che rappresentano il tipo di quota. I valori 32, 64 e 128 sono valori codificati in bit, che vengono aggiunti ai valori interi (il valore 32 è sempre impostato nella R13 e nelle release successive). 0 = Ruotata, orizzontale o verticale; 1 = Allineata 2 = Angolare; 3 = Diametro; 4 = Raggio; 5 = Angolare a 3 punti; 6 = Coordinata 32 = Indica che solo questa quota fa riferimento al riferimento di blocco (codice di gruppo 2). 64 = Tipo di ordinata. Questo è un valore codificato in bit (bit 7) utilizzato solo con il valore intero 6. Se è impostata, la coordinata è di tipo X; se non è impostata, la coordinata è di tipo Y. 128 = Questo è un valore codificato in bit (bit 8) aggiunto agli altri valori del gruppo 70 se il testo di quota è stato collocato in una posizione definita dall'utente invece che nella posizione di default.
1	Testo di quota esplicitamente inserito dall'utente. Opzionale; il valore di default è la misurazione. Se il valore è nullo o ""<>", viene inserita come testo la misurazione della quota, se il valore è " " (uno spazio vuoto), il testo viene eliminato. Qualsiasi altra cosa viene disegnata come testo.
53	Il codice di gruppo opzionale 53 è l'angolo di rotazione del testo di quota a partire dall'orientamento di default (la direzione della linea di quota).
51	Tutti i tipi di quota hanno un codice di gruppo 51 opzionale, che indica la direzione orizzontale per l'entità quota. Questo codice determina l'orientamento delle linee e del testo di quota per le quote lineari ruotate, orizzontali e verticali. Questo valore di gruppo è il valore negativo dell'angolo tra l'asse X OCS e l'asse X UCS. Si trova sempre nel piano XY dell'OCS.
210	Direzione di estrusione (opzionale; default = 0, 0, 1). DXF: X; APP: vettore 3D
220, 230	DXF: valore Y e Z della direzione di estrusione
3	Nome dello stile di quota

Un'entità quota può essere seguita dai dati estesi appartenenti all'ID applicazione "ACAD". Tali dati descrivono qualsiasi sostituzione applicata a questa entità. Vedere "Sostituzioni degli stili di quota."

Per tutti i tipi di quota, i seguenti codici di gruppo rappresentano i punti 3D WCS: (10, 20, 30), (13, 23, 33), (14, 24, 34) e (15, 25, 35).
Per tutti i tipi di quota, i seguenti codici di gruppo rappresentano i punti 3D OCS: (11, 21, 31), (12, 22, 32) e (16, 26, 36).

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

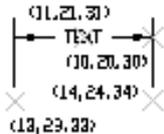
DIMENSION

Codici di gruppo per le quote allineate, lineari e ruotate

I seguenti codici di gruppo sono validi per le quote allineate, lineari e ruotate.

Codici di gruppo per le quote allineate, lineari e ruotate

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (<i>AcDbAlignedDimension</i>)
12	Punto di inserimento per i duplicati di una quota: Linea di base e Continua (in OCS). DXF: valore X; APP: punto 3D
22, 32	DXF: valore Y e Z del punto di inserimento per i duplicati di una quota: Linea di base e Continua (in OCS).
13	Punto di definizione per le quote lineari ed angolari (in WCS). DXF: valore X; APP: punto 3D
23, 33	DXF: valore Y e Z del punto di definizione per le quote lineari ed angolari (in WCS).
14	Punto di definizione per le quote lineari ed angolari (in WCS). DXF: valore X; APP: punto 3D
24, 34	DXF: valore Y e Z del punto di definizione per le quote lineari ed angolari (in WCS).



Il punto (13,23,33) specifica il punto iniziale della prima linea di estensione ed il punto (14,24,34) specifica il punto iniziale della seconda linea di estensione. Il punto (10,20,30) specifica la posizione della linea di quota. Il punto (11,21,31) specifica il punto medio del testo di quota.

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

DIMENSION

Codici di gruppo per le quote lineari e ruotate

I seguenti codici di gruppo sono validi per le quote lineari e ruotate.

Codici di gruppo per le quote lineari e ruotate

Codici di gruppo	Descrizione
50	Angolo di quote ruotate, orizzontali o verticali.
52	I tipi di quote lineari con un angolo obliquo hanno un codice di gruppo opzionale 52. Se aggiunto all'angolo di rotazione della quota lineare (codice di gruppo 50), fornisce l'angolo delle linee di estensione.
100	Contrassegno di sottoclasse (<i>AcDbRotatedDimension</i>)

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

DIMENSION

Codici di gruppo per le quote radiali e di diametro

I seguenti codici di gruppo sono validi per le quote radiali e di diametro.

Codici di gruppo per le quote radiali e di diametro

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (<i>AcDbRadialDimension</i> o <i>AcDbDiametricDimension</i>)
15	Punto di definizione per le quote di diametro, radiali ed angolari (in WCS). DXF: valore X; APP: punto 3D
25, 35	DXF: valore Y e Z del punto di definizione per le quote di diametro, radiali ed angolari (in WCS).
40	Lunghezza della direttrice per le quote di raggio e diametro.



Il punto (15,25,35) specifica il primo punto della linea di quota del cerchio/arco ed il punto (10,20,30) specifica il punto opposto al primo punto. Il punto (11,21,31) specifica il punto medio del testo di quota.



Il punto (15,25,35) specifica il primo punto della linea di quota del cerchio/arco ed il punto (10,20,30) specifica il centro del cerchio/arco. Il punto (11,21,31) specifica il punto medio del testo di quota.

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

DIMENSION

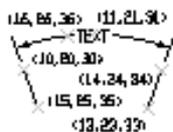
Codici di gruppo per quote angolari

I codici di gruppo seguenti sono validi per le quote angolari.

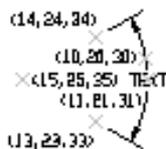
Codici di gruppo comuni per le quote angolari

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (<i>AcDb3PointAngularDimension</i>)
13	Punto di definizione per le quote lineari ed angolari (in WCS). DXF: valore X; APP: punto 3D
23, 33	DXF: valore Y e Z del punto di definizione per le quote lineari ed angolari (in WCS).
14	Punto di definizione per le quote lineari ed angolari (in WCS). DXF: valore X; APP: punto 3D

- 24, 34 *DXF: valore Y e Z del punto di definizione per le quote lineari ed angolari (in WCS).*
- 15 *Punto di definizione per le quote di diametro, radiali ed angolari (in WCS). DXF: valore X; APP: punto 3D*
- 25, 35 *DXF: valore Y e Z del punto di definizione per le quote di diametro, radiali ed angolari (in WCS).*
- 16 *Punto che definisce l'arco di quota per le quote angolari (in OCS). DXF: valore X; APP: punto 3D*
- 26, 36 *DXF: valore Y e Z del punto di definizione dell'arco di quota per le quote angolari (in OCS).*



Il punto (13,23,33) e (14,24,34) specificano i punti finali della linea utilizzati per determinare la prima linea di estensione ed i punti (10,20,30) e (15,25,35) specificano i punti finali della linea utilizzati per determinare la seconda linea di estensione. Il punto (16,26,36) specifica la posizione dell'arco della linea di quota. Il punto (11,21,31) specifica il punto medio del testo di quota.



Il punto (15,25,35) specifica il vertice dell'angolo. I punti (13,23,33) e (14,24,34) specificano i punti finali delle linee d'estensione. Il punto (10,20,30) specifica la posizione dell'arco della linea di quota ed il punto (11,21,31) specifica il punto medio del testo di quota.

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

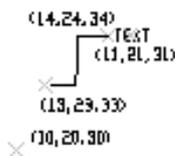
DIMENSION

Codici di gruppo per quote di coordinate

I seguenti codici di gruppo sono validi per le quote di coordinate.

Codici di gruppo per quote di coordinate

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (AcDbOrdinateDimension)
13	Punto di definizione per le quote lineari ed angolari (in WCS). DXF: valore X; APP: punto 3D
23, 33	DXF: valore Y e Z del punto di definizione per le quote lineari ed angolari (in WCS).
14	Punto di definizione per le quote lineari ed angolari (in WCS). DXF: valore X; APP: punto 3D
24, 34	DXF: valore Y e Z del punto di definizione per le quote lineari ed angolari (in WCS).



Il punto (13,23,33) specifica la posizione della funzione ed il punto (14,24,34) specifica il punto finale della direttrice. Il punto

(11,21,31) specifica il punto medio del testo di quota. Il punto (10,20,30) viene posizionato all'origine dell'UCS corrente al momento della creazione della quota.

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

DIMENSION

Sostituzioni degli stili di quota

Le sostituzioni degli stili di quota possono essere applicate alle entità dimension, leader e tolerance. Qualsiasi sostituzione applicata a queste entità viene memorizzata nell'entità come dati estesi. I codici di gruppo della variabile di quota sostituita ed i relativi valori si trovano nelle stringhe di controllo del gruppo 1002. L'esempio seguente mostra i dati estesi di un'entità dimension in cui le variabili DIMTOL e DIMCLRE sono state sostituite.

```
(setq diment (car (entsel))) ; Selezione dell'entità quota
(setq elst (entget diment '("ACAD"))) ; Elenco di definizione entità
(assoc -3 elst) ; Estrazione solo dati estesi
```

Questo codice restituisce quanto segue:

```
(-3 ("ACAD" ; Inizio della sezione ACAD APPID dei dati estesi
(1000 . "DSTYLE") (1002 . "{") ; Inizio della sottosezione dimstyle
(1070 . 177) (1070 . 3) ; Sostituzione DIMCLRE (codice 177) + valore (3)
(1070 . 71) (1070 . 1) ; Sostituzione DIMTOL (codice 71) + valore (1)
(1002 . "}") ) ; Fine sottosezione dimstyle e sezione ACAD
```

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

ELLIPSE

I seguenti codici di gruppo sono validi per le entità ellipse (ellisse).

Codici di gruppo Ellipse

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (AcDbEllipse)
10	Punto centrale (in WCS). DXF: valore X; APP: punto 3D
20, 30	DXF: valore Y e Z del punto centrale (in WCS)
11	Punto finale dell'asse maggiore (relativo al centro). DXF: X; APP: punto 3D
21, 31	DXF: valore e Z del punto finale dell'asse maggiore (relativo al centro)
210	Direzione di estrusione (opzionale; default = 0, 0, 1). DXF: X; APP: vettore 3D
220, 230	DXF: valore Y e Z della direzione di estrusione
40	Rapporto dell'asse minore rispetto all'asse maggiore
41	Parametro iniziale (questo valore è 0.0 per un'ellisse completa)
42	Parametro finale (questo valore è 2pi per un'ellisse completa)

I codici di gruppo 41 e 42 sono i valori iniziali e finali per la variabile u dell'equazione che segue. La grandezza del vettore dei codici 11, 21, 31 è pari ad $1/2$ dell'asse maggiore, che è il valore a nell'equazione. Il punto 10,20,30 è il valore c dell'equazione. Conoscendo tutti questi dati, è possibile calcolare il valore b per completare l'equazione.

Descrizione dell'opzione Parametro del comando ELLISSE

L'opzione Parametro del comando ELLISSE utilizza, per la definizione di un arco ellittico, l'equazione seguente.

$$p(u) = c + a \cdot \cos(u) + b \cdot \sin(u)$$

Le variabili a , b e c vengono determinate alla selezione dei punti finali per il primo asse e della distanza per il secondo asse. a è un valore negativo pari ad $1/2$ della lunghezza dell'asse maggiore, b è un valore negativo pari ad $1/2$ della lunghezza dell'asse minore e c è il punto centrale (2D) dell'ellisse.

Poiché questa equazione è effettivamente un'equazione vettoriale e la variabile c è in pratica un punto con valori X ed Y , dovrebbe essere scritta nel modo seguente:

$$p(u) = (Cx + a \cdot \cos(u)) \cdot i + (Cy + a \cdot \sin(u)) \cdot j$$

dove

Cx è il valore X del punto c

Cy è il valore Y del punto c

a è $-(1/2)$ della lunghezza dell'asse maggiore)

b è $-(1/2)$ della lunghezza dell'asse minore)

i e j rappresentano vettori di unità nelle direzioni X ed Y

In AutoCAD, dopo la selezione dei punti finali degli assi, non resta altro che specificare il punto iniziale e quello finale dell'arco ellittico.

Quando si seleziona l'opzione Parametro, viene richiesto di specificare un parametro iniziale ed uno finale. I suddetti valori vengono poi inseriti nell'equazione per determinare il punto iniziale e quello finale effettivi sull'ellisse. Il resto dell'ellisse viene riempito tra questi due punti in una direzione antioraria dal primo parametro al secondo. Il valore specificato per u è in gradi allo scopo di ottenere $\cos(u)$ and $\sin(u)$.

Ad esempio:

Prima estremità asse = 0,1

Seconda estremità dell'asse = 4,1

Distanza secondo asse = 2,0

Incluso parametro = 270

Fine parametro = 0

genererà il punto iniziale a 2,2 e quello finale a 0,1; l'ellisse verrà riempita da 2,2 a 0,1 in una direzione antioraria.

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

HATCH

I seguenti codici di gruppo sono validi per le entità hatch.

Codici di gruppo hatch

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbHatch)</i>
10	<i>Punto di elevazione (in OCS). DXF: valore X; APP: punto 3D (X ed Y sono sempre uguali a 0, Z rappresenta l'elevazione)</i>
20, 30	<i>DXF: valore Y e Z del punto di elevazione (in OCS). Se Y = 0, Z rappresenta l'elevazione.</i>
210	<i>Direzione di estrusione (opzionale; default = 0, 0, 1). DXF: X; APP: vettore 3D</i>
220, 230	<i>DXF: valore Y e Z della direzione di estrusione</i>
2	<i>Nome del modello di tratteggio</i>
70	<i>Flag del riempimento solido (riempimento solido = 1; riempimento a motivi = 0)</i>
71	<i>Flag dell'associatività (associativo = 1; non associativo = 0)</i>
91	<i>Numero di percorsi di contorno (cicli)</i>
varia	<i>Dati del percorso di contorno Ripete per il numero di volte specificato dal codice 91. Vedere "Dati del tracciato di contorno".</i>
75	<i>Stile di tratteggio 0 = tratteggia l'area "parità dispari" (stile Normale)</i>

	1 = tratteggia solo l'area più esterna (stile Esterno)
	2 = tratteggia l'intera area (stile Ignora)
76	Tipo del modello di tratteggio
	0 = definito dall'utente
	1 = predefinito
	2 = personalizzato
52	Angolo del modello di tratteggio (solo riempimento a motivi)
41	Spaziatura o scala del modello di tratteggio (solo riempimento a motivi)
77	Flag del modello di tratteggio doppio (doppio = 1, singolo = 0) (solo riempimento a motivi)
78	Numero di linee di definizione del modello
varia	Dati della linea modello. Ripete per il numero di volte specificato dal codice 78. Vedere "Dati del modello"..
47	Dimensioni in pixel (opzionale)
98	Numero di punti seme.
10	Punto seme (in OCS). DXF: valore X; APP: punto 2D (voci multiple)
20	DXF: valore Y del punto seme (in OCS) (voci multiple)

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

HATCH

Dati del tracciato di contorno

Il contorno di ciascun oggetto di tratteggio è definito da un tracciato (o *iterazione*) composto da uno o più segmenti. I dati relativi ai segmenti del tracciato dipendono dal tipo o dai tipi di entità che compongono il tracciato. Ciascun segmento del tracciato viene definito da un insieme di codici di gruppo specifico.

Codici di gruppo dei dati del tracciato del contorno

Codici di gruppo	Descrizione
92	Flag del tipo di tracciato del contorno (codificato bit)
	0 = default
	1 = esterno
	2 = polilinea
	4 = derivato
	8 = casella di testo
	16 = più esterno
varia	Dati del contorno di tipo polilinea (solo se contorno = polilinea)
93	Numero di limiti di questo tracciato (solo se il contorno non è una polilinea)
72	Tipo di limite (solo se il contorno non è una polilinea)
	1 = linea
	2 = arco circolare
	3 = arco ellittico
	4 = spline
varia	Dati del tipo di limite (solo se il contorno non è una polilinea). Vedere la tabella dei dati del limite appropriata.
97	Numero di oggetti di contorno di origine
340	Riferimento hard agli oggetti di contorno di origine (voci multiple)

Codici di gruppo dei dati del contorno polilinea

Codici di gruppo	Descrizione
72	Include il flag della curvatura
73	È un flag di chiusura
93	Numero di vertici della polilinea
10	Posizione del vertice (in OCS). DXF: valore X; APP: punto 2D (voci multiple)
20	DXF: valore Y della posizione del vertice (in OCS). (voci multiple)

42 *Curvatura (opzionale; default è 0).*

Codici di gruppo dei dati del limite linea

Codici di gruppo	<i>Descrizione</i>
10	<i>Punto iniziale (in OCS). DXF: valore X; APP: punto 2D</i>
20	<i>DXF: valore Y del punto iniziale (in OCS)</i>
11	<i>Punto finale (in OCS). DXF: valore X; APP: punto 2D</i>
21	<i>DXF: valore Y del punto finale (in OCS)</i>

Codici di gruppo dei dati del limite arco

Codici di gruppo	<i>Descrizione</i>
10	<i>Punto centrale (in OCS). DXF: valore X; APP: punto 2D</i>
20	<i>DXF: valore Y del punto centrale (in OCS)</i>
40	<i>Raggio</i>
50	<i>Angolo iniziale</i>
51	<i>Angolo finale</i>
73	<i>È un flag che indica senso antiorario.</i>

Codici di gruppo dei dati del limite ellisse

Codici di gruppo	<i>Descrizione</i>
10	<i>Punto centrale (in OCS). DXF: valore X; APP: punto 2D</i>
20	<i>DXF: valore Y del punto centrale (in OCS)</i>
11	<i>Punto finale dell'asse maggiore relativo al punto centrale (in OCS). DXF: valore X; APP: punto 2D</i>
21	<i>DXF: valore Y del punto finale dell'asse maggiore (in OCS)</i>
40	<i>Lunghezza dell'asse minore (percentuale della lunghezza dell'asse maggiore)</i>
50	<i>Angolo iniziale</i>
51	<i>Angolo finale</i>
73	<i>È un flag che indica senso antiorario.</i>

Codici di gruppo dei dati del limite spline

Codici di gruppo	<i>Descrizione</i>
94	<i>Gradi</i>
73	<i>Razionale</i>
74	<i>Periodico</i>
95	<i>Numero di nodi</i>
96	<i>Numero di punti di controllo</i>
40	<i>Valori dei nodi (voci multiple)</i>
10	<i>Punto centrale (in OCS). DXF: valore X; APP: punto 2D</i>
20	<i>DXF: valore Y del punto centrale (in OCS)</i>
42	<i>Pesi (opzionale; default = 1)</i>

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

HATCH

Dati del modello

I codici dei dati del modello seguenti vengono ripetuti per ogni linea di definizione del modello.

Codici di gruppo dei dati del modello di tratteggio

Codici di gruppo	<i>Descrizione</i>
53	<i>Angolo delle linee del modello.</i>
43	<i>Punto base delle linee del modello, componente X</i>
44	<i>Punto base delle linee del modello, componente Y</i>
45	<i>Sfalsamento delle linee del modello, componente X</i>
46	<i>Sfalsamento delle linee del modello, componente Y</i>
79	<i>Numero di elementi lunghezza del trattino</i>

49 Lunghezza del trattino (voci multiple)

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

IMAGE

I seguenti codici di gruppo sono validi per le entità image (immagine).

Codici di gruppo image

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (AcDbRasterImage)
90	Versione della classe
10	Punto di inserimento (in OCS). DXF: valore X; APP: punto 3D
20, 30	DXF: valore Y e Z del punto di inserimento (in OCS)
11	Vettore U di un singolo pixel: punti lungo la parte inferiore visibile dell'immagine, a partire dal punto di inserimento (in OCS). DXF: valore X; APP: punto 3D
21, 31	DXF: valore Y e Z del vettore U (in OCS)
12	Vettore V di un singolo pixel: punti lungo la parte sinistra visibile dell'immagine, a partire dal punto di inserimento (in OCS). DXF: valore X; APP: punto 3D
22, 32	DXF: valore Y e Z del vettore V (in OCS)
13	Dimensioni dell'immagine in pixel. DXF: valore U; APP: punto bidimensionale (valori U e V)
23	DXF: valore V, ovvero il valore delle dimensioni dell'immagine in pixel.
340	Riferimento hard all'oggetto imagedef
70	Proprietà della visualizzazione di immagini 1 = Mostra immagine 2 = Mostra immagine quando non è allineata allo schermo 4 = Usa contorno di ritaglio 8 = Attiva trasparenza
280	Stato ritaglio: 0 = disattivato, 1 = attivato
281	Valore della luminosità (da 0 a 100; default = 50)
282	Valore del contrasto (da 0 a 100; default = 50)
283	Valore della sbiaditura (da 0 a 100; default = 0)
360	Riferimento hard all'oggetto reattore_imagedef
71	Tipo di contorno di ritaglio: 1 = rettangolare, 2 = poligonale
91	Numero dei vertici del contorno di ritaglio che seguono.
14	Vertici del contorno di ritaglio (in OCS). DXF: valore X; APP: punto 2D (voci multiple) Note: 1) Per il tipo di contorno di ritaglio rettangolare, è necessario specificare due angoli opposti. Il default è (-0.5,-0.5), (dimensioni.x-0.5, dimensioni.y-0.5). 2) Per il tipo di contorno di ritaglio poligonale, è necessario specificare tre o più vertici. I vertici poligonali devono essere elencati in ordine consecutivo.
24	DXF: valore Y del vertice del contorno di ritaglio (in OCS) (voci multiple)

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

INSERT

I seguenti codici di gruppo sono validi per le entità insert (riferimento di blocco).

