AutoLISP 函数列表

运算符

+（加）

返回所有数的和

(+ [number number] ...)

如果在调用本函数时仅提供了一个 number 参数，本函数会返回它和 0 相加的结果，即返回该 number。如不提供参数，则返回 0。

(+ 1 2)　　返回 3

(+ 1 2 3 4.5)　　返回 10.5

(+ 1 2 3 4.0)　　返回 10.0

-（减）

将第一个数减去其他数的和并返回差值

(- [number number] ...)

调用本函数时，如果提供的 number 参数多于两个，则本函数返回从第一个数减去第二个数到最后一个数的和的差。如果仅提供一个 number 参数，本函数返回 0 减去它的差，即返回 -number。如不提供参数，则返回 0。

(- 50 40) 返回 10

(- 50 40.0)　　返回 10.0

(- 50 40.0 2.5) 返回 7.5

(- 8) 返回 -8

*（乘）

返回所有数的乘积

(* [number number] ...)

如果调用本函数时仅提供一个 number 参数，本函数返回它与 1 相乘的结果，即返回该 number。如不提供参数，则返回 0。

(* 2 3)　返回 6

(* 2 3.0) 返回 6.0

(* 2 3 4.0)　　返回 24.0

(* 3 -4.5) 返回 -13.5

(* 3)返回 3

/（除）

将第一个数除以其他数的乘积并返回商

(/ [number number] ...)

调用本函数时，如果提供的 number 参数多于两个，本函数用第一个数作为被除数，除以第二个数到最后一个数的乘积，返回最后得到的商。如果仅提供一个 number 参数，本函数返回它除以 1 的结果，即返回该 number。如不提供参数，则返回 0。

(/ 100 2) 返回 50

(/ 100 2.0)　　返回 50.0

(/ 100 20.0 2) 返回 2.5

(/ 100 20 2)　返回 2

(/ 4) 返回 4

=（等于）

如果所有参数值相等则返回 T ； 否则返回 nil

(= numstr [numstr] ...)

每个 numstr 参数既可以是数，也可以是字符串。

(= 4 4.0) 返回 T

(= 20 388)　　返回 nil

(= 2.4 2.4 2.4) 返回 T

(= 499 499 500)　　返回 nil

(= "me" "me") 返回 T

(= "me" "you")　　返回 nil

请参见　相关函数 eq 和 equal

/=（不等于）

如果各参数值不相等则返回 T ；如果所有参数在数值上相等则返回 nil

(/= numstr [numstr] ...)

每个 numstr 参数既可以是数，也可以是字符串。

(/= 10 20)　　返回 T

(/= "you" "you")　返回 nil

(/= 5.43 5.44) 返回 T

(/= 10 20)　　返回 T

<（小于）

如果每个参数值都小于它右边的参数则返回 T；否则返回 nil

(< numstr [numstr] ...)

每个 numstr 参数既可以是数，也可以是字符串。

(< 10 20)　　返回 T

(< "b" "c")　　返回 T

(< 357 33.2)　　返回 nil

(< 2 3 88)　　返回 T

(< 2 3 4 4)　　返回 nil

<=（小于或等于）

如果每个参数值都小于或等于它右边的参数则返回 T；否则返回 nil

(<= numstr [numstr] ...)

每个 numstr 参数既可以是数，也可以是字符串。

(<= 10 20)　　返回 T

(<= "b" "b")　返回 T

(<= 357 33.2) 返回 nil

(<= 2 9 9)　　返回 T

(<= 2 9 4 5)　返回 nil

>（大于）

如果每个参数值都大于它右边的参数则返回 T；否则返回 nil

(> numstr [numstr] ...)

每个 numstr 参数既可以是数，也可以是字符串。

(> 120 17)　　返回 T

(> "c" "b")　　返回 T

(> 3.5 1792)　　返回 nil

(> 77 4 2)　　返回 T

(> 77 4 4)　　返回 nil

>=（大于或等于）

如果每个参数值都大于或等于它右边的参数则返回 T；否则返回 nil

(>= numstr [numstr] ...)

每个 numstr 参数既可以是数，也可以是字符串。

(>= 120 17)　　返回 T

(>= "c" "c")　　返回 T

(>= 3.5 1792)　　返回 nil

(>= 77 4 4)　　返回 T

(>= 77 4 9)　　返回 nil

~（按位非）

返回参数的按位非（即 1 的补码）

(~ int)

(~ 3)　　返回 -4

(~ 100)　　返回 -101

(~ -4)　　返回 3

1+（增 1）

返回参数增 1 后的结果

(1+ number)

(1+ 5)　　返回 6

(1+ -17.5)　　返回 -16.5

1-（减 1）

返回参数减 1 后的结果

(1- number)

(1- 5)　　返回 4

(1- -17.5)　　返回 -18.5

A

abs

返回参数的绝对值

(abs number)

(abs 100)　　返回 100

(abs -100)　　返回 100

(abs -99.25)　　返回 99.25

acad_colordlg

显示标准的 AutoCAD 颜色选择对话框

(acad_colordlg colornum [flag])

函数语法格式中的 colornum 参数是一个整数，其取值范围是 0-256 （包括 0 和 256），它用于指定 AutoCAD 作为初始缺省值显示的颜色代码。 colornum 为 0 时代表 BYBLOCK（与所在块的缺省颜色一致）；colornum 为 256 时代表 BYLAYER（与所在图层的缺省颜色一致）。将可选参数 flag 设为 nil 时，禁用 BYLAYER 和 BYBLOCK 按钮；省略 flag 参数或将其设为非 nil 值，则可启用 BYLAYER 和 BYBLOCK 按钮。

acad_colordlg 函数返回用户所选择的颜色代码，如果用户取消该对话框则返回 nil。

下列代码以对话框形式提示用户选择一种颜色，并指定绿色为缺省颜色：

(acad_colordlg 3)

外部定义函数　acadapp ARX 应用程序

acad_helpdlg

启动帮助工具（已废弃）

(acad_helpdlg helpfile topic)

该外部定义函数已由内置的 help 函数所取代，提供该函数是为了和早期版本的 AutoCAD 兼容。关于本函数的详细介绍，请参见 help 函数。

外部定义函数　acadr14.lsp AutoLISP 函数

acad_strlsort

以 ASCII 码字母顺序对字符串表进行排序

(acad _strlsort　list)

list 参数是要进行排序的字符串表， acad_strlsort 函数返回排序后的相同字符串表。如果 list 参数是一个无效的表或者没有足够的内存来进行排序， acad_strlsort 函数返回 nil。

下列代码将一年中十二个月的月名的缩写字符串表按字母顺序进行排序：

```
(setq mos '("Jan" "Feb" "Mar" "Apr" "May" "Jun"
            "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"))
(acad_strlsort mos)
```

它返回如下字符串表：

```
("Apr" "Aug" "Dec" "Feb" "Jan" "Jul"
 "Jun" "Mar" "May" "Nov" "Oct" "Sep")
```

外部定义函数　acadapp ARX 应用程序

action_tile

为某一对话框控件指定一个动作表达式，用户在对话框中选中这个控件时，就会对该动作表达式进行求值

(action_tile key action-expression)

参数 key 和 action-expression 都是字符串。 key 参数是触发动作的控件关键字（该控件关键字由控件的 key 属性指定），它是区分大小写的。当该控件被选中时，就会对动作表达式 action-expression 进行求值。

由 action_tile 函数指定的动作将取代对话框的缺省动作（由 new_dialog 函数指定）或该控件的 action 属性（在指定了这些属性的情况下）。该动作表达式可以通过变量 $value 引用控件的当前值（即它的 value 属性），通过变量 $key 引用控件的关键字，通过变量 $data 引用控件的特定应用数据（由函数 client_data_tile 设置），通过变量 $reason 引用控件的回调原因，如果该控件是图像按钮的话，还可以通过变量 $x 和 $y 引用控件的图像坐标。

注意　在 action_tile 函数中不能调用 AutoLISP 的 (Command) 函数。

请参见　缺省值与 DCL 动作

add_list

在当前激活的对话框的列表框中增加一个字符串或修改其中的一个字符串

(add_list string)

在使用 add_list 函数之前，必须先调用 start_list 函数打开并初始化该列表框。根据 start_list 函数中指定的 operation 参数的不同，参数 string 被加入到当前列表框中或替换列表框的当前项。

假设当前激活的 DCL 文件中有一个名为 longlist 的列表框或弹出式列表框，下列代码将初始化它所用的表，并将该表加入到该列表框或弹出式列表框中。

```
(setq llist '("first line" "second line" "third line"))
(start_list "longlist")
(mapcar 'add_list llist)
(end_list)
```

定义该表之后，下列代码将该表第二行的字符串替换为 "2nd line"。

```
(start_list "longlist" 1 0)
(add_list "2nd line")
(end_list)
```

请参见　start_list 和 end_list 函数

ads

返回当前已加载的 ADS 应用程序名列表

(ads)

每一个已加载的 ADS 应用程序和它的路径都用双引号引起来作为表中的一项。

(ads)　　可能返回 ("files/progs/PROG1" "PROG2")

请参见　xload 和 xunload 函数

alert

显示一个警告框，其中显示一条出错或警告信息，该信息是以字符串形式由 string 参数提供的

(alert string)

警告框是带有单个 OK 按钮的对话框。

(alert "That function is not available.")

通过在 string 参数中使用换行符可以在一个警告框中显示多行字符。

(alert "That function\nis not available.")

注意　警告框中所能显示的警告信息的行数以及每行的长度依赖于所使用的平台、设备和窗口，AutoCAD 将自动切断超出其范围的任何字符。

alloc

将段长设置成指定的节点数

(alloc int)

本函数返回以前的段长。

请参见　手动分配

and

返回表达式的逻辑与 (AND) 运算结果

(and expr ...)

如果任何一个表达式的求值结果为 nil，本函数就停止进一步的求值并返回 nil，否则返回 T。

例如，给定如下初值：

```
(setq a 103 b nil c "string")
```

那么，

(and 1.4 a c)　返回 T
(and 1.4 a b c) 返回 nil

angle

以弧度为单位返回两点之间连线与 X 轴之间的夹角

(angle pt1 pt2)

返回的角度是从当前构造平面的 X 轴算起，以弧度为单位按逆时针方向计算的。如果指定的是三维点，则先将它们投影到当前构造平面上，再计算投影线与 X 轴的角度。

(angle '(1.0 1.0) '(1.0 4.0))　　返回 1.5708
(angle '(5.0 1.33) '(2.4 1.33))　　返回 3.14159

请参见　角度转换

angtof

按 mode 参数指定的格式解释字符串 string，将其转换成实数(浮点数)形式的弧度值

(angtof string [mode])

string 参数以 mode 参数指定的格式描述角度（用于指定角度字符串格式单位的 mode 参数的取值和含义如下表所示）。mode 参数的取值应与 AutoCAD 的系统变量 AUNITS 的允许取值对应，如果省略该参数，本函数则使用 AUNITS 的当前值。

角度单位值

Mode 值	字符串格式
0	（十进制）度
1	度/分/秒
2	百分度
3	弧度
4	勘测单位

string 必须是能够由 angtof 函数根据指定的 mode 参数进行正确分析的一个字符串。参数 string 的格式，既可以与 angtos 函数返回结果的格式相同，也可以是 AutoCAD 允许从键盘输入的角度格式。如果 string 带有明确的角度格式标记，如：angtof 函数将仅根据格式标记的类型解释 string。

angtof 和 angtos 函数是一对互补函数。如果将 angtos 函数创建的一个字符串作为参数传给函数 angtof，那么，angtof 函数保证会返回一个有效值，反之亦然（假定与 mode 参数相匹配）。

如果 angtof 函数调用成功，它将返回一个实数形式的弧度值，否则返回 nil。

angtos

将一个以弧度为单位的角度值转换成字符串

(angtos angle [mode [precision]])

angtos 函数以 angle 作参数（它是角度的弧度值），根据参数 mode、precision 和 AutoCAD 的系统变量 UNITMODE 以及尺寸标注变量 DIMZIN 的情况，将参数 angle 转换成字符串形式返回。参数 mode 和 precision 都是整数，分别指定角度单位的模式和精度，参数 mode 的值和含义如下表所示。

角度单位值

Mode 值	字符串格式
0	（十进制）度
1	度/分/秒
2	百分度
3	弧度
4	勘测单位

precision 参数是用于指定精度所需的小数位数的一个整数。参数 mode 和 precision 与 AutoCAD 的系统变量 AUNITS 和 AUPREC 是对应的。如果省略这两个参数，angtos 函数就会分别使用 AUNITS 和 AUPREC 这两个系统变量的当前值。

angtos 函数接受负的 angle 参数，但它总是在执行指定转换之前将其还原成 0 至 2p 弧度之间的正值。

(angtos 0.785398 0 4)　返回 "45.0000"
(angtos -0.785398 0 4)　返回 "315.0000"

如果选择勘测单位（mode 参数值为 4），系统变量 UNITMODE 会影响函数返回的结果字符串。如果 UNITMODE 为 0，结果字符串中会包含空格（例如 "N 45d E"）；相反，如果 UNITMODE 为 1，结果字符串中就不会包含空格（例如 "N45dE"）。

注意 使用 angtos 函数来显示任意角度值（与 ANGBASE 不相关的角度）的应用程序，应该检查和考虑系统变量 ANGBASE 的值。

请参见 字符串转换

append

将任意多个表组合成一个表

(append list ...)

(append '(a b) '(c d)) 返回 (A B C D)

(append '((a)(b)) '((c)(d))) 返回 ((A)(B)(C)(D))

apply

将参数表传给指定的函数

(apply 'function list)

apply 函数可以处理内置式 (subrs) 函数和用户（用 defun 或 lambda）定义的函数。

(apply '+ '(1 2 3)) 返回 6

(apply 'strcat '("a" "b" "c")) 返回 "abc"

arx

返回当前已加载的 ARX 应用程序名列表

(arx)

每个已加载的 ARX 应用程序和它的路径都用双引号引起来作为表中的一项。

(arx) 可能返回 ("files/progs/PROG1" "PROG2")

请参见 arxload 和 arxunload 函数

arxload

加载 ARX 应用程序

(arxload application [onfailure])

application 参数既可以是用双引号来的一个字符串，也可以是一个包含可执行文件名的变量。在加载 ARX 应用程序时，会对其有效性进行验证。

如果 arxload 函数操作失败，通常它会给出一个 AutoLISP 错误。但是，如果提供了 onfailure 参数， arxload 将返回该参数的值来响应错误而不给出错误信息。

如果加载应用程序成功，则返回该应用程序名。

(arxload "/myapps/appx") 如果成功，返回 "/myapps/appx"

如果试图加载一个已经加载的 ARX 应用程序，arxload 将发出如下一条信息并返回该应用程序名：

已加载应用程序“application”

在使用 arxload 加载一个应用程序之前，可以调用 arx 函数检查已加载的 ARX 应用程序。

arxunload

卸载 ARX 应用程序

(arxunload application [onfailure])

如果应用程序被成功卸载，本函数将返回应用程序名，否则，给出一个错误信息。

application 参数既可以是用双引号引起来的一个字符串，也可以是一个包含已用 arxload 函数加载的可执行文件名的变量。该应用程序名必须和调用 arxload 函数时的应用程序名完全一致。如果在调用 arxload 函数时给应用程序名指定了路径（即目录名），在调用 arxunload 函数卸载它时可以省略路径。

例如，下列代码将卸载先前用 arxload 函数加载的应用程序 appx：

(arxunload "appx") 如果成功，返回 "appx"

如果 arxunload 函数操作失败，通常它会给出一个 AutoLISP 错误。但是，如果提供了 onfailure 参数， arxunload 将返回该参数的值来响应错误而不给出错误信息。在这点上 arxunload 函数和 arxload 函数类似。

ascii

将字符串中的第一个字符转换成其 ASCII 码（一个整数）后返回

(ascii string)

(ascii "A") 返回 65
(ascii "a") 返回 97
(ascii "BIG") 返回 66
该函数的功能与 BASIC 语言中的 ASC 函数类似。

assoc

从关联表中搜索一个元素，如果找到则返回该关联表条目
(assoc item alist)
该函数以参数 item 为关键元素对关联表 alist 进行搜索并返回找到的关联表条目，如果找不到则返回 nil。
例如，如下所示，设置一个关联表：
(setq al '((name box) (width 3) (size 4.7263) (depth 5)))
然后，
(assoc 'size al) 返回 (SIZE 4.7263)
(assoc 'weight al) 返回 nil
关联表经常用于存储可以通过关键字进行访问的数据，利用 subst 函数可以方便地替换关联表中与关键字相关的值。

atan

返回一个数的反正切值（以弧度为单位）
(atan num1 [num2])
调用该函数时如果仅提供一个参数 num1，本函数返回数 num1 的以弧度为单位的反正切值。如果提供了两个参数 num1 和 num2，本函数返回 num1/
num2 的以弧度为单位的反正切值。如果 num2 为 0，该函数返回正的或负的 1.570796 弧度（+90 度或 -90 度），其正负取决于 num1 的正负。该函数返回的角度的范围是 -p/2 至 +p/2 （弧度）。
(atan 0.5) 返回 0.463648
(atan 1.0) 返回 0.785398
(atan -1.0) 返回 -0.785398
(atan 2.0 3.0) 返回 0.588003
(atan 2.0 -3.0) 返回 2.55359
(atan 1.0 0.0) 返回 1.5708
注意 angtos 函数可以将 atan 函数所返回的弧度值转换成一个字符串值。
(angtos (atan -1.0) 0 4) 返回 "315.0000"
(angtos (atan 2.0 3.0) 0 4) 返回 "33.6901"
(angtos (atan 2.0 -3.0) 0 4) 返回 "146.3099"
(angtos (atan 1.0 0.0) 0 4) 返回 "90.0000"

atof

将一个字符串转换成实数
(atof string)
(atof "97.1") 返回 97.1
(atof "3") 返回 3.0
(atof "3.9") 返回 3.9

atoi

将一个字符串转换成整数
(atoi string)
(atoi "97") 返回 97
(atoi "3") 返回 3
(atoi "3.9") 返回 3

atom

验证一个项是否是原子
(atom item)
如果参数 item 是一个表则返回 nil；否则返回 T。任何非表的参数均被认为是原子。
假设有如下初值：

(setq a '(x y z))
(setq b 'a)
那么
(atom 'a)　　返回 T
(atom a)　　返回 nil
(atom 'b)　　返回 T
(atom b)　　返回 T
(atom '(a b c))　　返回 nil
某些版本的 LISP 对 atom（原子）的解释有些不同，所以在使用移植的代码时应加以注意。

atoms-family

返回由当前已定义的符号组成的一个表
(atoms-family format [symlist])
format 参数是可取 0 或 1 的整数。如果 format 参数是 0，则 atoms-family 函数以表的形式返回符号名；如果 format 参数是 1，则 atoms-family 函数以字符串表的形式返回符号名。参数 symlist 用于指定符号名的字符串表，如果调用函数时提供了该参数，本函数就会在系统中对指定的符号名表进行搜索。atoms-family 函数返回一个由 format 参数所指定的类型（符号或字符串）的表，该表中包含了已经定义的那些符号名。对于没有定义的那些符号名，在返回的表中的对应位置上以 nil 表示。
(atoms-family 0)　　返回由当前已定义的符号组成的一个表
下列代码验证符号 CAR、CDR 和 XYZ 是否已经被定义，并以字符串表的格式返回它们：
(atoms-family 1
'("CAR" "CDR" "XYZ")) 返回 ("CAR" "CDR" nil)
这个返回的字符串表表明符号 XYZ 没有被定义。

autoarxload

预定义可自动加载某相关 ARX 应用程序的命令名
(autoarxload filename cmdlist)
在执行 cmdlist 参数定义的某一个命令之前，将自动加载 filename 参数指定的一个 .arx 文件。cmdlist 参数必须是一个字符串表。autoarxload
函数返回 nil。
下列代码定义了 C:APP1、C:APP2 和 C:APP3 函数将自动加载 bounsapp.arx 文件。在第一次运行 APP1、APP2 或 APP3 命令时，先自动加载相应 ARX
应用程序，然后继续执行该命令。
(autoarxload "BONUSAPP" '("APP1" "APP2" "APP3"))
警告　cmdlst 参数所含的命令必须在 filename 参数所指的文件中定义。
外部定义函数　acadr14.lsp AutoLISP 函数

autoload

预定义可自动加载某相关 AutoLISP 应用程序的命令名
(autoload filename cmdlist)
在执行 cmdlist 参数定义的某一个命令之前，将自动加载 filename 参数指定的一个 .lsp 文件。cmdlist 参数必须是一个字符串表。autoload
函数返回 nil。
下列代码定义了 C:APP1、C:APP2 和 C:APP3 函数将自动加载 bounsapp.lsp 文件。在第一次运行 APP1、APP2 或 APP3 命令时，先自动加载相应 AutoLISP 应用程序，然后继续执行该命令。
(autoload "BONUSAPP" '("APP1" "APP2" "APP3"))
警告　cmdlst 参数所含的命令必须在 filename 参数所指的文件中定义。
外部定义函数　acadr14.lsp AutoLISP 函数

autoxload

预定义可自动加载某相关 ADS 应用程序的命令名
(autoxload filename cmdlist)
在执行 cmdlist 参数定义的某一个命令之前，将自动加载 filename 参数指定的一个 ADS 应用程序。cmdlist 参数必须是一个字符串表。
autoxload 函数返回 nil。

下列代码定义了 C:APP1、C:APP2 和 C:APP3 函数将自动加载名为 bounsapp 的 ADS 应用程序。在第一次运行 APP1、APP2 或 APP3 命令时，先自动加载相应 ADS 应用程序，然后继续执行该命令。

(autoxload "BONUSAPP" '("APP1" "APP2" "APP3"))

警告 cmdlst 参数所含的命令必须在 filename 参数所指的文件中定义。

外部定义函数 acadr14.lsp AutoLISP 函数

B

Boole

用作一个通用的按位逻辑运算函数

(Boole func int1 int2 ...)

func 参数是 0 和 15 之间的整数，代表 16 种可能的双变量布尔函数。随后的整数参数则根据这个函数和如下的真值表进行按位方式（即逻辑方式）组合。

布尔真值表

Int1	Int2	Func 位值
0	0	8
0	1	4
1	0	2
1	1	1

参数 int1 的每一位与参数 int2 的对应位相匹配，指定真值表中的某一行。运算结果的该位可能是 0 或 1，这取决于真值表中这一行所对应的 func

位的设置情况。

如果相应 func 位和真值表一致，则结果位为 1；否则结果位为 0。func 参数的某些取值等效于标准布尔运算 AND（与）、OR（或）、XOR（异或）和 NOT（非）。

Boole 函数的位值

Func	运算	结果位为 1 的条件
1	AND（与）	两个输入位均为 1
6	XOR（异或）	两个输入位仅有一个为 1
7	OR（或）	有一个或两个输入位为 1
8	NOR（非）	两个输入位均为 0（1 的补）

下列代码进行数 12 和 5 的逻辑与 (AND) 运算：

(Boole 1 12 5) 返回 4

类似地，下列代码进行数 6 和 5 的逻辑异或 (XOR) 运算：

(Boole 6 6 5) 返回 3

也可以使用 func 的其他值来执行没有标准名的其他布尔运算。例如，如果 func 是 4，那么只有在 int2 的位为 1，而 int1 的对应位为 0 时，其结果位才为 1，因此

(Boole 4 3 14) 返回 12

boundp

检验符号是否被设置为某个值

(boundp sym)