Codici di gruppo insert

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbBlockReference)</i>
66	<i>Flag "seguono attributi della variabile" (opzionale; default: = 0); se il valore del flag "seguono attributi" è 1, Insert dovrebbe essere seguita da una serie di entità attrib e conclusa dall'entità seqend.</i>
2	<i>Nome blocco</i>
10	<i>Punto di inserimento (in OCS). DXF: valore X; APP: punto 3D</i>
20, 30	<i>DXF: valore Y e Z del punto di inserimento (in OCS)</i>
41	<i>Fattore di scala X (opzionale; default = 1)</i>
42	<i>Fattore di scala Y (opzionale; default = 1)</i>
43	<i>Fattore di scala Z (opzionale; default = 1)</i>
50	<i>Angolo di rotazione (opzionale, default = 0)</i>
70	<i>Calcolo colonna (opzionale; default = 1)</i>
71	<i>Calcolo riga (opzionale; default = 1)</i>
44	<i>Spaziatura colonna (opzionale; default = 0)</i>
45	<i>Spaziatura riga (opzionale; default = 0)</i>
210	<i>Direzione di estrusione (opzionale; default = 0, 0, 1). DXF: X; APP: vettore 3D</i>
220, 230	<i>DXF: valore Y e Z della direzione di estrusione</i>

Appendice C -- Codici di gruppo DXF **Sezione ENTITIES** **LEADER**

I seguenti codici di gruppo sono validi per le entità leader (direttrice).

Codici di gruppo leader

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbLeader)</i>
3	<i>Nome dello stile di quota</i>
71	<i>Flag della punta di freccia: 0 = disattivato; 1 = attivato</i>
72	<i>Tipo di percorso della direttrice: 0 = Segmenti di linea retta; 1 = Spline</i>
73	<i>Flag di creazione della direttrice (default = 3): 0 = direttrice creata con annotazione di testo 1 = creata con annotazione di tolleranza 2 = creata con annotazione di riferimento blocco 3 = creata senza alcuna annotazione</i>
74	<i>Flag di direzione della linea di aggancio: 0 = linea di aggancio (o fine della tangente per una direttrice spline) ha una direzione opposta al vettore orizzontale. 1 = linea di aggancio (o fine della tangente per una direttrice spline) ha la stessa direzione del vettore orizzontale (vedere il codice 75).</i>
75	<i>Flag di linea di aggancio: 0 = nessuna linea di aggancio; 1 = è presente una linea di aggancio</i>
40	<i>Altezza dell'annotazione del testo</i>
41	<i>Larghezza dell'annotazione del testo</i>
76	<i>Numero di vertici della direttrice (ignorato per DXFIN)</i>
10	<i>Coordinate dei vertici (una voce per ciascun vertice) DXF: X; APP: punto 3D</i>
20, 30	<i>DXF: valore Y e Z delle coordinate dei vertici</i>
77	<i>Colore da utilizzare se per la direttrice DIMCLRD=DABLOCCO</i>
340	<i>Riferimento hard all'annotazione associata (entità mtext, tolerance o insert)</i>
210	<i>Vettore normale. DXF: valore X; APP: vettore 3D</i>
220, 230	<i>DXF: valore Y e Z del vettore normal</i>
211	<i>Direzione "orizzontale" per la direttrice. DXF: X; APP: vettore 3D</i>
221,231	<i>DXF: valore Y e Z della direzione "orizzontale" per la direttrice</i>
212	<i>Sfalsamento del punto di inserimento del riferimento di blocco dall'ultimo vertice della direttrice. DXF: valore X; APP: vettore 3D</i>

222,232	<i>DXF: valore Y e Z dello sfalsamento del punto di inserimento del riferimento di blocco dall'ultimo vertice della direttrice.</i>
213	<i>Sfalsamento del punto di posizionamento dell'annotazione dall'ultimo vertice della direttrice. DXF: valore X; APP: vettore 3D</i>
223,233	<i>DXF: valore Y e Z dello sfalsamento del punto di posizionamento dell'annotazione dall'ultimo vertice della direttrice.</i>

Possono seguire dati estesi appartenenti all'ID applicazione "ACAD". Tali dati descrivono qualsiasi sostituzione applicata a questa entità. Vedere "Sostituzioni degli stili di quota."

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

LINE

I seguenti codici di gruppo sono validi per le entità line (linea).

Codici di gruppo line

Codici di gruppo	Descrizione
100	<i>Contrassegno di sottoclasse (AcDbLine)</i>
39	<i>Spessore (opzionale; default = 0)</i>
10	<i>Punto iniziale (in WCS). DXF: valore X; APP: punto 3D</i>
20, 30	<i>DXF: valore Y e Z del punto iniziale (in WCS)</i>
11	<i>Punto finale (in WCS). DXF: valore X; APP: punto 3D</i>
21, 31	<i>DXF: valore Y e Z del punto finale (in WCS)</i>
210	<i>Direzione di estrusione (opzionale; default = 0, 0, 1). DXF: X; APP: vettore 3D</i>
220, 230	<i>DXF: valore Y e Z della direzione di estrusione</i>

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

LWPOLYLINE

I seguenti codici di gruppo sono validi per le entità lwpolyline (polilinea ottimizzata).

Codici di gruppo lwpolyline

Codici di gruppo	Descrizione
100	<i>Contrassegno di sottoclasse (AcDbPolyline)</i>
90	<i>Numero di vertici.</i>
70	<i>Flag della polilinea (codificato in bit); default è 0: 1 = chiusa 128 = plinegen</i>
43	<i>Larghezza costante (opzionale; default = 0) Non usata se è impostata la larghezza variabile (codici 40 e/o 41).</i>
38	<i>Elevazione (opzionale; default = 0)</i>
39	<i>Spessore (opzionale; default = 0)</i>
10	<i>Coordinate del vertice (in WCS), voci multiple; una voce per ogni vertice. DXF: valore X; APP: punto 2D</i>
20	<i>DXF: valore Y del vertice (in WCS), voci multiple; una voce per ogni vertice</i>
40	<i>Larghezza iniziale (voci multiple; una voce per ogni vertice) (opzionale; default = 0; voci multiple). Non utilizzata se è impostata la larghezza costante (codice 43).</i>
41	<i>Larghezza finale (voci multiple; una voce per ogni vertice) (opzionale; default = 0; voci multiple). Non utilizzata se è impostata la larghezza costante (codice 43).</i>
42	<i>Curvatura (voci multiple; una voce per ogni vertice) (opzionale;</i>

	<i>default = 0).</i>
210	<i>Direzione di estrusione (opzionale; default = 0, 0, 1).</i>
	<i>DXF: X; APP: vettore 3D</i>
220, 230	<i>DXF: valore Y e Z della direzione di estrusione</i>

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

MLINE

I seguenti codici di gruppo sono validi per le entità di tipo mline (lineam).

Codici di gruppo mline

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbMline)</i>
2	<i>Stringa composta da un massimo di 32 caratteri. Il nome dello stile utilizzato per questa mline. Deve esistere una voce per questo stile nel dizionario MLINESTYLE.</i>
	Nota <i>Non modificare questo campo.</i>
340	<i>Puntatore-gestore/ID del dizionario MLINESTYLE</i>
40	<i>Fattore di scala</i>
70	<i>Giustificazione: 0=superiore, 1=zero, 2=inferiore</i>
71	<i>Flag di apertura/chiusura: 1=aperta, 3=chiusa</i>
72	<i>Numero di vertici</i>
73	<i>Numero di elementi nella definizione MLINESTYLE</i>
10	<i>Punto iniziale (in WCS). DXF: valore X; APP: punto 3D</i>
20, 30	<i>DXF: valore Y e Z del punto iniziale (in WCS)</i>
210	<i>Direzione di estrusione (opzionale; default = 0, 0, 1).</i>
	<i>DXF: X; APP: vettore 3D</i>
220, 230	<i>DXF: valore Y e Z della direzione di estrusione</i>
11	<i>Coordinate del vertice (voci multiple; una voce per ogni vertice).</i>
	<i>DXF: valore X; APP: punto 3D</i>
21, 31	<i>DXF: valore Y e Z delle coordinate dei vertici</i>
12	<i>Vettore di direzione del segmento avente inizio da questo vertice (voci multiple: una voce per ogni vertice).</i>
	<i>DXF: X; APP: vettore 3D</i>
22, 32	<i>DXF: valore del vettore di direzione del segmento avente inizio da questo vertice</i>
13	<i>Vettore di direzione di giunzione a questo vertice (voci multiple, una per ciascun vertice). DXF: valore X; APP: vettore 3D</i>
23, 33	<i>DXF: valore Y e Z del vettore di direzione di giunzione</i>
74	<i>Numero di parametri per questo elemento (si ripete per ciascun elemento del segmento)</i>
41	<i>Parametri degli elementi (si ripete in base al codice 74 precedente)</i>
75	<i>Numero di parametri di riempimento area per questo elemento (si ripete per ciascun elemento del segmento)</i>
42	<i>Parametri di riempimento area (si ripete in base al codice 75 precedente)</i>

I parametri del codice di gruppo 41 sono costituiti da un elenco di valori reali, un valore reale per ogni codice di gruppo 41. Il primo valore del codice di gruppo 41 è la distanza lungo il vettore di giunzione dal vertice del segmento al punto in cui il percorso dell'elemento linea interseca il vettore di giunzione. Il valore successivo del codice di gruppo 41 è la distanza lungo il percorso dell'elemento linea dal punto definito dal primo gruppo 41 all'inizio effettivo dell'elemento della linea. Il valore successivo è la distanza dall'inizio dell'elemento linea alla prima interruzione (o taglio) dell'elemento linea. I valori successivi del codice di gruppo 41 continuano l'elenco dei punti di inizio e fine dell'elemento linea in questo segmento della mline. I tipi di linea non influiscono sugli elenchi del gruppo 41.

Anche i parametri del codice di gruppo 42 sono un elenco di valori reali. Analogamente ai parametri del gruppo 41, questo elenco descrive i parametri dell'area di riempimento per questo segmento mline. I valori vengono interpretati nello stesso modo dei parametri del gruppo 41 e, se presi in blocco per tutti gli elementi di linea nel segmento mline, definiscono il contorno dell'area di riempimento per il segmento mline.

Un esempio comune dell'uso del meccanismo del codice di gruppo 42 si ha quando una lineam non riempita interseca una lineam riempita e viene utilizzato il comando editam1 per fare in modo che la lineam riempita appaia non riempita nell'area di intersecazione.

Ciò comporta in due gruppi del codice 42 per ciascun elemento linea del segmento mline, uno per l'interruzione e uno per l'inizio del riempimento.

I codici di gruppo 2 nelle entità mline e negli oggetti mlinestyle sono campi ridondanti. Questi gruppi *non* devono essere modificati in nessun caso, anche se è consigliabile leggerli ed utilizzarne i valori. I campi da modificare sono i seguenti:

Mline

Il gruppo 340 nello stesso oggetto, che indica l'oggetto mlinestyle appropriato.

Mlinestyle

Il valore del gruppo 3 nel dizionario mlinestyle che precede il gruppo 350 con il gestore o il nome entità mlinestyle corrente.

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

MTEXT

I seguenti codici di gruppo sono validi per le entità mtext (testom).

Codici di gruppo mtext

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (AcDbMText)
10	Punto di inserimento DXF: valore X; APP: punto 3D
20, 30	DXF: valore Y e Z del punto di inserimento
40	Altezza del testo di default
41	Larghezza del rettangolo di riferimento
71	Punto di unione: 1 = In alto a sinistra; 2 = In alto al centro; 3 = In alto a destra; 4 = In mezzo a sinistra; 5 = In mezzo al centro; 6 = In mezzo a destra 7 = In basso a sinistra; 8 = In basso al centro; 9 = In basso a destra
72	Direzione del disegno: 1 = Da sinistra a destra; 2 = Da destra a sinistra 3 = Dall'alto in basso; 4 = Dal basso in alto
1	Stringa di testo. Se la stringa di testo è inferiore a 250 caratteri, tutti i caratteri appaiono nel gruppo 1. Se la stringa di testo è maggiore di 250 caratteri, la stringa verrà divisa in frammenti di 250 caratteri ciascuno, che appaiono in uno o più codici di gruppo 3. Se vengono utilizzati i codici di gruppo 3, l'ultimo gruppo è un gruppo 1 ed ha meno di 250 caratteri.
3	Testo esteso (sempre in frammenti di 250 caratteri).
7	Nome dello stile di testo (se non è specificato: STANDARD)
210	Direzione di estrusione (opzionale; default = 0, 0, 1). DXF: X; APP: vettore 3D
220, 230	DXF: valore Y e Z della direzione di estrusione
11	Vettore di direzione dell'asse X (in WCS). DXF: X; APP: vettore 3D Nota: un codice di gruppo 50 (angolo di rotazione in radianti) passato come input DXF viene convertito nel vettore di direzione equivalente (se vengono passati sia un codice 50 che i codici 11,21,31, l'ultimo avrà la priorità). Questo codice consente di effettuare in modo conveniente le conversioni da oggetti di testo.
21, 31	DXF: valore Y e Z del vettore di direzione dell'asse X (in WCS)
42	Larghezza orizzontale dei caratteri che costituiscono l'oggetto testom. (Questo valore sarà sempre uguale o inferiore al codice di gruppo 41.)
43	Larghezza verticale dei caratteri che costituiscono l'oggetto testom.
50	Angolo di rotazione in radianti.

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

OLEFRAME

I seguenti codici di gruppo sono validi per le entità oleframe.

Codici di gruppo oleframe

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (AcDbOleFrame)
70	Numero di versione OLE
90	Lunghezza dei dati binari
310	Dati binari (più righe)
1	Fine dei dati Ole (la stringa "OLE")

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

OLE2FRAME

I seguenti codici di gruppo sono validi per le entità ole2frame. Queste informazioni sono di sola lettura. Quando si esegue DXFIN, i valori vengono ignorati, in quanto fanno parte dell'oggetto binario OLE, e vengono ottenuti tramite funzioni di accesso.

Codici di gruppo ole2frame

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (AcDbOle2Frame)
70	Numero di versione OLE
3	Lunghezza dei dati binari
10	Angolo superiore sinistro (WCS). DXF: valore X; APP: punto 3D
20, 30	DXF: valore Y e Z dell'angolo superiore sinistro (in WCS)
11	Angolo inferiore destro (WCS). DXF: valore X; APP: punto 3D
21, 31	DXF: valore Y e Z dell'angolo inferiore destro (in WCS)
71	Tipo di oggetto OLE, 1 = oggetto collegato, 2 = oggetto incorporato, 3 = oggetto statico
72	Descrittore della modalità finestra: 0 = l'oggetto è incluso in una finestra con modello a caselle 1 = l'oggetto è incluso in una finestra senza caselle (spazio carta o spazio a modello mobile)
90	Lunghezza dei dati binari
310	Dati binari (più righe)
1	Fine dei dati Ole (la stringa "OLE")

Esempio di output DXF:

```
OLE2FRAME
  5
  2D
  100
  AcDbEntity
    67
      1
        8
          0
            100
              AcDbOle2Frame
                70
                  2
                    3
                      Immagine Paintbrush
```

```

10
4.43116
20
5.665992
30
0.0
11
6.4188
21
4.244939
31
0.0
71
2
72
1
90
23680
310
0155764BD60082B91140114B08C8F9A916400000000000000000506DC0D0D9AC
310
1940114B08C8F9A9164000000000000000000000506DC0D0D9AC194002303E5CD1FA
310
1040000000000000000000000764BD60082B9114002303E5CD1FA1040000000000000
...
...

```

Esempio di output della funzione AutoLISP entnext:

```

Comando: (setq e (entget e3))
((-1 . <Nome entità: 7d50428>) (0 . "OLE2FRAME") (5 . "2D")
(100 . "AcDbEntity") (67 . 1) (8 . "0") (100 . "AcDbOle2Frame")
(70 . 2) (3 "Immagine Paintbrush") (10 4.43116 5.66599 0.0)
(11 6.4188 4.24494 0.0) (71 . 2) (72 . 1))

```

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

POINT

I seguenti codici di gruppo sono validi per le entità point (punto).

Codici di gruppo point

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbPoint)</i>
10	<i>Posizione del punto (in WCS). DXF: valore X; APP: punto 3D</i>
20, 30	<i>DXF: valore Y e Z della posizione del punto (in WCS)</i>
39	<i>Spessore (opzionale; default = 0)</i>
210	<i>Direzione di estrusione (opzionale; default = 0, 0, 1). DXF: X; APP: vettore 3D</i>
220, 230	<i>DXF: valore Y e Z della direzione di estrusione</i>
50	<i>Angolo dell'asse X dell'UCS effettivo quando è stato disegnato il punto (opzionale; default: 0). Utilizzato quando PDMODE è diverso da zero</i>

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

POLYLINE

I seguenti codici di gruppo sono validi per le entità polyline (polilinea).

Codici di gruppo polyline

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (<i>AcDb2dPolyline</i> o <i>AcDb3dPolyline</i>)
10	DXF: sempre 0 APP: un punto "dummy"; i valori X ed Y sono sempre 0 ed il valore Z è l'elevazione della polilinea (in OCS se 2D, in WCS se 3D)
20	DXF: sempre 0
30	DXF: elevazione della polilinea (in OCS se 2D, in WCS se 3D)
39	Spessore (opzionale; default = 0)
70	Flag della polilinea (codificato in bit); default è 0: 1 = Questa è una polilinea chiusa (o una mesh poligonale chiusa verso la direzione M). 2 = Sono stati aggiunti dei vertici adattati alla curva. 4 = Sono stati aggiunti dei vertici adattati alla spline. 8 = Questa è una polilinea 3D. 16 = Questa è una mesh poligonale 3D. 32 = La mesh poligonale è chiusa verso la direzione N. 64 = Questa polilinea è una mesh poliedrica. 128 = Il modello del tipo di linea è generato in maniera continua attorno al vertice di questa polilinea.
40	Larghezza iniziale di default (opzionale; default = 0)
41	Larghezza finale di default (opzionale; default = 0)
71	Calcolo del vertice M della mesh poligonale (opzionale; default = 0)
72	Calcolo del vertice N della mesh poligonale (opzionale; default = 0)
73	Densità della superficie levigata M (opzionale; default = 0)
74	Densità della superficie levigata N (opzionale; default = 0)
75	Tipo di superficie levigata e curve (opzionale; default = 0); codici interi, non codificati in bit: 0 = Nessuna superficie levigata adattata 5 = Superficie quadratica B-spline 6 = Superficie cubica B-spline 8 = Superficie di Bezier
210	Direzione di estrusione (opzionale; default = 0, 0, 1). DXF: X; APP: vettore 3D
220, 230	DXF: valore Y e Z della direzione di estrusione

Possono seguire dati estesi con l'ID applicazione "AUTOCAD_POSTSCRIPT_FIGURE". Questi dati contengono informazioni relative alle immagini PostScript e informazioni di riempimento PostScript.

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

POLYLINE

Mesh poliedriche

Una mesh poliedrica è rappresentata nei DXF come una variante di un'entità polyline. L'intestazione della polilinea introduce una mesh poliedrica quando il gruppo (70) dei flag della polilinea contiene il bit 64. Il gruppo 71 specifica il numero di vertici della mesh e il gruppo 72 specifica il numero di facce. Benché questi conteggi siano corretti per tutte le mesh create con il comando PFACE, non è necessario che le applicazioni inseriscano i valori corretti in questi campi. L'intestazione della polilinea è seguita da una sequenza di entità vertex che specifica le coordinate dei vertici e dalle facce che compongono la mesh.

La struttura delle entità di AutoCAD limita il numero di vertici che può essere specificato da una determinata entità face. È possibile rappresentare poligoni più complessi scomponendoli in cunei triangolari. È necessario rendere invisibili gli spigoli per impedire che vengano disegnate le parti visibili di questa suddivisione. Il comando POLIMESH esegue questa suddivisione automaticamente, ma quando le applicazioni generano direttamente mesh poliedriche, è necessario che le applicazioni eseguano tale suddivisione. Il numero di vertici per faccia è il parametro chiave di questa suddivisione. La variabile di sistema PFACEVMAX fornisce ad un'applicazione un determinato numero di vertici per l'elemento faccia. Questo valore è a sola lettura ed è impostato a 4.

Le mesh poliedriche create con il comando POLIMESH vengono sempre generate inserendo prima le entità per le coordinate dei vertici, seguite dalle entità di definizione delle facce. Il codice interno di AutoCAD che elabora le mesh poliedriche richiede questo ordine. I programmi che generano mesh poliedriche nei DXF devono generare tutti i vertici, poi tutte le facce. Tuttavia, i programmi che leggono le mesh poliedriche dai DXF devono avere un margine di tolleranza per un eventuale ordine diverso di facce e vertici.

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

RAY

I seguenti codici di gruppo sono validi per le entità ray (raggio).

Codici di gruppo ray

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (AcDbRay)
10	Punto iniziale (in WCS). DXF: valore X; APP: punto 3D
20, 30	DXF: valore Y e Z del punto iniziale (in WCS)
11	Vettore di direzione unità (in WCS). DXF: X; APP: vettore 3D
21, 31	DXF: valore Y e Z del vettore di direzione unità (in WCS)

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

REGION

I seguenti codici di gruppo sono validi per le entità region (regione).

Codici di gruppo region

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (AcDbModelerGeometry)
70	Numero di versione del formato del modellatore (correntemente = 1)
1	Dati di proprietà (più righe < 255 caratteri ciascuna)
3	Righe supplementari dei dati di proprietà (se una stringa del gruppo 1 era maggiore di 255 caratteri)

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

SEQEND

I seguenti codici di gruppo sono validi per le entità seqend.

Codici di gruppo seqend

Codici di gruppo	<i>Descrizione</i>
-2	<i>APP: nome dell'entità che ha iniziato la sequenza. Questa entità segna la fine del vertice (nome del tipo di vertice) per una polilinea o la fine delle entità attrib (nome di tipo attrib) per un'entità insert che ha degli attributi (indicati dalla presenza del gruppo 66 e da un valore diverso da zero nell'entità insert). Questo codice non viene salvato in un file DXF.</i>

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

SHAPE

I seguenti codici di gruppo sono validi per le entità shape (forma).

Codici di gruppo shape

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbShape)</i>
39	<i>Spessore (opzionale; default = 0)</i>
10	<i>Punto di inserimento (in WCS). DXF: valore X; APP: punto 3D</i>
20, 30	<i>DXF: valore Y e Z del punto di inserimento (in WCS)</i>
40	<i>Dimensione</i>
2	<i>Nome della forma</i>
50	<i>Angolo di rotazione (opzionale, default = 0)</i>
41	<i>Fattore di scala X relativo (opzionale; default = 1)</i>
51	<i>Angolo obliquo (opzionale, default = 0):</i>
210	<i>Direzione di estrusione (opzionale; default = 0, 0, 1). DXF: X; APP: vettore 3D</i>
220, 230	<i>DXF: valore Y e Z della direzione di estrusione</i>

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

SOLID

I seguenti codici di gruppo sono validi per le entità solid (solido).

Codici di gruppo solid

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbTrace)</i>
10	<i>Primo angolo. DXF: valore X; APP: punto 3D</i>
20, 30	<i>DXF: valore Y e Z del primo angolo</i>
11	<i>Secondo angolo. DXF: valore X; APP: punto 3D</i>
21, 31	<i>DXF: valore Y e Z del secondo angolo</i>
12	<i>Terzo angolo. DXF: valore X; APP: punto 3D</i>
22, 32	<i>DXF: valore Y e Z del terzo angolo</i>
13	<i>Quarto angolo (se vengono inseriti solo tre punti per definire SOLID, questo angolo prende i valori del terzo). DXF: X; APP: punto 3D</i>
23, 33	<i>DXF: valore Y e Z del quarto angolo</i>
39	<i>Spessore (opzionale; default = 0)</i>
210	<i>Direzione di estrusione (opzionale; default = 0, 0, 1). DXF: X; APP: vettore 3D</i>
220, 230	<i>DXF: valore Y e Z della direzione di estrusione</i>

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

SPLINE

I seguenti codici di gruppo sono validi per le entità spline.

Codici di gruppo spline

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (<i>AcDbSpline</i>)
210	Vettore normale (omesso se la spline non è planare) DXF: X; APP: vettore 3D
220, 230	DXF: valore Y e Z del vettore normale
70	Flag spline (codificato in bit): 1 = Spline chiusa 2 = Spline periodica 4 = Spline razionale 8 = Planare 16 = Lineare (è impostato anche il bit planare)
71	Grado della curva spline
72	Numero di nodi
73	Numero di punti di controllo
74	Numero di punti di adattamento (se presenti)
42	Tolleranza dei nodi (default = 0.0000001)
43	Tolleranza del punto di controllo (default = 0.0000001)
44	Tolleranza di adattamento (default = 0.000000001)
12	Tangente iniziale: può essere omessa (in WCS). DXF: X; APP: punto 3D
22, 32	DXF: valore Y e Z della tangente iniziale; possono essere omessi (in WCS).
13	Tangente finale: può essere omessa (in WCS). DXF: X; APP: punto 3D
23, 33	DXF: valore Y e Z della tangente finale; possono essere omessi (in WCS).
40	Valore del nodo (una voce per nodo)
41	Peso (se il valore è diverso da 1); con più coppie di gruppi, sono presenti se i valori sono tutti diversi da 1
10	Punti di controllo (in WCS); una voce per punto di controllo. DXF: X; APP: punto 3D
20, 30	DXF: valore Y e Z dei punti di controllo (in WCS) (una voce per punto di controllo)
11	Punti di adattamento (in WCS); una voce per punto di adattamento. DXF: X; APP: punto 3D
21, 31	DXF: valore Y e Z dei punti di adattamento (in WCS) (una voce per punto di adattamento)

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

TEXT

I seguenti codici di gruppi sono validi per le entità text (testo).

Codici di gruppo text

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (<i>AcDbText</i>)
39	Spessore (opzionale; default = 0)
10	Primo punto di allineamento (in OCS). DXF: X; APP: punto 3D

20, 30	<i>DXF: valore Y e Z del primo punto di allineamento (in OCS)</i>
40	<i>Altezza testo</i>
1	<i>Valore di default (la stringa stessa)</i>
50	<i>Rotazione testo (opzionale, default = 0):</i>
41	<i>Fattore di scala X relativo: larghezza (opzionale; default = 1). Questo valore viene regolato anche quando si utilizza il testo di tipo Adatta.</i>
51	<i>Angolo obliquo (opzionale, default = 0):</i>
7	<i>Nome dello stile di testo (opzionale, default = STANDARD)</i>
71	<i>Flag di generazione del testo (opzionale, default = 0): 2 = Testo rovesciato (speculare rispetto ad X), 4 = Testo capovolto (speculare rispetto a Y)</i>
72	<i>Tipo di giustificazione del testo orizzontale (opzionale, default = 0) codici interi (non codificati in bit) 0 = A sinistra; 1= In centro; 2 = A destra 3 = Allineato (se allineamento verticale = 0) 4 = Centrato (se allineamento verticale = 0) 5 = Adattato (se allineamento verticale = 0) Per maggiore chiarezza, vedere la seguente tabella</i>
11	<i>Secondo punto di allineamento (in OCS). DXF: X; APP: punto 3D Questo valore è significativo solo se il valore di un gruppo 72 o 73 è diverso da zero (se la giustificazione non è impostata su Linea di base/A sinistra)</i>
21, 31	<i>DXF: valore Y e Z del secondo punto di allineamento (in OCS)</i>
210	<i>Direzione di estrusione (opzionale; default = 0, 0, 1). DXF: X; APP: vettore 3D</i>
220, 230	<i>DXF: valore Y e Z della direzione di estrusione</i>
100	<i>Contrassegno di sottoclasse (AcDbText)</i>
73	<i>Tipo di giustificazione del testo verticale (opzionale, default = 0): codici interi (non codificati in bit) 0 = Linea di base; 1 = In basso; 2 = In centro; 3 = In alto Per maggiore chiarezza, vedere la seguente tabella.</i>

La tabella seguente descrive più dettagliatamente i codici dei gruppi 72 (allineamento orizzontale) e 73 (allineamento verticale).

Codici interi dei gruppi 72 e 73

	<i>Gruppo 72</i>					
Gruppo 73	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>
3 (in alto)	<i>ASinistra</i>	<i>ACentro</i>	<i>ADestra</i>			
2 (in centro)	<i>MSinistra</i>	<i>MCentro</i>	<i>MDestra</i>			
1 (in basso)	<i>BSinistra</i>	<i>BCentro</i>	<i>BDestra</i>			
0 (linea di base)	<i>A sinistra</i>	<i>Centrato</i>	<i>A destra</i>	<i>Allineato</i>	<i>In centro</i>	<i>Adattato</i>

Se i valori del gruppo 72 o 73 (o di entrambi) sono diversi da zero, i valori del primo punto di allineamento vengono ignorati ed AutoCAD calcola i nuovi valori in base al secondo punto di allineamento ed alla lunghezza ed all'altezza della stringa di testo stessa (dopo l'applicazione dello stile di testo). Se i valori 72 e 73 sono uguali a zero o mancanti, il secondo punto di allineamento non è significativo.

Appendice C -- Codici di gruppo DXF

 **Sezione ENTITIES**

 **TOLERANCE**

I seguenti codici di gruppo sono validi per le entità tolerance (tolleranza).

Codici di gruppo tolerance

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbFcf)</i>
3	<i>Nome dello stile di quota</i>
10	<i>Punto di inserimento (in WCS). DXF: valore X; APP: punto 3D</i>
20, 30	<i>DXF: valore Y e Z del punto di inserimento (in WCS)</i>
210	<i>Direzione di estrusione (opzionale; default = 0, 0, 1). DXF: X; APP: vettore 3D</i>

220, 230	<i>DXF: valore Y e Z della direzione di estrusione</i>
11	<i>Vettore di direzione dell'asse X (in WCS)</i>
	<i>DXF: X; APP: vettore 3D</i>
21, 31	<i>DXF: valore Y e Z del vettore di direzione dell'asse X (in WCS)</i>

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

TRACE

I seguenti codici di gruppo sono validi per le entità trace (traccia).

Codici di gruppo trace

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbTrace)</i>
10	<i>Primo angolo (in OCS). DXF: valore X; APP: punto 3D</i>
20, 30	<i>DXF: valore Y e Z del primo angolo (in OCS)</i>
11	<i>Secondo angolo (in OCS). DXF: valore X; APP: punto 3D</i>
21, 31	<i>DXF: valore Y e Z del secondo angolo (in OCS)</i>
12	<i>Terzo angolo (in OCS). DXF: valore X; APP: punto 3D</i>
22, 32	<i>DXF: valore Y e Z del terzo angolo (in OCS)</i>
13	<i>Quarto angolo (in OCS). DXF: valore X; APP: punto 3D</i>
23, 33	<i>DXF: valore Y e Z del quarto angolo (in OCS)</i>
39	<i>Spessore (opzionale; default = 0)</i>
210	<i>Direzione di estrusione (opzionale; default = 0, 0, 1).</i>
	<i>DXF: X; APP: vettore 3D</i>
220, 230	<i>DXF: valore Y e Z della direzione di estrusione</i>

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

VERTEX

I seguenti codici di gruppo sono validi per le entità vertex (vertice).

Codici di gruppo vertex

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbVertex)</i>
100	<i>Contrassegno di sottoclasse (AcDb2dVertex o AcDb3dPolylineVertex)</i>
10	<i>Posizione (in OCS se 2D, in WCS se 3D).</i>
	<i>DXF: X; APP: punto 3D</i>
20, 30	<i>DXF: valore Y e Z della posizione (in OCS se 2D, in WCS se 3D)</i>
40	<i>Larghezza iniziale (opzionale; default è 0)</i>
41	<i>Larghezza finale (opzionale; default è 0)</i>
42	<i>Curvatura (opzionale; default è 0). La curvatura è la tangente di un quarto dell'angolo incluso per un segmento di arco, resa negativa se la direzione dell'arco è in senso orario dal punto iniziale al punto finale. Una curvatura pari a 0 indica un segmento di linea retta e una curvatura pari a 1 è un semicerchio.</i>
70	<i>Flag di vertex:</i>
	<i>1 = Vertice supplementare creato per adattamento alla curva.</i>
	<i>2 = Tangente di adattamento alla curva definita per questo vertice. La direzione della tangente di adattamento con valore 0 può essere omessa dall'output del DXF, ma può risultare significativa se questo bit è impostato.</i>

	4 = Non utilizzato
	8 = Vertice spline di adattamento
	16 = Punto di controllo della cornice di spline
	32 = Vertice della polilinea tridimensionale
	64 = Vertice della mesh poligonale tridimensionale
	128 = Vertice della mesh poliedrica
50	Direzione della tangente di adattamento alla curva
71	Indice dei vertici della mesh poliedrica. Opzionale. Presente solo se il valore è diverso da zero.
72	Indice dei vertici della mesh poliedrica. Opzionale. Presente solo se il valore è diverso da zero.
73	Indice dei vertici della mesh poliedrica. Opzionale. Presente solo se il valore è diverso da zero.
74	Indice dei vertici della mesh poliedrica. Opzionale. Presente solo se il valore è diverso da zero.

Ciascun vertice facente parte di una mesh poliedrica ha il bit 128 dei flag dei vertici impostato. Se l'entità fornisce la coordinata di un vertice della mesh, è impostato anche il bit 64 ed i gruppi 10, 20, 30 forniscono la coordinata del vertice. I valori dell'indice dei vertici sono determinati dall'ordine in cui le entità dei vertici appaiono nella polilinea e la numerazione del primo è 1.

Se il vertice definisce una faccia della mesh, il relativo gruppo dei flag dei vertice ha il bit 128 impostato, ma non il bit 64. In questo caso, i gruppi 10, 20, 30 (posizione) dell'entità face sono irrilevanti e vengono sempre scritti come 0 in un file DXF. Gli indici dei vertici che definiscono la mesh vengono forniti dai codici di gruppo 71, 72, 73 e 74, i cui valori specificano uno dei vertici precedentemente definiti dall'indice. Se l'indice è negativo, lo spigolo che inizia con quel vertice è invisibile. Il primo vertice 0 indica la fine dei vertici della faccia.

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

VIEWPORT

I seguenti codici di gruppo sono validi per le entità viewport (finestra).

Codici di gruppo viewport

Codici di gruppo

	Descrizione
100	Contrassegno di sottoclasse (AcDbViewport)
10	Punto centrale (in WCS). DXF: valore X; APP: punto 3D
20, 30	DXF: valore Y e Z del punto centrale (in WCS)
40	Larghezza in unità di spazio carta
41	Altezza in unità di spazio carta
68	Campo di stato della finestra: 1 = On, ma non visualizzata oppure è una delle finestre non attive perché il conteggio di \$MAXACTVP è stato superato. 0 = Off <valore positivo > = On ed attiva. Il valore indica l'ordine di sovrapposizione delle finestre, dove 1 è la finestra attiva, 2 è la finestra successiva e così via.
69	Identificatore della finestra. Viene modificato ad ogni apertura di un disegno. Non viene mai salvato, tranne che per la finestra dello spazio carta, che è sempre 1.

La seguente tabella elenca i dati estesi che possono essere collegati alle entità viewport. Contrariamente ai normali dati delle entità, l'ordine dei codici di gruppo estesi è importante. Inoltre, i codici di gruppo possono essere presenti più volte.

Codici di gruppo dei dati estesi viewport

Codici di gruppo	Descrizione
1001	ID applicazione ("ACAD"). Inizia una sezione di dati estesi che descrive la finestra.
1000	Inizio dei dati della finestra. Questo campo è sempre la stringa MVIEW. In futuro, potranno essere presenti altri gruppi di dati.
1002	Inizio dei dati di descrizione della finestra. Questo campo è sempre la stringa.
1070	Numero di versione dei dati estesi. Sempre il numero intero 16.
1010	Punto di destinazione della vista (in WCS). DXF: valore X; APP: punto 3D

1020,1030	<i>DXF: valore Y e Z del punto di destinazione della vista (in WCS)</i>
1010	<i>Vettore di direzione della vista (in WCS).</i>
	<i>DXF: X; APP: vettore 3D</i>
1020,1030	<i>DXF: valore Y e Z del vettore di direzione della vista (in WCS)</i>
1040	<i>Angolo di inclinazione della vista</i>
1040	<i>Altezza della vista</i>
1040	<i>Valore X del punto centrale della vista (in DCS)</i>
1040	<i>Valore Y del punto centrale della vista (in DCS)</i>
1040	<i>Lunghezza dell'obiettivo della prospettiva</i>
1040	<i>Valore Z del piano di ritaglio anteriore</i>
1040	<i>Valore Z del piano di ritaglio posteriore</i>
1070	<i>Modalità visualizzazione</i>
1070	<i>Zoom cerchio</i>
1070	<i>Impostazione zoom veloce</i>
1070	<i>Impostazione ICONAUCS</i>
1070	<i>Snap ON/OFF</i>
1070	<i>Griglia ON/OFF</i>
1070	<i>Stile snap</i>
1070	<i>Snap ISOPAIR</i>
1040	<i>Angolo snap</i>
1040	<i>Valore X UCS del punto base dello snap della coordinata</i>
1040	<i>Valore Y UCS del punto base dello snap della coordinata</i>
1040	<i>Intervallo dello snap X</i>
1040	<i>Intervallo dello snap Y</i>
1040	<i>Spaziatura della griglia X</i>
1040	<i>Spaziatura della griglia Y</i>
1070	<i>Flag nascosto nella stampa</i>
1002	<i>Inizio dell'elenco dei layer congelati (possibilmente vuoto). Questo campo è sempre la stringa "{}".</i>
1003	<i>I nomi dei layer congelati in questa finestra. Questo elenco può comprendere layer dipendenti da riferimenti xref. Qui può essere presente qualsiasi numero di gruppi 1003.</i>
1002	<i>Fine dell'elenco dei layer congelati. Questo campo è sempre la stringa "}".</i>
1002	<i>Fine dei dati della finestra. Questo campo è sempre la stringa "}".</i>

Nota Il fattore ZOOM XP viene calcolato con la formula : $\text{group_41} / \text{2nd_group_1040}$ (o $\text{pspace_height} / \text{mspace_height}$).

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

XLINE

I seguenti codici di gruppo sono validi per le entità xline (xlinea).

Codici di gruppo xline

Codici di gruppo	Descrizione
100	<i>Contrassegno di sottoclasse (AcDbXline)</i>
10	<i>Primo punto (in WCS). DXF: valore X; APP: punto 3D</i>
20, 30	<i>DXF: valore Y e Z del primo punto (in WCS)</i>
11	<i>Vettore di direzione unità (in WCS).</i>
	<i>DXF: X; APP: vettore 3D</i>
21, 31	<i>DXF: valore Y e Z del vettore di direzione unità (in WCS)</i>

Appendice C -- Codici di gruppo DXF

Sezione ENTITIES

ACAD_PROXY_ENTITY

I seguenti codici di gruppo sono validi per le entità proxy.

Codici di gruppo Acad_proxy_entity

Codici di gruppo	Descrizione
100	DXF: AcDbProxyEntity
90	DXF: ID classe dell'entità proxy
91	DXF: ID classe dell'entità applicazione effettiva
92	DXF: Dimensioni dei dati del grafico in byte
310	DXF: Dati binari del grafico (possono comparire più voci)
93	DXF: Dimensioni dei dati dell'entità in bit
310	DXF: Dati binari dell'entità (possono comparire più voci)
330 o 340 o 350 o 360	DXF: Un ID oggetto (possono comparire più voci)
94	DXF: 0 (indica la fine della sezione dell'ID oggetto)

Appendice C -- Codici di gruppo DXF

Sezione OBJECTS

I codici di gruppo descritti in questa sezione sono validi per gli oggetti non grafici. Questi codici si trovano nella sezione ENTITIES di un file DXF e vengono utilizzati dalle applicazioni AutoLISP e ARX negli elenchi di definizioni delle entità.

I codici di gruppo riportati in questa sezione possono essere validi per i file DXF, per le applicazioni (AutoLISP, ADS o ARX) o per entrambi. Quando la descrizione di un codice è diversa per le applicazioni e per i file DXF (oppure è valida solo in uno dei due casi), essa è preceduta dai seguenti indicatori:

APP

Descrizione specifica per l'applicazione

DXF

Descrizione specifica per i file DXF

Se la descrizione è comune ai file DXF ed alle applicazioni, non è presente alcun indicatore. In caso contrario, è presente l'indicatore appropriato. I codici opzionali sono riportati in grigio.

Gli oggetti sono simili alle entità, con l'eccezione che essi non hanno un significato grafico o geometrico. In questa sezione sono contenuti tutti gli oggetti che non sono entità o record della tabella di simboli o tabelle di simboli. Questa sezione rappresenta un gruppo di oggetti omogeneo, con gli oggetti ordinati topologicamente in base alla proprietà, in modo che i proprietari compaiano sempre prima degli oggetti da loro posseduti.

Il proprietario principale di tutti gli oggetti che compaiono in questa sezione è il dizionario degli oggetti denominati, che quindi è sempre il primo oggetto che compare in questa sezione. Gli oggetti di questa sezione possono essere definiti da AutoCAD o da applicazioni che hanno accesso a ARX API. I nomi DXF dei tipi di oggetti definiti dalle applicazioni dovrebbero sempre essere associati al nome di una classe nella sezione CLASS del file DXF, altrimenti il record dell'oggetto non può essere collegato all'applicazione che lo interpreterà.

Appendice C -- Codici di gruppo DXF

Sezione OBJECTS

Dizionario degli oggetti denominati

Il dizionario degli oggetti denominati è il proprietario principale di tutti gli oggetti non grafici del disegno, ad eccezione degli oggetti e delle classi associati alla struttura DXF prima della Release 13. Come nel caso di altri dizionari, il record del dizionario degli oggetti denominati è costituito unicamente da coppie associate di nomi di voci e riferimenti di puntamento di proprietà hard all'oggetto associato.

AutoCAD aggiunge oggetti al dizionario con un nome (chiave) preceduto sempre dal prefisso ACAD_. Altre applicazioni Autodesk si riservano la prerogativa di usare qualsiasi prefisso che inizia per AC. Altri produttori di software devono utilizzare i propri prefissi di programmazione registrati per le loro voci, in modo da evitare conflitti tra i nomi. È possibile assegnare un prefisso ad oggetti di qualsiasi tipo.

Appendice C -- Codici di gruppo DXF

Sezione OBJECTS

Codici di gruppo di oggetti nei file DXF

Viene di seguito riportato un esempio della sezione OBJECTS di un file DXF:

```

0                Inizio della sezione OBJECTS
SECTION
2
OBJECTS
0                Gruppi di dati oggetto
<tipo di oggetto>
.
. <dati>
.
0                Fine della sezione OBJECTS
ENDSEC
    
```

Appendice C -- Codici di gruppo DXF

Sezione OBJECTS

Codici di gruppo di oggetti comuni

La seguente tabella mostra i codici di gruppo applicabili a tutte le entità non grafiche (oggetti). Quando si fa riferimento alla tabella dei codici di gruppo per tipo di entità, che elenca i codici associati a entità specifiche, tenere presente che possono esservi contenuti anche i codici riportati di seguito. Alcuni dei codici di gruppo sono inclusi in un'entità solo se l'entità contiene dei valori diversi da quelli di default per tali proprietà. Quando un gruppo viene omissso, il relativo valore di default al momento dell'input (quando si utilizza DXFIN) viene indicato nella terza colonna. Se il valore di un codice di gruppo è uguale a quello di default, viene omissso al momento dell'output (quando si utilizza DXFOUT).

Codici di gruppo di oggetti comuni

Codici di gruppo	<i>Descrizione</i>
0	<i>Tipo di entità (DICTIONARY o XRECORD)</i>
5	<i>Gestore</i>
102	<i>Inizio del gruppo definito dall'applicazione "{nome_applicazione}". Ad esempio, "{ACAD_REACTORS}" indica l'inizio del gruppo di reattori costanti di AutoCAD.</i>
Codici definiti dalle applicazioni	<i>I codici ed i valori dei gruppi 102 sono definiti dalle applicazioni.</i>
102	<i>Fine del gruppo, "}"</i>

La seguente tabella mostra i codici di gruppo risultanti se sono stati collegati ad un oggetto dei reattori costanti.