如果参数 sym 已被设置为非 nil 值，该函数返回 T。如果没有设置 sym 或它被设置为 nil，本函数返回 nil。如果 sym 是一个未定义的符号，则自动创建它，并将其设为 nil。

例如，初始化变量：

(setq a 2 b nil)

则，

(boundp 'a) 返回 T

(boundp 'b) 返回 nil

还可以用 atoms-family 函数来确定一个符号的是否存在，但它不自动创建该符号。

C

car 和 cdr

car 函数返回表中的第一个元素；cdr 函数则返回去掉了第一个元素的表

(car list) 和 (cdr list)

如果 list 为空，car 返回 nil。

(car '(a b c))　　返回 A
(car '((a b) c))　　返回 (A B)
(car '())　　返回 nil
如果 list 为空，cdr 返回 nil。
(cdr '(a b c))　　返回 (B C)
(cdr '((a b) c))　　返回 (C)
(cdr '())　　返回 nil
当 list 参数是点对表时（请参见 cons），cdr 返回点对表的第二个元素，而不是以表的形式返回它。
(cdr '(a . b))　　返回 B
(cdr '(1 . "Text"))　　返回 "Text"
AutoLISP 支持的 car 和 cdr 的连续调用可深达四层，以下列出的均为有效的函数名：

caaaar	cadaar	cdaaar	cddaar
caaadr	cadadr	cdaadr	cddadr
caaar	cadar	cdaar	cddar
caadar	caddar	cdadar	cdddar
caaddr	cadddr	cdaddr	cddddr
caadr	caddr	cdadr	cdddr
caar	cadr	cdar	cddr

上述连续函数与嵌套调用 car 和 cdr 函数是等效的。其中每一个 a 表示对 cdr 函数的一次调用；每一个 d 则表示对 cdr 函数的一次调用。
(caar x)　　等效于 (car (car x))
(cdar x)　　等效于 (cdr (car x))
(cadar x)　　等效于 (car (cdr (car x)))
(cadr x)　　等效于 (car (cdr x))
(cddr x)　　等效于 (cdr (cdr x))
(caddr x)　　等效于　(car (cdr (cdr x)))
在 AutoLISP 中，cadr 函数经常用于获得二维或三维点的 Y 坐标（两个或三个实数组成的表中的第二个元素）。类似地，caddr 函数可用于获得三维点的 Z 坐标。假设有如下初始值：
(setq pt2 '(5.25 1.0))　　一个二维点
(setq pt3 '(5.25 1.0 3.0))　　一个三维点
则，
(car pt2)　　返回 5.25
(cadr pt2)　　返回 1.0
(caddr pt2)　　返回 nil
(car pt3)　　返回 5.25
(cadr pt3)　　返回 1.0
(caddr pt3)　　返回 3.0

chr

将代表字符的 ASCII 码的整数转换成相应的单一字符的字符串
(chr integer)
(chr 65)　　返回 "A"
(chr 66)　　返回 "B"
(chr 97)　　返回 "a"
该函数的功能与 BASIC 语言中的 chr$ 函数相似。

client_data_tile

将特定应用数据与一个对话框控件联系起来
(client_data_tile key clientdata)
key 参数是用于指定控件的字符串，它是区分大小写的。数据是由 clientdata 参数指定的字符串，动作表达式或回调函数可以通过 $data 变量引用它。

close

关闭一个已打开的文件

(close file-desc)

file-desc 参数是在 open 函数打开文件时获得的一个文件描述符。用 close 函数关闭文件后，该文件描述符并没有改变，但它已不再有效。在未关闭文件之前，加入已打开文件中的数据并没有真正写入文件。如果 file-desc 参数有效，close 函数返回 nil；否则它返回一个错误信息。

例如，下列代码可获得文件 somefile.tx 的行数并将其值赋给变量 ct。

```
(setq fil "SOMEFILE.TXT")
(setq x (open fil "r") ct 0)
(while (read-line x)
   (setq ct (1+ ct))
)
(close x)
```

command

执行一条 AutoCAD 命令

(command [arguments] ...)

arguments 参数表示要执行的 AutoCAD 命令名和所需的响应。command 函数的参数可以是字符串、实数、整数或点，但必须与要执行的命令所需的参数一致。空字符串 ("") 表示从键盘敲 ENTER 键。不带参数调用 command 相当于敲 ESC 键，这样可取消大多数 AutoCAD 命令。command 函数返回

nil。

command 函数对每一个参数求值并顺序传给 AutoCAD 以响应提示。它以字符串形式提交命令名和选项；以两个实数组成的表的形式提交二维点；以三个实数组成的表的形式提交三维点。command 函数不能识别命令名。

注意　如果在 acad.lsp 或 MNL 文件中使用 command 函数，则只能在 defun 表达式中调用。应使用 S::STARTUP 函数定义那些图形初始化命令。

下例先将变量 pt1 和 pt2 分别设为点 (1,1) 和点 (1,5)，然后调用 command 函数执行 LINE 命令并将这两个点的值传给 AutoCAD。

```
(setq pt1 '(1 1) pt2 '(1 5))
(command "line" pt1 pt2 "")
```

如果应用程序的运行平台是外语版 AutoCAD，则 command 函数发送的命令名前必须带下划线 (_)，这样命令才能被转换。如果使用圆点前缀来避免重定义命令，圆点前缀和下划线前缀可以用任意顺序组合：如 "._line" 和 "_.line" 都是有效命令。

如果系统变量 CMDECHO（可通过 setvar 和 getvar 函数存取）设为 0，通过 command 函数执行的命令将不会在命令行中显示。

getxxx 等用户输入函数（如 getangle、getstring、getint、getpoint 等）不能在 command 函数里使用，如果这样做，将会导致如下错误信息并中断正在执行的命令：

错误: AutoCAD 拒绝的函数

如果需要用户输入，可以在调用 command 函数前调用 getxxx 函数，或在相邻的 command 函数调用之间调用 getxxx 函数。

对于那些需要选取一个对象的 AutoCAD 命令（如 BREAK 和 TRIM 命令），可以提供一个用 entsel 函数获得的表来取代为选取一个对象而需要的点，这方面的例子可参见将拾取点传给 AutoCAD。

AutoCAD 的 DTEXT 和 SKETCH 命令直接读取键盘和数字化仪，因此不能在 AutoLISP 的 command 函数中使用这两个命令。如果在函数中使用 'SCRIPT

命令，它必须是出现在该 AutoLISP 程序的最后。

每一个通过 command 函数执行的命令都会创建一个 UNDO 组。如果用户在运行 AutoLISP 程序后键入 U（或 UNDO），只有最后一个命令被撤销。如果继续键入 UNDO，将把程序中的命令以从后到前的顺序撤销。如果想将一些命令（或整个程序中的命令）视为一组，可使用“UNDO 开始”和“

UNDO 结束”选项。

使用 PAUSE 符号

如果一个 AutoCAD 命令处于激活状态，并且有预定义的 PAUSE 符号作为函数的参数，这时 command 函数就会被暂时挂起，等待用户输入。

PAUSE 符号被定义为仅包含单个反斜杠字符的字符串，用户可以用一个反斜杠来代替 PAUSE 符号。然而，如果在菜单项中调用 command 函数，反斜杠会暂缓菜单项的读入，从而导致 AutoLISP 的局部求值。另外，在 AutoLISP 后续版本中，暂停机制可能需要一个不同的触发值。所以建议使用 PAUSE 符号而不是直接使用反斜杠字符。

如果字符串中需要使用反斜杠字符 (\)，该反斜杠前必须再加一个反斜杠字符 (\\)。

如果一个命令请求输入文本字符串或属性值时遇到 PAUSE 符号，则只有当系统变量 TEXTEVAL 的值为非零时，AutoCAD 才会暂停，以使用户输入文本字符串或属性值。否则，AutoCAD 不会暂停而直接使用 PAUSE 符号的文本值（单个反斜杠字符）。

当 command 函数暂停下来让用户输入时，command 函数仍是激活的，所以用户不能输入其他要进行求值的 AutoLISP 表达式。

下列代码示范了如何使用 PAUSE 符号。应该注意的是，在运行该段代码前，图形中应已存在 NEW_LAY 图层和 MY_BLOCK 块。

```
(setq blk "MY_BLOCK")
(setq old_lay (getvar "clayer"))
(command "layer" "set" "NEW_LAY" "")
(command "insert" blk pause "" "" pause)
(command "layer" "set" old_lay "")
```

上述代码先将 NEW_LAY 置为当前图层，接着暂停下来让用户选择 MY_BLOCK 块的插入点（其 X 轴和 Y 轴的插入比例因子为 1），接下来的第二次暂停是为了让用户选择图块插入的旋转角度。最后将当前图层恢复为块插入之前的状态。

如果通过 command 函数执行 SELECT 命令，并为它指定了一个 PAUSE 符号，且有一个激活的 PICKFIRST 选择集，则该命令将获得 PICKFIRST 选择集而不会暂停以让用户输入。

警告 Dim 提示符下的 Radius 和 Diameter 子命令在某些情况下会发出一些附加提示，这可能导致 11 版以前用这些命令写的 AutoLISP 程序发生故障。

请参见 关于 command 函数的详细信息，请参见执行 AutoCAD 命令。

cond

多条件多结果处理函数

(cond (test1 result1 ...) ...)

cond 函数可接受任意数目的表作为参数。它按顺序对每一个表的第一项求值，直至其中之一的返回值不是 nil 为止。该函数接着对该项后续的其他表达式求值，并返回该子表中的最后一个表达式的值。如果该子表中只有一项（即没有 result 部分），则返回表达式 test 的值。

下例用 cond 函数计算一个数的绝对值。

```
(cond
((minusp a) (- a))
    (t a)
)
```

如果变量 a 被设为 -10，上述代码将返回 10。

正如所示，cond 函数能用作 case 类型的函数，而且经常将 T 作为最后一个（缺省） test 表达式。下面是一个简单例子，它可用于测试用户响应的字符串 s： 如果 s 是 Y 或 y，它返回 1；如果是 N 或 n，它返回 0；否则它返回 nil。

```
(cond
    ((= s "Y") 1)
    ((= s "y") 1)
    ((= s "N") 0)
    ((= s "n") 0)
    (t nil)
)
```

cons

表构造函数

(cons new-first-element list)

cons 函数把第一个参数 new-first-element 加到第二个参数 list 表的开始，构成一个新表后返回。参数 new-first-element 可以是一个原子或一个表。

(cons 'a '(b c d)) 返回 (A B C D)

(cons '(a) '(b c d))　　返回 ((A) B C D)

cons 函数也可以接受原子形式的参数以构造称为点对的结构。当显示一个点对时，AutoLISP 会在第一个元素和第二个元素之间显示一个圆点。可以使用 cdr 函数返回点对的第二个元素。

(cons 'a 2)　　返回 (A . 2)
(car (cons 'a 2))　　返回 A
(cdr (cons 'a 2))　　返回 2

点对是一种特殊类型的表，一些处理普通表的函数通常不能接受点对作为参数。

cos

以实数形式返回一个以弧度为单位的角度的余弦值

(cos ang)
(cos 0.0)　　返回 1.0
(cos pi)　　返回 -1.0

cvunit

将某值从一种度量单位转换成另一种度量单位

(cvunit value from to)

value 参数可以是一个要作单位转换的数，也可以是包含了两个或三个数的二维或 三维点表。from 参数表示要被转换值的单位，而 to 参数则表示转换后值的单位。from 和 to 参数可以是文件 acad.unt 中的任何单位类型名。

如果成功，cvunit 函数返回转换后的值。如果 from 和 to 参数中有某个单位名是未知的（在文件 acad.unt 中找不到），或两个单位不相容（例如将“克”转换成“年”），cvunit 函数返回 nil。

(cvunit 1 "minute" "second") 返回 60.0
(cvunit 1 "gallon" "furlong") 返回 nil
(cvunit 1.0 "inch" "cm")　　返回 2.54
(cvunit 1.0 "acre" "sq yard") 返回 4840.0
(cvunit '(1.0 2.5) "ft" "in")　　返回 (12.0 30.0)
(cvunit '(1 2 3) "ft" "in")　　返回 (12.0 24.0 36.0)

注意　如果有若干个值要作同一类型的转换，可以先取数 1.0 作该种类型的转换，然后将返回值作为类型转换的转换系数即可，这样效率更高。但要注意的是，这种方法只适应于除温度之外的其他所有单位的转换，因为温度转换要另外引入一个偏移量。

请参见　单位转换

D

defun

定义一个函数

(defun sym argument-list expr ...)

defun 函数定义一个以 sym 参数为函数名的函数。函数名后是参数列表（可能为空），其中可以用斜杠隔开，描述该函数的一个或多个形参和局部变量（这是可选的）。该斜杠与最前面的形参和最后面的局部变量之间要以至少一个空格隔开。如果不申明任何参数和局部变量，必须在函数名后提供一对空括号或者 nil。

下列例子说明了哪些参数列表是有效的，哪些是无效的：

(defun myfunc (x y) ...)　　函数有两个形参
(defun myfunc (/ a b) ...)　　函数有两个局部变量
(defun myfunc (x / temp) ...) 一个形参，一个局部变量
(defun myfunc () ...)　　没有形参或局部变量

函数不能有两个或多个同名的形参，但两个局部变量可以同名，局部变量也可以和形参同名：

(defun fubar (a a / b) ...) 非法定义
(defun fubar (a b / a a b) ...)　合法的，但没有实际作用

如果形参/局部变量表中有多个同名符号，则只使用最先出现的一个而把随后的同名符号忽略。

执行函数时，参数表和局部变量表后的一个或多个表达式将会被求值。

defun 函数返回被定义的函数名。但调用定义的函数时，调用表达式中的实参将会把值赋给形参。在函数中可以自由地使用局部变量而不必担心其在函数之外的影响。函数返回最后一个表达式的值。函数中所有表达式的影响都局限在该函数内部。

下例用 defun 函数定义了一个新函数并说明了该函数的返回值：

```
(defun add10 (x)
    (+ 10 x)
)                    返回 ADD10
(add10 5)            返回 15
(add10 -7.4)             返回 2.6
```

下列代码也定义了一个新函数：

```
(defun dots (x y / temp)
    (setq temp (strcat x "..."))
    (strcat temp y)
)                    返回 DOTS
(dots "a" "b")           返回 "a...b"
(dots "from" "to")       返回 "from...to"
```

警告　不要使用 AutoLISP 的内置函数名或符号名作为用户自定义的函数名，这会造成无法调用该内置函数。关于如何获得内置函数和已定义函数名的列表，请参见 atoms-family。

请参见　符号和函数处理

dictadd

将一个非图形对象加入到指定的词典中

(dictadd ename symbol newobj)

将对象 newobj 加入到词典 ename 中。symbol 参数是待加对象的关键字，它必须是唯一的，也就是说，在词典中不能有两个同样的关键字。参数 newobj 所指的对象必须是非图形对象。

一般的规则是，加入到某词典的对象在该词典中必须是唯一的。在将组对象加入到组词典时这可能会引起问题。在组词典中用不同的关键字重复加入同一个组对象可能会使 dictnext 函数陷入死循环。

dictnext

在词典中查找下一项

(dictnext ename [rewind])

ename 参数指定要搜索的词典对象的图元名。

当重复调用 dictnext 函数时，它每次返回指定词典中的下一个条目。dictsearch 函数指定要检索的条目。如果在调用 dictnext 函数时提供了 rewind 参数并且其求值结果不为 nil，词典会被回绕。该函数将检索词典的第一个条目。当词典中不再有条目时，该函数返回 nil。它不返回已被删除的词典条目。

关于主词典图元名的介绍，请参见 namedobjdict。

注意　一旦开始检索一个词典的内容，传递另一个词典名给 dictnext 函数将会导致丢失原词典中的位置。也就是说，dictnext 函数只能保证使用一个全局枚举器。

如果找到了一个条目，dictnext 函数会返回含 DXF 类型码和值的一个点对表。

dictremove

从指定词典中删除一个条目

(dictremove ename symbol)

从 ename 参数指定的词典中删除 symbol 参数指定的词典条目。

如果 ename 无效或没有找到 symbol 指定的条目，dictremove 函数返回 nil。如果成功，dictremove 函数返回所删条目的图元名。

缺省情况下，从词典中删除某条目并不会将其从数据库中删除。将其从数据库中删除需使用 entdel 函数。但当前该规则的例外包括组和多线样式。实现这些功能的代码要求数据库和词典都是最新的，这样才可以在（用 dictremove 函数）从词典中删除某条目时，自动从数据库中删除该条目。

当一个多线样式正被数据库中的一个多线引用时，dictremove 函数不能将其从多线样式词典中删除。

dictrename

重命名词典条目

(dictrename ename oldsym newsym)

将 ename 参数指定的词典中关键字为 oldsym 参数的条目重命名为 newsym。

如果 oldname 在词典中不存在，或 ename 无效，或 newname 无效，或词典中已经存在

newname，dictrename 函数都返回 nil。

dictsearch

在词典中搜索某个项

(dictsearch ename symbol [setnext])

ename 参数指定要搜索的词典对象的图元名。symbol 参数是指定词典中某个项的字符串。

如果 dictsearch 函数找到了给定项的一个条目，它会按照与 dictnext 函数同样的格式返回该条目。如果没有找到这样的条目，它将返回 nil。

通常情况下，dictsearch 函数不影响 dictnext 函数检索条目的顺序。但是如果 dictsearch 函数调用成功，而且为其提供了其值非空的 setnext 参数，那么 dictnext 函数的条目计数器就会被调整，使得接下来的 dictnext 函数调用会返回此次 dictsearch 调用返回条目的下一个条目。

下列例子使用 dictsearch 函数来寻找组 G2 的定义行（该代码假定当前图形中存在 G2 组）。

(setq grp (dictsearch (namedobjdict) "ACAD_GROUP"))

(setq g2 (dictsearch (cdr (assoc -1 grp)) "G2"))

关于主词典图元名的介绍，请参见 namedobjdict。

dimx_tile 和 dimy_tile

以对话框单位为单位返回控件的尺寸

(dimx_tile key) 和 (dimy_tile key)

dimx_tile 函数返回控件的宽度，dimy_tile 函数返回控件的高度。key 参数是用来指定控件的一个字符串，它是区分大小写的。

这两个函数返回的坐标值都是指定控件所允许的最大值。由于控件的坐标值是从 0 开始计算的，所以，假设 X 和 Y 是总宽度和总高度，那么，由这两个函数返回的坐标值不会超过 X-1 和 Y-1。dimx_tile 和 dimy_tile 在与 vector_image、fill_image 和 slide_image 等函数配套使用时，可为这些函数提供指定控件大小的绝对坐标。

distance

返回两个点之间的三维距离

(distance pt1 pt2)

(distance '(1.0 2.5 3.0) '(7.7 2.5 3.0)) 返回 6.7

(distance '(1.0 2.0 0.5) '(3.0 4.0 0.5)) 返回 2.82843

如果提供的参数中有一个或两个二维点，distance 函数会忽略所提供的任何三维点的 Z 坐标，而返回将这些点投影到当前构造平面上后所得的点之间的二维距离。

请参见 几何实用函数

distof

将一个表示实（浮点）数的字符串转换成一个实数

(distof string [mode])

mode 参数指定了字符串所用格式的单位，它的值应与 AutoCAD 的系统变量 LUNITS 的允许取值相对应，其取值如下表所示。如果省略 mode 参数，

distof 函数使用系统变量 LUNITS 的当前值。

线性单位值

Mode 值 字符串格式

1 科学记数格式

2 十进制格式

3 工程记数格式（英尺和十进制英寸）

4 建筑记数格式（英尺和分数英寸）

5 分数单位格式

string 参数必须是一个字符串，且能够按照 mode 参数所指定的方式正确地进行语法分析。string 参数既可以与 rtos 函数返回字符串具有相同的格式，也可以是 AutoCAD 允许从键盘输入的那种格式。distof 函数和 rtos 函数是一对互补函数。如果将 rtos 函数的返回值作为参数传给 distof

函数，则 distof 函数保证会返回一个有效值，反之亦然（假定 mode 参数的值相同）。

注意 distof 函数把方式代码 3 和方式代码 4 同等看待。也就是说，如果方式代码指定为 3（工程）或 4（建筑），且 string 参数为这两种格式之一的字符串，distof 函数均会返回正确的实数值。

如果 distof 函数调用成功，它返回一个实数；否则它返回 nil。

done_dialog

终止一个对话框

(done_dialog [status])

必须在动作表达式或回调函数中调用 done_dialog 函数（请参见 action_tile）。

如果指定了可选参数 status，它必须是一个正整数，该正整数将由函数 start_dialog 返回（否则 start_dialog 函数在选择 OK 时返回 1，而在选择 Cancel 时返回 0）。任何大于 1 的 status 值的含义取决于应用程序。

done_dialog 函数返回一个二维点表，该点表是退出对话框时该对话框的位置坐标 (X,Y)。可以将该点传给随后调用的 new_dialog 函数，这可以让该对话框再次被打开时处于用户选择的位置。

注意 如果为关键字为 "accept" 或 "cancel" 的按钮（通常是 OK 和 Cancel 按钮）提供了一个回调函数，那么该回调函数必须调用 done_dialog 函数，否则用户将陷入对话框中出不来。如果不为这些按钮提供回调函数而使用标准的退出按钮，AutoCAD 将自动处理它们。另外，为 "accept" 按钮提供的显式回调函数必须在调用 done_dialog 函数时将参数设为 1（或由应用程序定义的其他值），否则，start_dialog 函数会返回缺省值 0，而 0

意味着用户取消了该对话框。

E

end_image

结束创建当前激活的对话框图像

(end_image)

该函数是 start_image 函数的配套函数。

end_list

结束对当前激活的对话框的表的处理

(end_list)

该函数是 start_list 函数的配套函数。

entdel

删除对象（图元）或恢复先前已被删除的对象

(entdel ename)

如果由 ename 参数指定的图元在当前图形中，entdel 函数将删除该图元。如果在本次编辑会话中该图元已被删除，entdel 函数将恢复该图元（使其重新回复到图形中）。只有在图形退出图形编辑环境时，被删除的图元才会真正从图形中清除。entdel 函数既可以删除图形对象，又可以删除非图形对象。

```
(setq e1 (entnext))        ;将 e1 设为图形中第一个图元的
                           ;图元名
(entdel e1)                ;删除图元 e1
(entdel e1)                ;恢复已被删除的图元 e1
```

entdel 函数仅能处理主图元。属性和多段线的顶点不能独立于它们的父图元而被删除。可以通过调用 command 函数执行 ATTEDIT 或 PEDIT 命令来修改子图元。

不能删除块定义中的图元。但可以调用 entmake 函数来完全重新定义一个块，以去掉想要删除的图元。

entget

获得对象（图元）的定义数据

(entget ename [applist])

entget 函数返回一个包含了图元定义数据的表，这同时适用于图形图元和非图形图元。如果提供了可选参数 applist(已注册的应用名表)，entget

函数还会返回与这些应用程序相关的扩展数据。

entget 函数以关联表的形式返回数据，可以使用 assoc 函数提取各个数据项。该关联表中的每一项都被指定了一个 AutoCAD 的 DXF 组码，来描述图元的各种性质。

假定图形中最后创建的那个对象是从点 (1,2) 到点 (6,5) 的一条直线。可以通过调用 entlast 函数获得最后一个对象的图元名，并将其传给 entget

函数以获取图元数据。

(entget (entlast))