Record ACAD_REACTORS

Codice di gruppo	Descrizione
102	"{ACAD_REACTORS" indica l'inizio del gruppo di reattori costanti di AutoCAD
330	ID/gestore puntatore soft per il dizionario di proprietà
102	Fine del gruppo, "]"

La tabella seguente mostra i codici di gruppo risultanti se è stato collegato ad un oggetto un dizionario di estensione.

Record ACAD_XDICTIONARY

Codice di gruppo	Descrizione
102	"{ACAD_XDICTIONARY" indica l'inizio di un gruppo di dizionari estesi.
360	ID/gestore proprietario hard per il dizionario di proprietà
102	Fine del gruppo, "]"

Appendice C -- Codici di gruppo DXF

Sezione OBJECTS

DICTIONARY

I seguenti codici di gruppo sono comuni a tutti gli oggetti dictionary (dizionario).

Codici di gruppo di oggetti comuni

Codici di gruppo	Descrizione
100	Contrassegno di sottoclasse (AcDbDictionary)
3	Nome della voce (uno per ciascuna voce)
350	Gestore dell'oggetto voce (uno per ciascuna voce)

AutoCAD conserva gli elementi come gli stili lineam e le definizioni di gruppo come oggetti nei dizionari. Le seguenti sezioni descrivono i codici di gruppo di oggetti AutoCAD contenuti nei dizionari. Tuttavia, altre applicazioni hanno la possibilità di creare ed utilizzare i propri dizionari nel modo desiderato. Notare che il prefisso "ACAD_" è riservato alle applicazioni AutoCAD.

Appendice C -- Codici di gruppo DXF

Sezione OBJECTS

DICTIONARY

Dizionario GROUP: ACAD_GROUP

I seguenti codici di gruppo sono utilizzati dai gruppi AutoCAD.

Codici di gruppo ACAD_GROUP

Codici di gruppo	Descrizione
0	Nome oggetto (GROUP)
5	Gestore
102	Inizio di un gruppo di reattori costanti; sempre "{ACAD_REACTORS" (il gruppo di reattori costanti compare in tutti i dizionari, ad eccezione di quello principale)
330	ID/gestore puntatore soft per il dizionario di proprietà
102	Fine del gruppo di reattori costanti, sempre "]"
100	Contrassegno di sottoclasse (AcDbGroup)

300	<i>Descrizione del gruppo</i>
70	<i>Flag "senza nome": 1 = senza nome; 0 = con nome</i>
71	<i>Flag di selezione: 1 = selezionabile; 0 = non selezionabile</i>
340	<i>Gestore dell'entità nel gruppo (una voce per oggetto)</i>

Appendice C -- Codici di gruppo DXF

Sezione OBJECTS

DICTIONARY

Dizionario MLINESTYLE: ACAD_MLINESTYLE

I seguenti codici di gruppo definiscono gli stili lineam.

Codici di gruppo ACAD_MLINESTYLE

Codici di gruppo	<i>Descrizione</i>
0	<i>Nome oggetto (MLINESTYLE)</i>
5	<i>Gestore</i>
102	<i>Inizio di un gruppo di reattori costanti; sempre "{ACAD_REACTORS" (il gruppo di reattori costanti compare in tutti i dizionari, ad eccezione di quello principale)</i>
330	<i>ID/gestore puntatore soft per il dizionario di proprietà</i>
102	<i>Fine del gruppo di reattori costanti, sempre "}"</i>
100	<i>Contrassegno di sottoclasse (AcDbMlineStyle)</i>
2	<i>Nome dello stile lineam</i>
70	<i>Flag (codificati in bit):</i> 1 = riempimento attivo 2 = visualizzazione giunzioni 16 = Inizio estremità (linea) a squadra 32 = Inizio estremità archi interni 64 = Inizio estremità arrotondata (archi esterni) 256 = Fine estremità (linea) a squadra 512 = Fine archi interni 1024 = Fine estremità arrotondata (archi esterni)
3	<i>Descrizione stile (stringa, massimo 255 caratteri)</i>
62	<i>Colore riempimento (numero intero, default = 256). Possono esistere più voci; una voce per ciascun elemento.</i>
51	<i>Angolo iniziale (reale, default è 90 gradi)</i>
52	<i>Angolo finale (reale, default è 90 gradi)</i>
71	<i>Numero di elementi</i>
49	<i>Sfalsamento elementi (reale, senza default). Possono esistere più voci; una voce per ciascun elemento.</i>
62	<i>Colore elementi (numero intero, default = 0). Possono esistere più voci; una voce per ciascun elemento.</i>
6	<i>Tipo linea elementi (stringa, default = DALAYER). Possono esistere più voci; una voce per ciascun elemento.</i>

I codici di gruppo 2 nelle entità mline e negli oggetti mlinestyle sono campi ridondanti. Questi gruppi *non* devono essere modificati in nessun caso, anche se è consigliabile leggerli ed utilizzarne i valori. I campi da modificare sono i seguenti:

Mline

Il gruppo 340 nello stesso oggetto, che indica l'oggetto mlinestyle appropriato.

Mlinestyle

Il valore del gruppo 3 nel dizionario MLINESTYLE, che precede il gruppo 350 con il gestore o il nome entità dell'MLINESTYLE corrente.

Appendice C -- Codici di gruppo DXF

Sezione OBJECTS

XRECORD

I seguenti codici di gruppo sono comuni a tutti gli oggetti xrecord (record estesi).

Codici di gruppo xrecord

Codici di gruppo	<i>Descrizione</i>
100	<i>Contrassegno di sottoclasse (AcDbXrecord)</i>
Da 1 a 369 (tranne 5 e 105)	<i>Questi valori possono essere utilizzati da un'applicazione in qualsiasi modo.</i>

Gli oggetti xrecord vengono utilizzati per memorizzare e gestire dati arbitrari o liberi. Sono composti da codici di gruppo DXF con gruppi di "oggetti normali" ossia codici di gruppo di dati non estesi, compresi nell'intervallo 1-369 per gli intervalli supportati. Questo oggetto è concettualmente simile ai dati estesi, ma non ha limiti di ordine o di dimensioni.

Gli oggetti xrecord sono progettati in modo da non danneggiare le release da R13c0 a R13c3 di AutoCAD. Tuttavia, se letti in una release precedente a R13c4, gli oggetti xrecord scompaiono.

Appendice C -- Codici di gruppo DXF

Sezione OBJECTS

ACAD_PROXY_OBJECT

I seguenti codici di gruppo sono validi per gli oggetti proxy.

Codici di gruppo Acad_proxy_object

Codici di gruppo	<i>Descrizione</i>
100	<i>DXF: AcDbProxyObject</i>
90	<i>DXF: ID classe dell'oggetto proxy</i>
91	<i>DXF: ID classe dell'oggetto applicazione effettiva</i>
93	<i>DXF: Dimensioni dei dati dell'oggetto in bit</i>
310	<i>DXF: Dati binari dell'oggetto (possono comparire più voci).</i>
330 o 340 o 350 o 360	<i>DXF: Un ID oggetto (possono comparire più voci).</i>
94	<i>DXF: 0 (indica la fine della sezione dell'ID oggetto)</i>

Il campo 92 non viene utilizzato per AcDbProxyObject. Gli oggetti di questa classe non contengono mai grafica.

Appendice C -- Codici di gruppo DXF

Problemi DXF avanzati

Questa sezione contiene concetti avanzati relativi ai codici di gruppo DXF.

Appendice C -- Codici di gruppo DXF

Problemi DXF avanzati

Oggetti di database

I disegni AutoCAD sono principalmente costituiti da contenitori strutturati per gli oggetti di database. Ciascuno degli oggetti di database contiene quanto segue:

- Un gestore, il cui valore è univoco per il disegno/file DXF ed è costante per l'intera vita utile del disegno. Questo formato esiste a partire dalla Release 10 di AutoCAD, ma a partire dalla Release 13 di AutoCAD i gestori sono sempre attivi.
- Una tabella di dati estesi opzionale, come quella che esiste per le entità dalla Release 11 di AutoCAD.
- Una tabella dei reattori costanti opzionale.
- Un puntatore di proprietà opzionale che punta ad un dizionario di estensione, che a sua volta possiede oggetti secondari inseriti al suo interno da un'applicazione.

Le tabelle dei simboli ed i record delle tabelle dei simboli sono oggetti di database e quindi hanno un gestore. Inoltre, possono avere dati estesi e reattori costanti nei propri record DXF.

Appendice C -- Codici di gruppo DXF

Problemi DXF avanzati

Gestori di riferimento costanti tra oggetti

Una serie di intervalli di codici di gruppo consente agli oggetti di specificare direttamente riferimenti ad altri oggetti nello stesso disegno/file DXF. I quattro tipi di relazione che è possibile specificare sono associati a quattro intervalli:

- Gestore di puntatore soft.
- Gestore di puntatore hard.
- Gestore di proprietà soft.
- Gestore di proprietà hard

Questi tipi di gestore sono presenti come nomi di entità in AutoLISP, come valori `ads_name` in ADS e come classi con nome simile derivate da ARX. Questi valori sono sempre conservati durante le operazioni `inser`, `xrif` e `mblocco`, per consentire che i riferimenti tra oggetti in un gruppo in corso di copia siano aggiornati in modo da puntare agli oggetti copiati, mentre i riferimenti ad altri oggetti non vengono modificati.

Inoltre, viene definito un intervallo di codici di gruppo per i gestori "liberi", in modo da consentire la memorizzazione adeguata dei valori dei gestori che *non* vengono convertiti in nomi di entità e quindi non vengono aggiornati nelle operazioni `inser`, `xrif` o `mblocco`.

Nota Se si utilizzano i codici di gruppo `xdata 1005` per memorizzare i gestori, essi vengono considerati come gestori di puntatori soft. Ciò significa che quando i gruppi di oggetti vengono copiati o inseriti in un altro disegno, i riferimenti tra gli oggetti interessati vengono convertiti. Sebbene gli elementi `xdata 1005` siano sempre restituiti come gestori in AutoLISP ed ADS, tutti gli intervalli di codici di gruppo di gestori di riferimento vengono rappresentati come "nomi entità" in AutoLISP e strutture `ads_name` in ADS.

Appendice C -- Codici di gruppo DXF

Problemi DXF avanzati

Gestori di riferimento costanti tra oggetti

Riferimenti di puntatore e proprietà

Un puntatore è un riferimento che indica l'uso ma non la proprietà o la responsabilità di un altro oggetto. I riferimenti di puntatore indicano che un oggetto utilizza in qualche modo l'altro oggetto e ne condivide l'accesso.

Un riferimento di proprietà significa che un oggetto proprietario è responsabile per gli oggetti per cui ha un gestore proprietario. I riferimenti di proprietà sovrintendono alla scrittura di tutti i file DWG e DXF in modo generico, ad esempio iniziando la scrittura di un gestore proprietario a partire da alcuni oggetti chiave principali.

Un oggetto può avere un numero qualsiasi di riferimenti di puntatore, ma può avere un solo proprietario.

Appendice C -- Codici di gruppo DXF

Problemi DXF avanzati

Gestori di riferimento costanti tra oggetti

Hard e soft

I riferimenti hard, sia puntatore che proprietario, proteggono un oggetto dall'eliminazione, al contrario dei riferimenti soft.

In AutoCAD, le definizioni di blocco e le entità complesse sono proprietari hard dei propri elementi. Una tabella dei simboli ed i dizionari sono proprietari soft dei propri elementi. Le entità polyline sono proprietari hard delle proprie entità vertex e seqend. Le entità insert sono proprietari hard delle proprie entità attrib e seqend.

Quando si stabilisce un riferimento ad un altro oggetto, si raccomanda di considerare se il riferimento deve proteggere un oggetto dal comando ELIMINA.

Appendice C -- Codici di gruppo DXF

Problemi DXF avanzati

Gestori di riferimento costanti tra oggetti

Gestori liberi

I gestori liberi si distinguono in quanto non vengono convertiti internamente in identificatori costanti di sessione, in nomi di entità in AutoLISP e così via, ma vengono memorizzati come gestori. Quando i valori dei gestori vengono convertiti in operazioni di unione di disegni, i gestori liberi vengono ignorati.

In tutti gli ambienti, i gestori liberi possono essere scambiati con nomi di entità del disegno corrente tramite le funzioni **handent**. I gestori liberi vengono comunemente usati per i riferimenti ad oggetti in file DXF e DWG esterni.

Appendice C -- Codici di gruppo DXF

Problemi DXF avanzati

Gestori di riferimento costanti tra oggetti

Codici di gruppo 1005

I codici di gruppo di dati estesi 1005 hanno lo stesso comportamento e la stessa semantica dei puntatori soft e quindi vengono convertiti ogni volta che l'oggetto host viene unito ad un altro disegno. Tuttavia, gli elementi 1005 non vengono convertiti in identificativi costanti di sessione o nomi di entità interni in AutoLISP e ADSRX, ma vengono memorizzati come gestori.

Appendice C -- Codici di gruppo DXF

Problemi DXF avanzati

Contrassegni di sottoclasse

Un problema nell'archiviazione di un flusso di dati di gruppo è quello che un singolo oggetto può essere costituito da diversi membri filer, uno per ciascun livello di adozione in cui viene effettuata l'archiviazione, che può essere eseguita da diversi programmatori. Poiché ciascun livello di adozione può evolversi separatamente, è necessario separare i dati di ciascun membro filer di classi.

Tutti i membri filer di classi devono essere preceduti dalla propria parte di dati di esempio specifica della classe, con un

contrassegno di "sottoclasse": un codice di gruppo 100 seguito da una stringa con il nome della classe. Ciò non influisce sullo stato necessario per definire lo stato dell'oggetto, ma fornisce ai parser di file DXF un metodo per dirigere i codici di gruppo al corrispondente software applicativo.

Ad esempio, un oggetto con dati provenienti da classi derivate diverse verrebbe rappresentato nel seguente modo:

```

999
FOOGRANDCHILD, definito dalla classe AcDbSonOfSonOfFoo, che
999
  proviene da AcDbSonOfFoo
    0
FOOGRANDCHILD
  5
C2
100
AcDbFoo
999
Utilizza i codici di gruppo 10/20/30
  10
  1.1
  20
  2.3
  30
  7.3
100
AcDbSonOfFoo
999
Utilizza i codici di gruppo 10/20/30, per uno scopo diverso
  10
  1.1
  20
  2.3
  30
  7.3
100
AcDbSonOfSonOfFoo
999
Utilizza i codici di gruppo 10/20/30, per un altro scopo ancora
  10
  13.2
  20
  23.1
  30
  31.2
999
Ora per i dati estesi
1001
APP_1
1070
45
1001
APP_2
1004
18A5B3EF2C199A

```

Appendice C -- Codici di gruppo DXF

Problemi DXF avanzati

Dizionario di estensione e reattori costanti

Il dizionario di estensione è una sequenza opzionale che memorizza il gestore di un oggetto di dizionario appartenente all'oggetto corrente, il quale a sua volta può contenere voci. Questa funzione consente il collegamento di oggetti di database liberi a qualsiasi

oggetto di database. Qualsiasi oggetto o entità può avere questa sezione.

I reattori costanti sono una sequenza opzionale che memorizza i gestori di oggetti che si registrano come reattori sull'oggetto corrente. Qualsiasi oggetto o entità può avere questa sezione.

Appendice C -- Codici di gruppo DXF

Problemi DXF avanzati

Dati estesi

I dati estesi vengono creati dalle applicazioni AutoLISP o ARX. Se un'entità contiene dati estesi, segue i dati di definizione normali dell'entità. I codici di gruppo da 1000 a 1071 descrivono i dati estesi. Viene di seguito riportato un esempio di un'entità contenente dati estesi in formato DXF.

Dati di definizione entità normali:

```

0
INSERT
5
F11
100
AcDbEntity
8
TOP
100
AcDbBlockReference
2
BLOCK_A
10
0.0
20
0.0
30
0.0

```

Dati di definizione estesi

```

1001
AME_SOL
1002
{
1070
0
1071
1.95059E+06
1070
519
1010
2.54717
1020
2.122642
1030
2.049201
1005
ECD
1005
EE9
1005
0
1040
0.0
1040
1.0

```

1000
MILD_STEEL

Il codice di gruppo 1001 indica l'inizio dei dati estesi. Al contrario dei dati di entità normali, nei dati estesi lo stesso codice di gruppo può comparire più volte e l'ordine è importante.

I dati estesi sono raggruppati in base al nome dell'applicazione registrata. Il gruppo di ogni applicazione registrata inizia con un codice di gruppo 1001, con il nome dell'applicazione come valore di stringa. I nomi delle applicazioni registrate corrispondono alle voci della tabella dei simboli APPID.

Un'applicazione può utilizzare un numero qualsiasi di nomi APPID, secondo la necessità. I nomi APPID sono permanenti, ma possono essere eliminati se non sono attualmente utilizzati nel disegno. Ciascun nome APPID non può avere più di un gruppo di dati collegati a ciascuna entità. All'interno del gruppo di un'applicazione, la sequenza di gruppi di dati estesi ed il loro significato sono definiti dall'applicazione.

I codici di gruppo dei dati estesi elencati nella seguente tabella hanno il valore descritto:

Codici di gruppo e descrizioni dei dati estesi

Nome entità	Codice di gruppo	Descrizione
Stringa	1000	Le stringhe nei dati estesi possono avere una lunghezza massima di 255 byte (con il 256esimo byte riservato al carattere nullo).
Nome applicazione	1001 anche un valore di stringa	I nomi delle applicazioni possono avere una lunghezza massima di 31 byte (il 32esimo byte è riservato al carattere nullo).
Stringa di controllo	1002	Nota: non aggiungere un gruppo 1001 ai propri dati estesi perché AutoCAD lo rileva come inizio di un gruppo di dati estesi di una nuova applicazione. Una stringa di controllo dei dati estesi può essere "{ " o "} ". Queste parentesi graffe consentono all'applicazione di organizzare i propri dati suddividendoli in elenchi. La parentesi aperta inizia un elenco, mentre la parentesi chiusa termina l'ultimo elenco creato. Gli elenchi possono essere nidificati.
Nome layer	1003	Quando AutoCAD legge i dati estesi per un'applicazione in particolare, verifica che le parentesi siano bilanciate.
Dati binari	1004	Nome del layer associato ai dati estesi. I dati binari sono organizzati in frammenti di lunghezza variabile. Ogni parte può essere lunga, al massimo, 127 byte. Nei file DXF ASCII, i dati binari sono rappresentati come una stringa di cifre esadecimali, due per byte binario.
Gestore di database	1005	I gestori di entità nel database del disegno. Nota: durante l'importazione di un disegno con gestori e gestori di dati estesi in un altro disegno utilizzando INSERT, INSERT *, XREF Bind, XBIND DXFIN parziale, i gestori dei dati estesi vengono convertiti allo stesso modo dei corrispondenti gestori di entità, mantenendo in questo modo la propria unione. Ciò viene effettuato anche nell'operazione ESPLODI blocco o in qualsiasi altra operazione AutoCAD. Quando AUDIT rileva un gestore di dati estesi che non corrisponde al gestore di un'entità nel file del disegno, lo considera un errore. Se VERIFICA sta correggendo le entità, il gestore viene impostato su 0.
3 valori reali	1010, 1020, 1030	Tre valori reali, nell'ordine X, Y, Z. Possono essere utilizzati come un record di punto o di vettore. AutoCAD non modifica mai il loro valore.
Posizione globale	1011, 1021, 1031	A differenza di un semplice punto tridimensionale, le coordinate globali vengono spostate, scalate, ruotate e rese speculari insieme all'entità correlata a cui appartengono i dati estesi. Anche la posizione globale viene stirata quando viene applicato il comando STIRA all'entità correlata e questo punto si trova all'interno della finestra di selezione.
Spostamento globale	1012, 1022, 1032	Un altro punto tridimensionale che viene scalato, ruotato e reso speculare insieme all'entità correlata (ma non viene spostato o stirato).
Direzione globale	1013, 1023, 1033	Un altro punto tridimensionale che viene ruotato e reso speculare insieme all'entità correlata (ma non viene spostato, scalato o stirato).
Reale	1040	Un valore reale.
Distanza	1041	Un valore reale che viene scalato insieme all'entità correlata.
Fattore di scala	1042	Un altro valore reale che viene scalato insieme all'entità correlata. La differenza tra una distanza ed un fattore di scala è definito dall'applicazione.

Numero intero 1070 Un numero intero a 16 bit (con e senza segno).
 Lungo 1071 Un numero intero 32 bit (lungo) con segno.

Appendice C -- Codici di gruppo DXF

Problemi DXF avanzati

OCS (Sistema di Coordinate Oggetto)

Per occupare meno spazio nel database del disegno (e nel file DXF), i punti associati a ciascuna entità vengono espressi in base all'OCS (Sistema di Coordinate Oggetto) dell'entità. In precedenti release di AutoCAD, l'OCS veniva chiamato ECS. Con l'OCS, le sole informazioni aggiuntive necessarie per descrivere la posizione dell'entità nello spazio tridimensionale sono il vettore tridimensionale che descrive l'asse Z dell'OCS ed il valore di elevazione.

Per una data direzione (o estrusione) dell'asse Z, esiste un numero infinito di sistemi di coordinate, definito tramite la conversione dell'origine nello spazio tridimensionale e ruotando gli assi X ed Y intorno all'asse Z. Tuttavia, per la stessa direzione dell'asse Z, esiste un solo OCS, con le seguenti proprietà:

- L'origine coincide con quella del WCS.
- L'orientamento degli assi X ed Y all'interno del piano XY viene calcolato in modo arbitrario ma coerente. AutoCAD effettua questo calcolo utilizzando l'algoritmo dell'asse arbitrario (vedere "Algoritmo dell'asse arbitrario").