可能返回如下关联表：

((-1 . <图元名: 60000014>) 图元名
  (0 . "LINE") 对象类型
  (8 . "0") 图层
  (10 1.0 2.0 0.0) 起点
  (11 6.0 5.0 0.0) 终点
)

AutoLISP 所用的 DXF 组码与 DXF 文件中的组码稍有不同。关于 AutoLISP 的 DXF 组码的详细信息，请参见附录 C“DXF 组码”。

和 DXF 文件相同的是，图元的头部项（颜色、线型、厚度、属性跟随标志和图元句柄）只有在具有非缺省值时才被输出。和 DXF 文件不同的是，可选的图形定义域不论其值是否与缺省值相同都会被输出。这使程序处理起来更简单：对处理这些域的通用算法程序可以假定总是提供这些域的数据。还有一点与 DXF 文件不同的是，相关的 X、Y 和 Z 坐标被组合成一个点，象 (10 1.0 2.0 3.0) 这样的形式，而不象 DXF 文件中那样用三个分隔的组 10、20 和 30

来表示。

关联表的第一项 -1 组包含了该图元的图元名。可以用 assoc 函数提取表示图元数据的各个点对，然后用 cdr 函数取出其中的值。

表示点信息的子表与其他点对表不同。由于系统约定子表的 cdr 是该组的组值，因为点是一个由两个（或三个）实数组成的表，所以整个表必须是一个有三个（或四个）元素的表（包括第一个元素：组码值）。该表的 cdr 是代表点的一个子表，这保证了“子表的 cdr 返回该组组值”这个约定的正确性。

当写程序处理这些图元数据表时，必须保证程序与子表的顺序无关。使用 assoc 函数可以做到这点。包含图元名的 -1 组，使得修改操作可以承认图元表，从而避免了在平行结构中保持图元名的要求。在多段线和一组属性结尾处的 SEQEND 图元包含了一个 -2 组，它的 cdr 是该图元的主图元图元名。这样使得从任何子图元出发，向后搜索找到 SEQEND 图元后总可以获得主图元：对 -2 组调用 cdr 函数便可获得相应主图元的图元名。

警告　在对多段线的顶点执行 entget 操作之前，应该读入或写入该多段线图元的头部。如果最近处理过的多段线图元不同于该顶点所在的多段线图元，可能会丢失线宽信息（40 和 41 组）。

一个对象所涉及的所有点均以该对象的对象坐标系 (OCS) 表示。OCS 可以从 WCS 和对象的延伸方向（210 组）中导出。当处理在非 WCS 中绘制的对象时，可能需要调用 trans 函数将它们的坐标转换成 WCS 或当前 UCS。

entlast

返回图形中最后那个未被删除的主对象（图元）名

(entlast)

entlast 函数经常用于获得最近用 command 函数加入到图形中的一个新图元的图元名。该函数并不要求返回的图元显示在屏幕上，也不要求它处于解冻状态。

(setq e1 (entlast)) 获得图形中最后一个主图元名，
                     并赋给 e1
(setq e2 (entnext e1)) 将 e2 设为 nil （也可能是一个属性或子图元名）

在用户的应用程序中，如果需要获得最后一个未被删除的图元名（主图元和子图元），可用以下函数代替 entlast 函数。

```
(defun lastent (/ a b)
  (if (setq a (entlast))          获得最后的主图元
    (while (setq b (entnext a))     如果随后有子图元，循环
      (setq a b)                    直至最后一个子图元
    )
  )
  a                               返回最后一个主图元
)                                 或子图元
```

entmake

在图形中创建一个新图元（图形对象）

(entmake [elist])

elist 参数必须是一个图元定义数据表，该表格式应与 entget 函数所返回表的格式相似，该参数

必须包含所欲创建图元的全部必要数据。本函数可以创建图形和非图形图元。如果省略了任何一个必须的定义数据，该函数返回 nil 并拒绝创建该图元。如果省略了可选的定义数据（如图层），该函数使用缺省值。

如果 entmake 成功创建了新图元，则返回该图元的定义数据表；如果没有成功创建该图元，则返回 nil。

也可以用以下方法创建新图元：先调用 entget 函数获得一个图元的定义数据，修改后传给 entmake 函数。在创建新图元之前，entmake 函数先验证是否提供了合法的图层名、线型名和颜色代码。如果引入了一个新的图层名，该函数会自动创建一个新图层。如果所创建的图元类型要求提供块名、尺寸样式名、字体名和形名，该函数也会检查它们。

图元类型（如 CIRCLE 或 LINE）必须是 elist 参数所指的表中的第一项或第二项。如果是第二项，那么第一项必须是图元名。这正是 entget 函数所返回表的那种格式。在这种情况下，entmake 函数创建图元时会忽略图元名。如果 elist 参数中包含了图元句柄，该句柄也会被忽略。

下列代码创建一个以点 (4,4) 为圆心、以 1 为半径的红色圆，可选的图层和线型项被省略而采用缺省值。

```
(entmake
  '((0 . "CIRCLE")      图元类型
    (62 . 1)            颜色
    (10 4.0 4.0 0.0)        圆心
    (40 . 1.0)          半径
  )
)
```

注意　创建在冻结图层上的对象，在该图层被解冻之前不会被重新生成。

复杂图元

可以通过若干次调用 entmake 函数来创建一个复杂图元（块定义、多段线或包含有属性的一个块引用）。当 entmake 函数发现是创建一个复杂图元时，它会创建一个临时文件来收集定义数据。每次调用 entmake 函数时都会检查是否存在临时文件，如果存在，新数据将会增加到该文件中。当复杂图元的定义完成时（通过 entmake 函数增加一个合适的 SEQEND 或 ENDBLK 图元），会重新检查提供的数据并将复杂图元加入到图形中。完成块定义后（用 entmake 函数附加一个 ENDBLK 图元），该函数将返回块名而不是通常的图元定义数据表。

注意　不能用 entmake 函数创建视口对象。

如果其中某个图元的数据无效，则该函数会拒绝创建该图元和整个复杂图元。块定义不允许嵌套，也不允许引用它本身，但可以包含对其他块定义的引用。一个复杂图元的所有子图元要么全在模型空间，要么全在图纸空间，但不能两个空间中各有一部分。

只有插入图元才承认 66 组码，表示属性跟随。多段线的 66 组码被强制性地置为 1 以表示顶点跟随，其他图元该组码的缺省值为 0。紧随多段线图元后的只能是顶点图元。

在完成复杂图元定义之前，不会显示该图元的任何部分。可以通过不带参数调用 entmake 函数来取消复杂图元的创建，这将清除临时文件并返回 nil

。

BLOCK 和 ENDBLK 图元可用于创建新的块定义。新定义的块将被自动加入到符号表中以使得它能被别的对象引用。

应用程序可以用具有任意多个边的多面网格来表示多边形。然而， AutoCAD 的图元结构对一个给定的面图元所能指定的顶点数目作了一定限制，但可以通过将多边形划分为三角形楔块来表示更复杂的多边形。 AutoCAD 用四顶点面来表示三角形楔块，当然其中有两个顶点的值相同。它们的边应设为不可见状态以避免这种划分成为可见的。PFACE 命令可自动实现这种划分，但如果应用程序直接生成多面网格，它必须自己完成这个工作。

在这种划分的处理过程中，每个面的顶点数目是一个关键的参数。系统变量 PFACEVMAX 为应用程序提供了每个面图元的顶点数目。该变量是只读的，其值为 4。

警告　用 entmake 函数创建块定义时可能会覆盖已有块。该函数不检查块定义表中块名是否有冲突，所以在用它创建块之前，最好先调用 tblsearch 函数以确保新块的块名是不冲突的。然而按下节所述用 entmake 函数重定义无名块，却是非常有用的。

无名块

一个图形中的块定义表可以包含多个无名块。创建无名块通常是为了支持填充图案和关联尺寸标注。应用程序也可以调用 entmake 函数创建无名块来达到自己的目的，通常是包含用户不能直接访问的图元。

调用 entmake 函数时，尺寸图元的组码 2（块名部分）是可选的。如果在图元定义数据表中省略了块名，AutoCAD 将创建一个新块；否则它用提供的块名创建尺寸。

无名块的块名（组码 2）是 *Unnn，其中 nnn 是 AutoCAD 生成的一个数。另外，无名块的块类型标志（组码 70）的低位被设为 1。当 entmake 函数创建一个块名以 * 开头并且其无名标志位（组码 70 的低位）为 1 的块时，entmake 函数视之为无名块并给它分配一个块名。块名字符串中 * 后的字符将被忽略。块被创建后， entmake 函数返回块名。如果是多次调用 entmake 函数来创建块，将在成功执行下列函数调用后返回块名。

(entmake "endblk")

无论何时打开图形，其中所有未被引用的无名块都将从块定义表中清除出去，而被引用（插入）的无名块则不会被清除。可以用 entmake 函数创建对无名块的引用（插入），但不能将无名块传入 INSERT 命令中，还可以用 entmake 函数重新定义块。可以用 entmod 函数修改块中的图元（但不是块本身）。

注意 尽管被引用的无名块会被永久保存，但其块名的数字部分在不同的编辑会话期间可能不一样。应用程序不能依赖于无名块的块名保持不变。

entmakex

创建一个新图元或对象，赋给它一个句柄和图元名（但不指定所有者），返回新图元的图元名

(entmakex [elist])

elist 参数必须是一个图元定义数据表，该表格式应与 entget 函数所返回表的格式相似，该参数必须包含需创建图元的全部必要数据。该函数可以创建图形和非图形图元。如果省略了任何一个必须的定义数据，该函数返回 nil 并拒绝创建该图元。如果省略了可选的定义数据（如图层），该函数使用缺省值。

如果 entmakex 成功创建了新图元，则返回该图元的定义数据表；如果没有成功创建该图元，则返回 nil。

警告 没有所有者的图元和对象将不会写入 .dwg 或 .dxf 文件。在调用 entmakex 函数后的某处一定要给新建的对象和图元设置所有者。例如，可调用 dictadd 函数设置某词典拥有某对象。

entmod

修改对象（图元）的定义数据

(entmod elist)

entmod 函数需要传递一个与函数返回表相同格式的表 elist，它更新由 -1 组中的值指定的图元名的数据库信息。通过 AutoLISP 更新数据库信息的基本方法是，先用 entget 函数获得图元的定义数据，修改这些数据后调用 entmod 函数来更新数据库中的该图元。entmod 函数既可以修改图形对象，又可以修改非图形对象。

```
(setq en (entnext))          将 en 设为图形中第一个图元名
(setq ed (entget en))            将 ed 设为图元 en 的图元数据
(setq ed
  (subst (cons 8 "0")
    (assoc 8 ed)             将 ed 的图层设为 0
    ed
  )
)
(entmod ed)                  修改图形中图元 en 的图层
```

entmod 函数对它所能完成的修改作了某些限制。首先，它不能改变一个图元的类型和句柄。如果一定要这样做，只能先调用 entdel 函数删除它，然后调用 command 或 entmake 函数创建一个新图元。在执行 entmod 函数之前，图元数据表引用的所有对象对 AutoCAD 来说都必须是已知的。因此，字体、线型、形和块名必须先在图形中定义才能在 entmod 函数中的图元数据表中使用。但图层名例外：如果在图元数据表中用了未定义的新图层名，

entmod 函数将用 LAYER 命令的 New 选项采用标准的缺省值来创建该图层。

对于那些使用浮点值的图元域（如图元的厚度），entmod 函数可以接受整数，并将整数转换成浮点数。与此类似，对于使用整数的图元域（如颜色代码），如果提供的是浮点数，entmod 函数会切除

其小数部分，将其转换成整数。

entmod 函数会对提供给它的关联表作一致性检查。如果检测到严重错误，则 entmod 函数不会更新数据库而返回。否则，entmod 函数返回作为它参数的关联表。另外，entmod 函数不能修改那些内部域，如 SEQEND 图元的 -2 组中的图元名，如果试图作这样的修改，将会被系统忽略。

当 entmod 函数修改一个主图元时，它会修改该图元并更新图元在屏幕上的图像（包括子图元）。但当 entmod 函数修改一个子图元（如多段线顶点或块属性）时，它只修改该子图元的数据而不会更新它在屏幕上的图像。在完成对某主图元的所有子图元的修改后，可以调用 entupd 函数来更新它在屏幕上的图像。

注意　不能用 entmod 函数修改视口图元，但可以将图元的空间可见性改为 0 或 1（视口对象除外）。如果用 entmod 函数修改了块定义中的图元，该修改会影响图形中所有引用该块的场所。

在用 entmod 函数修改顶点图元前，应先读出或写入多段线的头部。如果最近处理的多段线图元不是该顶点所在的多段线，可能会丢失宽度信息（40

和 41 组）。

警告　可以用 entmod 函数修改块定义中的图元，但这样做可能会生成一个引用自身的块，这将导致 AutoCAD 系统崩溃。

entnext

返回图形中的下一个对象（图元）名

(entnext [ename])

如果不带参数调用 entnext 函数，它返回数据库中第一个未被删除的图元名。如果带参数 ename 调用该函数，它返回数据库中 ename 图元后的下一个未被删除的图元名。如果数据库中不存在下一个图元，它返回 nil。 entnext 函数既可以返回主图元，又可以返回子图元。

ssget 函数所形成的选择集中只包含主图元，不包含块属性或多段线顶点。可以通过调用 entnext 函数遍历复杂图元的子图元来访问复杂图元的内部结构。获取子图元名后，可以象处理其他图元一样处理子图元。获取子图元名后，还可以用以下方法找到它的主图元：调用 entnext 函数直至找到 SEQEND 图元，该图元的 -2 组中即包含了它的主图元名。

(setq e1 (entnext))　　将 e1 设为图形中第一个图元名

(setq e2 (entnext e1))　　将 e2 设为 e1 后的下一个图元名

entsel

提示用户通过指定一个点来选择单个对象（图元）

(entsel [msg])

entsel 函数返回一个表，其中第一个元素是用户所选对象的图元名，第二个元素是用户选择对象时指定的拾取点的坐标值（用当前 UCS 表示）。如果指定了可选参数 msg（一个字符串），则该字符串将作为提示用户选择对象的提示信息。否则，该提示为其缺省值：“选择对象:”。

注意　entsel 函数返回的拾取点不一定在所选对象上，它返回的是选择对象时十字光标的位置。拾取点和对象之间的关系依赖于拾取框的尺寸和当前缩放比例。

entsel 函数返回的表可以供 AutoCAD 响应任何选择对象的提示。AutoCAD 将其视为指定某点来拾取对象。

下列 AutoCAD 命令说明了如何使用 entsel 函数以及 entsel 函数返回的表：

命令：　line

起点：　1,1

下一点：　6,6

下一点：　ENTER

命令：　(setq e (entsel "请选择一个对象: "))

请选择一个对象：　3,3

(<图元名: 60000014> (3.0 3.0 0.0))

有时候在操作一个对象时，不但需要选中一个对象，同时还需要知道指定的点。例如在 AutoCAD 中的 Object Snap 以及 BREAK、TRIM 和 EXTEND 命令中就是这样。entsel 使得 AutoLISP 程序可以实现这种操作。它通过选取点的方式来选取单个对象。除非在函数中作特别申明，否则 initget 函数将忽略当前 Osnap 设置。entsel 函数支持它前面通过调用 initget 函数设置的关键字。

entupd

更新对象（图元）的屏幕图像

(entupd ename)

当调用 entmod 函数修改多段线顶点和块属性时，整个复杂图元在屏幕上的图像并没有被更新。可以使用 entupd 函数来更新被修改的多段线或块的屏幕图像。调用 entupd 函数时指定的 ename 参数，可以是多段线或块的任何部分的图元名，而不要求一定是该对象的头部图元名。虽然 entupd 函数主要用来更新多段线或带属性的块的屏幕图像，但它可以用于任何图元。它将更新图元的屏幕图像，包括所有的子图元。

注意　如果 entupd 函数用于嵌套图元（块中的图元）或包含嵌套图元的块，某些图元可能不会被重新生成。为了确保完全重新生成，必须执行 REGEN

命令。

假定图形中的第一个图元是有若干个顶点的多段线，那么：

```
(setq e1 (entnext))          将 e1 设为多段线的图元名
(setq e2 (entnext e1))           将 e2 设为第一个顶点
(setq ed (entget e2))            将 ed 设为顶点数据
(setq ed
  (subst '(10 1.0 2.0)
    (assoc 10 ed)          将 ed 中的顶点位置
    ed                 改为点 (1,2)
  )
)
(entmod ed)                  移动图形中的顶点
(entupd e1)                  重新生成多段线图元 e1
```

eq

确定两个表达式是否有相同的约束

(eq expr1 expr2)

eq 函数确定两个表达式 expr1 和 expr2 是否被设置为同一个对象（例如用 setq 函数设置）。如果是，它返回 T；否则它返回 nil。

可以用本函数确定两个表是否相同。例如，给定如下表达式：

```
(setq f1 '(a b c))
(setq f2 '(a b c))
(setq f3 f2)
```

那么

```
(eq f1 f3)           返回 nil，因为 f1 和 f3 不是同一个表
(eq f3 f2)           返回 T，因为 f3 和 f2 是同一个表
```

请参见　=（等于）和 equal 函数

equal

确定两个表达式的值是否相等

(equal expr1 expr2 [fuzz])

equal 函数确定两个表达式 expr1 和 expr2 的求值结果是否相等。当比较两个实数（或由实数组成的表，如点表）时，即使是恒等的两个数，如果采用不同的计算方法，结果也有可能稍有差别。因此，提供可选参数 fuzz（一个数字）让用户能指定表达式 expr1 和 expr2 之间的最大允许误差，误差在此范围内时，仍然认为二者相等。

例如给定下列赋值：

```
(setq f1 '(a b c))
(setq f2 '(a b c))
(setq f3 f2)
(setq a 1.123456)
(setq b 1.123457)
```

那么

```
(equal f1 f3)                返回 T
(equal f3 f2)                返回 T
(equal a b)                  返回 nil
(equal a b 0.000001)             返回 T
```

尽管 equal 认为相等的两个表 eq 不一定认为它们相等，但 equal 认为相等的两个原子，eq 一

定也认为它们相等，而且，eq 认为相等的两个表或原子，equal 一定也认为它们相等。

请参见 =（等于）和 eq 函数

error

用户可定义的错误处理函数

(*error* string)

如果 *error* 函数不为 nil，那么当 AutoLISP 的错误条件出现时，就会执行它，并由 AutoCAD 传给它一个包含了错误信息的字符串作为参数，否则将执行 AutoLISP 标准的出错处理。

下面的函数完成和 AutoLISP 的标准错误处理程序同样的事情：打印 "错误：" 和错误信息。

```
(defun *error* (msg)
  (princ "错误: ")
  (princ msg)
  (princ)
)
```

用户的 *error* 函数中可以不带参数调用 command 函数（例如 (command)），这样可以中断前面由 command 函数执行的 AutoCAD 命令。

请参见 错误处理，其中有一个错误处理程序的例子，它检测由 exit 和 quit 函数返回的字符串。

eval

返回 AutoLISP 表达式的求值结果

(eval expr)

例如，假定下列赋值：

```
(setq a 123)
(setq b 'a)
```

则，

```
(eval 4.0)          返回 4.0
(eval (abs -10))        返回 10
(eval a)            返回 123
(eval b)            返回 123
```

exit

强行使当前应用程序退出

(exit)

如果调用 exit 函数，它会返回错误信息 "quit/exit abort"，并返回到 AutoCAD 的命令提示处。

请参见 quit 函数

exp

返回常数 e (2.718282...) 的指定次幂的值

(exp num)

```
(exp 1.0)           返回 2.71828
(exp 2.2)           返回 9.02501
(exp -0.4)          返回 0.67032
```

expand

通过请求指定数目的段数来分配节点空间

(expand int)

请参见 关于 expand 函数的详细信息，请参见手动分配

expt

返回以某指定数为底数的若干次幂的值

(expt number power)

如果两个参数都是整数，结果也是整数，否则结果为实数。实例如下：

(expt 2 4)返回 16

(expt 3.0 2.0) 返回 9.0

F

fill_image

在当前激活的对话框图像控件上画一个填充矩形

(fill_image x1 y1 wid hgt color)

fill_image 函数必须在 start_image 和 end_image 两个函数调用之间调用。color 参数可以是 AutoCAD
的一个颜色代码，也可以是下表中的逻辑颜色代码之一：
颜色属性所用的符号名称
颜色代码　　ADI 助记符　说明
-2　BGLCOLOR　AutoCAD 图形屏幕的当前背景颜色
-15　DBGLCOLOR　　当前对话框的背景颜色
-16　DFGLCOLOR　　当前对话框（文本）的前景颜色
-18　LINELCOLOR　　当前对话框线的颜色
(x1,y1) 坐标指定填充矩形第一个角（左上角）的位置，该填充矩形的第二个角（右下角）由从第一个角的相对距离 (wid,hgt) 确定（wid 和 hgt 必须是正数）。原点 (0,0) 表示图像控件的左上角，可以调用 dimx_tile
和 dimy_tile 函数来获取其右下角的坐标。

findfile

在 AutoCAD 的库目录范围内搜索指定文件
(findfile filename)
findfile 函数对要搜索的文件类型或 filename 的扩展名不作假定。如果在 filename 中没有指定驱动器/目录前缀，findfile 函数在 AutoCAD 的库目录范围内搜索。如果指定了驱动器/目录前缀，findfile
仅在指定的目录下搜索。findfile 函数总是返回“驱动器/目录/文件名”形式的路径全名和文件名；如果没有找到指定文件，它返回 nil。
下例中假定当前目录是 /acad 且该目录下有文件 abc.lsp，那么：
(findfile "abc.lsp")　　返回 "/acad/abc.lsp"
如果当前正在编辑的图形文件在 /acad/drawings 目录下，且环境变量 ACAD 被设为 /acad/support，而文件 xyz.txt 仅存在于 /acad/support 目录下，那么：
(findfile "xyz.txt")　　返回 "/acad/support/xyz.txt"
如果在库目录范围内不存在文件 nosuch，那么：
(findfile "nosuch")　　返回 nil
findfile 函数返回的带全路径名的文件名可以供 open 函数使用。

fix

截去实数的小数部分，将它转换成整数后返回该整数
(fix num)
fix 函数截去数 number 的小数部分，而返回它的整数部分。
(fix 3)　　返回 3
(fix 3.7)　　返回 3
注意　如果 number 大于最大可能的整数或小于最小可能的整数（在 32 位的平台上分别是 +2,147,483,647 和 -2,147,483,648），fix 函数返回 number 被截去小数部分后得到的实数（尽管 AutoLISP 和 AutoCAD 之间的整数传送被限制在 16 位值范围内）。

float

将一个数转换成实数后返回
(float num)
(float 3)　　返回 3.0
(float 3.75)　　返回 3.75

foreach

将表中的所有成员带入表达式求值
(foreach name lst expr...)
遍历表 lst，将其中每一个元素依次赋给 name，并以之对每一个表达式 exprs 求值。可以指定任意多个 exprs 表达式，foreach 函数返回最后一个 expr 表达式求值的结果。
(foreach n '(a b c) (print n))
等价于：
(print a)
(print b)

(print c)　　并返回 c

值得注意的是，foreach 函数仅返回最后一个表达式的求值结果。

G

gc

强制执行无用数据收集，即释放那些不再使用的节点

(gc)

请参见　关于无用数据收集的详细信息，请参见节点空间

gcd

返回两个整数的最大公约数

(gcd int1 int2)

int1 和 int2 参数必须是正整数。

(gcd 81 57)　　返回 3

(gcd 12 20)　　返回 4

get_attr

获取对话框属性的 DCL 定义值

(get_attr key attribute)

参数 key 是一个用于指定控件的字符串，它是区分大小写的。参数 attribute 指定出现在该控件的 DCL 描述中的某个属性。这两个参数都是字符串。本函数返回的值是由 DCL 文件为该控件属性定义的初始值，而不反映由于用户输入或调用 set_tile 函数对该控件属性作的修改。本函数返回的属性值是一个字符串。

get_tile

返回对话框控件的当前运行时的值

(get_tile key)

参数 key 是一个用于指定控件的字符串，它是区分大小写的。它以字符串形式返回控件的当前值。

getangle

暂停以等待用户输入一个角度，然后以弧度形式返回该角度

(getangle [pt] [msg])

pt 参数是以当前 UCS 表示的一个二维基点，而 msg 参数是用作提示信息显示的字符串。如果指定了 pt 参数，那么就假定它是两个点中的第一个，这样，用户可以通过再指定一个点来给 AutoLISP 输入一个角度。也可以提供三维基点，但角度的度量都是在当前构造平面上进行的。

getangle 函数以逆时针方向测量零弧度方向（由系统变量 ANGBASE 设置）和用户指定的两点确定的直线之间的角度。所返回的角度以弧度表示，是相对于当前构造平面（当前标高处的当前 UCS 的 XY 平面）来测量的。

用户可以通过输入一个以 AutoCAD 的当前角度单位格式表示的数来指定角度，虽然当前角度单位格式可能是度、百分度或其他单位，但本函数总是以弧度为单位返回角度值。用户也可以通过在图形屏幕上指定两个点来指定角度。AutoCAD 从第一个点到当前十字光标画一条橡皮线，以帮助用户确定角度。

理解输入的角度与 getangle 函数所返回的角度之间的区别是非常重要的。传送给 getangle 函数的角度，是基于系统变量 ANGDIR 和 ANGBASE 的当前设置而确定的。然而，一旦输入了一个角度，它就以 ANGBASE

的当前设置为零弧度按逆时针方向来测量（忽略 ANGDIR 的设置）。下列代码说明了如何使用不同的参数调用该函数：

(setq ang (getangle))

(setq ang (getangle '(1.0 3.5)))

(setq ang (getangle "Which way? "))

(setq ang (getangle '(1.0 3.5) "Which way? "))

用户不能输入另一个 AutoLISP 表达式来响应 getangle 函数的请求。

请参见　getorient 函数的说明以及二者的比较

getcfg

从 acad.cfg 文件的 AppData 区域中检索应用数据

(getcfg cfgname)

cfgname 参数是指定要检索的段和参数值的一个字符串（长度不能超过 347 个字符）。如果该参数不正确，

getcfg 函数返回 nil。cfgname 参数必须是如下形式的一个字符串：

"AppData/application_name/section_name/.../param_name"

例如，假定在 AppData/ArchStuff 段中的 WallThk 参数的值是 8，则：

(getcfg "AppData/ArchStuff/WallThk") 返回 "8"

请参见 setcfg 函数

getcname

返回 AutoCAD 命令的本地化名或英文名

(getcname cname)

cname 参数指定本地化命令名或带下划线的英文命令名，其长度不能超过 64 个字符。如果 cname 参数前没有下划线，它将被认为是本地化命令名，getcname 函数返回带下划线的英文命令名。如果 cname 参数前有下划线，它将被认为是英文命令名，getcname 函数返回本地化命令名。如果 cname 参数不是一个有效的命令名，本函数将返回 nil。

例如，在 AutoCAD 的法文版中，将有：

(getcname "ETIRER") 返回 "_STRETCH"

(getcname "_STRETCH") 返回 "ETIRER"

getcorner

暂停以等待用户输入矩形第二个角的坐标

(getcorner pt [msg])

getcorner 函数需要一个以当前 UCS 表示的基点作参数，当用户在屏幕上移动十字光标时，它会从这个基点开始画出一个矩形。 msg 参数是用作提示信息显示的字符串。本函数和 getpoint 函数类似，返回一个以当前 UCS 表示的点。如果用户提供的点是三维点，本函数将忽略其 Z 坐标，而使用当前标高作 Z 坐标。

用户不能输入一个 AutoLISP 表达式来响应 getcorner 函数的请求。

getdist

暂停以等待用户输入一个距离

(getdist [pt] [msg])

pt 参数是以当前 UCS 表示的一个二维或三维基点。如果提供了该参数，那么，它就作为两点中的第一点，这时仅提示用户输入第二点。msg 参数是用作提示信息显示的字符串。

用户可以通过选择两个点来指定距离，如果提供了基点的话，则只需选择第二个点。用户还可以通过输入一个以 AutoCAD 的当前距离单位格式表示的数来指定距离。虽然当前距离单位格式可能是以英尺和英寸（建筑单位制）表示的，getdist 函数总是以实数形式返回这个距离值。

getdist 函数从第一个点到当前十字光标位置显示一条橡皮线，以帮助用户确定距离值。

如果所提供的是三维点，那么返回的值就是一个三维距离。然而，可以在调用 getdist 函数之前先调用 initget 函数，并将其标志位设置为 64，那么它就会通知 getdist 函数，要它忽略三维点的 Z 坐标，而返回一个二维距离。

(setq dist (getdist))

(setq dist (getdist '(1.0 3.5)))

(setq dist (getdist "How far "))

(setq dist (getdist '(1.0 3.5) "How far? "))

用户不能输入一个 AutoLISP 表达式来响应 getdist 函数的请求。

getenv

以字符串方式返回指定的环境变量的值

(getenv variable-name)

variable-name 参数是一个字符串，它指定要读取的变量名。如果该变量不存在，getenv 函数返回 nil。变量名必须用双引号引起来，而且其中环境变量的拼写必须与系统注册表中的拼写完全一致（包括大小写）。

例如，假定系统变量 ACAD 被设为 /acad/support ，且没有一个名为 NOSUCH 的变量，那么：

(getenv "ACAD") 返回 "/acad/support"

(getenv "NOSUCH") 返回 nil

假定环境变量 MaxArray 被设为 10000，那么：

(getenv "MaxArray")　　返回 "10000"

请参见　setenv 函数

getfiled

用标准的 AutoCAD 文件对话框提示用户输入一个文件名，并返回该文件名

(getfiled title default ext flags)

title 参数是一个用于指定对话框标题的字符串；default 参数指定使用的缺省文件名（也可以是空字符串 [""]）；ext 参数指定缺省文件扩展名，如果是空字符串 [""]，其缺省值为 *（所有的文件类型）。如果 ext 参数中包含了 dwg 文件类型，则该函数会在对话框中显示一个图形预览框。flags 参数是一个整数（按位编码），它控制对话框的行为。为了一次设置一个以上的条件，可以将几个位值加在一起生成一个 0 和 15 之间的标志值（包括 0 和 15）。

Flag value = 1 （位 0）

当希望提示用户输入一个要新创建的文件名时设置该位。如果输入文件名是为了打开一个已存在的文件，请不要设置该位。因为在后面这种情况下，如果用户输入的是一个不存在的文件名，对话框将会在对话框的底部显示一条错误信息。

如果设置了该位而用户选择了一个已存在的文件，AutoCAD 会显示一个警告框并让用户选择继续进行或取消该操作。

Flag value = 4 （位 2）

允许用户输入任意的文件扩展名，或者干脆不输入文件扩展名。

如果不设置该位，getfiled 函数将仅接受 ext 参数中指定的扩展名，并在用户没有输入扩展名时会自动给文件名加上该扩展名。

Flag value = 8 （位 3）

如果设置了该位而第 0 位没有设置，getfiled 函数将在库目录范围内搜索输入的文件名。如果在库目录下发现了该文件，它将截去路径部分而仅返回文件名（但如果在另一个路径下也发现了同名的文件，它就不会截去路径部分）。

如果没有设置该位，getfiled 函数返回包括全路径名的文件名。

如果要用对话框打开一个已存在的文件并想将其文件名存入图形文件（或其他数据库），就应该设置该位。

如果对话框从用户那里获得了一个文件名，getfiled 函数就以字符串形式返回该文件名；否则它返回 nil

。

下列代码调用 getfiled 函数显示 "Select a Lisp File" 对话框：

(getfiled "Select a Lisp File" "/acadr14/support/" "lsp" 8)

选择文件的对话框实例

getfiled 函数显示的对话框中包含了一个由指定类型（由扩展名指定）的文件名组成的一个表。用户可以用该对话框来浏览不同驱动器和目录中的文件，并选择一个现有文件或指定一个新文件名。

getint

暂停以等待用户输入一个整数并返回该整数

(getint [msg])

可选参数 msg 是用作提示信息的一个字符串。getint 函数返回该整数的值或 nil。

(setq num (getint))

(setq num (getint "Enter a number: "))

传给 getint 函数的数的范围是从 -32,768 到 +32,767。用户不能输入一个 AutoLISP 表达式来响应 getint

函数的请求。

请参见　getxxx 函数和 if 函数

getkword

暂停以等待用户输入一个关键字并返回该关键字

(getkword [msg])

在调用 getkword 函数之前必须先调用 initget 函数设置合法的关键字。参数 msg 是用作提示信息的一个字符串。该函数以字符串的形式返回与用户输入相匹配的关键字。如果用户输入的不是一个关键字，AutoCAD

会让用户再来一次。如果用户输入为空（即仅键入 ENTER 键），而 getkword 函数又允许空输

入（亦由 initget 函数设置），本函数返回 nil。如果在调用该函数之前，没有调用 initget 函数设置一个或多个关键字，该函数也返回 nil。

下例先调用 initget 函数创建一个关键字列表（"Yes" 和 "No"），并且不允许随后的 getkword 调用接受空输入（将 bits 设为 1），然后调用 getkword 函数。

(initget 1 "Yes No")
(setq x (getkword "Are you sure? (Yes or No) "))

这段代码提示用户输入关键字，并根据用户输入将变量 x 设为 Yes 或 No。如果用户的输入和关键字不匹配或用户给出的是空回答（即仅键入 ENTER 键），AutoCAD 将再次显示由参数 msg 提供的字符串提示用户重新输入。如果没有提供该参数，AutoCAD 将显示如下提示：

重试：

用户不能输入一个 AutoLISP 表达式来响应 getkword 函数的请求。

请参见 getxxx 函数和 if 函数

getorient

暂停以等待用户输入一个绝对的角度并返回该角度

(getorient [pt] [msg])

本函数与 getangle 函数类似，但与它稍有不同之处在于 getorient 函数返回的角度值不受系统变量 ANGBASE 和 ANGDIR 的影响。然而，用户输入的角度仍然基于 ANGBASE 和 ANGDIR 的当前值。

pt 参数是一个以当前 UCS 表示的二维基点，而 msg 参数则是用作提示信息的字符串。如果指定了 pt 参数，它将作为两个点中的第一个，允许用户指定另一个点来给 AutoLISP 指定角度。也可以提供三维基点，但量测角度总是在当前构造平面上进行。

getorient 函数以逆时针方向测量由用户指定的两点所确定的直线与零弧度方向（正东方，即时钟三点钟位置）之间的角度。和 getangle 函数一样，getorient 函数相对于当前构造平面以弧度的形式返回角度值。给 getorient 函数输入的角度是以系统变量 ANGDIR 和 ANGBASE 的当前值为基准的。然而，一旦该角度值被输入，对它的测量则是相对于零弧度（正东方）按逆时针方向进行的，而忽略系统变量 ANGDIR 和 ANGBASE 的设置。因此，如果已通过 UNITS 命令或通过设置系统变量 ANGDIR 和 ANGBASE，选择了一种不同的零度基准方向或角度增量方向，在使用 getorient 函数的过程中就必然会发生某种转换。

当需要一个旋转量（相对角度）时应使用 getangle 函数，而需要一个指定方向（绝对角度）时则应使用 getorient 函数。

用户不能输入一个 AutoLISP 表达式来响应 getorient 函数的请求。

请参见 getxxx 函数以及 getangle 和 if 函数

getpoint

暂停以让用户输入一个点并返回该点

(getpoint [pt] [msg])

pt 参数是一个以当前 UCS 表示的二维基点，而 msg 参数则是用作提示信息的字符串。用户既可以通过拾取点来指定点，又可以通过输入以当前单位格式表示的坐标来指定点。如果提供了 pt 参数，那么 AutoCAD 会从该点到当前十字光标位置画一条橡皮线。该函数的返回值是当前 UCS 中的一个三维点。

(setq p (getpoint))
(setq p (getpoint "Where? "))
(setq p (getpoint '(1.5 2.0) "Second point: "))

用户不能输入一个 AutoLISP 表达式来响应 getpoint 函数的请求。

请参见 getxxx 函数以及 getcorner 和 if 函数

getreal

暂停以让用户输入一个实数并返回该实数

(getreal [msg])

msg 参数是用作提示信息的字符串。

(setq val (getreal))
(setq val (getreal "Scale factor: "))

用户不能输入一个 AutoLISP 表达式来响应 getreal 函数的请求。

getstring

暂停以等待用户输入一个字符串并返回该字符串

(getstring [cr] [msg])

如果提供了 cr 参数且其值不为 nil，那么输入的字符串可以包括空格且必须以 ENTER 键结束。否则，输入的字符串以空格键或 ENTER 键结束。 msg 参数是用作提示信息的字符串。

如果输入的字符串长度超过 132 个字符，它仅返回前面的 132 个字符。如果输入的字符串中包含了斜杠 (\)，那么该斜杠会被转换成两个斜杠 (\\)。这样做是因为用户输入的字符串中可能包括其他函数要使用的文件路径名。

(setq s (getstring "What's your first name? "))

用 John 来响应，则将 s 设为 "John"

(setq s (getstring T "What's your full name? "))

用 John Doe 来响应，则将 s 设为 "John Doe"

(setq s (getstring "Enter filename: "))

用 \acad\mydwg 来响应，则将 s 设为 "\\acad\\mydwg"

用户不能输入一个 AutoLISP 表达式来响应 getstring 函数的请求。

请参见 如果应用程序要求用户输入若干个已知选项（关键字）之一，应使用 getkword 函数

getvar

获取一个 AutoCAD 系统变量的值

(getvar varname)

varname 参数是指定系统变量名的字符串，如果它不是一个有效的系统变量名，该函数返回 nil。

例如，假定系统变量 FILLETRAD 的当前值为 0.25 个单位，那么：

(getvar "FILLETRAD") 返回 0.25

关于当前版本 AutoCAD 系统变量的列表，请参见 AutoCAD 命令参考中的“系统变量”。

请参见 setvar 函数

graphscr

切换到 AutoCAD 的图形屏幕

(graphscr)

graphscr 函数总是返回 nil。

本函数的功能等价于 GRAPHSCR 命令或按切换屏幕的功能键。textscr 函数是 graphscr 函数的对应函数。

grclear

已废弃的函数，以前用于清除当前视口

(grclear)

为了和早期版本 AutoLISP 保持有限的兼容性，grclear 函数返回 nil。

grdraw

在当前视口中的两个点之间显示一个矢量

(grdraw from to color [highlight])

from 和 to 参数是以当前 UCS 表示的两个二维或三维点（由两个或三个实数组成的表），它们用于指定矢量的端点。grdraw 函数将该矢量以 color 参数指定的颜色显示，如果该参数为 -1，则指定为 XOR （异或）操作方式，即用当前绘图颜色与所经过位置处的当前颜色做异或操作，而不是简单地覆盖原有颜色。如果提供了可选参数 highlight（该参数是整数类型）且其值不为 0，那么就使用显示设备的醒目显示方法处理该矢量（通常是虚线）。如果省略了 highlight 参数或其值为 0，函数使用普通的显示方式。该函数总是返回 nil。

请参见 绘制多个矢量应使用函数 grvecs

grread

从 AutoCAD 的任何一种输入设备中读取数值

(grread [track] [allkeys [curtype]])

只有特殊用途的 AutoLISP 应用程序才需要调用本函数，AutoLISP 的大多数输入应使用各种 getxxx 函数来完成。

如果提供了 track 参数且其值不是 nil，那么当输入设备移动时，本函数能从定点设备中返回坐标。如果提供了 allkeys 参数， grread 函数所执行的功能取决于该参数的值。curtype 参数用于控制所显示光标的类型。allkeys 和 curtype 参数都是整数，其意义解释如下：

allkeys

可以通过将 allkeys 的几个位值相加来获得组合功能。

1（位 0） 返回“拖动模式”坐标。如果设置了该位，而且用户只是移动定点设备而没有按下按钮或键盘，grread 函数就返回一个表，其第一个成员是类型代码 5，第二个成员是当前定点设备（鼠标或数字化仪）的位置坐标 (X,Y)，这就是 AutoCAD 实现拖动的方法。

2（位 1） 返回所有的键值，包括功能键和光标键代码，并且在用户按下光标键时并不移动光标。

4（位 2） 使用 curtype 参数传来的值来控制光标的显示。

8（位 3） 在用户按下 CTRL+C 键时不显示相应错误信息。

curtype

只有当 allkeys 参数的第二个位为 1 时 curtype 参数才有效。该参数只控制在当前的 grread 函数调用时所显示的光标类型。

0 显示普通的十字光标。

1 不显示光标（即没有十字光标）。

2 显示对象选择光标

注意 AutoCAD 的后续版本中可能会定义其他的控制位。

grread 函数返回一个表，该表的第一个元素是说明输入类型的代码，第二个元素既可能是整数，又可能是点，这取决于输入的类型。其返回值列表如下：

grread 函数的返回值

第一个元素 第二个元素

值 输入类型 值 说明

2 键盘输入 各种值 字符代码

3 选取点 三维点 点坐标

4 通过定点设备选取屏幕菜单项或下拉式菜单项 0 到 999

1001 到 1999

2001 到 2999

3001 到 3999

... 如此直到

16001 到 16999 屏幕菜单项号

POP1 菜单项号

POP2 菜单项号

POP3 菜单项号

...

POP16 菜单项号

5 定点设备（仅当指定了跟踪标志时才返回） 三维点 拖动模式坐标

6 BUTTONS 菜单项 0 到 999

1001 到 1999

2001 到 2999

3001 到 3999 BUTTONS1 菜单按钮号

BUTTONS2 菜单按钮号

BUTTONS3 菜单按钮号

BUTTONS4 菜单按钮号

7 TABLET1 菜单项 0 到 32767 数字化仪菜单的单元号

8 TABLET2 菜单项 0 到 32767 数字化仪菜单的单元号

9 TABLET3 菜单项 0 到 32767 数字化仪菜单的单元号

10 TABLET4 菜单项 0 到 32767 数字化仪菜单的单元号

11 AUX 菜单项 0 到 999

1001 到 1999

2001 到 2999

3001 到 3999 AUX1 菜单按钮号

AUX2 菜单按钮号

AUX3 菜单按钮号

AUX4 菜单按钮号

12　定点设备按钮（紧跟在类型 6 或后面返回）　三维点　点的坐标

当一个 grread 函数调用处于激活状态时，按下 ESC 键便可以通过键盘中断 AutoLISP 程序的运行（除非指定的 allkeys 参数不允许这样做）。任何其他的输入都被直接传给 grread 函数，这使得应用程序能控制所有的输入设备。

如果用户在屏幕菜单项或下拉式菜单项上按下定点设备按钮，grread 函数返回一个类型为 6 或 11 的代码，但在随后的调用中，它并不返回类型代码 12。因为只有在屏幕的图形区中按下定点设备按钮，类型代码 12 才会跟随在类型代码 6 或 11 之后返回。

在试图用定点设备按钮或辅助按钮执行另一操作之前，将类型代码 12 的数据从缓冲区中清除，是非常重要的。为了做到这一点，可以执行如下形式的一个嵌套的 grread 函数调用：

(setq code_12 (grread (setq code (grread))))

上述代码就象从输入流设备上获取输入一样，获取类型代码 12 的值表。

注意　由于由 AutoCAD 支持的各种平台在输入处理上的区别，grread 函数可能返回预想不到的结果。

在缺省定点设备是系统鼠标的平台上，返回的类型代码是 11 而不是 6。

在 Macintosh 平台上，弹出式菜单返回代码 11 而不是 4。另外，在 Macintosh 平台上，在当前视口中双击将返回代码 11（而不是 6），并且后跟代码 5 的坐标对。相反，在当前视口以外的其他视口双击将返回代码 3 的坐标对，并后跟一个代码 11。

grtext

将文本写到状态行或屏幕菜单区

(grtext [box text [highlight]])

box 参数是一个整数，它指定待写文本的位置。text 参数是要写到状态行或屏幕菜单区的字符串。如果该字符串太长，则它将被截断以使指定的区域能容纳下它。highlight 参数是一个整数，用于标识是否突出显示屏幕菜单项。这些参数的值会根据书写文本的屏幕区域的不同而变化。

该函数将所提供的文本显示在菜单区，它并不改变基础菜单项。不带参数调用 grtext 函数将使所有文本区恢复成缺省值。如果调用成功，grtext 函数返回由参数 text 指定的字符串，否则它返回 nil。

屏幕菜单区

将 box 参数设为正数或 0 可以指定一个屏幕菜单位置，该参数的有效取值范围是从 0 到最大的屏幕菜单行号减 1。通过系统变量 SCREENBOXES 可以获得最大的屏幕菜单行号。如果在调用函数时提供的 highlight 参数是一个正整数，本函数将突出显示指定位置的文本。突出显示一个菜单项，将自动使其他已突出显示的菜单项解除突出显示。如果该参数为 0，菜单项会被解除突出显示。如果该参数为负，它将被忽略。在某些平台上，文本必须先用不带 highlight 参数的 grtext 函数写到屏幕菜单区，然后再将它突出显示。仅当光标不在该菜单区域时，才能突出显示该菜单区域的菜单项。

状态行区域

如果调用 grtext 函数时将 box 参数的值取为 -1，则该函数会将文本写入到模式状态行区域。模式状态行的长度依显示设备的不同而不同（大多数显示设备的模式状态行至少允许容纳 40 个字符）。下列代码用 DIESEL 表达式 $(linelen) 来获取模式状态行的长度：

(setq modelen (menucmd "M=$(linelen)"))

如果将 box 参数设为 -2，本函数就将文本写入到坐标状态行区域。如果坐标状态行的坐标显示跟踪开关是打开的，那么，在定点设备传过来新的坐标集时，它将会覆盖由本函数写入到该区域的文本。box

参数为 -1 或 -2 时，忽略 highlight 参数。

grvecs

在图形屏幕上绘制多个矢量

(grvecs vlist [trans])

vlist 参数是一个矢量表，它由一系列的可选颜色代码（整数）和两个点表组成。trans 参数是一个转换矩阵，可以用它来改变定义在矢量表中的矢量位置或比例。该转换矩阵是由四个子表组成的一个表，每个子表由四个实数组成。

参数 vlist 的格式如下：

([color1] from1 to1 [color2] from2 to2 ...)