Per alcune entità, l'OCS equivale al WCS e tutti i punti (gruppi DXF 10-37) sono espressi in coordinate globali. Vedere la seguente tabella.

Sistemi di coordinate associati ad un tipo di entità

Entità	<i>Note</i>
Entità tridimensionali come line, point, 3dface, 3D polyline, 3D vertex, 3D mesh, 3D mesh vertex	<i>Queste entità non si trovano in un piano particolare. Tutti i punti sono espressi in coordinate globali. Di tutte queste entità, solo le linee ed i punti possono essere estrusi. La loro direzione di estrusione può differire dall'asse Z globale.</i>
Entità bidimensionali come circle, arc, solid, trace, text, attrib, attdef, shape, insert, 2D polyline, 2D vertex, lwpolyline, hatch, image	<i>Queste entità sono planari per natura. Tutti i punti sono espressi in coordinate di oggetto. Tutte queste entità possono essere estruse. La loro direzione di estrusione può differire dall'asse Z globale.</i>
Dimension	<i>Alcuni dei punti di una dimensione sono espressi in WCS ed alcuni in OCS.</i>
Viewport	<i>Espressa in coordinate globali.</i>

Quando AutoCAD ha stabilito l'OCS per una data entità, l'OCS funziona nel seguente modo:

Il valore di elevazione memorizzato con un'entità indica il valore dello spostamento del piano XY lungo l'asse Z (dall'origine del WCS) per farlo coincidere con il piano che contiene l'entità. Non è importante quale percentuale di questo valore è l'elevazione definita dall'utente.

Qualsiasi punto bidimensionale immesso tramite l'UCS viene trasformato nei punti bidimensionali corrispondenti nell'OCS, che viene spostato e ruotato rispetto all'UCS.

Seguono alcune varianti di questo processo:

- Non è possibile risalire in modo attendibile all'UCS attivo al momento dell'acquisizione dell'entità.
- Quando si immettono le coordinate XY di un'entità in un dato UCS e quindi si esegue un comando DXFOUT, probabilmente non sarà possibile riconoscere tali coordinate XY nel file DXF. È necessario conoscere il metodo utilizzato da AutoCAD per calcolare gli assi X ed Y per poter lavorare con questi valori.
- Il valore di elevazione memorizzato con un'entità e indicato nei file DXF è la somma della differenza della coordinata Z tra il piano XY dell'UCS ed il piano XY dell'OCS ed il valore dell'elevazione che l'utente ha specificato al momento in cui è stata disegnata l'entità.

Appendice C -- Codici di gruppo DXF

Problemi DXF avanzati

OCS (Sistema di Coordinate Oggetto)

Algoritmo dell'asse arbitrario

L'algoritmo dell'asse arbitrario viene utilizzato internamente da AutoCAD per implementare la generazione arbitraria ma coerente di sistemi di coordinate oggetto per tutte le entità che utilizzano le coordinate oggetto.

Dato un vettore con lunghezza unitaria da utilizzare come asse Z di un sistema di coordinate, l'algoritmo dell'asse arbitrario genera un asse X corrispondente per il sistema di coordinate. L'asse Y segue in base alla convenzione della mano destra.

Il metodo è quello di esaminare l'asse Z dato (chiamato anche *vettore normale*) e controllare se è vicino all'asse Z globale positivo o negativo. In questo caso, far attraversare l'asse Y globale dall'asse Z dato per ottenere l'asse X arbitrario. In caso contrario, far attraversare l'asse Z globale dall'asse Z dato per ottenere l'asse X arbitrario. Il limite in cui viene effettuata la decisione è stato scelto per essere sia conveniente che trasportabile tra una macchina e l'altra. Questo è possibile grazie ad una specie di calotta polare "quadrata", i cui limiti sono 1/64, e che può essere specificata esattamente con sei cifre a frazione decimale e sei bit a frazione binaria.

L'algoritmo esegue le seguenti operazioni. Si presuppone che tutti i vettori si trovino nello spazio tridimensionale e che siano specificati nel Sistema di Coordinate Globali.

Si supponga che il vettore normale dato sia denominato N .

Si supponga che l'asse Y globale sia denominato W_y , che è sempre $(0,1,0)$.

Si supponga che l'asse Z globale sia denominato W_z , che è sempre $(0,0,1)$.

Si desiderano ottenere gli assi X ed Y arbitrari associati alla normale N . Verranno denominati A_x ed A_y . N potrebbe essere anche denominato A_z (l'asse Z arbitrario) nel seguente modo:

```
If (abs (Nx) < 1/64) and (abs (Ny) < 1/64) then
  Ax = Wy X N (dove "X" è l'operatore del prodotto vettoriale).
Altrimenti,
  Ax = Wz X N.
  Scalare Ax alla lunghezza dell'unità.
```

Il metodo per ottenere il vettore A_y è il seguente:

```
Ay = N X Ax. Scalare Ay alla lunghezza dell'unità.
```

Index

—3—

3D, distanza; 289; 305
 3D, punti; 330
 3D, punti, nei dati estesi; 209

—A—

a cascata, menu. Vedere menu, a discesa; 98; 99
 a discesa, menu. Vedere menu, a discesa; 98; 99
 a discesa, menu. Vedere menu, a discesa; 98; 99
 a gruppi di immagini, menu. Vedere menu, a gruppi di immagini; 110
 a icone, menu. Vedere menu, a gruppi di immagini; 110
 abbreviazioni dei comandi. Vedere comandi, alias dei; 43
 acad, comando; 134
 ActiveX, Automazione. Vedere Automazione; 136
 ADS (AutoCAD Development System), applicazioni, caricamento; 142; 165; 166
 ADS (AutoCAD Development System), applicazioni, elenco delle applicazioni correntemente caricate; 272; 273
 ADS (AutoCAD Development System), definizione; 142
 ADS, applicazioni, caricamento; 280
 ADSRX, controllo delle finestre di dialogo; 413
 ADSRX, gruppi di selezione, passaggio ad AutoLISP; 197
 algoritmo dell'asse arbitrario; 537
 alias, creazione; 43
 alias, uso del prefisso - (simbolo meno); 43
 allocazione manuale della memoria; 300
 ambiente, variabili, in AutoLISP; 169
 American Standard Code for Information Interchange. Vedere ASCII; 473
 ANGBASE, variabile di sistema; 173; 181; 339
 ANGDIR, variabile di sistema; 173
 angoli, conversione, gradi in radianti; 182
 angoli, conversione, in stringa; 274; 275
 angoli, conversione, radianti in gradi; 182
 angoli, conversione, stringa in radianti; 274
 angoli, radianti; 182
 angoli, specificati dall'utente; 303; 304; 308
 ANNULLA, comando; 134
 ANNULLA, comando, ed errori; 163
 anonime, funzioni, definizione; 318
 anonimi, blocchi, *Unnn; 294
 anonimi, blocchi, creazione; 294
 apertura, file; 329
 apertura, finestre di dialogo; 320; 328
 APERTURE, variabile di sistema; 177
 AppData, sezione di acad.cfg; 304; 337; 338
 applicazione, nomi applicazione nei dati estesi; 209
 APPLOAD, comando; 138
 argomenti opzionali, AutoLISP; 162
 ARX; 141
 ARX, applicazioni, caricamento; 141; 165; 166

ARX, applicazioni, caricamento automatico; 141; 142
 ARX, file, caricamento; 279
 ARX, gruppi di selezione, passaggio ad AutoLISP; 197
 ARX, linguaggio di programmazione; 133; 134
 ASCII; 473
 ASCII, codici; 466
 ASCII, funzioni; 277; 282; 334
 ASEADMIN, comando; 365
 ASEEXPORT, comando; 366
 ASELINKS, comando; 366
 ASEROWS, comando; 366
 ASESELECT, comando; 366
 ASESQLED, comando; 366; 367; 369; 370
 asterisco (*), nei, nomi di blocchi anonimi; 294
 attributi, indicazione del valore DCL; 303
 AUNITS, variabile di sistema; 181
 AUPREC, variabile di sistema; 181
 AutoCAD Runtime Extension. Vedere ARX; 144
 AutoCAD, ambiente; 37
 AutoCAD, file di guida; 40
 AutoCAD, file di Guida, collegamenti ipertesto; 51
 AutoCAD, file di Guida, conversione da R12 in R13; 51; 52
 AutoCAD, file di Guida, direttive; 49
 AutoCAD, file di Guida, formato; 49
 AutoCAD, file di Guida, formattazione speciale; 51
 AutoCAD, file di Guida, Release 12, conversione; 51; 52
 AutoCAD, file di Guida, stringhe di contesto; 49
 AutoCAD, file di supporto personalizzabili; 40
 AutoCAD, programma di visualizzazione della Guida; 49
 AutoCAD, struttura della directory; 37
 AutoLISP; 143; 144
 AutoLISP,; 406
 AutoLISP, applicazioni, accesso ai comandi definiti esternamente; 363
 AutoLISP, argomenti, funzioni con; 162
 AutoLISP, argomenti, opzionali; 162
 AutoLISP, argomenti, simboli come; 162
 AutoLISP, caricamento; 165
 AutoLISP, caricamento automatico dei comandi; 140
 AutoLISP, caricamento, file DCL; 216
 AutoLISP, codici di errore; 406
 AutoLISP, commenti nei file; 149
 AutoLISP, controllo della configurazione; 169
 AutoLISP, controllo della visualizzazione di testo e disegno; 172
 AutoLISP, controllo delle finestre di dialogo; 413
 AutoLISP, conversione ASCII; 183
 AutoLISP, conversioni di unità; 184
 AutoLISP, conversioni, angoli; 182
 AutoLISP, conversioni, sistemi di coordinate; 186
 AutoLISP, coppie puntate; 158
 AutoLISP, corso interattivo; 232
 AutoLISP, dati estesi, gestione dell'utilizzo della memoria; 213
 AutoLISP, dati estesi, gestori; 213
 AutoLISP, dati estesi, recupero; 211
 AutoLISP, dati estesi, unione ad una entità; 212

- AutoLISP, definizione; 138
- AutoLISP, elenchi dei filtri di entità; 193
- AutoLISP, elenchi di associazione; 158
- AutoLISP, elenchi punto; 157
- AutoLISP, espressioni; 144
- AutoLISP, espressioni DIESEL; 122; 123; 126; 159
- AutoLISP, espressioni DIESEL, nei menu; 124
- AutoLISP, file di programma; 149
- AutoLISP, file di programma, caricamento; 165
- AutoLISP, file di programma, commenti; 149
- AutoLISP, file di programma, indentazione; 150
- AutoLISP, file, caricamento; 279; 280
- AutoLISP, finestra di dialogo, apertura; 216
- AutoLISP, finestra di dialogo, azioni di default; 218; 220
- AutoLISP, finestra di dialogo, caselle; 218
- AutoLISP, finestra di dialogo, caselle di modifica; 224
- AutoLISP, finestra di dialogo, chiusura; 216; 224
- AutoLISP, finestra di dialogo, cluster di scelta; 222
- AutoLISP, finestra di dialogo, codici di causa delle funzioni di richiamo; 220
- AutoLISP, finestra di dialogo, corso interattivo; 239
- AutoLISP, finestra di dialogo, dati specifici dell'applicazione; 230
- AutoLISP, finestra di dialogo, disattivazione di caselle; 221; 222
- AutoLISP, finestra di dialogo, dispositivi di scorrimento; 223
- AutoLISP, finestra di dialogo, funzioni di gestione di attributi; 218
- AutoLISP, finestra di dialogo, funzioni di gestione di caselle; 218; 221
- AutoLISP, finestra di dialogo, funzioni di richiamo; 218
- AutoLISP, finestra di dialogo, funzioni di richiamo, dai dispositivi di scorrimento; 223
- AutoLISP, finestra di dialogo, funzioni disattivate mentre sono attive; 216
- AutoLISP, finestra di dialogo, funzioni per la gestione delle caselle di riepilogo; 227
- AutoLISP, finestra di dialogo, gruppi di immagini; 228
- AutoLISP, finestra di dialogo, modifica di modalità; 221; 222
- AutoLISP, finestra di dialogo, nascondere; 216; 224
- AutoLISP, finestra di dialogo, nidificare; 224
- AutoLISP, finestra di dialogo, parole d'ordine; 226
- AutoLISP, finestra di dialogo, pulsanti immagine; 228
- AutoLISP, finestra di dialogo, riepilogo delle funzioni; 231
- AutoLISP, finestra di dialogo, variabili delle espressioni delle azioni; 218
- AutoLISP, finestra di dialogo, variabili delle espressioni delle azioni, \$data; 230
- AutoLISP, finestra di dialogo, variabili delle espressioni delle azioni, \$reason; 220
- AutoLISP, funzione di conversione di stringhe; 181
- AutoLISP, funzioni C, XXX; 159
- AutoLISP, funzioni con argomenti; 162
- AutoLISP, funzioni condizionali; 156
- AutoLISP, funzioni condizionali, sommario; 246
- AutoLISP, funzioni definite dall'utente; 162
- AutoLISP, funzioni definite esternamente; 260
- AutoLISP, funzioni dei dati dell'entità; 202; 206
- AutoLISP, funzioni di accesso ai dispositivi; 189
- AutoLISP, funzioni di accesso ai dispositivi, sommario; 254
- AutoLISP, funzioni di accesso ai dizionari; 214
- AutoLISP, funzioni di apertura e chiusura delle finestre di dialogo, sommario; 257
- AutoLISP, funzioni di comando; 166
- AutoLISP, funzioni di comando, sommario; 250
- AutoLISP, funzioni di controllo della visualizzazione; 169
- AutoLISP, funzioni di controllo della visualizzazione sommario; 251
- AutoLISP, funzioni di conversione; 180; 181
- AutoLISP, funzioni di conversione, sommario; 253
- AutoLISP, funzioni di esempio, BASE (converte un valore decimale su altra base); 183
- AutoLISP, funzioni di esempio, C:ASCII (stampa i codici ASCII); 183
- AutoLISP, funzioni di esempio, C:HELLO (esempio di finestra di dialogo); 216
- AutoLISP, funzioni di esempio, C:12M (pollici in metri); 184
- AutoLISP, funzioni di esempio, C:LISTXRECORD (elenca i dati di xrecord); 213
- AutoLISP, funzioni di esempio, C:MAKEXRECORD (crea un oggetto xrecord); 213
- AutoLISP, funzioni di esempio, C:PRINTDXF (visualizza informazioni DXF per l'ultima entità); 202
- AutoLISP, funzioni di esempio, DTR (degrees to radians, gradi in radianti); 182
- AutoLISP, funzioni di esempio, GETPASS (richiede una parola d'ordine tramite una finestra di dialogo); 226
- AutoLISP, funzioni di esempio, MK_LIST; 227
- AutoLISP, funzioni di esempio, RTD (radians to degrees, radianti in gradi); 182
- AutoLISP, funzioni di gestione degli elenchi; 156
- AutoLISP, funzioni di gestione degli errori, sommario; 249
- AutoLISP, funzioni di gestione degli oggetti; 197
- AutoLISP, funzioni di gestione dei file; 170
- AutoLISP, funzioni di gestione dei file, sommario; 254
- AutoLISP, funzioni di gestione dei gruppi di selezione; 192
- AutoLISP, funzioni di gestione dei simboli; 159
- AutoLISP, funzioni di gestione dei simboli, sommario; 247; 248
- AutoLISP, funzioni di gestione della memoria, sommario; 259
- AutoLISP, funzioni di gestione dell'applicazione, sommario; 249
- AutoLISP, funzioni di gestione delle funzioni; 159
- AutoLISP, funzioni di gestione delle funzioni, sommario; 248
- AutoLISP, funzioni di gestione delle stringhe, sommario; 245
- AutoLISP, funzioni di gestione delle tabelle di simboli; 214
- AutoLISP, funzioni di input utente; 173
- AutoLISP, funzioni di input utente, sommario; 252
- AutoLISP, funzioni di manipolazione delle liste, sommario; 246; 247
- AutoLISP, funzioni di richiesta; 166
- AutoLISP, funzioni di richiesta, sommario; 250
- AutoLISP, funzioni di uguaglianza; 156
- AutoLISP, funzioni di uguaglianza, sommario; 246

AutoLISP, funzioni di utilità generali; 166
 AutoLISP, funzioni di utilità geometriche; 177
 AutoLISP, funzioni geometriche, sommario; 252; 253
 AutoLISP, funzioni getxxx; 173
 AutoLISP, funzioni per i calcoli numerici; 152
 AutoLISP, funzioni per la gestione degli errori; 163
 AutoLISP, funzioni per la gestione delle applicazioni; 164
 AutoLISP, funzioni per la gestione delle stringhe; 152; 153
 AutoLISP, funzioni riguardanti i nomi di entità; 198
 AutoLISP, gestione degli errori; 173
 AutoLISP, gestione della memoria; 402
 AutoLISP, gruppi di selezione; 192
 AutoLISP, gruppi di selezione, gestione; 196
 AutoLISP, gruppi di selezione, passaggio ad ADSRX; 197
 AutoLISP, gruppi di selezione, rilascio dalla memoria; 192
 AutoLISP, immissione di testo alla riga di comando; 144
 AutoLISP, memoria; 322
 AutoLISP, messaggi di errore; 408
 AutoLISP, messaggi di errore, errori di programmazione; 408
 AutoLISP, messaggi di errore, errori interni; 412
 AutoLISP, messaggi di errore, formato; 163
 AutoLISP, punto esclamativo (!), uso; 151
 AutoLISP, schermo di disegno; 206
 AutoLISP, simboli locali nelle funzioni; 161; 162
 AutoLISP, simboli uso della memoria; 402
 AutoLISP, simboli, come argomenti; 162
 AutoLISP, sintassi delle funzioni; 148
 AutoLISP, sommario delle funzioni; 243
 AutoLISP, sottomissione di comandi; 167
 AutoLISP, stringhe, barra rovesciata (); 181; 221; 222
 AutoLISP, stringhe, carattere con codice ottale (); 154
 AutoLISP, stringhe, carattere con codice ottale (nnn); 154
 AutoLISP, stringhe, carattere di tabulazione (t); 154
 AutoLISP, stringhe, carattere Escape (e); 154
 AutoLISP, stringhe, corrispondenza con un modello di caratteri jolly; 155
 AutoLISP, stringhe, sequenza di caratteri a byte multiplo, (M+NXXXX); 154
 AutoLISP, stringhe, sequenza di caratteri Unicode, U+XXXX; 154
 AutoLISP, stringhe, virgolette; 154; 181
 AutoLISP, supporto per altre lingue; 167
 AutoLISP, tipi di dati; 145
 AutoLISP, tipi di dati, descrittori di file; 148
 AutoLISP, tipi di dati, nomi di entità; 147
 AutoLISP, tipi di dati, simboli; 148
 AutoLISP, tipi di dati, stringhe, caratteri di controllo; 154
 AutoLISP, tipi di dati, variabili; 148
 AutoLISP, uscita tranquilla; 154
 AutoLISP, uso delle parentesi; 144
 AutoLISP, uso nei menu; 95
 AutoLISP, variabili; 150
 AutoLISP, variabili d'ambiente; 169
 AutoLISP, variabili di sistema; 169
 AutoLISP, variabili predefinite; 151
 AutoLISP, variabili predefinite, PAUSE; 151
 AutoLISP, variabili predefinite, PI; 151

AutoLISP, variabili predefinite, T; 151; 152; 153
 AutoLISP, variabili, alla riga di comando; 151
 AutoLISP, variabili, valore nil; 150
 AutoLISP, Vedere anche codici di gruppo DXF; 478
 AutoLISP, visualizzazione di messaggi; 170
 Automazione; 136
 Automazione, applicazioni, uso; 136

—B—

barra rovesciata (), come simbolo PAUSA, AutoLISP; 284
 barra rovesciata (), come simbolo PAUSE, AutoLISP; 151
 barra rovesciata (), e stampa dei caratteri di controllo; 331
 barra rovesciata (), nelle stringhe; 221; 222; 284; 309
 barra rovesciata (), nelle stringhe, AutoLISP; 181
 barra. Vedere barra rovesciata (); 284
 big font; 68
 blocchi, anonimi; 294
 blocchi, definizioni, creazione; 294
 blocchi, definizioni, modifica; 295
 blocchi, definizioni, nidificate; 294; 325
 blocchi, elementi base; 474
 blocchi, riferimenti, con attributi, aggiornamento dell'immagine sullo schermo; 297
 blocchi, riferimenti, con attributi, creazione; 294
 blocchi, riferimenti, con attributi, dati di definizione; 325
 blocchi, riferimenti, con attributi, modifica; 295; 297
 blocchi, riferimenti, con attributi, selezione; 325
 blocchi, riferimenti, dati di definizione; 325
 blocchi, riferimenti, selezione; 325
 blocco, ottenuto dal file DXB; 41
 bonus, directory; 138

—C—

CAL, comando; 369
 cancellazione, finestra; 310
 caratteri di controllo, nelle stringhe, AutoLISP; 154
 caratteri jolly, - (trattino); 358
 caratteri jolly, # (cancelletto); 358
 caratteri jolly, * (asterisco); 358
 caratteri jolly, , (virgola); 358
 caratteri jolly, . (punto); 358
 caratteri jolly, ... (ellissi); 358
 caratteri jolly, ? (punto interrogativo); 358
 caratteri jolly, @ (chiocciola); 358
 caratteri jolly, ` (apice inverso); 358
 caratteri jolly, ~ (tilde); 358
 caratteri jolly, corrispondenza tra stringa e modello, AutoLISP; 155
 CARICA, comando; 60
 caricamento, applicazioni ADS; 142
 caricamento, applicazioni ARX; 141; 276
 caricamento, applicazioni AutoLISP; 138; 165
 caricamento, file DCL; 320
 caricamento, file DXB; 41
 caselle DCL; 413; 437; 450