颜色值可以作用于随后的所有矢量，直到 vlist 指定另一种颜色为止。AutoCAD 颜色代码的取

值范围是 0 到 255。如果颜色代码值大于 255，则后续矢量会以 XOR （异或）方式显示，也就是以取补颜色生成被该矢量覆盖的对象。而且当矢量被覆盖时，它自身会消失。如果颜色代码小于 0，将醒目显示后续矢量。醒目显示的方式取决于显示设备，大多数显示设备用虚线表示突出显示，但某些显示设备是用一种有区别的颜色来表示突出显示。

两个点表 from 和 to 用来指定矢量的两个端点，它们是以当前 UCS 表示的。这些点既可以是二维点，也可以是三维点。必须把矢量的两个端点作为一对点（即两个连续的点表）来传送，否则，对 grvecs

函数的调用将会失败。

AutoCAD 会将矢量进行裁剪以适应屏幕。如果 grvecs 函数调用成功，它返回 nil。

下列代码在图形屏幕上显示五条垂直矢量，而且每条颜色各不相同：

```
(grvecs '(1 (1 2)(1 5)          从 (1,2) 到 (1,5) 画一条红线
           2 (2 2)(2 5)         从 (2,2) 到 (2,5) 画一条黄线
           3 (3 2)(3 5)         从 (3,2) 到 (3,5) 画一条绿线
           4 (4 2)(4 5)         从 (4,2) 到 (4,5) 画一条青线
           5 (5 2)(5 5)         从 (5,2) 到 (5,5) 画一条蓝线
) )
```

以下矩阵表示以 1.0 的比例作统一的比例缩放（实际上是不变）并作偏移量为 (5.0,5.0,0.0) 的平移。如果将该矩阵作用到上述矢量上，它们将被平移 (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)
)
```

请参见 关于转换矩阵的详细信息，请参见 nentselp 函数

H

handent

根据对象（图元）的句柄返回它的对象（图元）名

(handent handle)

根据给定的图元句柄字符串参数 handle，handent 函数返回在当前编辑会话期间与该图元句柄相关联的图元的图元名。handent 函数既可以返回图形图元的名称，又可以返回非图形图元的名称。

如果传给 handent 函数一个无效句柄，或没有被当前图形的任何图元使用的句柄，它会返回 nil。该函数可以返回在当前编辑会话期间被删除的图元的名称，然后调用 entdel 函数恢复它。

在不同的编辑会话期间同一个图元的图元名可能会不同，但图元的句柄却保持不变。在某个编辑会话期间，代码

(handent "5A2") 可能返回 <图元名: 60004722>

在另一个编辑会话期间，对同一个图形，同样地执行上述代码可能会返回一个不同的图元名。一旦获取到图元名，就可以用它来调用与图元有关的函数来处理该图元。

help

调用帮助工具

(help [helpfile [topic [command]]])

helpfile 参数是一个用于指定帮助文件的字符串。如果指定的是一个 AutoCAD 的帮助文件 (.ahp)，help

函数将使用 AutoCAD 的 Help 浏览器来显示该文件。如果指定的是一个 Windows 的帮助文件 (.hlp)，help

函数将使用 WinHelp 程序来显示该文件。如果 helpfile 参数是空字符串 ("")，或省略了该参数，AutoCAD

将使用缺省的 AutoCAD 帮助文件。参数 topic 是一个关键字，用于指定帮助工具最初显示的主题。如果

topic 参数是空字符串 ("")，帮助工具将显示帮助文件的绪论部分。command 参数是一个指定帮助窗口初始状态的字符串，其取值如下表所列：

command 参数的取值

字符串 说明

HELP_CONTENTS　显示帮助文件中的第一个主题

HELP_HELPONHELP 显示的主题为“如何使用帮助的帮助”

HELP_PARTIALKEY 显示搜索对话框，将 topic 参数传过来的字符串作为初始搜索文本

如果指定的是 Windows 帮助文件，command 参数也可以是由 WinHelp() 函数的 fuCommand 参数所使用的一个字符串，WinHelp() 函数的定义可参见 Microsoft Windows SDK 中的 WinHelp API。

helpfile 参数中并不强制性地要求有文件扩展名。如果提供了扩展名，AutoCAD 仅搜索该文件；如果没有提供扩展名，将采用如下搜索规则：先加 .hlp 扩展名，如搜索不到再加 .ahp 扩展名搜索。

help 函数返回给应用程序的唯一错误条件是由参数 helpfile 指定的文件不存在。所有其他的错误条件都通过对话框报告给用户。如果调用成功，help 函数返回 helpfile 字符串，否则它返回 nil。如果不带任何参数调用 help 函数，那么成功时它返回空字符串 (""), 否则它返回 nil。

下列代码调用 help 函数显示帮助文件 achelp.ahp 中 MYCOMMAND 主题的信息：

(help "achelp.ahp" "mycommand")

请参见　关于如何创建 AutoCAD 帮助文件的详细信息，请参见自定义联机文档，而 setfunhelp 函数用于将上下文相关帮助（用户按下 F1 键时得到的帮助）与一个用户自定义命令联系起来。

if

根据对条件的判断，来对不同的表达式求值

(if testexpr thenexpr [elseexpr])

如果 testexpr 的值不是 nil，它对 thenexpr 求值，否则它对 elseexpr 求值。if 函数返回所选择的表达式的值。如果没有　elseexpr 表达式且 testexpr 为 nil，if 函数返回 nil。

(if (= 1 3) "YES!!" "no.")　　返回 "no."

(if (= 2 (+ 1 1)) "YES!!")　　返回 "YES!!"

(if (= 2 (+ 3 4)) "YES!!")　　返回 nil

请参见　progn 函数

initdia

强制显示下一个命令的对话框

(initdia [dialogflag])

dialogflag 参数是一个整数，如果没有提供该参数或提供的该参数不为 0，下一个（也仅有下一个）命令将使用该命令的对话框而不是命令提示行。当前能使函数 initdia 有效的命令只有 BHATCH、LAYER、IMAGE、

IMAGEADJUST、LINETYPE、MTEXT、PLOT、STYLE 和 TOOLBAR。

如果 dialogflag 参数为 0，那么该调用将清除以前对该函数的调用，恢复显示命令行界面的缺省状态。

本函数没有返回值。

外部定义函数　acadapp ARX 应用程序

initget

为随后的用户输入函数调用创建关键字

(initget [bits] [string])

能够接受关键字输入的函数有 getint、getreal、getdist、getangle、getorient、getpoint、

getcorner、getkword、entsel、nentsel 和 nentselp，但 getstring 函数是唯一不能接受关键字输入的用户输入函数。

bits 参数是一个按位编码的整数，用于控制是否允许某些类型的用户输入。而 string 参数定义一个关键字表。在随后调用用户输入函数时，如果用户输入的不是相应类型（例如与 getpoint 函数相对应的类型是一个点），该函数将检索关键字表来确定用户是否键入了一个关键字。如果用户的输入和表中的一个关键字相匹配，函数将以字符串的形式返回该关键字。应用程序可以对返回的关键字进行检测，并对每一个关键字执行相应动作。如果用户的输入不是相应类型且和表中任何一个关键字都不匹配，AutoCAD

将要求用户再次输入。initget 函数的位编码值与关键字表仅对紧随其后的那个用户输入函数有效。

initget 函数总是返回 nil。

如果用 initget 函数设置了一个控制位，而该控制位对应用程序随后调用的那个用户输入函数来说没有意义，则将忽略该控制位。按位编码的控制位参数 bits 中的控制位可以任意组合（即把各位加起来），其组合值范围是从 0 到 225。如果没有指定参数 bits，则假定它是 0，即没有任何控制条件。如果用户的输入不满足一个或多个指定条件（例如在不允许输入零值时输入了零值），AutoCAD 将显示一条信息，并要求用户再次输入。

initget 函数设置的输入选项

位值　　说明

1（位 0）　禁止用户仅按 ENTER 键来响应输入请求。

2（位 1）　禁止用户输入零值来响应输入请求。

4（位 2）　禁止用户输入负值来响应输入请求。

8（位 3）　允许用户在当前图形的极限之外输入一个点，即使 AutoCAD 的系统变量 LIMCHECK 当前被设置为开 (ON)， 本条件也照样对随后调用的用户输入函数有效。

16（位 4）　（目前尚未使用）

32（位 5）　画橡皮线或拉伸方框时用虚线。对那些可以由用户在图形屏幕上通过选择位置来指定一个点的那些函数，设置该控制位将使橡皮线和拉伸方框显示为虚线而不是实线（某些显示驱动程序用一种醒目颜色的线来代替虚线）。如果系统变量 POPUPS 设置为 0，AutoCAD 将忽略该控制位。

64（位 6）　在使用 getdist 函数时，本控制位可禁止输入 Z 坐标，以使应用程序确保该函数返回的是二维距离。

128（位 7）　在尊重任何其他的控制位和所列出的关键字的情况下，允许任意的输入，就好象它是一个关键字一样。该位的优先权高于位 0：如果同时设置了位 7 和位 0，那么用户仅键入 ENTER 键时，将返回空字符串。

注意　AutoLISP 的后续版本可能会使用 initget 的其他控制位，所以要避免设置本表中没有列出的位。

对一个特定的 getxxx 函数来说，只有特定的控制位才有意义，下表对此作了全面说明。

用户输入函数和可用的控制位

函数　　关键字
是否可用　　控制位
之值

	非空(1)	非零(2)	非负(4)	不检查极限(8)	使用虚线(32)	二维距离(64)	任意输入(128)
getint							
getreal							
getdist							
getangle							
getorient							
getpoint							
getcorner							
getkword							
entsel							
nentsel							
nentselp							

关键字说明

string 参数按如下规则进行解释：

每个关键字与随后的关键字之间用一个或多个空格分隔。例如， "Width Height Depth" 定义了三个关键字

关键字只能由字母、数字和连字符 (-) 组成。

关键字有如下两种缩写办法：

关键字的必须部分用大写字母表示，而其余部分用小写字母表示。大写的缩写部分可以位于关键

字的任何位置（例如， "LType"、"eXit" 或 "toP"）。

整个关键字用大写字母表示，其后紧跟一个逗号，然后再跟随其必须部分（例如，"LTYPE,LT"）。这种情况下，关键字的必须部分必须包含关键字的第一个字符，这意味着 "EXIT,X" 是无效的。

"LType" 和 "LTYPE,LT" 这两种关键字缩写方式是等价的。如果用户键入 LT（无论是大写还是小写），都可以被识别为这个关键字。用户还可以输入关键字必须部分之后的字符，这样它们就不会与缩写规则相冲突了。在本例中，用户可以输入 LTY 或 LTYP，但只输入 L 是不够的。

如果 string 参数完全以大写或小写字符给出，其后没有逗号，也没有跟随必须部分，则只有当用户完整输入这个关键字时 AutoCAD 才能识别。

initget 函数支持本地化的关键字。关键字字符串的下列语法说明允许输入本地化的关键字，而返回与语言无关的关键字：

"local1 local2 localn _indep1 indep2 indepn"

在这里，从 local1 到 localn 是本地化的关键字，而从 indep1 到 indepn 是与语言无关的关键字。

本地化关键字和与语言无关的关键字的数目必须相同，而且第一个与语言无关的关键字的前面必须有一个下划线，如下例所示：

(initget "Abc Def _Ghi Jkl")
(getkword "\nEnter an option (Abc/Def): ")
键入 A 返回 Ghi，而键入 _J 则返回 Jkl。
请参见 用户输入函数的条件控制

inters

求两条直线的交点坐标

(inters pt1 pt2 pt3 pt4 [onseg])

pt1 和 pt2 参数是第一条直线的端点，而 pt3 和 pt4 参数是第二条直线的端点。如果提供了 onseg 参数且其值为 nil，则由四个点定义的两条线被认为是无限长的，inters 函数返回交点坐标，即使这个交点不在其中的一条线（或者两条线）的端点范围之内。如果省略 onseg 参数或者其值不是 nil，则交点必须同时位于两条线上，否则 inters 函数将返回 nil。如果这两条线没有交点， inters 函数返回 nil。

所有的点都以当前 UCS 表示，如果所提供的四个点都是三维点，inters 函数检查三维交点。只要所提供的点中有一个是二维点，inters 函数就将这两条线投影到当前构造平面上，仅检查它的二维交点。

(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) 返回 nil
(inters a b c d T) 返回 nil
(inters a b c d nil) 返回 (4.0 4.0)

itoa

将整数转换成字符串，并返回转换结果

(itoa int)

int 参数用于指定一个整数。

(itoa 33) 返回 "33"
(itoa -17) 返回 "-17"

L

lambda

定义一个无名函数

(lambda arguments expr...)

在经常使用某一表达式，而又觉得把它定义成一个新函数开销太大时可使用 lambda 函数。lambda 将定义的函数放在要使用它的位置，还可以使程序员的意图表达得更清楚。 lambda 函数返回它最后的一个 expr 的值，并且它常与 apply 和（或）mapcar 函数连用，以便对表中的元素执行某个操作。

```
(apply '(lambda (x y z)
          (* x (- y z))
```

```
        )
        '(5 20 14)
)                   返回 30
```

而

```
(setq counter 0)
(mapcar '(lambda (x)
           (setq counter (1+ counter))
           (* x 5)
         )
         '(2 4 -6 10.2)
)                   返回 (10 20 -30 51.0)
```

last

返回表中的最后那个元素

(last lst)

(last '(a b c d e))　　返回 E

(last '(a b c (d e)))　　返回 (D E)

如上所示，last 函数可以返回原子或表。

注意　乍看上去，用 last 函数去获取点的 Y 坐标，似乎是一种很好的方法。这对于二维点（由两个实数组成的表）来说确实如此，但对于三维点，用 last 函数返回的却是 Z 坐标。为了使函数在处理二维点和三维点时都能正确工作，建议调用 cadr 函数去获取 Y 坐标而调用 caddr 函数去获取 Z 坐标。

length

以整数形式返回表中元素的数目

(length lst)

(length '(a b c d))　　返回 4

(length '(a b (c d)))　　返回 3

(length '())　　返回 0

list

将任意数目的表达式组合成一个表

(list expr...)

(list 'a 'b 'c)　　返回 (A B C)

(list 'a '(b c) 'd)　　返回 (A (B C) D)

(list 3.9 6.7)　　返回 (3.9 6.7)

在 AutoLISP 中，本函数常用于定义二维或三维点变量（由两个或三个实数组成的表）。

如果表中没有变量和未定义的项，还可以用 quote 函数定义一个表，这和调用 list 函数效果相同。单引号 (') 被定义为 quote 函数：

'(3.9 6.7)　　等价于　(list 3.9 6.7)

这对创建关联表和定义点来说非常有用。

请参见　quote 函数

listp

检查某个项是否是表

(listp item)

如果 item 是表，该函数返回 T；否则它返回 nil。

(listp '(a b c))　　返回 T

(listp 'a)　　返回 nil

(listp 4.343)　　返回 nil

注意　由于 nil 既是原子又是表，所以传给 listp 函数的参数为 nil 时，它返回 T。

(listp nil)　　返回 T

load_dialog

加载一个 DCL 文件

(load_dialog dclfile)

dclfile 参数是用于指定要加载的 DCL 文件的字符串。如果 dclfile 参数没有指定文件扩展名，则假设为 .dcl。如果成功，本函数返回一个正整数 (dcl_id)，如果无法打开指定文件，它返回一个负

整数。该 dcl_id 在随后调用 new_dialog 和 unload_dialog 函数时可以用作被加载的 DCL 文件的句柄。

load_dialog 函数按照 AutoCAD 的库搜索路径来搜索指定的 DCL 文件。

本函数是 unload_dialog 函数的配套函数。应用程序可以通过多次调用 load_dialog 函数来加载多个 DCL 文件。

load

对一个文件中的 AutoLISP 表达式求值

(load filename [onfailure])

filename 参数是表示文件名的一个字符串。如果 filename 参数没有指定文件扩展名，则假设是 .lsp。如果 load 函数失败，它返回 onfailure 参数的值。然而，如果没有提供 onfailure 参数，load 函数的调用失败将引发一个 AutoLISP 错误。如果调用成功，load 函数返回文件中最后一个表达式的值。

filename 参数可以包括一个目录前缀，如 "/function/test1"（在 DOS 平台上，还可以有驱动器字符），一个斜杠 (/) 或两个反斜杠 (\\) 都是有效的目录分隔符。如果 filename 字符串中没有包含目录前缀，

load 函数将在 AutoCAD 的库搜索路径中搜索指定的文件。如果在这个路径中的某个地方找到了指定文件，

load 函数就会加载它。

如果 onfailure 参数是一个有效的 AutoLISP 函数，它会被求值。在大多数情况下，onfailure 参数应该是一个字符串或原子。这样就允许 AutoLISP 应用程序在调用 load 函数时如发生故障可以执行某个动作。

例如，假设文件 /fred/test1.lsp 包括：

```
(defun MY-FUNC1 (x)
        ...函数体...
)
(defun MY-FUNC2 (x)
        ...函数体...
```

而文件 test2.lsp 并不存在，那么：

```
(load "/fred/test1")          返回 MY-FUNC2
(load "\\fred\\test1")        返回 MY-FUNC2
(load "/fred/test1" "bad")    返回 MY-FUNC2
(load "test2" "bad")          返回 "bad"
(load "test2")                返回一个 AutoLISP 错误
```

load 函数可以用在另一个 AutoLISP 函数中，甚至可以在被它调用的文件中递归地调用 load 函数。

请参见 defun 函数和符号和函数处理

log

返回一个实数的自然对数

(log num)

```
(log 4.5)          返回 1.50408
(log 1.22)         返回 0.198851
```

logand

返回一个整数表中的各数按位逻辑与 (AND) 的结果

(logand int int...)

```
(logand 7 15 3)    返回 3
(logand 2 3 15)    返回 2
(logand 8 3 4)     返回 0
```

logior

返回一个整数表中的各数按位逻辑或 (OR) 的结果

(logior int int...)

```
(logior 1 2 4)     返回 7
(logior 9 3)       返回 11
```

lsh

返回某整数作指定次逻辑移位后的结果

(lsh int numbits)

如果 numbits 是正数，则数 int 向左移位 numbits 次；如果 numbits 是负数，则数 int 向右移位 numbits 次。在这两种情况下，移入位为 0，移出位丢弃。如果移位运算之后符号位（位号为 31）是 0，则本函数返回的值是正数；否则它返回的值是负数。

(lsh 2 1)　　返回 4

(lsh 2 -1)　　返回 1

(lsh 40 2)　　返回 160

M

mapcar

将作为本函数参数的一个或多个表的各个元素提供给指定函数进行求值，并将由求值结果构成的表返回

(mapcar function list1... listn)

表的数目必须和函数 function 所要求的参数数目相匹配。

```
(setq a 10 b 20 c 30)
(mapcar '1+ (list a b c))        返回 (11 21 31)
```

等价于

```
(1+ a)
(1+ b)
(1+ c)
```

唯一不同的是函数 mapcar 返回的是由结果组成的表。

lambda 函数可以指定一个由 mapcar 函数执行的无名函数。当某些函数参数是常量或由其他方法提供时，这种办法很有用。

```
(mapcar '(lambda (x)
            (+ x 3)
         )
         '(10 20 30)
)                    返回 (13 23 33)
```

max

返回给定的各数中的最大者

(max number number...)

(max 4.07 -144)　　返回 4.07

(max -88 19 5 2)　　返回 19

(max 2.1 4 8)　　返回 8.0

mem

显示 AutoLISP 内存的当前状态

(mem)

显示 AutoLISP 内存的当前状态并返回 nil。它显示如下信息：

节点 是指已分配的节点总数，它等于节点段的节点数目和段数的乘积。

可用节点 是指当前自由存储表中的节点数，它是无用数据收集操作的结果。

段 是指已分配的节点段数。

分配 是指当前节点段的大小（即段中的节点数目）。

收集 是指收集的无用数据的数目，无论是自动收集或强制收集均被考虑在内。

请参见　第十五章“内存管理”

member

搜索一个表中是否包含某表达式，并从该表达式的第一次出现处返回表的其余部分

(member expr lst)

如果在表 lst 中没有出现表达式 expr，本函数返回 nil。

(member 'c '(a b c d e))　　返回 (C D E)

(member 'q '(a b c d e))　　返回 nil

menucmd

发出菜单命令，或设置并检索菜单项状态

(menucmd string)

string 参数是指定菜单区和要赋给这个菜单区的值的一个字符串，该参数的形式如下所示：

"menu_area=value"

下表列出了 menu_area 的允许值，它们与菜单文件中的子菜单引用格式完全相同，详细信息请参见引用下拉菜单和光标菜单。

Menu_area 字符串的取值

Menu_area 字符串菜单区域

B1-B4 按钮菜单 1 到 4

A1-A4 辅助菜单 1 到 4

P0-P16 下拉式菜单 0 到 16

I 图像控件菜单

S 屏幕菜单

T1-T4 数字化仪菜单 1 到 4

M DIESEL 字符串表达式

Gmenugroup.nametag 指定一个菜单组和名称标志

value 参数指定要赋给 menu_area 的值。

menucmd 函数可以在 AutoCAD 菜单的各个子页间进行切换，该函数还可以强制显示菜单。这使得 AutoLISP 程序可以使用图像控件菜单和显示其他菜单让用户选择。 AutoLISP 程序还可以启用或禁用一些菜单项，或给菜单项加上标记。

下列代码将显示图像控件菜单 MOREICONS：

```
(menucmd "I=moreicons")    加载图像控件菜单 MOREICONS
(menucmd "I=*")            显示该菜单
```

下列代码检查下拉式菜单区 POP11 中的第三个菜单项的状态，如果当前它是可启用的，menucmd 函数将禁用它。

```
(setq s (menucmd "P11.3=?"))      获取菜单项的状态
(if (= s "")                      如果该状态是空字符串，
   (menucmd "P11.3=~")            禁用该菜单项
)
```

上述代码不是在任何情况下都正确，因为菜单项除了可处于启用或禁用状态外，还可以有标记。代码 (menucmd "P11.3=?") 可能会返回 "!."，这说明该菜单项当前已打上了一个复选标记。这种情况下，上述代码会认为该菜单项已经被禁用而不禁用它就继续执行。如果该代码中包含了对 wcmatch 函数的调用，它还可以检查波浪号 (~) 是否在状态字符串中出现，从而执行相应的动作。

menucmd 函数还允许 AutoLISP 程序发挥 DIESEL 字符串表达式语言的优越性。某些事情用 DIESEL 来完成可能比用等价的 AutoLISP 代码来完成更容易。下列代码返回包含了当前日期和时间的一个字符串：

```
(menucmd "M=$(edtime,$(getvar,date),DDDD\",\" D MONTH YYYY)")
                    返回 "Sunday, 16 July 1995"
```

请参见 关于如何利用 AutoLISP 存取菜单标签状态的详细信息，请参见第六章“编程接口”，关于如何使用 DIESEL 的详细信息，请参见第五章“状态行配置和 DIESEL -- 字符串表达式语言”

menugroup

检查是否加载了某菜单组

(menugroup groupname)

groupname 参数是用于指定菜单组名称的一个字符串。如果 groupname 与某个已加载的菜单组相匹配，本函数返回 groupname 字符串，否则它返回 nil。

min

返回给定的各数中的最小者

(min number number...)