- caselle DCL, attive (predefinite); 450
- caselle DCL, attributi definiti dall'utente; 449
- caselle DCL, attributi predefiniti; 418; 437
- caselle DCL, attributi predefiniti, larghezza; 449
- caselle DCL, attributi predefiniti, password_char; 447
- caselle DCL, attributi predefiniti, small_increment; 447; 448
- caselle DCL, attributi predefiniti, tab_truncate; 448
- caselle DCL, attributi predefiniti, tabs; 448
- caselle DCL, attributi predefiniti, valore; 448
- caselle DCL, attributi predefiniti, big_increment; 440
- caselle DCL, attributi predefiniti, children_alignment; 440
- caselle DCL, attributi predefiniti, children_fixed_height; 440
- caselle DCL, attributi predefiniti, children_fixed_width; 441
- caselle DCL, attributi predefiniti, color; 441
- caselle DCL, attributi predefiniti, edit_limit; 441; 442
- caselle DCL, attributi predefiniti, edit_width; 442
- caselle DCL, attributi predefiniti, fixed_height; 442
- caselle DCL, attributi predefiniti, fixed_width; 442
- caselle DCL, attributi predefiniti, fixed_width_font; 443
- caselle DCL, attributi predefiniti, height; 443
- caselle DCL, attributi predefiniti, initial_focus; 443
- caselle DCL, attributi predefiniti, is_bold; 443; 444
- caselle DCL, attributi predefiniti, is_cancel; 444
- caselle DCL, attributi predefiniti, is_default; 444
- caselle DCL, attributi predefiniti, is_enabled; 444
- caselle DCL, attributi predefiniti, is_tab_stop; 445
- caselle DCL, attributi predefiniti, key; 445
- caselle DCL, attributi predefiniti, label; 445
- caselle DCL, attributi predefiniti, layout; 446
- caselle DCL, attributi predefiniti, list; 446
- caselle DCL, attributi predefiniti, max_value; 446
- caselle DCL, attributi predefiniti, min_value; 446
- caselle DCL, attributi predefiniti, mnemonic; 447
- caselle DCL, attributi predefiniti, password_char; 226
- caselle DCL, boxed_row; 453; 454
- caselle DCL, button; 454
- caselle DCL, casella di scorrimento. Vedere caselle DCL, dispositivo di scorrimento; 413
- caselle DCL, caselle di modifica; 224
- caselle DCL, cluster; 422
- caselle DCL, cluster di scelta; 222
- caselle DCL, column; 454; 455
- caselle DCL, concatenation; 455
- caselle DCL, definizioni; 417
- caselle DCL, dialog; 455
- caselle DCL, disabilitazione; 429; 444
- caselle DCL, dispositivo di scorrimento; 223
- caselle DCL, dispositivo di scorrimento, codice di causa delle funzioni di richiamo; 220
- caselle DCL, edit_box; 456
- caselle DCL, edit_box, codice di causa delle funzioni di richiamo; 220
- caselle DCL, edit_box, variabile \$reason; 220
- caselle DCL, errtile; 456
- caselle DCL, image; 456; 457
- caselle DCL, image_button; 457
- caselle DCL, image_button, codice di causa delle funzioni di richiamo; 220
- caselle DCL, image_button, variabile \$reason; 220
- caselle DCL, limitate; 451; 452
- caselle DCL, list_box; 227; 457; 458
- caselle DCL, list_box, codice di causa delle funzioni di richiamo; 220
- caselle DCL, list_box, variabile \$reason; 220
- caselle DCL, nomi; 417
- caselle DCL, ok_cancel; 459
- caselle DCL, ok_cancel_help; 459
- caselle DCL, ok_cancel_help_errtile; 459
- caselle DCL, ok_cancel_help_info; 459; 460
- caselle DCL, ok_only; 458
- caselle DCL, paragraph; 460
- caselle DCL, per i pulsanti di uscita; 451
- caselle DCL, per la visualizzazione degli errori; 435
- caselle DCL, popup_list; 460
- caselle DCL, predefinite; 431
- caselle DCL, prototipi; 413
- caselle DCL, radio_button; 461
- caselle DCL, radio_button, valori; 222
- caselle DCL, radio_column; 462
- caselle DCL, radio_row; 462
- caselle DCL, riferimenti; 416; 417
- caselle DCL, row; 463
- caselle DCL, slider; 463
- caselle DCL, slider, variabile \$reason; 220
- caselle DCL, sottoinsiemi; 413
- caselle DCL, spacer; 465
- caselle DCL, spacer_0; 465
- caselle DCL, spacer_1; 466
- caselle DCL, text; 463; 464
- caselle DCL, text_part; 464
- caselle DCL, toggle; 464
- caselle DCL, valori di default; 430
- caselle di modifica; 441; 442; 456
- caselle visualizzate in grigio, Vedere caselle DCL, disabilitazione; 417
- caselle, Vedere anche caselle DCL; 413
- chiusura, file; 283
- chiusura, finestre di dialogo; 290
- cluster, caselle DCL; 422
- cluster, predefiniti; 431
- cluster, Vedere anche caselle DCL; 413
- CMDACTIVE, variabile di sistema; 216
- codici di gruppo dizionario ACAD_GROUP; 529
- codici di gruppo DXF; 193; 202; 478
- codici di gruppo DXF, dizionario ACAD_GROUP; 529
- codici di gruppo DXF, elenco dei tipi di oggetto; 498
- codici di gruppo DXF, elenco in ordine numerico; 479; 480
- codici di gruppo DXF, entità 3dface; 500
- codici di gruppo DXF, entità 3dsolid; 500
- codici di gruppo DXF, entità acad_proxy_entity; 526; 527
- codici di gruppo DXF, entità arc; 500
- codici di gruppo DXF, entità attdef; 501
- codici di gruppo DXF, entità attrib; 502
- codici di gruppo DXF, entità body; 502; 503
- codici di gruppo DXF, entità circle; 503
- codici di gruppo DXF, entità dimension; 503
- codici di gruppo DXF, entità ellipse; 508
- codici di gruppo DXF, entità endblk; 498
- codici di gruppo DXF, entità hatch; 509
- codici di gruppo DXF, entità image; 512

- codici di gruppo DXF, entità insert; 512
- codici di gruppo DXF, entità leader; 513
- codici di gruppo DXF, entità line; 514
- codici di gruppo DXF, entità lwpolyline; 514
- codici di gruppo DXF, entità mline; 515
- codici di gruppo DXF, entità mtext; 516
- codici di gruppo DXF, entità ole2frame; 517
- codici di gruppo DXF, entità oleframe; 516; 517
- codici di gruppo DXF, entità point; 518
- codici di gruppo DXF, entità polyline; 519
- codici di gruppo DXF, entità proxy; 526; 527
- codici di gruppo DXF, entità seqend; 520
- codici di gruppo DXF, entità shape; 521
- codici di gruppo DXF, entità solid; 521
- codici di gruppo DXF, entità spline; 521; 522
- codici di gruppo DXF, entità text; 522
- codici di gruppo DXF, entità tolerance; 523
- codici di gruppo DXF, entità trace; 524
- codici di gruppo DXF, entità vertex; 524
- codici di gruppo DXF, entità viewport; 525
- codici di gruppo DXF, entità xline; 526
- codici di gruppo DXF, tabella APPID; 490
- codici di gruppo DXF, tabella BLOCK_RECORD; 491
- codici di gruppo DXF, tabella DIMSTYLE; 491
- codici di gruppo DXF, tabella LAYER; 492
- codici di gruppo DXF, tabella LTYPE; 492; 493
- codici di gruppo DXF, tabella STYLE; 493
- codici di gruppo DXF, tabella UCS; 494
- codici di gruppo DXF, tabella VIEW; 494
- codici di gruppo DXF, tabella VPORT; 495
- codici di gruppo entità 3dface; 500
- codici di gruppo entità 3dsolid; 500
- codici di gruppo entità acad_proxy_entity; 526; 527
- codici di gruppo entità arc; 500
- codici di gruppo entità attdef; 501
- codici di gruppo entità attrib; 502
- codici di gruppo entità body; 502; 503
- codici di gruppo entità circle; 503
- codici di gruppo entità dimension; 503
- codici di gruppo entità ellipse; 508
- codici di gruppo entità endblk; 498
- codici di gruppo entità hatch; 509
- codici di gruppo entità image; 512
- codici di gruppo entità insert; 512
- codici di gruppo entità leader; 513
- codici di gruppo entità line; 514
- codici di gruppo entità lwpolyline; 514
- codici di gruppo entità mline; 515
- codici di gruppo entità mtext; 516
- codici di gruppo entità ole2frame; 517
- codici di gruppo entità oleframe; 516; 517
- codici di gruppo entità point; 518
- codici di gruppo entità polyline; 519
- codici di gruppo entità proxy; 526; 527
- codici di gruppo entità seqend; 520
- codici di gruppo entità shape; 521
- codici di gruppo entità solid; 521
- codici di gruppo entità spline; 521; 522
- codici di gruppo entità text; 522
- codici di gruppo entità tolerance; 523
- codici di gruppo entità trace; 524
- codici di gruppo entità vertex; 524
- codici di gruppo entità viewport; 525
- codici di gruppo entità xline; 526
- codici di gruppo tabella APPID; 490
- codici di gruppo tabella BLOCK_RECORD; 491
- codici di gruppo tabella DIMSTYLE; 491
- codici di gruppo tabella LAYER; 492
- codici di gruppo tabella LTYPE; 492; 493
- codici di gruppo tabella STYLE; 493
- codici di gruppo tabella UCS; 494
- codici di gruppo tabella VIEW; 494
- codici di gruppo tabella VPORT; 495
- codici di gruppo, dati estesi; 535
- codici di gruppo, Vedere anche codici di gruppo DXF; 478
- Comandi ARX, e gestione degli errori; 163
- comandi definiti esternamente comandi, accesso; 363
- comandi definiti esternamente comandi, sommario; 363
- comandi esterni, definizione; 41
- comandi trasparenti, AutoLISP; 159
- comandi, aggiunta di nuovi; 159
- comandi, alias dei; 43
- comandi, ANNULLA; 134; 163
- comandi, APPLOAD; 138
- comandi, ASEADMIN; 365
- comandi, ASEEXPORT; 366
- comandi, ASELINKS; 366
- comandi, ASEROWS; 366
- comandi, ASESELECT; 366
- comandi, ASESQLED; 366; 367; 369; 370
- comandi, ATTDEF; 151
- comandi, CAL; 369
- comandi, CARICA; 60
- comandi, caricamento automatico, AutoLISP; 140
- comandi, definiti esternamente; 363
- comandi, di tipo script; 133; 134
- comandi, DXFIN; 473
- comandi, DXFOUT; 473
- comandi, EDITATT; 202
- comandi, EDITPL; 202; 206
- comandi, eseguire; 263; 283
- comandi, FORMA; 60
- comandi, guida per definiti dall'utente; 338
- comandi, INSERT; 294
- comandi, LIBMAT; 379; 380
- comandi, LIBPAES; 375; 376
- comandi, LINEA; 151
- comandi, LUCE; 370
- comandi, MAPPAGGIO; 400
- comandi, MATERIALE; 385; 386
- comandi, NPAES; 379
- comandi, POLIMESH; 294
- comandi, procedura per la ricerca dei comandi; 39
- comandi, PSIN; 80; 381
- comandi, PSMOTIVO; 381
- comandi, PSOUT; 76
- comandi, PSTRASCINA; 380
- comandi, RENDER; 381; 382
- comandi, RENDERUPDATE; 385
- comandi, REPLAY; 385
- comandi, richiamo di nomi; 304
- comandi, ridefinizione; 159
- comandi, RIDIS; 172
- comandi, RIGEN; 206
- comandi, RPREF; 394

comandi, RUOTA3D; 393
 comandi, SALVAIMM; 396
 comandi, SCENA; 397
 comandi, SCRIPT; 134
 comandi, SFONDO; 367
 comandi, SHOWMAT; 401
 comandi, SOLPROFILO; 401
 comandi, SPECCHIO3D; 380
 comandi, STILE; 67; 73
 comandi, TAVOLET; 118
 comandi, TESTO; 151
 comandi, TESTODIN; 151
 comandi, TLINEA; 53
 comandi, versioni in altre lingue; 263; 283
 commenti, interni alla riga, AutoLisp; 149
 commenti, nei file AutoLISP; 149
 commenti, nei file DCL; 418; 419
 commenti, nei file di script; 134
 commenti, nel file di definizione dell'unità; 186
 commutazione. Vedere modifica; 295
 complessi, tipi di linea; 53
 compressione di font di testo; 67
 concatenazione, espressioni; 319
 concatenazione, funzioni car e cdr; 281
 concatenazione, liste; 275; 285; 319
 configurazione, controllo, AutoLISP; 169
 contrassegni di sottoclasse; 533
 controllo degli errori, AutoLISP; 173
 controllo delle finestre di dialogo. Vedere AutoLISP, finestra di dialogo; 216
 convalida. Vedere verifica; 278
 convenzioni dei caratteri minuscoli/maiuscoli parole chiave; 316
 conversione di numeri reali, AutoLISP; 181
 conversione di stringhe, AutoLISP; 181
 coordinate, OCS; 537
 coppie puntate; 285
 coppie puntate, AutoLISP; 158
 corso interattivo, AutoLISP; 232
 corso interattivo, DCL; 239
 corso interattivo, funzioni di finestra di dialogo AutoLISP; 239
 curve, polilinee; 206
 CVPORT, variabile di sistema; 339

D

DABLOCCO, pulsante, abilitazione/disabilitazione; 271
 DALAYER, pulsante, abilitazione/disabilitazione; 271
 dati arbitrari; 213
 dati binari, nei dati estesi; 209
 dati di definizione, oggetti (entità); 291; 292; 293; 295; 325
 dati estesi, codici; 193
 dati estesi, codici di gruppo; 209
 dati estesi, descrizione; 473
 dati estesi, formato DXF; 535
 dati estesi, gestori; 213
 dati estesi, organizzazione; 209
 dati estesi, recupero; 209
 dati estesi, recupero con AutoLISP; 211
 dati estesi, unione ad una entità; 212

dati per la trasformazione delle coordinate; 199
 DCL; 413
 DCL, attributi predefiniti, multiple_select; 447
 DCL, caricamento di file; 216
 DCL, corso interattivo; 239
 DCL, file di esempio, ciao.dcl; 216
 DCL, file di esempio, ddgp.dcl; 239
 DCL, file di esempio, getpass.dcl; 226
 DCL, finestra di dialogo, abbreviazioni; 428; 429
 DCL, inclusione di commenti; 418; 419
 DCL, modifica di modalità con AutoLISP; 221; 222
 DCL, sintassi; 416; 417
 DCS (Display coordinate system, sistema di coordinate di visualizzazione); 186
 DCS dello Spazio Carta (DCSSC); 186
 DCSSC (DCS dello Spazio Carta); 186
 DEFATT, comando; 151
 descrittori di file in AutoLISP; 148
 Diametro, comando secondario; 284
 diapositive di immagini; 110
 DIESEL; 120; 121
 DIESEL, catalogo delle funzioni; 126
 DIESEL, espressioni AutoLISP; 126; 159
 DIESEL, espressioni di menu; 323
 DIESEL, espressioni, incluse nei menu; 94
 DIESEL, funzioni,; 128
 DIESEL, funzioni, != (not equal to); 128
 DIESEL, funzioni, (maggiore di); 128
 DIESEL, funzioni, = (maggiore di o uguale a); 128
 DIESEL, funzioni, and; 128
 DIESEL, funzioni, angtos; 129
 DIESEL, funzioni, edtime; 129
 DIESEL, funzioni, eq; 130
 DIESEL, funzioni, eval; 130
 DIESEL, funzioni, fix; 130
 DIESEL, funzioni, getenv; 130
 DIESEL, funzioni, getvar; 131
 DIESEL, funzioni, if; 131
 DIESEL, funzioni, index; 131
 DIESEL, funzioni, linelen; 131
 DIESEL, funzioni, nth; 132
 DIESEL, funzioni, or; 132
 DIESEL, funzioni, rtos; 132
 DIESEL, funzioni, strlen; 132
 DIESEL, funzioni, substr; 132; 133
 DIESEL, funzioni, upper; 133
 DIESEL, funzioni, xor; 133
 DIESEL, immissione di espressioni alla riga di comando; 126
 DIESEL, in AutoLISP; 122; 123
 DIESEL, messaggi di errore; 133
 DIESEL, messaggi di errore, \$(++); 133
 DIESEL, messaggi di errore, \$?; 133
 DIESEL, nei menu; 124
 digitalizzatori, impostazione dei menu; 118
 Dim, messaggio di richiesta, Diametro, comando secondario; 284
 Dim, messaggio di richiesta, Raggio, comando secondario; 284
 DIMZIN, variabile di sistema; 181
 directory sample; 138
 directory, barra come delimitatore; 320
 directory, bonus; 138

directory, delimitatori, (barra rovesciata); 320; 329
 directory, delimitatori, / (barra); 320
 directory, delimitatori, barra (/); 138; 141; 142
 directory, delimitatori, barra rovesciata (); 138; 141; 142
 direttiva include; 416
 direzione globale, nei dati estesi; 209
 disegni; 80
 disegno grafico vettoriale, visualizzazione nella finestra di dialogo; 456; 457
 dispositivi di puntamento, personalizzazione dei menu per; 96
 dispositivi di scorrimento, nelle finestre di dialogo; 433; 434; 440; 446; 463
 distanza, nei dati estesi; 209
 dizionari; 215
 dizionari, oggetto nome; 325
 dizionari, ricerca; 289
 dizionari, trovare l'elemento successivo; 288
 dizionario degli oggetti denominati; 527
 dizionario delle estensioni; 534
 DOS, comandi, richiamo; 41
 Drawing Interchange Binary. Vedere file DXB; 467
 DXFIN, comando; 473
 DXFOUT, comando; 473

—E—

ECS (Entity coordinate system, sistema di coordinate di entità). Vedere OCS. ; 186
 EDITATT, comando; 202
 EDITPL, comando; 202; 206
 elementi curvatura; 474
 elementi estensione di linea; 474
 elementi larghezza; 474
 elementi modo numerico; 474
 elementi, aggiungere alle liste; 285
 elementi, elemento n delle liste; 328
 elementi, numero nelle liste; 319
 elementi, ultimo elemento delle liste; 318
 elenchi di associazione, AutoLISP; 158
 elenchi punto, AutoLISP; 157
 elenchi, associazione, AutoLISP; 158
 elenchi, coppie puntate, AutoLISP; 158
 elenchi, lettura; 333; 334
 elenco, funzioni; 279
 elenco, simboli; 279
 eliminazione, oggetti; 291
 eliminazione, oggetti, da gruppi di selezione; 341
 ENAME, tipo di simbolo; 356
 entità, elenchi di definizione; 192
 entità, nomi; 192
 entità. Vedere oggetti (entità); 293
 ERRNO, variabile di sistema; 406
 esci/continua, AutoLISP; 299
 esecuzione di comandi; 263; 283
 espansione di font di testo; 67
 espressioni, concatenazione; 319
 espressioni, scrittura in un file; 331
 espressioni, stampa sulla riga di comando; 331
 espressioni, valutazione; 299; 302; 320
 espressioni, Vedere anche liste; 299
 estensioni di file, .ahp; 40; 314

estensioni di file, .ccp; 40
 estensioni di file, .cus; 40
 estensioni di file, .dcl; 40; 320; 413
 estensioni di file, .dxb; 474
 estensioni di file, .dxt; 40
 estensioni di file, .hlp; 314
 estensioni di file, .lin; 40; 53
 estensioni di file, .lsp; 40; 138; 149; 165; 320
 estensioni di file, .mln; 40
 estensioni di file, .mnd; 40
 estensioni di file, .mnl; 40; 140; 141; 142; 149
 estensioni di file, .mns; 40
 estensioni di file, .mnu; 40
 estensioni di file, .pat; 40
 estensioni di file, .pcp; 40
 estensioni di file, .pfb; 61
 estensioni di file, .rpf; 40
 estensioni di file, .scr; 40; 134
 estensioni di file, .shp; 40; 61
 estensioni di file, .shx; 61
 estensioni di file, .slb; 229
 estensioni di file, .sld; 229
 EXSUBR, tipo di simbolo; 356

—F—

fattore di scala; 474
 fattore di scala, nei dati estesi; 209
 file batch, richiamo; 41
 file binari di interscambio dei disegni. Vedere file DXB; 473
 file DCL, caricamento; 320
 file DCL, gerarchia; 413
 file DCL, riferimenti ad altri file DCL; 416
 file DCL, verifica semantica; 419
 file dei parametri di programma; 41
 file della mappa dei font (fontmap.ps); 80
 file di definizione delle unità, commenti dell'utente; 186
 file di font, Vedere file di forma; 61
 file di forma, big font; 68
 file di forma, codice di direzione; 62
 file di forma, codici, archi specificati da curvature; 66
 file di forma, codici, arco frazionario; 65
 file di forma, codici, arco ottante; 65
 file di forma, codici, flag relativo al testo verticale; 66
 file di forma, codici, forma secondaria; 64
 file di forma, codici, spostamenti X-Y; 64
 file di forma, compilazione; 61
 file di forma, descrizione dei font di testo; 67
 file di forma, file di definizione; 61
 file di forma, font Unicode; 73; 74
 file di forma, font, creazione; 67
 file di forma, lunghezza dei vettori; 62
 file di Guida della Release 12, conversione; 51; 52
 file di Guida, AutoCAD; 40
 file di modelli; 58
 file di script, commenti; 134
 file di script, creazione; 134
 file di supporto personalizzabili; 40
 file di testo, lettura dei comandi; 134
 file DLC multipli, caricamento; 320
 file DXB; 467; 473

- file DXB caricamento; 473
- file DXB da definire in blocco; 41
- file DXB scrittura; 473
- file DXB, caricamento; 41
- file DXB, formato; 474
- file DXB, gruppo nuovo colore; 474
- file DXF; 467; 468; 473
- file DXF binari; 467; 473
- file DXF binari, Vedere anche file DXF; 467
- file DXF, algoritmo dell'asse arbitrario; 537
- file DXF, binari; 473
- file DXF, codici degli oggetti e delle entità; 481; 482
- file DXF, codici di gruppo, dati estesi; 535
- file DXF, codici di gruppo, sezione BLOCKS; 495; 496
- file DXF, codici di gruppo, sezione CLASSES; 487
- file DXF, contrassegni di sottoclasse; 533
- file DXF, dati estesi di entità; 535
- file DXF, dizionario delle estensioni; 534
- file DXF, oggetti di database; 531
- file DXF, programmi interfaccia; 470; 471
- file DXF, reattori costanti; 534
- file DXF, scrittura di programmi interfaccia; 470; 471
- file DXF, sezione BLOCKS; 495; 496
- file DXF, sezione CLASSES; 487
- file DXF, sezione ENTITIES; 498
- file DXF, sezione HEADER; 482
- file DXF, sezione OBJECTS; 527
- file DXF, sezione TABLES; 488
- file Encapsulated PostScript (EPS), esportazione dei disegni; 76
- file immagine. Vedere caselle DCL, image_button; 417
- file, \$cmd.dxb; 41
- file, acad.ase; 40
- file, acad.cfg; 304; 337; 338
- file, acad.cfg, sezione AppData; 169
- file, acad.dce; 419
- file, acad.dcl; 40
- file, acad.lin; 40; 53
- file, acad.lsp; 40; 122; 123; 140; 404
- file, acad.mnl; 40; 140
- file, acad.mns; 40
- file, acad.mnu; 40
- file, acad.pat; 40; 57; 58
- file, acad.pgp; 40; 41; 43
- file, acad.psf; 40; 76
- file, acad.rx; 40; 141; 142
- file, acad.unt; 40; 184
- file, acadfull.mnu; 40
- file, acadpsd.ps; 80
- file, acadpsf.ps; 80
- file, acadpsi.ps; 80
- file, acadpss.ps; 80
- file, ai_utils.lsp; 149
- file, apertura; 329
- file, ascii.txt; 466
- file, base.dcl; 424; 450
- file, bmake.lsp, programma di esempio; 224
- file, caricamento; 320
- file, chiusura; 283
- file, CTRL-Z contrassegni di fine file; 329
- file, ddgp.dcl; 239
- file, diapositiva; 229
- file, fontmap.ps; 40; 80
- file, lettura, caratteri di; 334
- file, lettura, stringhe di; 334
- file, nomi, specificati dall'utente; 305; 306
- file, personalizzabili; 40
- file, ricerca; 301
- file, scrittura, espressioni; 331
- FILE, tipo di simbolo; 356
- filtraggio di gruppi di selezione, AutoLISP; 192
- filtri del gruppo di selezione; 343
- finestra di disegno, controllo con AutoLISP; 172
- finestra di testo, controllo con AutoLISP; 172
- finestra, cancellazione; 310
- finestra, disegno di vettori; 311; 313
- finestra, funzione entmake e; 202
- finestra, funzione entmod e; 202
- finestra, ridisegnare; 335
- finestre di dialogo, aggiunta; 328
- finestre di dialogo, apertura; 320; 328
- finestre di dialogo, attributi; 303
- finestre di dialogo, AutoLISP; 133; 134
- finestre di dialogo, caselle, associazione dei dati gestiti dall'applicazione; 283
- finestre di dialogo, chiusura; 290
- finestre di dialogo, componenti. Vedere caselle; 413
- finestre di dialogo, default, layout; 421
- finestre di dialogo, default, valori; 430
- finestre di dialogo, definizione di nuove; 455
- finestre di dialogo, gestione degli errori; 435
- finestre di dialogo, impostazione dello spazio tra le caselle; 422
- finestre di dialogo, inclusione della funzione di guida in linea; 427
- finestre di dialogo, inizializzazione; 328
- finestre di dialogo, menu a gruppi di immagini; 110
- finestre di dialogo, nascondere; 430
- finestre di dialogo, nidificate; 429; 430
- finestre di dialogo, progettazione; 425
- finestre di dialogo, progettazione, per interfacce GUI; 425
- finestre di dialogo, progettazione, per utenti disabili; 427; 428
- finestre di dialogo, progettazione, uso delle lettere maiuscole; 428
- finestre di dialogo, pulsanti; 431
- finestre di dialogo, Seleziona il file di forma; 61
- finestre di dialogo, Selezione colore; 229; 271
- finestre di dialogo, testo dei pulsanti di uscita; 424
- finestre di dialogo, traduzione in altre lingue; 431
- finestre di dialogo, Vedere anche caselle DCL; 413
- font Unicode; 73; 74
- font, big font; 68
- font, creazione; 67
- font, descrizione dei font di testo; 67
- font, PostScript; 61
- font, Unicode; 73; 74
- FORMA, comando; 60
- formati dei file; 467
- formati di interscambio dei file; 467
- formato dei file; 474
- funzione script; 134
- funzione, tasti, Grafico/Testo; 310
- funzioni AutoLISP definite dall'utente; 162
- funzioni AutoLISP,; 156; 268; 269; 281

funzioni AutoLISP, (maggiore di); 269
 funzioni AutoLISP, ' (quote). Vedere funzioni AutoLISP, quote; 157
 funzioni AutoLISP, - (sottrazione); 152
 funzioni AutoLISP, conversioni, angoli; 274
 funzioni AutoLISP, conversioni, ASCII; 282
 funzioni AutoLISP, conversioni, numeri interi; 278
 funzioni AutoLISP, conversioni, numeri reali; 278
 funzioni AutoLISP, decremento (1-); 270
 funzioni AutoLISP, entmake; 293
 funzioni AutoLISP, get_attr; 303
 funzioni AutoLISP, getenv; 305
 funzioni AutoLISP, graphscr; 310
 funzioni AutoLISP, incremento (1+); 270
 funzioni AutoLISP, lambda; 318
 funzioni AutoLISP, last; 318
 funzioni AutoLISP, list; 319
 funzioni AutoLISP, logand; 321
 funzioni AutoLISP, maggiore di o uguale a (=); 269
 funzioni AutoLISP, massimo comune denominatore; 302
 funzioni AutoLISP, namedobjdict; 325
 funzioni AutoLISP, operatori relazionali, maggiore di o uguale a (=); 269
 funzioni AutoLISP, (end_list); 291
 funzioni AutoLISP, (maggiore di); 156
 funzioni AutoLISP, * (moltiplicazione); 144; 152
 funzioni AutoLISP, *error*; 163; 298; 408
 funzioni AutoLISP, / (divisione); 152; 268
 funzioni AutoLISP, /= (diverso da); 156
 funzioni AutoLISP, /= (diverso da); 268
 funzioni AutoLISP, ~ (NOT a livello bit); 152
 funzioni AutoLISP, ~ (not); 270
 funzioni AutoLISP, + (addizione); 144; 152
 funzioni AutoLISP, = (maggiore di o uguale a); 156; 269
 funzioni AutoLISP, = (uguale a); 156; 268
 funzioni AutoLISP, 1+ (decremento); 152; 270
 funzioni AutoLISP, 1+ (incremento); 152; 270
 funzioni AutoLISP, abs; 152; 270
 funzioni AutoLISP, acad_colordlg; 166; 271
 funzioni AutoLISP, acad_helpdlg; 271
 funzioni AutoLISP, acad_strlsort; 271
 funzioni AutoLISP, action_tile; 218; 220; 221; 222; 231; 272
 funzioni AutoLISP, add_list; 227; 231; 272
 funzioni AutoLISP, ads; 164; 272; 273
 funzioni AutoLISP, alert; 163; 273
 funzioni AutoLISP, alloc; 273; 404
 funzioni AutoLISP, and; 156; 273
 funzioni AutoLISP, angle; 177; 274
 funzioni AutoLISP, angtof; 152; 153; 180; 181; 274
 funzioni AutoLISP, angtos; 152; 153; 180; 181; 274; 275
 funzioni AutoLISP, append; 156; 275
 funzioni AutoLISP, apply; 159; 275
 funzioni AutoLISP, arcotangente; 277
 funzioni AutoLISP, arx; 164; 276
 funzioni AutoLISP, arxload; 141; 164; 276
 funzioni AutoLISP, arxunload; 141; 164; 276
 funzioni AutoLISP, ascii; 152; 153; 180; 181; 183
 funzioni AutoLISP, assoc; 156; 158; 199; 202; 277
 funzioni AutoLISP, atan; 152; 277
 funzioni AutoLISP, atof; 152; 153; 180; 181; 182; 278
 funzioni AutoLISP, atoi; 152; 153; 180; 181; 221; 222; 227; 278
 funzioni AutoLISP, atom; 159; 278
 funzioni AutoLISP, atoms_family; 159; 279
 funzioni AutoLISP, autoarxload; 164; 279
 funzioni AutoLISP, autoload; 140; 164; 279; 280
 funzioni AutoLISP, autoxload; 164; 280
 funzioni AutoLISP, barra (/), divisione; 268
 funzioni AutoLISP, Boole; 156; 193; 280
 funzioni AutoLISP, boundp; 159; 281
 funzioni AutoLISP, caddr; 157
 funzioni AutoLISP, cadr; 157
 funzioni AutoLISP, car; 156; 157; 158
 funzioni AutoLISP, cdr; 156; 158; 285
 funzioni AutoLISP, chr; 152; 153; 180; 181; 183; 282
 funzioni AutoLISP, client_data_tile; 230; 231; 283
 funzioni AutoLISP, close; 283
 funzioni AutoLISP, command; 138; 151; 159; 198; 202; 216; 236; 263; 283
 funzioni AutoLISP, cond; 156; 285
 funzioni AutoLISP, cons; 156; 158; 193; 285
 funzioni AutoLISP, conversioni, ASCII; 277
 funzioni AutoLISP, conversioni, conversione in lettere minuscole; 349
 funzioni AutoLISP, conversioni, numeri interi; 282; 301; 317; 318
 funzioni AutoLISP, conversioni, numeri reali; 290; 301
 funzioni AutoLISP, conversioni, sistemi di coordinate; 325; 327
 funzioni AutoLISP, conversioni, unità di misura; 286
 funzioni AutoLISP, cos; 152; 286
 funzioni AutoLISP, cvunit; 180; 181; 184; 286
 funzioni AutoLISP, defun; 159; 165; 263; 283; 286; 405
 funzioni AutoLISP, dictnext; 214; 215; 288
 funzioni AutoLISP, dictsearch; 214; 215; 289
 funzioni AutoLISP, dimx_tile; 228; 289
 funzioni AutoLISP, dimy_tile; 228; 289
 funzioni AutoLISP, distance; 177; 289
 funzioni AutoLISP, distof; 152; 153; 180; 181; 290
 funzioni AutoLISP, diverso da (/=); 268
 funzioni AutoLISP, divisione (/); 268
 funzioni AutoLISP, done_dialog; 216; 224; 231; 290
 funzioni AutoLISP, done_dialogfunzioni AutoLISP, load_dialog; 216
 funzioni AutoLISP, end_image; 228; 231; 291
 funzioni AutoLISP, end_list; 227; 231
 funzioni AutoLISP, entdel; 197; 202; 214; 215; 216; 291
 funzioni AutoLISP, entget; 186; 193; 197; 199; 202; 214; 291; 292; 293
 funzioni AutoLISP, entget, corrispondenza con caratteri jolly; 155
 funzioni AutoLISP, entget, ed i record della tabella dei simboli; 207
 funzioni AutoLISP, entget, recupero dei dati estesi; 211
 funzioni AutoLISP, entlast; 147; 196; 197; 198; 292; 293
 funzioni AutoLISP, entmake; 186; 197; 202; 214; 215; 216
 funzioni AutoLISP, entmake, creazione dei dati estesi; 211
 funzioni AutoLISP, entmakex; 295
 funzioni AutoLISP, entmod; 186; 197; 202; 206; 214;

- 216; 295
- funzioni AutoLISP, entmod, modifica dei dati estesi; 211
- funzioni AutoLISP, entnext; 196; 197; 199; 206; 296
- funzioni AutoLISP, entsel; 173; 174; 198; 199; 216; 296; 315
- funzioni AutoLISP, entupd; 197; 206; 216; 295; 297
- funzioni AutoLISP, eq; 156; 297
- funzioni AutoLISP, equal; 156; 298
- funzioni AutoLISP, eval; 159; 299
- funzioni AutoLISP, exit; 163; 299
- funzioni AutoLISP, exp; 152; 299; 300
- funzioni AutoLISP, expand; 300; 404
- funzioni AutoLISP, expt; 152; 300
- funzioni AutoLISP, fill_image; 228; 229; 231; 300
- funzioni AutoLISP, findfile; 301
- funzioni AutoLISP, fix; 152; 301
- funzioni AutoLISP, float; 152; 301; 302
- funzioni AutoLISP, foreach; 156; 302
- funzioni AutoLISP, gc; 192; 302; 402
- funzioni AutoLISP, gcd; 152; 302
- funzioni AutoLISP, get_attr; 218; 221; 222; 231
- funzioni AutoLISP, get_tile; 218; 221; 222; 231; 303
- funzioni AutoLISP, getangle; 173; 216; 303; 315
- funzioni AutoLISP, getcfg; 166; 169; 173; 304
- funzioni AutoLISP, getcorner; 173; 216; 304; 315
- funzioni AutoLISP, getdist; 173; 216; 305; 315
- funzioni AutoLISP, getenv; 166; 169; 173
- funzioni AutoLISP, getfiled; 173; 305; 306
- funzioni AutoLISP, getint; 173; 175; 216; 307; 315
- funzioni AutoLISP, getkeyword; 173; 216; 307; 308; 315
- funzioni AutoLISP, getorient; 173; 216; 308; 315
- funzioni AutoLISP, getpoint; 173; 216; 309; 315
- funzioni AutoLISP, getreal; 173; 216; 309; 315
- funzioni AutoLISP, getstring; 173; 216; 309; 315
- funzioni AutoLISP, getvar; 163; 166; 169; 173; 238; 310
- funzioni AutoLISP, graphscr; 169; 172; 216
- funzioni AutoLISP, grclear; 169; 216; 310
- funzioni AutoLISP, grdraw; 169; 172; 216; 311
- funzioni AutoLISP, grread; 189; 216; 311
- funzioni AutoLISP, grtext; 169; 172; 216; 313
- funzioni AutoLISP, grvecs; 169; 172; 199; 216; 313
- funzioni AutoLISP, handent; 197; 213; 214; 314
- funzioni AutoLISP, help; 314; 427
- funzioni AutoLISP, if; 156; 236; 315
- funzioni AutoLISP, initget; 159; 173; 174; 198; 315
- funzioni AutoLISP, initiget; 173
- funzioni AutoLISP, inters; 177; 317
- funzioni AutoLISP, itoa; 152; 153; 180; 181; 317; 318
- funzioni AutoLISP, lambda; 159
- funzioni AutoLISP, last; 156
- funzioni AutoLISP, length; 156
- funzioni AutoLISP, list; 156; 157; 193
- funzioni AutoLISP, listp; 156; 319
- funzioni AutoLISP, load; 138; 164; 165; 320
- funzioni AutoLISP, load_dialog; 216; 231; 320; 419
- funzioni AutoLISP, log; 152; 321
- funzioni AutoLISP, logand; 152
- funzioni AutoLISP, logior; 152; 321
- funzioni AutoLISP, lsh; 152; 321
- funzioni AutoLISP, maggiore di (); 269
- funzioni AutoLISP, mapcar; 156; 227; 321; 322
- funzioni AutoLISP, max; 152; 322
- funzioni AutoLISP, mem; 322; 405
- funzioni AutoLISP, member; 156; 323
- funzioni AutoLISP, menucmd; 126; 169; 216; 323
- funzioni AutoLISP, min; 152; 324
- funzioni AutoLISP, minusp; 152; 324
- funzioni AutoLISP, mode_tile; 218; 221; 222; 231; 325
- funzioni AutoLISP, mode_tile, modalità; 221; 222
- funzioni AutoLISP, namedobjdict; 214
- funzioni AutoLISP, nentsel; 173; 174; 198; 199; 216; 315; 325
- funzioni AutoLISP, nentselp; 173; 174; 199; 315; 327
- funzioni AutoLISP, new_dialog; 216; 218; 220; 224; 231; 328; 419
- funzioni AutoLISP, not; 159; 328
- funzioni AutoLISP, nth; 156; 328
- funzioni AutoLISP, null; 159; 329
- funzioni AutoLISP, numberp; 159; 329
- funzioni AutoLISP, open; 329
- funzioni AutoLISP, operatori relazionali, diverso da (/=); 268
- funzioni AutoLISP, operatori relazionali, maggiore di (); 269
- funzioni AutoLISP, operatori relazionali, uguale a (=); 268
- funzioni AutoLISP, oppure; 156
- funzioni AutoLISP, or; 330
- funzioni AutoLISP, osnap; 177; 216; 330
- funzioni AutoLISP, polar; 177; 236; 331
- funzioni AutoLISP, prin1; 152; 153; 165; 169; 331
- funzioni AutoLISP, prin1, visualizzazione delle informazioni di debug; 216
- funzioni AutoLISP, princ; 140; 152; 153; 154; 163; 169; 238
- funzioni AutoLISP, princ, e messaggi di errore; 163
- funzioni AutoLISP, princ, visualizzazione delle informazioni di debug; 216
- funzioni AutoLISP, print; 152; 153; 165; 169
- funzioni AutoLISP, print, visualizzazione delle informazioni di debug; 216
- funzioni AutoLISP, progn; 159
- funzioni AutoLISP, prompt; 152; 153; 154; 169; 216
- funzioni AutoLISP, quit; 163
- funzioni AutoLISP, quote; 157; 159; 319
- funzioni AutoLISP, read; 227; 333; 334
- funzioni AutoLISP, read-char; 170; 189; 334
- funzioni AutoLISP, read-line; 170; 189; 334
- funzioni AutoLISP, redraw; 169; 172; 216; 335
- funzioni AutoLISP, rem; 152; 335; 336
- funzioni AutoLISP, repeat; 156
- funzioni AutoLISP, reverse; 156
- funzioni AutoLISP, rtos; 152; 153; 180; 181
- funzioni AutoLISP, S::STARTUP; 263; 283
- funzioni AutoLISP, set; 159
- funzioni AutoLISP, set_tile; 218; 221; 231
- funzioni AutoLISP, setcfg; 166; 169; 337; 338
- funzioni AutoLISP, setfunhelp; 159; 338
- funzioni AutoLISP, setq; 148; 150; 151; 159; 236; 238; 339
- funzioni AutoLISP, setvar; 166; 169; 238
- funzioni AutoLISP, sin; 152
- funzioni AutoLISP, slide_image; 229; 231
- funzioni AutoLISP, svalid; 214; 340
- funzioni AutoLISP, sqrt; 152
- funzioni AutoLISP, ssadd; 192; 196; 198; 341

funzioni AutoLISP, ssdel; 192; 196; 341
 funzioni AutoLISP, ssgget; 192; 216
 funzioni AutoLISP, ssgget, corrispondenza con caratteri jolly; 155
 funzioni AutoLISP, sslength; 192; 196
 funzioni AutoLISP, ssmemb; 192; 196
 funzioni AutoLISP, ssname; 192; 196
 funzioni AutoLISP, start_dialog; 216; 218; 224; 231
 funzioni AutoLISP, start_image; 229; 231
 funzioni AutoLISP, start_list; 227; 231
 funzioni AutoLISP, startapp; 164
 funzioni AutoLISP, strcase; 152; 153; 349
 funzioni AutoLISP, strcat; 152; 153
 funzioni AutoLISP, strlen; 152; 153
 funzioni AutoLISP, subst; 156
 funzioni AutoLISP, substr; 152; 153; 156
 funzioni AutoLISP, tablet; 189
 funzioni AutoLISP, tblnext; 207; 214
 funzioni AutoLISP, tblobjname; 207; 214; 352
 funzioni AutoLISP, tsearch; 207; 214
 funzioni AutoLISP, term_dialog; 216; 224
 funzioni AutoLISP, terpri; 169
 funzioni AutoLISP, textbox; 177
 funzioni AutoLISP, textpage; 169; 172; 216
 funzioni AutoLISP, textscr; 169; 172; 216
 funzioni AutoLISP, trace; 159
 funzioni AutoLISP, trans; 180; 181; 186; 354; 355
 funzioni AutoLISP, trans, codici del sistema di coordinate; 186
 funzioni AutoLISP, type; 159; 356
 funzioni AutoLISP, uguale a (=); 268
 funzioni AutoLISP, unload_dialog; 216; 231; 419
 funzioni AutoLISP, untrace; 159
 funzioni AutoLISP, vector_image; 229; 231
 funzioni AutoLISP, ver; 166
 funzioni AutoLISP, vmon; 405
 funzioni AutoLISP, vports; 169
 funzioni AutoLISP, wcmatch; 152; 153; 155
 funzioni AutoLISP, while; 156; 236
 funzioni AutoLISP, write-char; 170
 funzioni AutoLISP, write-line; 170
 funzioni AutoLISP, xdroom; 213
 funzioni AutoLISP, xdsiz; 213
 funzioni AutoLISP, xload; 141; 142; 164; 362
 funzioni AutoLISP, xunload; 142; 164
 funzioni AutoLISP, zerop; 152
 funzioni condizionali, AutoLISP; 180; 181
 funzioni di accesso ai dispositivi, AutoLISP; 189
 funzioni di controllo della visualizzazione, AutoLISP; 169
 funzioni di gestione degli elenchi, AutoLISP; 156
 funzioni di gestione degli oggetti, AutoLISP; 197
 funzioni di gestione dei file, AutoLISP; 170
 funzioni di gestione dei gruppi di selezione, AutoLISP; 192
 funzioni di gestione della memoria,; 402; 404
 funzioni di gestione dell'applicazione, applicazioni ADS; 272; 273
 funzioni di gestione dell'applicazione, applicazioni ARX; 276
 funzioni di gestione dell'applicazione, associazione dei dati gestiti dall'applicazione alle caselle delle finestre di dialogo; 283

funzioni di input utente, AutoLISP; 173
 funzioni di utilità generali, AutoLISP; 166
 funzioni di utilità geometriche, AutoLISP; 177
 funzioni per gestione delle stringhe, AutoLISP; 152; 153
 funzioni per i calcoli numerici, AutoLISP; 152
 funzioni per la gestione degli errori, AutoLISP; 163
 funzioni per la gestione delle applicazioni, AutoLISP; 164
 funzioni riguardanti i nomi di entità, AutoLISP; 198
 funzioni,; 405
 funzioni, anonime; 318
 funzioni, applicazioni ADS; 272; 273
 funzioni, applicazioni ARX; 276
 funzioni, define; 338
 funzioni, definire; 286
 funzioni, definite esternamente; 260
 funzioni, elenco; 279
 funzioni, g; 283
 funzioni, gestione, AutoLISP; 159
 funzioni, gruppo di selezione; 341
 funzioni, incorporate; 286
 funzioni, lista restituita da; 321; 322
 funzioni, parole chiave; 315
 funzioni, ricorsività; 320
 funzioni, sintassi in AutoLISP; 148