```
(min 683 -10.0)          返回 -10.0
(min 73 2 48 5)          返回 2
(min 2 4 6.7)            返回 2.0
```

minusp

检查某个数是否是负数

(minusp num)

如果 number 是负数，该函数返回 T，否则返回 nil。

(minusp -1)　返回 T

(minusp -4.293)　返回 T

(minusp 830.2)　返回 nil

mode_tile

设置某个对话框控件的状态

(mode_tile key mode)

key 参数是用于指定某个控件的一个字符串，它是区分大小写的。mode 参数是一个整数，其含义如下表所述：

Mode 参数值

值　说明

0　启用该控件

1　禁用该控件

2　将焦点设置到该控件

3　选择编辑框中的内容

4　切换是否突出显示图像的开关

N

namedobjdict

返回当前图形的命名对象词典的图元名，它是所有非图形对象的根

(namedobjdict)

通过调用词典访问函数并使用本函数返回的图元名，应用程序可以访问图形中的非图形对象。

nentsel

提示用户通过指定一个点来选择一个对象（图元），从而可以存取包含在复杂对象内的定义数据

(nentsel [msg])

msg 参数是用作提示信息的一个字符串。如果省略该参数，AutoCAD 将给出“选择对象”提示。

nentsel 函数提示用户选择一个对象，它忽略当前的对象捕捉模式（除非用户专门指定了它）。为了给命令行提供更多的支持，nentsel 函数支持在前面用 initget 函数定义的关键字。

当用户选择的不是复杂对象（如多段线或块等）时，nentsel 函数返回的信息与 entsel 函数相同。然而，如果所选对象是多段线，nentsel 函数返回一个表，该表包含了子图元（顶点）名和拾取点坐标。这与 entsel 函数返回的表类似，不同的是 nentsel 函数返回的是多段线的所选顶点名，而不是多段线的头部图元名。nentsel 函数总是返回所选取的多段线的那一小段的起始顶点，例如，拾取某多段线的第三段，返回的是该多段线的第三个顶点。nentsel 函数从不返回多段线的 SEQEND 图元。

注意　在图形数据库中轻量多段线被定义为一个简单图元，它不包含子图元。

选取一个块引用中的一个属性时，nentsel 返回的是该属性的属性名和拾取点坐标。当选取的是块引用的一个部件而不是属性时，nentsel 函数返回一个包含四个元素的表。

拾取了块中的一个对象而返回的表的第一个元素是被选中图元的图元名。第二个元素是一个表，该表包含了用户拾取该对象时指定点的坐标。

第三个元素称为模型坐标系到世界坐标系的转换矩阵，它是一个包含四个子表的表，四个子表中的每一个都包含了一个坐标集。该矩阵可以用来将图元定义数据中的点从称之为模型坐标系 (MCS) 的内部坐标系转换为世界坐标系 (WCS)。MCS 的原点，是包含所选图元的块的插入点，而 MCS 轴的方向则由创建该块时 UCS 的方向决定。

第四个元素是一个表，它包含了用户所选对象所在的块的图元名。若所选的对象内含在一个嵌套块中（即块中块），则该表会包括内含该对象的所有块的图元名。该表的排列顺序是从最内层的块开始，向外层推，直到遇到插入图形中的最外层块才结束。

(<图元名: ename1>　　图元名

(Px Py Pz)　　拾取点

((X0 Y0 Z0)　　模型坐标系到世界坐标系的转换矩阵

(X1 Y1 Z1)

(X2 Y2 Z2)

(X3 Y3 Z3)

)
(<图元名:ename2>　　包含了所选图元的最深嵌套块
.　　的图元名
.
.
<图元名:enamen>)　　包含了所选图元的最外层块
)　　的图元名

一旦获得了图元名和模型坐标系到世界坐标系的转换矩阵，就可以把图元定义数据中的点从 MCS 转换到 WCS。调用 entget 和 assoc 函数，可以从图元名获得以 MCS 坐标表示的定义点。由 nentsel 函数返回的从 MCS 到 WCS 的转换矩阵与 nentselp 函数返回的转换矩阵用途相同，但是它是一个 4 x 3 的矩阵（它是作为四个点的一个数组来传递的），而且约定它的每一行（而不是每一列）表示一个点。这种坐标转换可以用如下的矩阵乘法来描述：

由上式可以推导出新坐标的计算公式如下：

上述公式中的 Mij (0　i, j　2)，是 MCS 到 WCS 的转换矩阵的系数， X、Y、Z 则是以 MCS 坐标表示的图元定义数据中的点，而 X'、Y'、Z' 是转换后的以 WCS 坐标表示的图元定义数据中的点。

注意　nentsel 是唯一使用上述这种类型转换矩阵的 AutoLISP 函数，而 nentselp 函数返回的矩阵类似于其他 AutoLISP 函数和 ADSRX 函数使用的矩阵。

请参见　图元名称函数以及 entsel 和 if 函数

nentselp

在没有用户输入的情况下，本函数提供与 nentsel 函数类似的功能

(nentselp [msg] [pt])

除了可选参数 msg 外，nentselp 函数还可以接受作为可选参数的一个选择点 pt，这就允许在没有用户输入的情况下实现对象选择。函数返回一个 4 x 4 的转换矩阵，其定义如下：

该矩阵的前三列指定缩放比例和旋转角度，第四列是一个转换矢量。

使用这种类型的矩阵的函数将点看成是一个四维的列矢量。点用齐次坐标表示，点矢量的第四个元素是一个比例因子，通常它的值被设置为 1.0。该矩阵的最后一行，也就是 [M30 M31 M32 M33]，有标准值 [0 0 0 1]，使用这种矩阵格式的函数通常会忽略它。按照这种约定，对一个点作转换，实际上是作如下的矩阵乘法运算：

这实际上给出了点的单个坐标值的计算公式如下：

正如上述公式所表明的，比例因子和矩阵的最后一行对计算没有影响，可以忽略它们。

new_dialog

开始一个新的对话框并显示该对话框，而且能指定一个缺省动作

(new_dialog dlgname dcl_id [action [screen-pt]])

dlgname 参数是用于指定对话框的一个字符串，而 dcl_id 参数用于指定 DCL 文件（必须先调用 load_dialog 函数获取其值）。

如果指定了 screen-pt 参数，就必须指定 action 参数。action 参数是一个字符串，它包含了用来表示缺省动作的一个 AutoLISP 表达式。如果不想定义缺省动作，可以传给 action 参数一个空字符串 ("")。

screen-pt 参数是一个二维点表，它用于指定对话框在屏幕上的位置的 X、Y 坐标。该点通常指定的是对话框的左上角，但它还与平台有关，其值通常用系统单位来表示。如果将其指定为 '(-1 -1)，那么打开对话框时它会显示在缺省位置上（即 AutoCAD 的图形屏幕的中心位置）。

如果 new_dialog 函数调用成功，它返回 T；否则它返回 nil。

应用程序在调用 start_dialog 函数之前，必须先调用 new_dialog 函数。所有的对话框初始化工作，如设置控件值、创建图像、创建列表框的表和将各个动作与特定的控件联系起来（用 action_tile 函数完成）等，都必须在调用 new_dialog 函数之后和调用 start_dialog 函数之前完成。

当用户拾取了某个激活的控件，而该控件既没有通过调用 action_tile 函数显式地为它指定一个动作或回调函数，也没有在 DCL 文件中为它指定动作，那么，由 new_dialog 函数指定的缺省动作就会被求值。

注意　在应用程序中应该总是检查由 new_dialog 函数返回的状态，当 new_dialog 函数调用失败时调用

start_dialog 函数可能会导致无法预料的后果。

not

检查一个项的求值结果是否为 nil

(not item)

如果 item 的求值结果为 nil，本函数返回 T；否则它返回 nil。

(setq a 123 b "string" c nil)

(not a) 返回 nil

(not b) 返回 nil

(not c) 返回 T

(not '()) 返回 T

通常用法是，null 函数用于测试表，而 not 函数则用于测试其他数据类型和某些类型的控制函数。

nth

返回表中的第 n 个元素

(nth n lst)

n 参数是表中要返回元素的序号（表中元素的编号从 0 开始），如果 n 大于表中最后那个元素的序号，nth

函数返回 nil。

(nth 3 '(a b c d e)) 返回 D

(nth 0 '(a b c d e)) 返回 A

(nth 5 '(a b c d e)) 返回 nil

null

检查某个项的值是否被设置为 nil

(null item)

如果 item 被设置为 nil，本函数返回 T；否则它返回 nil。

(setq a 123 b "string" c nil)

(null a) 返回 nil

(null b) 返回 nil

(null c) 返回 T

(null '()) 返回 T

numberp

检查某个项是否是实数或整数

(numberp item)

如果 item 是实数或整数，本函数返回 T；否则它返回 nil。

(setq a 123 b 'a)

(numberp 4) 返回 T

(numberp 3.8348) 返回 T

(numberp "Howdy") 返回 nil

(numberp a) 返回 T

(numberp b) 返回 nil

(numberp (eval b)) 返回 T

O

open

打开一个文件，供其他 AutoLISP I/O 函数访问

(open filename mode)

filename 参数是一个字符串，它指定要打开的文件的文件名和扩展名。 mode 参数是一个读/写标志，它必须是一个包含了单个小写字符的字符串，下表列出了 mode 参数的有效取值。

open 函数的 Mode 参数的有效取值

打开方式 说明

"r" 打开文件用于读操作，若 filename 不存在，open 返回 nil。

"w" 打开文件用于写操作，若 filename 不存在，则创建一个新文件并打开它；若 filename 已存在，则覆盖它已有的数据。

"a" 打开文件用于追加操作，若 filename 不存在，则创建一个新文件并打开它；若 filename 存在，则打开该文件并把文件指针移到现有数据的尾部，这样，用户写入文件的数据都将被追加到现有数据的后面。

传给一个已打开文件的数据，只有在用 close 函数关闭文件后才会真正被写入文件中。

注意 在 DOS 平台上，某些程序和文本编辑器在写入文本文件时会在文本尾部加上一个文件结束标记（

CTRL Z，十进制 ASCII 码 26）。在读入文件时，当碰到 CTRL Z 标记时， 便返回文件结束状态，而不管其后是否还有其他数据。如果想用 OPEN 函数的 "a"（追加）方式在其他程序所建立的文本文件后面追加数据，则必须保证这些程序没有在其文本文件尾部插入 CTRL Z 结束标记。

open 函数返回一个可由其他 I/O 函数使用的文件描述符。该文件描述符必须用 setq 函数赋给某变量。

(setq a (open "file.ext" "r"))

假定下面实例中所用的文件都不存在，

(setq f (open "new.tst" "w")) 返回 <File #nnn>

(setq f (open "nosuch.fil" "r")) 返回 nil

(setq f (open "logfile" "a")) 返回 <File #nnn>

filename 参数中可以包含一个目录前缀，如 /test/func3（在 DOS 平台上，还允许使用驱动器字符），并且还可以使用反斜杠 (\) 来代替斜杠 (/)，但应记住要使用两个反斜杠 (\\)，这样字符串中才会有一个反斜杠。

(setq f (open "/x/new.tst" "w")) 返回 <File #nnn>

(setq f (open "nosuch.fil" "r")) 返回 nil

or

返回一个表达式的逻辑或 (OR) 运算结果

(or expr...)

or 函数对表达式表中的表达式从左到右进行求值，并查找一个非 nil 表达式。如果找到了这样一个表达式，它就停止进一步求值，并返回 T；如果表达式表中的所有表达式都是 nil，or 函数返回 nil。

(or nil 45 '()) 返回 T

(or nil '()) 返回 nil

osnap

将某种对象捕捉模式作用于指定点而获得一个三维点，并返回该三维点

(osnap pt mode)

mode 参数是一个字符串，其中包含了一个或多个有效的对象捕捉模式标志符（如 mid、cen 等），各标志符之间用逗号隔开。

(setq pt1 (getpoint))

(setq pt2 (osnap pt1 "cen"))

(setq pt3 (osnap pt1 "end,int"))

由 osnap 函数返回的点取决于当前三维视图和系统变量 APERTURE 的设置。

P

polar

在 UCS 坐标系下，求某点的指定角度和指定距离处的三维点，并返回该三维点

(polar pt ang dist)

ang 参数表示一个以弧度为单位的角度值，它是相对于 X 轴按逆时针方向计算的。尽管 pt 可以是三维点，角度 ang 总是在当前构造平面内计算的。

(polar '(1 1 3.5) 0.785398 1.414214) 返回 (2.0 2.0 3.5)

prin1

在命令行打印一个表达式或将该表达式写入一个已打开的文件中

(prin1 [expr [file-desc]])

并不要求 expr 参数是字符串。如果提供了 file-desc 参数并且是一个以写入方式打开的文件的描述符，expr 会被准确地写到文件中，就象它出现在屏幕上的那样。本函数仅仅打印指定的 expr，不包括换行和空格。

(setq a 123 b '(a))
(prin1 'a)　　打印 A　　并返回 A
(prin1 a)　　打印 123　　并返回 123
(prin1 b)　　打印 (A)　　并返回 (A)
(prin1 "Hello")　　打印 "Hello" 并返回 "Hello"

因为没有指定 file-desc，前面的所有例子都显示在屏幕上。假定 f 是一个有效的以写入方式打开的文件的描述符，那么，

(prin1 "Hello" f)

将把 "Hello" 写入指定文件中并返回 "Hello"。

如果 expr 是一个包含了控制字符的字符串，prin1 将按原样显示这些字符，不用它们的扩展功能，下表给出了可用的控制字符。

控制代码

代码	说明
\\	\ 字符
\"	" 字符
\e	Escape 字符
\n	换行符
\r	回车符
\t	制表符
\nnn	八进制代码为 nnn 的字符
\U+XXXX	Unicode 序列
\M+NXXXX	Unicode 序列

也可以不带参数调用 prin1 函数，这时它返回（和打印）空字符串。如果在用户定义的函数中把 prin1

函数（不带参数）作为最后一个表达式，那么当函数执行完成时仅会打印一个空行，使应用程序静默退出。

请参见　在命令行中显示信息

princ

在命令行打印一个表达式或将该表达式写入一个已打开的文件中

(princ [expr [file-desc]])

本函数的功能和 prin1 函数几乎相同，只是本函数将使用 expr 中的控制字符的功能而不是照原样打印它们。

print

在命令行打印一个表达式或将该表达式写入一个已打开的文件中

(print [expr [file-desc]])

本函数的功能和 prin1 函数几乎相同，只是本函数在 expr 之前打印一个换行符而在 expr 之后再打印一个空格。

progn

顺序地对每一个表达式进行求值，并返回最后那个表达式的值

(progn [expr]...)

在仅能使用一个表达式充当操作数，却需要对好几个表达式求值的地方，可以使用 progn 函数来达到目的。

```
(if (= a b)
(progn
        (princ "\nA = B ")
        (setq a (+ a 10) b (- b 10))
    )
)
```

通常情况下，if 函数在测试结果不为 nil 的情况下，仅对一个 then 表达式进行求值。在本例中由于使用了

progn 函数，可以实现对两个表达式的求值。

prompt

在屏幕提示区显示一个字符串

(prompt msg)

在双屏幕的 AutoCAD 配置中，prompt 函数在两个屏幕上都显示字符串 msg，因此，它比 princ 函数更可取。

(prompt "New value: ")　　将 New value: 显示在单屏或双屏上

prompt 函数返回 nil。

Q

quit

强制退出当前应用程序

(quit)

如果调用本函数，它会返回错误信息 "错误: quit/exit abort"，并退出当前程序，返回到 AutoCAD 的命令行提示处。

请参见 exit 函数

quote

返回一个表达式而不对它求值

(quote expr)

这也可以写成：

'expr

(quote a)　　返回 A

(quote cat)　　返回 CAT

(quote (a b))　　返回 (A B)

'a　　返回 A

'cat　　返回 CAT

'(a b)　　返回 (A B)

不能直接从键盘上输入上述最后三个表达式来响应 AutoCAD 的提示，应记住，这样的输入必须用字符 "(" 或 "!" 打头，AutoLISP 才会认为它们是表达式。

R

read

返回从一个字符串中获得的第一个表或第一个原子

(read [string])

string 参数不能在表或字符串外包含空格。read 函数将其参数转换成相应的数据类型后返回：

(read "hello")　　返回原子　　HELLO

(read "hello there")　　返回原子　　HELLO

(read "\"Hi Y'all\"")　　返回字符串　　"Hi Y'all"

(read "(a b c)")　　返回表　　(A B C)

(read "(a b c) (d)")　　返回表　　(A B C)

(read "1.2300")　　返回实数　　1.23

(read "87")　　返回整数　　87

(read "87 3.2")　　返回整数　　87

read-char

从键盘输入缓冲区或已打开的文件中读入一个字符，并将该字符转换成十进制的 ASCII 码值后返回

(read-char [file-desc])

如果没有指定 file-desc 参数，且键盘输入缓冲区中没有字符，read-char 函数将等待用户从键盘输入（随后以 ENTER 键结束）。例如，假设键盘输入缓冲区为空，那么

(read-char)

将等待用户输入。如果用户输入 ABC，后跟 ENTER 键，read-char 函数返回 65（字符 A 的十进制 ASCII 码值）。随后执行的三个 read-char 函数调用将分别返回 66、67 和 10（换行）。如果再调用 read-char

函数，它将再次等待用户输入。

AutoCAD 运行其上的各种操作系统，对文本文件使用了不同的行结束字符。例如，在 UNIX 系

统上使用单字符的换行符（换行 [LF]，即 ASCII 码 10），而在 DOS 系统上却使用两个字符（回车换行 [CR]/LF，即 ASCII 码 13 和 10）作为行结束字符序列。为了便于 AutoLISP 应用程序的开发，read-char 函数接受所有这些约定，在碰到行结束字符（或字符序列）时，它总是返回单个换行符（ASCII 码 10）。

read-line

从键盘输入缓冲区或已打开的文件中读取一个字符串后返回该字符串

(read-line [file-desc])

如果 read-line 函数遇到了文件结束标志，它返回 nil；否则它返回所读取的那个字符串。

例如，假设 f 是一个有效的已打开文件的指针，那么

(read-line f)

将返回文件的下一个输入行，如果已经到达文件结束处则返回 nil。

redraw

重画当前视口或当前视口中的一个指定对象（图元）

(redraw [ename [mode]])

redraw 函数调用的效果取决于所提供的参数。如果不带参数调用 redraw 函数，它重画当前视口。如果调用它时提供了 ename 参数（图元名），它将重画该指定图元。

mode 参数是一个整数，它控制图元是否可见和突出显示。

redraw 的 mode 参数

Redraw mode 对应动作

1 显示图元

2 隐藏图元（使其不可见）

3 突出显示图元

4 解除图元的突出显示

如果 ename 是复杂图元（多段线或带属性的块引用）的头部图元名，那么在 mode 参数为正时，redraw

函数对主图元和它的所有子图元进行处理，而在 mode 参数为负时，它仅处理头部图元。

redraw 函数总是返回 nil。

注意 使图元突出显示（模式 3）后必须使图元解除突出显示（模式 4）。

REDRAW 命令对已突出显示或隐藏的图元无效，然而 REGEN 可强制使图元以正常模式重新显示。

regapp

为当前 AutoCAD 图形注册一个应用名，为使用扩展对象数据作准备

(regapp application)

application 参数是一个最长可达 31 个字符的字符串，它必须遵守符号的命名规则。应用程序名可以包含字符、数字和特殊符号（美元符 ($)、连字符 (-) 和下划线 (_)），但不可以包含空格。应用程序名中的字符将被转换成大写。

如果已经注册了一个同名的应用名，该函数返回 nil；否则它返回应用名。

如果注册成功，该应用名会被加入到 APPID 符号表中，该表维护了所有要在该图形中使用扩展数据的应用名。

(regapp "ADESK_4153322344")

(regapp "DESIGNER-v2.1-124753")

注意 建议使用唯一的应用名，一个可以保证其唯一性的命令方法是使用公司和产品名以及一个唯一的数字串（如您的电话号码或当前日期/时间）。也可以将产品的版本号加入到应用名中或存储在应用程序定义数据的一个单独的整数或实数域中（如 (1040 2.1)）。

rem

将第一个数除以第二个数，并返回余数

(rem number number...)

如果调用本函数时提供的参数多于两个，则 rem 函数返回 (rem ... (rem (rem num1 num2) num3)...) 的结果。

(rem 42 12) 返回 6

(rem 12.0 16) 返回 12.0

(rem 26 7) 返回 5

(rem 5 2)　　返回 1
(rem 26 7 2)　　返回 1

repeat

对每一个表达式进行指定次数的求值计算，并返回最后一个表达式的值
(repeat int expr...)
int 参数必须是正数。
(setq a 10 b 100)
(repeat 4
(setq a (+ a 10))
(setq b (+ b 100))
)　　将 a 设为 50，将 b 设为 500，返回 500

reverse

将表的元素顺序倒置后返回
(reverse lst)
(reverse '((a) b c))　　返回 (C B (A))

rtos

将数转换成字符串
(rtos number [mode [precision]])
根据参数 mode、precision 以及系统变量 UNITMODE 和 DIMZIN 的设置情况，rtos 函数返回一个代表数
number 的字符串。参数 mode 和 precision 都是整数，它们指定了线性单位格式和精度，其取值如下表所示。
线性单位值

Mode 值	字符串格式
1	科学记数格式
2	十进制格式
3	工程记数格式(英尺和十进制英寸)
4	建筑记数格式(英尺和分数英寸)
5	分数单位格式

参数 mode 和 precision 对应于系统变量 LUNITS 和 LUPREC。如果省略这两个参数，rtos 函数使用系统变量 LUNITS 和 LUPREC 的当前值。当选择了工程记数格式、建筑记数格式或分数单位格式时（即 mode 值为 3、4 或 5），系统变量 UNITMODE 对返回的字符串有影响。
请参见　关于 rtos 函数的详细信息，请参见字符串转换

S

set

将一个被引起来的符号名的值设置成一个表达式的值
(set sym expr)
本函数返回表达式的值。
(set 'a 5.0)　　返回 5.0 并设置符号　A
(set (quote b) 'a)　　返回 A　并设置符号　B
如果用函数 set 设置一个未被引起来的符号名，它可以间接地为另一个符号赋一个新值。例如，在执行上例两个命令后，
(set b 640)　　返回 640
并将符号 a 赋值为 640，因为符号 b 的内容是 a。
请参见　setq 函数

set_tile

为一个对话框控件设置值
(set_tile key value)
key 参数是指定控件的一个字符串，value 参数则是指定控件新值的一个字符串（控件的初始值由 value
属性设置）。

setcfg

将应用数据写到 acad.cfg 文件的 AppData 区域中

(setcfg cfgname cfgval)

cfgname 参数是一个字符串（最长可达 132 个字符），它指定段和要被设置为 cfgval（最长可达 347 个字符）的值的参数。如果 cfgname 无效，setcfg 函数返回 nil。cfgname 参数必须是如下格式的字符串：

"AppData/application_name/section_name/.../param_name"

下列代码将 AppData/ArchStuff 段中的 WallThk 参数的值设置成 8，并返回字符串 "8"：

(setcfg "AppData/ArchStuff/WallThk" "8")　返回 "8"

请参见　getcfg 函数

setenv

将一个系统环境变量设为指定值

(setenv varname value)

varname 参数是一个字符串，它指定要设置的系统环境变量名。必须将参数 varname 和 value 用双引号引起来。环境变量名在拼写和大小写上必须和系统注册表完全一致。可能要等到下一次启动 AutoCAD 时该设置才生效。

例如，下列代码将 MaxArray 环境变量设为 10000：

(setenv "MaxArray" "10000")

请参见　getenv 函数

setfunhelp

给帮助工具注册一个用户定义函数，这样，当用户在命令行请求帮助时，就会调用正确的帮助文件和主题

(setfunhelp c:fname [helpfile [topic [command]]])

c:fname 参数是一个字符串，它指定用户定义的命令名（C:XXX 函数），该命令必须包括 C: 前缀。与传给 defun 函数的函数名写法不同，setfunhelp 函数的 function 参数必须是加双引号引起来的字符串。其余的三个可选参数指定帮助文件的调用方式，它们也是字符串类型的，且与调用 help 函数所指定的参数相同。如果调用成功，setfunhelp 函数返回作为 c:fname 参数传递的字符串；否则它返回 nil。

不要求 helpfile 参数带文件扩展名。如果提供了文件扩展名，AutoCAD 仅搜索指定文件。如果没有提供文件扩展名，则按如下搜索规则搜索：如果是在 Windows/NT 版的 AutoCAD 下，加上 .hlp 扩展名，否则加上 .ahp 扩展名。如果没有找到 <filename>.ahp 文件，搜索不带扩展名的 <filename> 文件。注意是在最后搜索不带扩展名的文件，所以在 UNIX 系统上，在搜索 acad 文件之前先搜索 acad.ahp 文件。

应该注意的是，setfunhelp 函数仅检查 c:fname 参数是否带有前缀 c:。它并不检查 函数是否存在或其他参数是否正确。

当用 defun 函数定义一个 C:XXX 类型的函数时，它会取消由 setfunhelp 注册的同名函数（如果存在的话）。因此，只能在调用 defun 函数定义一个用户定义的命令后，才能调用 setfunhelp 函数。

下面的例子用 defun 函数定义一个用户定义命令 MYFUN，用 setfunhelp 函数注册该函数名，并使它与 myhelp.ahp 文件的 myfun 主题联系起来：