—G—

gestione degli errori DCL; 419
 gestione della memoria; 402
 gestione della memoria, allocazione manuale; 300
 gestione della memoria, heap; 402
 gestione della memoria, ripulitura dello spazio; 302
 gestione della memoria, statistiche; 322
 gestori; 191
 gestori database, nei dati estesi; 209
 gestori, e la funzione entmake; 207
 gestori, e la funzione entmod; 207
 gestori, nei dati estesi; 213
 globale, posizione, nei dati estesi; 209
 globale, spostamento, nei dati estesi; 209
 gradi, conversione in radianti; 182
 grafica, a basso livello, AutoLISP; 172
 grafica, disegno di vettori, sullo schermo grafico; 311; 313
 grafica, ridisegnare, finestra; 335
 grafica, ridisegnare, oggetto; 335
 grafica, visualizzazione dello schermo grafico; 310
 Grafico/Testo, tasto funzione; 310
 gruppi di immagini; 433
 gruppi di selezione, creazione con AutoLISP; 192
 gruppi di selezione, funzioni; 341
 guida, per i comandi definiti dall'utente; 338
 guida, richiamo; 271; 314
 guida, specifica di menu; 118; 119

—H—

heap; 402
 heap, spazio per i nodi; 402

heap, spazio per le stringhe; 404

— I —

immagine grafica, visualizzazione nella finestra di dialogo; 457
 importazione, file Encapsulated PostScript (EPS); 80
 inclinazione di font di testo; 67
 incorporate, funzioni; 286
 indentazione, nei file AutoLISP; 150
 inizializzazione di finestre di dialogo; 328
 INSER, comando; 294
 INT, tipo di simbolo; 356
 interfaccia di programmazione definibile dall'utente; 133; 134
 interruttore. Vedere modifica; 295
 interruttori nelle finestre di dialogo; 434
 intestazioni di applicazioni AutoLISP; 138

— L —

lettere minuscole, Vedere convenzioni dei caratteri minuscoli/maiuscoli; 349
 lettura, caratteri; 334
 lettura, dispositivi di input AutoCAD; 311
 lettura, stringhe; 333; 334
 LIBMAT, comando; 379; 380
 LIBPAES, comando; 376
 limiti, lunghezza della stringa; 309
 limiti, numeri interi; 307
 limiti, valore della variabile di sistema; 339
 limiti, Vedere anche funzioni AutoLISP, max; 307
 limiti, Vedere anche funzioni AutoLISP, min; 307
 limiti, vertici per elemento faccia; 294
 LINEA, comando; 151
 linee, intersezione di; 317
 linee, punteggiate; 53
 linee, tratteggiate; 53
 linee, Vedere anche intersezione di; 317
 linguaggio DCL. Vedere DCL; 413
 lingue, uso nelle finestre di dialogo; 431
 LISP (Addison-Wesley); 143; 144
 LIST, tipo di simbolo; 356
 liste, associazione; 277
 liste, coppie puntate; 285
 liste, costruzione/concatenazione; 275; 285; 319
 liste, elemento n; 328
 liste, lunghezza; 319
 liste, ordinamento; 271
 liste, restituzione di liste da parte di funzioni; 321; 322
 liste, resto di; 323
 liste, ricerca; 277; 323
 liste, ultimo elemento; 318
 liste, valutazione delle espressioni di tutti i membri; 302
 liste, verifica; 319
 Looking at LISP (Addison-Wesley); 143; 144
 LUCE, comando; 370
 lunghezza, liste; 319
 lunghezza, stringhe, limite; 309
 LUNITS, variabile di sistema; 181

LUPREC, variabile di sistema; 181

— M —

macro di menu; 82
 MAPPAGGIO, comando; 400
 massimi, Vedere limiti; 307
 MATERIALE, comando; 385; 386
 matrice di trasformazione da Modello a Globale; 199; 325
 matrice, funzioni di moltiplicazione; 187
 matrice, trasformazione; 187; 325; 327
 menu, a cursore, numero massimo di voci; 98; 99
 menu, a cursore, Vedere anche menu, a discesa; 98; 99
 menu, a discesa; 98; 99
 menu, a discesa, caricamento; 105
 menu, a discesa, controllo della visualizzazione di etichette; 101
 menu, a discesa, etichette, separazione; 101
 menu, a discesa, numero massimo di voci; 98; 99
 menu, a discesa, riferimento; 103
 menu, a discesa, scambio; 105
 menu, a discesa, sottomenu; 101
 menu, a gruppi di immagini; 110
 menu, a gruppi di immagini, preparazione di diapositive; 113
 menu, a gruppi di immagini, sintassi; 110
 menu, a icone; 110
 menu, annullamento di un comando; 91
 menu, ausiliari; 96
 menu, caratteri di controllo; 91
 menu, caratteri per la terminazione delle voci di menu; 94
 menu, con il mouse di sistema; 96
 menu, di pulsanti; 96
 menu, di schermo; 114
 menu, di schermo, etichette di voci; 117
 menu, di tavoletta; 118
 menu, espressioni DIESEL; 94; 124; 323
 menu, file di; 83
 menu, guida sui; 118; 119
 menu, macro; 82
 menu, personalizzazione; 82
 menu, ripetizione dei comandi; 92
 menu, scrittura di testo nell'area del menu di schermo; 313
 menu, selezione singola di oggetti; 93
 menu, sezione Accelerators; 119
 menu, sintassi; 87
 menu, soppressione di visualizzazioni e messaggi di richiesta; 91
 menu, sottomenu di schermo; 117
 menu, stato di voci; 323
 menu, supporto per altre lingue; 90
 menu, uso di AutoLISP nei; 95
 menu, Vedere anche funzioni AutoLISP, menucmd; 323
 MENUCTL, variabile di sistema; 117
 MENUCHO, variabile di sistema; 91
 messaggi di errore, \; 216; 263; 283; 362; 404; 405
 messaggi di errore, definiti dall'utente; 298
 messaggi di errore, DIESEL; 133

messaggi di errore, formato, AutoLISP; 163
 messaggi, Vedere anche messaggi di errore; 263; 283
 messaggi, visualizzazione sulla riga di comando, AutoLISP; 170
 minimi, Vedere limiti; 307
 modalità snap ad oggetto; 296; 330; 342
 modalità Snap ad oggetto, da AutoLISP; 177
 modalità, riga di stato. Vedere riga di stato; 121
 modelli di tratteggio; 57; 58
 modelli di tratteggio, con linee tratteggiate; 59
 modelli di tratteggio, costruzione; 57; 58
 modelli di tratteggio, modifica; 202
 MODEMACRO, variabile di sistema; 121
 MODPAES, comando; 375

—N—

nidificate, definizioni di blocco; 294; 325
 nidificate, entità; 297
 nidificate, finestre di dialogo; 429; 430
 nil, variabili AutoLISP; 150
 nil, verifica dell'elemento, limitato solo al valore nil; 329
 nil, verifica dell'elemento, valore nil; 328
 nodi,; 402; 404
 nodi, allocazione manuale; 300
 nodi, liberi; 322; 402
 nodi, statistiche; 322
 nome entità; 191
 nomi applicazione, nei dati estesi; 209
 nomi di entità in AutoLISP; 147
 nomi layer, nei dati estesi; 209
 nomi, blocchi anonimi; 294
 nomi, di caselle DCL; 417
 nomi, file; 305; 306
 nomi, oggetti; 292; 293; 296; 314
 NPAES, comando; 379
 numeri interi lunghi, nei dati estesi; 209
 numeri interi, funzioni di conversione; 278; 282; 301; 317; 318
 numeri interi, limiti; 307
 numeri interi, nei dati estesi; 209
 numeri interi, specificati dall'utente; 307
 numeri interi, Vedere anche numeri; 307
 numeri interi, verifica; 329
 numeri negativi, impedire l'immissione di; 315
 numeri negativi, verifica; 324
 numeri reali, funzioni di conversione; 278; 290; 301
 numeri reali, nei dati estesi; 209
 numeri reali, specificati dall'utente; 309
 numeri reali, verifica; 329

—O—

OCS; 186; 354; 355; 537
 oggetti (entità), aggiornamento dell'immagine sullo schermo; 297
 oggetti (entità), codici di gruppo DXF; 478
 oggetti (entità), complessi; 294; 325
 oggetti (entità), creazione; 293
 oggetti (entità), dati di definizione; 291; 292; 293; 325

oggetti (entità), definizione entità; 295
 oggetti (entità), eliminazione; 291; 341
 oggetti (entità), gruppi di selezione; 341
 oggetti (entità), nidificati; 297
 oggetti (entità), nomi; 292; 293; 296; 314
 oggetti (entità), ridisegnare; 335
 oggetti (entità), ripristino; 291
 oggetti (entità), selezione; 296; 325
 oggetti complessi (entità); 294; 325
 oggetti di database; 531
 oggetti xrecord; 213
 oggetti xrecord, creazione; 213
 oggetti, riferimenti tra oggetti, codici di gruppo 1005; 533
 oggetti, riferimenti tra oggetti, gestori liberi; 533
 OLE, Automazione. Vedere Automazione; 136
 operatori di raggruppamento; 195
 operatori logici di raggruppamento; 195
 orientamento. Vedere funzioni AutoLISP, getorient; 308

—P—

PAGETB, tipo di simbolo; 356
 parentesi (), caratteri racchiusi tra; 358
 parentesi in AutoLISP; 144
 parentesi quadre {}, caratteri racchiusi tra; 358
 parole chiave, definizione per chiamate di funzioni di input utente; 315
 parole chiave, specificate dall'utente; 307; 308
 PAUSA, simbolo; 284
 PAUSE, variabile predefinita, AutoLISP; 151
 PFACEVMAX, variabile di sistema; 294
 PI, variabile predefinita, AutoLISP; 151
 piattaforme GUI, effetto sulle finestre di dialogo; 413
 PICKADD, variabile di sistema; 134; 138
 PICKAUTO, variabile di sistema; 134; 138
 PICKSET, tipo di simbolo; 356
 pila; 402
 PLINEA, comando; 236
 polilinee; 474
 polilinee curve; 206
 polilinee spline; 206
 polilinee, aggiornamento dell'immagine s; 297
 polilinee, creazione; 294
 polilinee, dati di definizione; 325
 polilinee, funzione entget e; 291; 292
 polilinee, funzione entmod e; 295
 polilinee, modifica; 295
 polilinee, selezione; 325
 POLIMESH, comando; 294
 PostScript, compilazione dei font; 61
 PostScript, definizione dei modelli di riempimento; 78
 PostScript, esportazione di file EPS; 76
 PostScript, file, modifiche al colore; 80
 PostScript, file, modifiche al layer; 79
 PostScript, file, modifiche al tipo di linea; 80
 PostScript, font, compilazione; 61
 PostScript, importazione di file Encapsulated PostScript (EPS); 80
 PostScript, materiale di riferimento; 82
 PostScript, sezione introduttiva; 79
 predefinite, variabili, AutoLISP; 151

prefissi dei comandi, _ (trattino di sottolineatura); 167
 prefissi per comandi, - (simbolo meno); 43
 prefissi per comandi, . (punto); 263; 283
 prefissi per comandi, .(punto); 167
 prefissi per comandi, _ (trattino di sottolineatura); 263; 283
 prefisso costituito da un punto nei comandi; 167
 programmi di indentazione del codice LISP; 150
 PSIN, comando; 80; 381
 PSMOTIVO, comando; 381
 PSOUT, comando; 76
 PSPROLOG, variabile di sistema; 79
 PSQUALITY, variabile di sistema; 80
 PSTRASCINA, comando; 380
 pulsanti; 431; 444
 pulsanti immagine; 433
 pulsanti, righe e colonne di scelta; 433
 punti, 3D; 330
 punti, coordinata Y; 187; 281; 318
 punti, coordinata Z; 187; 281; 318
 punti, definizione delle variabili; 319
 punti, specificati dall'utente; 309; 315
 punti, trasformazione di sistemi di coordinate; 325; 327
 punti, Vedere anche funzioni AutoLISP, distance; 309
 punti, Vedere anche funzioni AutoLISP, vectors; 309

—Q—

quote, modifica; 202

—R—

Raggio, comando secondario; 284
 REAL, tipo di simbolo; 356
 reattori costanti; 534
 RENDER, comando; 381; 382
 RENDERUPDATE, comando; 385
 REPLAY, comando; 385
 resto della funzione list; 323
 rettangoli, angolo specificato; 304
 rettangoli, nei gruppi di immagini della finestra di dialogo; 300
 ricerca, dizionari; 289
 ricerca, liste di associazione; 277
 richiamo dei comandi DOS; 41
 ricorsività, funzione load; 320
 ricorsività, uso della memoria; 402
 RIDIS, comando; 172
 riempimento a motivi, Vedere modelli di tratteggio; 57; 58
 riga di comando, stampa, espressione su; 331
 riga di comando, visualizzazione di messaggi, AutoLISP; 170
 riga di stato, configurazione; 121
 riga di stato, scrittura di testo; 313
 RIGEN, comando; 206; 297
 ripristino, oggetti; 291
 ripulitura dello spazio; 302; 402; 404
 RPREF, comando; 394
 RUOTA3D, comando; 393

—S—

SALVAIMM, comando; 396
 sample, directory; 138
 scaricamento, applicazioni ARX; 276
 scaricamento, Vedere anche caricamento; 356; 357
 SCENA, comando; 397
 schermo, menu di Vedere menu, di schermo; 114
 schermo, scrittura, testo; 313
 schermo, visualizzazione, schermo grafico; 310
 SCHGRAF, comando; 310
 SCHIZZO, comando; 263; 283
 SCM (Sistema di Coordinate di Movimento); 199; 325
 SCREENBOXES, variabile di sistema; 117
 script, Annulla, funzione; 134
 SCRIPT, comando; 134; 263; 283
 scrittura, AppData, sezione (acad.cfg); 337; 338
 scrittura, espressioni in un file; 331
 SELEZ, comando; 284
 Seleziona file Lisp, finestra di dialogo; 305; 306
 selezione di oggetti; 296; 325
 sezione AppData di acad.cfg; 169
 sezione introduttiva, di PostScript; 79
 SFONDO, comando; 367
 SHOWMAT, comando; 401
 simboli in AutoLISP; 148
 simboli locali nelle funzioni, AutoLISP; 161; 162
 simboli, elenco; 279
 simboli, gestione, AutoLISP; 159
 simboli, lettura; 333; 334
 simboli, locali, AutoLISP; 161; 162
 simboli, PAUSA; 284
 simboli, tipi; 356
 simboli, verifica; 278; 281; 328
 Sistema di Coordinate di Movimento (SCM); 199; 325
 Sistema di Coordinate Oggetto. Vedere OCS.; 354; 355
 sistema, variabili in AutoLISP; 169
 sistemi di coordinate, trasformazione; 325; 327
 SNAPANG, variabile di sistema; 339
 SOLPROFILO, comando; 401
 SORTENTS, variabile di sistema; 177
 sottomenu di schermo; 117
 spazi vuoti, e simboli locali, AutoLISP; 161; 162
 spazi vuoti, nel codice AutoLISP; 150
 spazio per i nodi; 402
 SPECCHIO3D, comando; 380
 spline, polilinee; 206
 SPLINESEGS, variabile di sistema; 206
 stato, voci di menu; 323
 STILE, comando; 67; 73
 STR, tipo di simbolo; 356
 stringhe di controllo, nei dati estesi; 209
 stringhe, barra rovesciata (); 284; 309
 stringhe, conversione dei caratteri minuscoli/maiuscoli; 349
 stringhe, funzioni di conversione; 349
 stringhe, length; 309
 stringhe, lettura; 333; 334
 stringhe, negli elenchi delle finestre di dialogo; 272
 stringhe, nei dati estesi; 209
 stringhe, ordinare liste di; 271
 stringhe, read; 334
 stringhe, specificate dall'utente; 309

SUBR, tipo di simbolo; 356
 supporto per altre lingue, AutoLISP; 167
 SYM, tipo di simbolo; 356

—T—

T, variabile predefinita, AutoLISP; 151
 tabella di simboli APPID; 207
 tabella di simboli BLOCKS; 207
 tabella di simboli DIMSTYLE; 207
 tabella di simboli LAYER; 207
 tabella di simboli LINETYPE; 207
 tabella di simboli STYLE; 207
 tabella di simboli VPORT; 207; 214
 tabelle di simboli; 191; 214
 tabelle di simboli, e la funzione entmake; 207
 tabelle di simboli, e la funzione entmod; 207
 tabulazioni nei file AutoLISP; 150
 tangente. Vedere funzioni AutoLISP, arcotangente; 277
 tasti di scelta; 119; 430
 tastiera, lettura, carattere da; 334
 tastiera, lettura, stringhe di; 334
 tastiera, scelte rapide, nelle finestre di dialogo; 430
 tasto di accettazione; 430; 458
 tasto di annullamento; 430
 TAVOLET, comando; 118
 tavoletta, menu di; 118
 test. Vedere verifica; 278
 TESTO, comando; 151
 testo, font; 67
 testo, nelle finestre di dialogo; 434
 testo, scrittura, nella riga di stato; 313
 testo, scrittura, nelle aree dei menu di schermo; 313
 TESTODIN, comando; 151; 263; 283
 TEXTEVAL, variabile di sistema; 151
 tipi di dati, AutoLISP; 145
 tipi di dati, variabili AutoLISP; 150
 tipi di linea complessi; 53
 tipi di linea semplici; 53
 tipi di linea, file di definizione; 53
 tipi di linea, inclusione di forme; 55
 tipi di linea, semplici; 53
 tipo di entità finestra, creazione; 294
 tipo di entità finestra, modifica; 295
 TLINEA, comando; 53
 trasformazione del sistema di coordinate; 325; 327
 trasformazione di punti; 187
 trattino di sottolineatura (_), comandi nelle versioni in altre lingue; 263; 283
 trattino di sottolineatura (_), comandi per una versione in un'altra lingua; 167
 troncatura; 301
 trovare. Vedere ricerca; 301

—U—

UCS; 186; 199; 331; 354; 355
 UNDOCTL, variabile di sistema; 163
 UNDOMARKS, variabile di sistema; 163
 unità di altezza carattere; 443

unità di misura, AutoLISP; 184
 unità di misura, conversione; 286
 UNITMODE, variabile di sistema; 181
 uscita tranquilla, AutoLISP; 154
 User Coordinate System, Sistema di Coordinate Utente. Vedere UCS; 199
 uso delle lettere maiuscole nel testo delle finestre di dialogo; 428

—V—

valutazione di espressioni; 299; 302; 320
 variabili d'ambiente, in AutoLISP; 169
 variabili d'ambiente, richiamo; 305
 variabili di sistema, ANGBASE; 173; 181; 339
 variabili di sistema, ANGDIR; 173
 variabili di sistema, APERTURE; 177
 variabili di sistema, AUNITS; 181
 variabili di sistema, AUPREC; 181
 variabili di sistema, CMDACTIVE; 216
 variabili di sistema, CVPORT; 339
 variabili di sistema, DIMZIN; 181
 variabili di sistema, ERRNO; 406
 variabili di sistema, in AutoLISP; 169
 variabili di sistema, LUNITS; 181
 variabili di sistema, LUPREC; 181
 variabili di sistema, MENUCTL; 117
 variabili di sistema, MENU ECHO; 91
 variabili di sistema, MODEMACRO; 121
 variabili di sistema, PFACEVMAX; 294
 variabili di sistema, PICKADD; 134; 138
 variabili di sistema, PICKAUTO; 134; 138
 variabili di sistema, PSPROLOG; 79
 variabili di sistema, PSQUALITY; 80
 variabili di sistema, recupero; 310
 variabili di sistema, SCREENBOXES; 117
 variabili di sistema, SNAPANG; 339
 variabili di sistema, SORTENTS; 177
 variabili di sistema, SPLINESEGS; 206
 variabili di sistema, TEXTEVAL; 151
 variabili di sistema, UNDOCTL; 163
 variabili di sistema, UNDOMARKS; 163
 variabili di sistema, UNITMODE; 181
 variabili di sistema, Vedere anche variabili d'ambiente; 310
 variabili in AutoLISP; 148
 variabili predefinite, AutoLISP; 151
 variabili, AutoLISP; 150
 verifica semantica per i file DCL; 419
 verifica, atomi; 278
 verifica, liste; 319
 verifica, nomi delle tabelle di simboli; 340
 verifica, numeri interi; 329
 verifica, numeri reali; 329
 verifica, simboli; 278; 281; 328
 verifiche dei filtri, raggruppamento logico; 195
 verifiche relazionali, nei gruppi di selezione; 195
 versioni in altre lingue, nomi dei comandi; 263; 283
 vertici per elemento faccia, massimo; 294
 vettori di base; 187
 vettori multipli, disegno; 313
 vettori, disegno sullo schermo grafico; 311; 313

virgolette, nelle stringhe AutoLISP; 154; 181

—W—

WCS; 186; 199; 325; 354; 355
WCS, direzione, nei dati estesi; 209
WCS, posizione, nei dati estesi; 209
WCS, spostamento, nei dati estesi; 209
World Coordinate System, Sistema di Coordinate
Globali. Vedere WCS; 199

—Y—

Y coordinata, ottenere; 281; 318

—Z—

Z coordinata, ottenere; 281; 318
zero, impedire l'immissione di; 315