```
(defun c:myfun ()
...
    (getint "gimme: ")
    ...
)
(setfunhelp "c:myfun" "myhelp.ahp" "myfun")
```

命令：　myfun

gimme：　'help

假设 AutoCAD 的帮助文件 myhelp.ahp 存在于支持路径中，则 AutoCAD 会显示包含 myhelp.ahp 文件中的 myfun 主题的帮助对话框。

请参见　defun 和 help 函数

setq

将一个或多个符号的值设置为相应表达式的值

(setq sym1 expr1 [sym2 expr2]...)

这是 AutoLISP 的一个基本赋值函数。可以在对 setq 函数的一次调用中给多个变量赋值，但它仅返回最后那个表达式的值。

(setq a 5.0)　　返回 5.0

并将变量 a 的值设置为 5.0，以后无论何时对变量 a 求值，都会返回实数 5.0。

(setq b 123 c 4.7)　　返回 4.7

(setq s "it")　　返回 "it"

(setq x '(a b))　　返回 (A B)

请参见　AutoLISP 变量

setvar

设置 AutoCAD 系统变量值

(setvar varname value)

如果成功，setvar 函数返回该系统变量的值。调用本函数时指定的系统变量名必须用双引号引起来。

(setvar "FILLETRAD" 0.50) 返回 0.5

并将 AutoCAD 圆角半径设置为 0.5 个单位。在设置值为整数的那些系统变量时，提供的 value 参数的值必须在 -32,768 和 +32,767 之间。

某些 AutoCAD 命令在给出提示之前，就已经获得了系统变量的值。如果一条命令正在执行时使用 setvar

函数为某个系统变量设置一个新值，该新值可能要等到执行下一个 AutoCAD 命令时才有效。

当用 setvar 函数改变 AutoCAD 系统变量 ANGBASE 时，value 参数解释为弧度值。这与 AutoCAD 命令 SETVAR 不同，SETVAR 命令将参数按度来解释。当用 setvar 函数改变 AutoCAD 系统变量 SNAPANG 时，

参数是按相对于 AutoCAD 的隐含 0 度方向（即正东或时钟三点钟位置）的弧度值解释的。这也与 SETVAR 命令不同，SETVAR 命令将该参数解释成相对于系统变量 ANGBASE 的角度且单位为度。

注意　UNDO 命令不能撤销由 setvar 函数对系统变量 CVPORT 作出的修改。

关于当前版本 AutoCAD 系统变量的列表，请参见 AutoCAD 命令参考中的“系统变量”。

请参见　getvar 函数

sin

以实数形式返回一个以弧度为单位表示的角度的正弦值

(sin ang)

ang 参数必须是一个以弧度为单位的角度值。

(sin 1.0)　　返回 0.841471

(sin 0.0)　　返回 0.0

setview

为指定视口建立一个视图

(setview view_descriptor [vport_id])

view_descriptor 参数是一个图元定义数据表，它与对 VIEW 符号表使用 tblsearch 函数返回的表类似。可选参数 vport_id 指定获得新视图的视口，可以通过 CVPORT 系统变量来获取 vport_id 参数。如果 vport_id 为 0，当前视口获得新视图。如果成功，setview 函数返回参数 view_descriptor。

slide_image

在当前激活的对话框图像控件上显示一个 AutoCAD 幻灯片

(slide_image x1 y1 wid hgt sldname)

幻灯片既可以是一个幻灯文件 (.sld)，也可以是幻灯库文件 (.slb) 中的一个幻灯片。sldname 参数指定幻灯片的方法和在 VSLIDE 命令或菜单文件中指定幻灯片的方法一样（请参见创建图像），可采用如下两种格式之一：

sldname 或 libname(sldname)

幻灯的第一个角（左上角），也就是它的插入点，其坐标是 (x1,y1)，而它的第二个角（右下角）由距第一个角的相对距离 (wid,hgt) 确定（wid 和 hgt 必须是正数）。原点 (0,0) 是图像控件的左上角，可以通过调用尺寸函数 dimx_tile 和 dimy_tile 获得右下角的坐标。

snvalid

检查组成符号表名的各字符的有效性

(snvalid sym_name [flag])

sym_name 参数是指定符号表名的一个字符串，可选参数 flag 指定 sym_name 中是否可以包含竖线 (|)，它可以是 1 或 0（缺省是 0）。如果 sym_name 是有效的符号表名，snvalid 函数返回 T；否则它返回 nil。

符号表名必须仅由字母、数字字符和美元符 ($)、下划线 (_) 和连字符 (-) 等特殊字符组成。空字符串不是有效的符号名。

sqrt

以实数形式返回一个数的平方根

(sqrt num)

(sqrt 4)　　返回 2.0

(sqrt 2.0)　　返回 1.41421

ssadd

将一个对象（图元）加入到选择集中，或创建一个新的选择集

(ssadd [ename [ss]])

如果不带参数调用 ssadd 函数，它将创建一个不含任何成员的选择集（空选择集）。如果调用本函数时，仅提供一个图元名参数 ename，它将创建一个仅含该图元的选择集。如果调用本函数时，同时提供了一个图元名参数 ename 和选择集参数 ss，它将把指定图元 ename 加入到该选择集 ss 中。ssadd 函数总是返回新创建的或修改过的选择集。

在选择集中增加一个图元时，新图元将被加入到已有选择集中，并返回由参数 ss 传入的选择集。这样，如果该选择集被赋给其他变量，它也会反映新增的内容。如果要增加的图元已存在于选择集中，则将忽略 ssadd

函数的操作，且不报告任何错误。

(setq e1 (entnext))　　将 e1 设为图形中第一个图元的图元名

(setq ss (ssadd))　　将 ss 设为空选择集

(ssadd e1 ss)　　将图元 e1 加入 ss 后返回该选择集

(setq e2 (entnext e1))　　获取 e1 后面的那个图元

(ssadd e2 ss)　　将图元 e2 加入 ss 后返回该选择集

ssdel

从选择集中删除一个对象（图元）

(ssdel ename ss)

ssdel 函数从选择集 ss 中删除名为 ename 的图元，并返回 ss 的选择集名。要注意的是，确实从选择集中删除了图元 ename，返回的选择集中不再包含该图元。如果要删除的图元不在该选择集中，本函数返回 nil

。

例如，假设图元 e1 是选择集 ss1 的成员而图元 e2 不是它的成员，那么：

(ssdel e1 ss1)　　返回删除 e1 后的选择集

(ssdel e2 ss1)　　返回 nil（不改变选择集 ss1）

ssget

提示用户选择对象（图元），并返回一个选择集

(ssget [mode] [pt1 [pt2]] [pt-list] [filter-list])

mode 参数是一个用于指定对象选择方式的字符串。合法的 mode 值有 "W"、"WP"、"C"、"CP"、"L"、

"P"、"I" 和 "F"，它们分别对应于 Window、WPolygon、Crossing、CPolygon、Last、Previous、Implied 和 Fence 选择方式。另一种可选的 mode 值是 "X"，它用于选择整个数据库。参数 pt1 和 pt2 指定与选择有关的点。调用 ssget 函数时仅提供一个点而不提供 mode 参数，等价于拾取单个点来选择对象。ssget

函数忽略 Object Snap（对象捕捉模式）的当前设置（除非在调用本函数时专门指定它）。filter-list

参数是指定对象特征的一个关联表，只有与 filter-list 相匹配的那些对象才被加入到选择集中。如果调用 ssget 函数时省略了所有的参数，则 ssget 函数给出“选择对象”提示，允许用户交互式地构造选择集。

选择集中可以包含图纸空间和模型空间两个空间中的对象，但该选择集用于某操作时，那些在当前无效的空间中的对象会被过滤掉。由 ssget 函数返回的选择集中仅包含主图元（不包含属性和多段

线顶点）。

(ssget) 用普通的对象选择方式选择对象，创建一个选择集

(ssget "P") 创建一个选择集，该选择集由最近所选择的对象组成

(ssget "L") 创建一个选择集，该选择集由数据库中最新的可见对象组成

(ssget "I") 创建一个选择集，该选择集由隐含选择集中的对象（PICKFIRST 生效时所选择的那些对象）组成

(ssget '(2 2)) 创建一个选择集，该选择集由通过点 (2,2) 的对象组成

(ssget "W" '(0 0) '(5 5)) 创建一个选择集，该选择集由从 (0,0) 到 (5,5) 的窗口中包含的所有对象组成

(ssget "C" '(0 0) '(1 1)) 创建一个选择集，该选择集由交叉窗选而成，交叉窗口的对角顶点是 (0,0) 和 (1,1)

(ssget "X") 创建一个选择集，该选择集由数据库中的的所有对象组成

(ssget "X" filter-list) 搜索图形数据库，创建一个选择集，其中包括与 filter-list 相匹配的所有对象

(ssget filter-list) 用普通的对象选择方式选择对象，但只将选择的对象中与 filter-list 相匹配的对象放入创建的选择集中

(ssget "P" filter-list) 创建一个选择集，该选择集由最近所选择的对象中与 filter-list 相匹配的那些对象组成

下例中的 ssget 函数需要传入一个点序列，由点序列 pt_list 定义的线中不能有长度为 0 的段。

```
(setq pt_list '((1 1)(3 1)(5 2)(2 4)) )
```

(ssget "WP" pt_list) 创建一个选择集，该选择集由 pt_list 所定义的多边形包含的所有图元组成

(ssget "CP" pt_list) 创建一个选择集，该选择集由 pt_list 所定义的多边形包含或与之相交的所有图元组成

(ssget "F" pt_list) 创建一个选择集，其中包含所有与 pt_list 所定义的多段线相交的所有图元

(ssget "WP" pt_list filter-list) 创建一个选择集，该选择集由 pt_list 所定义的多边形包含的所有图元中与 filter-list 相匹配的图元组成

只有在调用 ssget 函数不带参数时，被选取的对象才被突出显示。选择集会消耗 AutoCAD 的临时文件存储区，AutoLISP 不允许同时打开多于 128 个选择集。如果选择集的数目达到了这个极限，AutoCAD 会拒绝再创建新的选择集，以后调用 ssget 函数只会返回 nil。可以通过将选择集设为 nil 来关闭不再需要的选择集。

在任何可以用 Last 选择方式来响应 AutoCAD 的选择对象提示的地方，都可以通过向 AutoCAD 传递一个选择集变量来响应之，它将选择该选择集变量中的所有对象。

本部分内容包括：

选择集过滤器

在任何选择方式下都可以使用选择集过滤器序列。可以通过它获得一个选择集，其中包含给定类型或在给定图层上或给定颜色的全部对象。

以下例子返回一个选择集，其中包含了隐含选择集（PICKFIRST 生效时所选择的那些对象）中所有的蓝色直线：

```
(ssget "I" '((0 . "LINE") (62 . 5)))
```

还可以通过 ssget 的过滤器序列来获取包含某应用程序扩展数据的全部对象，要实现之，必须如下所示，使用组码 -3：

(ssget "P" '((0 . "CIRCLE") (-3 ("APPNAME"))))
上述代码将选择所有含应用程序 "APPNAME" 的数据的圆。
详细信息请参见选择集过滤器序列和过滤扩展数据。
本部分内容包括：

关系测试

如无特别指定，对过滤器序列 filter-list 中的每一项都隐含使用“相等”测试。对于数值组（整数、实数、点和向量），可以通过包含一个专用的 -4 组指定一个关系运算符来指定其他关系。 -4 组的值是一个字符串，它指定作用于过滤器序列中随后的组的测试运算符。
(ssget "X" '((0 . "CIRCLE") (-4 . ">=") (40 . 2.0)))
上述代码选择所有半径（组码 40）大于或等于 2.0 的圆。
下表列出了所有可能的运算符。
选择集过滤器序列中的关系运算符

运算符	说明
"*"	任何情况均为真
"="	等于
"!="	不等于
"/="	不等于
"<>"	不等于
"<"	小于
"<="	小于或等于
">"	大于
">="	大于或等于
"&"	按位与 (AND)（仅适用于值为整数的组）
"&="	按位屏蔽相等（仅适用于值为整数的组）

关系运算符的使用取决于要测试的组的类型：
除按位运算符（"&" 和 "&="）之外的所有关系运算符对值为实数和整数的组都有效。
按位运算符 "&" 和 "&=" 只对值为整数的组有效。如果 ((integer_group & filter) /= 0)，也就是说 integer_group 屏蔽位中有任意一位设置为 1 时，按位与 "&" 的结果就为真。如果 ((integer_group & filter) = filter)—也就是说，integer_group 的所有屏蔽位全设置为 1 时，按位屏蔽相等 "&=" 的结果才为真（在 integer_group 中可能也设置了整数的其他位，但不对它们作检测）。
对于表示点坐标的组，对坐标 X、Y 和 Z 的测试可以组合成单个字符串，只需将每一个运算符用逗号分隔开即可（例如 ">,>,*"）。如果字符串中省略了某个运算符（例如 "=,<>" 省略了 Z 坐标的测试），则假定使用任意情况均可的 "*" 运算符。
方向矢量（210 组）仅能使用 "*"、"=" 和 "!="运算符（或任意与“不等于”等价的运算符）进行比较。
组值为字符串的组不能用关系运算符进行测试，而应使用通配符。

过滤器的逻辑分组

上节讨论的关系运算符是二进制运算符。也可以通过创建使用逻辑分组运算符的多层嵌套布尔表达式来对组进行测试。与关系运算符类似，分组运算符也由 -4 组指定。它们是成对的，必须在过滤器序列中正确匹配，否则 ssget 调用将失败。这些运算符能包括的操作数的数目取决于所进行的运算。
选择集过滤器序列中的分组运算符

起始运算符	包含的内容	结束运算符
"<AND"	一个或多个操作数	"AND>"
"<OR"	一个或多个操作数	"OR>"
"<XOR"	两个操作数	"XOR>"

"<NOT" 一个操作数 "NOT>"
由分组运算符所使用的操作数，是某个图元域中的组，或一个关系运算符后跟一个图元域中的组，或由这些运算符组成的嵌套表达式。下面是一个在过滤器序列中使用分组运算符的例子：

```
(ssget "X" '((-4 . "<OR")
(-4 . "<AND")
            (0 . "CIRCLE")
            (40 . 1.0)
        (-4 . "AND>")
        (-4 . "<AND")
            (0 . "LINE")
            (8 . "ABC")
        (-4 . "AND>")
    (-4 . "OR>"))
)
```

这将选择所有半径为 1.0 的圆和所有在图层 "ABC" 上的直线。
由于分组运算符不区分大小写，所以也可以使用与大写等价的小写形式："<and"、"and>"、"<or"、"or>"、"<xor"、"xor>"、"<not" 和 "not>"。

ssgetfirst

判断哪些对象是被选取的和被夹取的

(ssgetfirst)

与传给 sssetfirst 函数的参数类似，本函数返回一个含两个选择集的表。表中的第一个元素是由那些被夹取而没有被选取的图元构成的选择集，第二个元素是由那些既被夹取又被选取的图元构成的选择集。表中的两个选择集中的任意一个都可以是 nil，也可以同时为 nil。

注意 本函数只能分析当前图形中模型空间和图纸空间中的图元，而不能分析非图形对象和其他块定义中的图元。

sslength

求出一个选择集中的对象（图元）数目，并将其作为一个整数返回

(sslength ss)

如果该数目大于 32,767，本函数以实数形式返回它。选择集中永远也不会包含重复的对象。

(setq sset (ssget "L")) 将最后一个图元放入选择集 sset 中

(sslength sset) 返回 1

ssmemb

测试某对象（图元）是否是某选择集的成员

(ssmemb ename ss)

如果图元 ename 是选择集 ss 的成员，ssmemb 函数返回图元名 (ename)。否则它返回 nil。例如，假设图元 e1 是选择集 ss1 的成员而图元 e2 不是它的成员，那么：

(ssmemb e1 ss1) 返回图元名 e1

(ssmemb e2 ss1) 返回 nil

ssname

返回选择集中由序号指定的那个对象（图元）的图元名

(ssname ss index)

如果 index 是一个负数或大于选择集中图元的最大序号，那么本函数返回 nil。选择集中的第一个图元的序号是 0。由 ssget 函数获得的选择集中的图元名总是主图元名，而不会返回子图元（属性和多段线顶点），只能通过 entnext 函数访问子图元（请参见 entnext）。

(setq sset (ssget)) 创建一个名为 sset 的选择集

(setq ent1 (ssname sset 0)) 获取 sset 中第一个图元的图元名

(setq ent4 (ssname sset 3)) 获取 sset 中第四个图元的图元名

为了访问选择集中第 32767 个以后的图元，必须提供实数形式的 index 参数，例如：

(setq entx (ssname sset 50843.0)) 获取 sset 中第 50844 个图元的图元名

ssnamex

获取关于选择集创建方式的信息

(ssnamex ss [index])

本函数除了返回序号 index 指定的图元名外，还返回描述该图元是如何被选中的数据。如果没有提供 index 参数，本函数将返回一个表，表中包括该选择集中的每个元素的图元名和描述该图元是如何被选中的数据。

ssnamex 函数返回的数据是一个表，该表的格式如下所述：表中的每个子表要么包含描述图元和其选择方式的信息，要么包含描述用来选择一个或多个图元的多边形的信息。每一个描述选择集中某个图元的子表都包括三个部分：选择方式的 ID 号（大于或等于 0 的整数），被选图元的图元名，和与图元选择方式有关的特定数据（它用于描述图元是如何被选中的）：

((sel_id1 ename1 (data))(sel_id2 ename2 (data)) ...)

下表列出了选择方式的 ID 号。

选择方式的 ID 号

ID 号	说明
0	没有特殊信息（如 Last、All 等）
1	Pick 方式
2	Window 或 WPolygon 方式
3	Crossing 或 CPolygon 方式
4	Fence 方式

用来描述选择图元所用多边形的子表格式如下：先是一个多边形 ID 号（小于 0 的整数），接着是点的描述符：

(polygon_id point_description_1 point_description_n...)

多边形 ID 号从 -1 开始并每个以 -1 递减。由于和显示位置有关，点用以下几种方式描述：一个无限长线方式、一个射线方式或一个线段方式。每个点的描述符包括三个部分：一个点的描述符 ID 号（要描述的项的类型）、该项的起点和一个可选的单位向量，该单位向量或者用来描述无限长线的方向，或者用来描述线段另外一端的偏移量。

(point_descriptor_id base_point [unit_or_offset_vector])

下表列出了有效的点描述符的 ID 号：

点描述符的 ID 号

ID 号	说明
0	无限长线
1	射线
2	线段

如果视点不是 0,0,1，将返回 unit_or_offset_vector。

与 Pick（类型 1）方式选择图元相关联的数据 data 是单个点描述符。例如，对在 WCS 的平面视图中拾取点 (1,1) 而选到的一个图元，返回的表可能如下所示：

(1 <图元名: 60000064> (0 (1.0 1.0 0.0)))

与 Window、WPolygon、Crossing 或 CPolygon 方式选择图元相关联的数据 data 是选择图元所用的多边形的整数 ID 号。将多边形标志符关联起来，以及在多边形与它所选择的图元间建立联系，则取决于应用程序。例如，对由 Crossing 方式选取的一个图元，返回的表可能如下所示（请注意多边形 ID 号是 -1）：

((3 <图元名: 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))))

与 Window、WPolygon、Crossing 或 CPolygon 方式选择图元相关联的数据 data 是一个由点描述符组成的表，这些点描述符描述的是选择栏和图元直观上的交点。例如，对一条近似垂直且与 Z 型选择栏相交三次的直线，返回的表可能如下所示：

((4 <图元名: 60000024> (0 (5.28135 6.25219 0.0))
(0 (5.61868 2.81961 0.0)) (0 (5.52688 3.75381 0.0))))

注意　本函数只能查找当前图形中模型空间和图纸空间中的图元，而不能查找非图形对象和其他块定义中的图元。

sssetfirst

设置哪些对象既是被选取的又是被夹取的

(sssetfirst gripset [pickset])

gripset 参数指定的选择集中的对象将成为被夹取的，而 pickset 参数指定的选择集中的对象将

成为既被夹取的又被选取的。如果两个选择集有公共部分（即某些对象同时属于这两个选择集），sssetfirst

函数将只选择和夹取 pickset 指定的选择集（而不夹取 gripset 选择集）。如果 gripset 为 nil，sssetfirst 函数将选择和夹取 pickset 选择集。 sssetfirst 函数返回由作参数传入的两个选择集构成的表。

注意　在 AutoCAD 运行某命令过程中时不要调用 ads_ssetfirst() 函数。

startapp

启动一个 Windows 应用程序

(startapp appcmd [file])

appcmd 参数是指定要执行的应用程序名的一个字符串。如果 appcmd 没有包含全路径名，它将按照环境变量 PATH 设置的路径来搜索该应用程序。file 参数是指定要打开的文件名的一个字符串。如果函数调用成功，它返回一个大于 0 的整数，否则它返回 0。

下列代码启动 Windows 中的 Notepad，并打开 acad.lsp 文件。

(startapp "notepad" "acad.lsp")

如果一个参数内含空格，它必须用两层双引号引起来。例如，用 Notepad 编辑 my stuff.txt 文件需要执行下列代码：

(startapp "notepad.exe" "\"my stuff.txt\"")

外部定义函数　acadapp ARX 应用程序

start_dialog

显示一个对话框并开始接受用户输入

(start_dialog)

调用本函数之前，必须先调用 new_dialog 函数初始化对话框。对话框一直保持激活状态，直到一个动作表达式或回调函数调用了 done_dialog 函数。通常，done_dialog 函数与关键字为 "accept" 的那个控件（通常是 OK 按钮）和关键字为 "cancel" 的那个控件（通常是 Cancel 按钮）相关联。

start_dialog 函数没有参数，它返回一个传给 done_dialog 函数的可选参数 status。缺省情况下，用户按下 OK 按钮时返回 1，用户按下 Cancel 按钮时返回 0，而在用 term_dialog 函数终止所有对话框时返回 -1。但是，如果传给 done_dialog 函数一个大于 1 的整数形式的状态代码，start_dialog 函数就会将这个值返回，它的含义取决于应用程序。

start_image

开始对话框控件中的一个图像控件的处理

(start_image key)

调用本函数之后，可以调用 fill_image、slide_image 和 vector_image 等函数对图像控件进行各种处理，直到应用程序调用 end_image 函数才结束这种处理。key 参数是指定对话框控件的一个字符串，它是区分大小写的。

注意　不要在调用 start_image 和 end_image 函数之间调用 set_tile 函数。

start_list

开始处理对话框中的一个列表框或弹出式列表框中的列表

(start_list key [operation [index]])

key 参数是指定对话框控件的一个字符串，它是区分大小写的。operation 参数是一个指定操作模式的整数，其含义总结于下表：

start_list 函数所用的操作模式代码

值　说明

1　改变所选择表的内容

2　追加新的表项

3　删除旧表并创建新表（缺省方式）

只有在使用修改模式 (1) 调用 start_list 函数时，才不会忽略 index 参数，此时它指定用随后调用的 add_list 函数修改的表项，index 参数的取值从零开始。如果不指定 operation，其缺省值为 3（创建新表），如果指定了 operation 而没有指定 index，index 的缺省值为 0。

随后调用的 add_list 函数将影响由 start_list 函数启动的表，直到应用程序调用 end_list 函数为止。

注意　不要在调用 start_list 和 end_list 函数之间调用 set_tile 函数。

strcase

将字符串中的所有字母转换成大写或小写后返回

(strcase string [which])

如果省略 which 参数或其求值结果为 nil，string 中的所有字符将会被转换成大写。如果提供了 which 参数且其值不为 nil，string 中的所有字符将会被转换成小写。

(strcase "Sample") 返回 "SAMPLE"

(strcase "Sample" T) 返回 "sample"

strcase 函数将正确处理当前所配置的字符集的大小写映射。（请参见外语版支持）。

strcat

将多个字符串拼接成一个长字符串后返回

(strcat string1 [string2]...)

(strcat "a" "bout") 返回 "about"

(strcat "a" "b" "c") 返回 "abc"

(strcat "a" "" "c") 返回 "ac"

strlen

以整数形式返回一个字符串中字符的个数

(strlen [string]...)

如果提供了多个 string 参数，本函数返回所有字符串的长度之和。如果省略参数或提供一个空字符串，本函数将返回整数 0。

(strlen "abcd") 返回 4

(strlen "ab") 返回 2

(strlen "one" "two" "four") 返回 10

(strlen) 返回 0

(strlen "") 返回 0

subst

在表中搜索某旧项，并将表中出现的每一个旧项用新项代替，然后返回修改后所得的表

(subst newitem olditem lst)

如果在表 lst 中没有找到 olditem，subst 函数返回没有被修改的表 lst。

(setq sample '(a b (c d) b))

(subst 'qq 'b sample) 返回 (A QQ (C D) QQ)

(subst 'qq 'z sample) 返回 (A B (C D) B)

(subst 'qq '(c d) sample) 返回 (A B QQ B)

(subst '(qq rr) '(c d) sample) 返回 (A B (QQ RR) B)

(subst '(qq rr) 'z sample) 返回 (A B (C D) B)

与 assoc 函数连用，subst 函数可以方便地替换关联表中与某关键字相关的值。

(setq who '((first john) (mid q) (last public)))

(setq old (assoc 'first who) 将 old 设为 (FIRST JOHN)

new '(first j) 将 new 设为 (FIRST J)

)

(subst new old who) 返回 ((FIRST J)(MID Q)(LAST PUBLIC))

substr

返回字符串中的一个子字符串

(substr string start [length])

substr 函数从字符串 string 中取出一个子字符串后返回，截取的子字符串的起点在第 start 个字符处，且其长度为 length。如果没有提供 length 参数，则子字符串将一直延续到 string 的结束处才结束。

start 和 length 参数都必须是正整数。

string 中的第一个字符的序号为 1，这和其他那些处理表元素的函数（如 nth 和 ssname）不同：它们将第一个元素的序号视为 0。

(substr "abcde" 2) 返回 "bcde"

(substr "abcde" 2 1) 返回 "b"

(substr "abcde" 3 2)　　返回 "cd"

T

tablet

获取和设置数字化仪校准

(tablet code [row1 row2 row3 direction])

根据 code 参数所指定的整数的不同，tablet 函数既可以获取当前数字化仪的校准，又可以设置数字化仪的校准。如果 code 为 0，tablet 函数返回当前校准，如果 code 为 1，它后面必须跟随新的校准设置值 row1、row2、row3 和 direction。

code

一个整数。如果 code 为 0，tablet 函数返回当前校准，这种情况下，必须省略其余参数。如果 code 为 1，tablet 函数根据后面提供的参数设置校准，这种情况下，必须提供其余参数。

row1, row2, row3

三个三维点。这三个参数指定数字化仪转换矩阵的三行。

direction

一个三维点。这是一个垂直于代表数字化仪表面那个平面的矢量（以世界坐标系 (WCS) 表示）。

注意 如果指定的 direction 没有被规格化，tablet 函数会修正它，所以设置校准后返回的 direction 可能与传入该函数的 direction 不同。与此类似，在 row3 中的第三个元素 (Z) 应该总是为 1，如果在 row3 中的第三个元素处指定了一个不为 1 的数，tablet 函数也会在返回时将其值设为 1。

如果 tablet 函数调用失败，它返回 nil，并设置系统变量 ERRNO 以指出失败原因（请参见第十六章“AutoLISP 错误代码和错误信息”）。这种情况可能在数字化仪不是 tablet 时发生。

用 tablet 函数建立的最简单的变换是恒等变换：

(tablet 1 '(1 0 0) '(0 1 0) '(0 0 1) '(0 0 1))

随着这个变换的生效，AutoCAD 将有效地接受来自数字化仪的原始数字化仪坐标。例如，如果拾取了数字化仪的坐标点 (5000,15000)，那么，在用户的图形中 AutoCAD 仍用该坐标显示这个点。

系统变量 TABMODE 使得 AutoLISP 应用程序可以将数字化仪设置为开或关。

请参见 关于数字化仪变换矩阵的详细信息，请参见数字化仪的校准

tblnext

在符号表中查找下一项

(tblnext table-name [rewind])

table-name 参数是用于指定符号表名的一个字符串，其有效取值为 "LAYER"、"LTYPE"、"VIEW"、

"STYLE"、"BLOCK"、"UCS"、"APPID"、"DIMSTYLE" 和 "VPORT"。指定上述符号表名时，不区分大小写。

注意 由于 vports 函数可以返回当前 VPORT 表信息，所以使用 vports 函数来检索 VPORT 信息可能比用 tblnext 函数更方便一些。

当重复使用 tblnext 函数时，它每次通常会返回指定表中的下一个条目（tblsearch 函数可以设置要检索的下一个条目）。然而，如果在调用 tblnext 函数时提供了可选参数 rewind 且其值不为 nil ，那么就会回绕到该符号表的起始位置，从而获得它的第一个条目。如果符号表中没有条目了，tblnext 函数返回 nil。另外，本函数永远也不会返回已经被删除的条目。

如果找到了一个条目，tblnext 函数就会将该条目的信息以点对表的形式返回其 DXF 类型码和值。

(tblnext "layer" T)　　检索第一个图层的信息

可能返回

((0 . "LAYER")　　符号类型

(2 . "0")　　符号名

(70 . 0)　　标志

(62 . 7)　　颜色代码，如果图层是被关闭的，颜色代码为负值

(6 . "CONTINUOUS")　　线型名

)

注意上述返回表中没有 -1 组。AutoCAD 会记住从每个符号表中最近一次所返回的条目，每次对某符号表调用 tblnext 函数时它会返回该表中的下一个条目。所以，每次开始扫描一个表时，应该确保提供了不为 nil 的第二个参数 ，这样才能回绕到数据库中该表的开头，从而获得该表中的第一

个条目。

由块表中检查出的条目中包含了一个 -2 组，其组值是该块定义中的第一个图元的图元名（如果有的话）。因此，假定当前图形中有一个名为 BOX 的块，那么：

```
(tblnext "block")          检索块定义
可能返回
((0 . "BLOCK")                  符号类型
  (2 . "BOX")              符号名
  (70 . 0)             标志
  (10 9.0 2.0 0.0)         原点坐标 X,Y,Z
  (-2 . <图元名: 40000126>) 第一个图元的图元名
)
```

-2 组中的图元名，可以被函数 entget 和 entnext 接受，但其他的图元访问函数不接受该图元名。例如，不能用 ssadd 函数将其加入到一个选择集中。通过以 -2 组中的图元名为参数调用 entnext 函数，可以扫描组成块定义的各个图元。在块定义的最后一个图元之后，再调用 entnext 函数将返回 nil。

注意　如果一个块定义中不包含图元，那么 tblnext 函数所返回的 -2 组的组值是该块的 ENDBLK 图元的图元名。

tblobjname

返回指定符号表条目的图元名

(tblobjname table-name symbol)

tblobjname 函数在符号表 table-name 中搜索符号 symbol，并返回该符号表条目的图元名。

entget 和 entmod 函数可以用由 tblobjname 函数返回的图元名作参数。

tblsearch

在指定的符号表中搜索一个指定的符号名

(tblsearch table-name symbol [setnext])

tblsearch 函数在符号表 table-name 中搜索符号 symbol，这两个名称都会自动转换成大写。如果 tblsearch 函数找到了符号名为指定符号名的一个条目，它就会以与 tblnext 函数相同的格式返回该条目。如果没有找到这样的条目，它返回 nil。

```
(tblsearch "style" "standard") 检索文本样式
可能返回
((0 . "STYLE")                  符号名
  (70 . 0)             标志
  (40 . 0.0)               固定高度
  (41 . 1.0)               宽度因子
  (50 . 0.0)               倾斜角度
  (71 . 0)             生成标志
  (3 . "txt")              原字体文件名
  (4 . "")             大字体文件名
)
```

通常，tblsearch 函数对 tblnext 函数检索条目的顺序没有影响，然而，如果 tblsearch 函数调用成功，且提供了值不为 nil 的 setnext 参数，那么 tblnext 函数的条目计数器就会被调整，这样，随后调用 tblnext 函数时，它就会返回由该 tblsearch 调用所返回的那个条目之后的条目。

term_dialog

终止当前所有的对话框，就象用户取消了它们一样

(term_dialog)

如果在某个 DCL 文件被打开时终止某应用程序的运行，AutoCAD 将自动调用 term_dialog 函数。本函数主要用于中断嵌套对话框。term_dialog 函数总是返回 nil。

terpri

在命令行上输出一个换行符

(terpri)

terpri 函数不能用于文件 I/O。往文件中写入换行符需要使用函数 prin1、princ 或 print。

textbox

测量一个指定文本对象的尺寸，并返回围住该文本的一个矩形框的对角坐标

(textbox elist)

elist 参数是一个图元定义数据表，其格式与 entget 函数返回的表的格式相同。它定义的必须是一个文本对象。如果在 elist 表中定义文本参数（而不是文本本身）的那些域被省略，则使用当前值或缺省值。如果 textbox 函数调用成功，它返回一个包括两个点的表；否则它返回 nil。

textbox 函数可以接受的最小的表是文本本身：

(textbox '((1 . "Hello world."))) 可能返回 ((0.0 0.0 0.0) (0.8 0.2 0.0))

在这种情况下，textbox 函数将使用当前缺省值来为文本提供其余参数。

textbox 函数返回的点描述了文本对象的边框（假定该文本对象的插入点是 (0,0,0)，而它的旋转角度是 0）。通常由 textbox 函数返回的第一个子表是点 (0.0 0.0 0.0)，除非该文本对象是倾斜的、或者是垂直书写的、或者它包含的字符串中有下行字符（如 g 和 p）。第一个点表的值指定从文本的插入点到围住该文本的最小矩形的左下角的偏移量。第二个点表的值指定该矩形框的右上角。无论被测量文本的方向怎样，返回的两个点表总是描述围住文本的矩形框的左下角和右上角。

textpage

切换至文本屏幕

(textpage)

textpage 函数等价于 textscr 函数。本函数总是返回 nil。

textscr

切换至文本屏幕

(textscr)

textscr 函数总是返回 nil。

trace

调试 AutoLISP 程序时的辅助函数

(trace function...)

trace 函数为指定的函数设置跟踪标志。每当对指定函数进行求值时，会显示一条跟踪信息表示流程进入该函数（信息按调用深度的不同进行缩排）。此外，还会打印出该函数的执行结果。

(trace my-func) 返回 MY-FUNC

并为 MY-FUNC 函数设置跟踪标志。trace 函数返回传给它的最后的函数名。

请参见 untrace 函数

trans

将一个点（或位移量）从一个坐标系转换成另一个坐标系

(trans pt from to [disp])

pt 参数是由三个实数组成的一个表，它既可以被解释成一个三维点，又可以被解释成一个三维位移（矢量）。from 参数指定 pt 的坐标系，而 to 参数指定返回点的坐标系。如果提供了可选参数 disp 且其值不为 nil，则将 pt 作为三维位移而不是三维点看待。from 和 to 参数可以是一个整数代码（其取值和含义如下表所示），也可以是一个图元名或一个三维拉伸矢量。

坐标系代码

代码 坐标系

0 世界坐标系 (WCS)

1 用户坐标系 （当前 UCS）

2 显示坐标系：

与代码 0 或代码 1 一起使用时，表示当前视口的 DCS

与代码 3 一起使用时，表示当前模型空间视口的 DCS

3 图纸空间的 DCS（仅与代码 2 一起使用）

如果 from 或 to 参数是图元名，它必须是由 entnext、entlast、entsel、nentsel 和 ssname 等函数返回的图元名。这样就可以将某个特定对象的对象坐标系 (OCS) 中的点来回进行转换（对于某些对象，

OCS 等价于 WCS，对这些对象，OCS 和 WCS 之间的转换是空操作）。使用三维拉伸矢量（三个实数组成的一个表）是来回转换对象 OCS 的另一种方法。但是，对于 OCS 等价于 WCS 的那些对象，这种转换也不做任何操作。

trans 函数返回由参数 to 指定的那种坐标系表示的一个三维点（或位移）。例如，给定绕 WCS 的

Z 轴逆时针方向旋转 90 度的一个 UCS，那么，

(trans '(1.0 2.0 3.0) 0 1) 返回 (2.0 -1.0 3.0)

(trans '(1.0 2.0 3.0) 1 0) 返回 (-2.0 1.0 3.0)

关于坐标系的详细信息，请参见坐标系统转换。

例如，为了从一行文本的插入点画一条直线，需要将文本对象的插入点从文本对象的 OCS 转换到 UCS：

(trans text-insert-point text-ename 1)

然后就可以将结果传给“起点:”提示。

相反地，将点值送到 entmod 函数中去之前，必须将该点（或位移）值转换成用该对象的 OCS 表示。例如，如果要将圆相对 UCS 偏移 (1,2,3)（不使用 MOVE 命令），就需要将该偏移量从 UCS 转换成圆的 OCS：

(trans '(1 2 3) 1 circle-ename)

然后就可以将结果偏移量加到圆的圆心上去。

例如，如果用户输入了一个点，且想要找出一条直线的哪一个端点看起来离该点更近，这样就先得将用户输入的点从 UCS 转换到 DCS：

(trans user-point 1 2)

然后再将直线的每一个端点从 OCS 转换到 DCS：

(trans endpoint line-ename 2)

这样就可以计算出用户输入的点与直线每一个端点间的距离（忽略 Z 坐标），从而确定哪一个端点看起来距用户输入的点更近。

trans 函数也可以转换二维点，这需要通过给 Z 坐标赋一个适当的值来实现。所使用的 Z 分量取决于所指定的 from 坐标系和该值是作为一个点还是作为一个位移。如果是作为一个位移，那么其 Z 值总是为 0.0

；如果是作为一个点，那么其 Z 值由下表确定。

转换二维点时的 Z 坐标值

原来的坐标系　　所填入的 Z 坐标值

WCS　　0.0

UCS当前标高

OCS0.0

DCS投影到当前构造平面

（UCS XY 平面 + 当前标高）

PSDCS　投影到当前构造平面

（UCS XY 平面 + 当前标高）

type

返回指定项的类型

(type item)

所返回的类型是下表所列的几种原子之一：

符号类型

类型	说明		类型	说明
REAL	浮点数		SUBR	内部函数
FILE	文件描述符		EXSUBR	外部函数 (ARX)
STR	字符串		PICKSET	选择集
INT	整数		ENAME	图元名
SYM	符号		PAGETB	函数分页表
LIST	表（包括用户定义函数）			

对求值结果为 nil 的项（如未定义的符号），本函数返回 nil。

例如，假设对变量进行如下赋值：

(setq a 123 r 3.45 s "Hello!" x '(a b c))

(setq f (open "name" "r"))

那么，

(type 'a)　　返回 SYM

(type a)　　返回 INT

(type f)　　返回 FILE
(type r)　　返回 REAL
(type s)　　返回 STR
(type x)　　返回 LIST
(type +)　　返回 SUBR
(type nil)　　返回 nil
下面的例子中使用了 type 函数：

```
(defun isint (a)
    (if (= (type a) 'INT)        是否整数？
      T              是，返回 T
      nil            不是，返回 nil
    )
)
```

U

unload_dialog

卸载一个 DCL 文件
(unload_dialog dcl_id)
卸载与 dcl_id（在先前调用 new_dialog 函数时获得）相关联的 DCL 文件。
本函数总是返回 nil。

untrace

清除指定函数的跟踪标志
(untrace function...)
本函数返回最后一个函数的名称。
下列代码清除函数 MY-FUNC 的跟踪标志：
(untrace my-func)　　返回 MY-FUNC
请参见 trace 函数

V

vector_image

在当前激活的对话框图像控件上显示一个矢量
(vector_image x1 y1 x2 y2 color)
本函数在当前激活的对话框图像控件(由 start_image 函数打开)上从点 (x1,y1) 到点 (x2,y2) 显示一个矢量。color 参数指定显示该矢量时所使用的颜色，它可以是一个 AutoCAD 的颜色代码，或下表所列的逻辑颜色代码之一：
颜色属性的符号名
颜色代码　　ADI 助记符　说明
-2　BGLCOLOR　AutoCAD 图形屏幕的当前背景颜色
-15　DBGLCOLOR　　当前对话框的背景颜色
-16　DFGLCOLOR　　当前对话框（文本）的前景颜色
-18　LINELCOLOR　　当前对话框线的颜色
原点 (0,0) 表示图像控件的左上角，可以调用 dimx_tile 和 dimy_tile 函数来获取其右下角的坐标。

ver

返回一个包含了当前 AutoLISP 版本号的字符串
(ver)
ver 函数应与 equal 函数连用来检查程序的兼容性。它返回的字符串具有如下格式：
"AutoLISP Release X.X (nn)"
这里 X.X 是当前版本号，而 nn 是说明版本语言的两个字符。
(ver)　　可能返回 "AutoLISP Release 14.0 (en)"
说明版本语言的两个字符的一些例子如下：
(en) 美国/英国　(es) 西班牙　(fr) 法国
(de) 德国　(it) 意大利

vports

返回表示当前视口配置的视口描述符表

(vports)

每个视口描述符都是一个表，该表由视口识别代码和该视口的左下角和右上角的坐标组成。

如果 AutoCAD 的系统变量 TILEMODE 设置为 1（打开状态），所返回的表描述的是由 AutoCAD 的 VPORTS 命令创建的视口配置。视口的角点用 0.0 和 1.0 之间的值表示，用 (0.0, 0.0) 来表示显示屏幕的图形区的左下角，用 (1.0, 1.0) 表示右上角。如果 AutoCAD 的系统变量 TILEMODE 设置为 0（关闭状态），那么所返回的表描述的是由 AutoCAD 的 MVIEW 命令创建的视口对象。视口的角点用图纸空间坐标表示。当 TILEMODE 为关闭状态时，视口号为 1 的视口总是图纸空间。

例如，系统变量 TILEMODE 为打开状态时，给定单个视口配置，vports 函数可能返回：

((1 (0.0 0.0) (1.0 1.0)))

同样，系统变量 TILEMODE 为打开状态时，假设在屏幕的四个角上给定四个相同大小的视口，vports 函数可能返回：

```
(     (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)) )
```

当前视口的描述符总是处于所返回的表的开头位置。在上例中，当前视口的视口号为 5。

W

wcmatch

将模式字符串与某指定的字符串进行匹配比较

(wcmatch string pattern)

wcmatch 函数将字符串 string 与模式字符串 pattern 进行比较，看它们是否相匹配。如果匹配，它返回 T；否则它返回 nil。string 和 pattern 这两个参数既可以是由双引号引起来的字符串，也可以是变量。pattern 参数中可以包含下表所列的通配符。本函数仅对 string 和 pattern 中最前面的 500 个字符进行比较，超过 500 个字符之后的那些字符会被忽略。

通配符

通配符	说明
#	匹配任意单个数字字符
@	匹配任意单个字母字符
.（圆点）	匹配任意单个非字母数字字符
*（星号）	匹配任意字符序列，包括空字符串，它可以出现在任何位置，包括开头、中间和结尾处
?（问号）	匹配任意单个字符
~（波浪号）	如果它是模式字符串的第一个字符，则匹配除此字符串之外的任意字符串
[...]	匹配括号中的任意一个字符
[~...]	匹配不在括号中的任意单个字符
-（连字符）	用在括号里面，用来指明单个字符的取值范围
,（逗号）	分隔两个模式字符串
`（反引号）	特殊转义字符（按字义读取随后的字符）

(wcmatch "Name" "N*") 返回 T

该例子测试字符串 "Name" 是否以字符 N 开头。可以在模式字符串中使用逗号来分隔多个测试条件，下例就进行三个比较：

(wcmatch "Name" "???,~*m*,N*") 返回 T

只要字符串匹配上述三个模式字符串中的任何一个，wcmatch 函数就返回 T。在本例中进行的测试有：

"Name" 有三个字符（为假）；"Name" 中不含字符 m（为假）；"Name" 以字符 N 开头（为真）。由于满足了其中至少一个条件，所以表达式返回 T。

本函数进行的比较是区分大小写的，所以其大写和小写必须匹配。在 string 和 pattern 参数中，可以使用由 AutoLISP 函数返回的变量和值。

为了测试一个字符串中的通配符，必须使用单个反引号 (`) 来转义字符。转义是指跟在单个反引

号后面的字符不当作通配符读取，而是按其表面值进行比较。例如，为了测试在字符串 "Name" 中是否包含逗号，可以使用如下表达式：

(wcmatch "Name" "*`,*") 返回 nil

注意 由于 AutoLISP 的后续版本中可能会加入其他的通配符，所以建议在样本中给所有非字母数字字符加上转义字符转义，以确保它的向上兼容性。

由于 C 和 AutoLISP 程序设计语言都使用反斜杠 (\) 作为转义字符，所以在字符串中必须使用两个反斜杠 (\\)

才能产生一个反斜杠。为了测试字符串 "Name" 中是否包含一个反斜杠，可以使用如下表达式：

(wcmatch "Name" "*`*") 返回 nil

所有包含在括号 ([. . .]) 中的字符都按字义读取，所以不必使用转义字符。但有如下例外：波浪号 (~) 只有当它不是括号里的第一个字符（如 "[A~BC]"）时才按字义读取；否则作为一个“非”字符读取它，这意味着 wcmatch 函数将匹配跟随在波浪号之后那些字符（如 "[~ABC]"）之外的所有字符。而连字符 (-)

则只有当它是括号中的第一个字符或最后一个字符（如 "[-ABC]" 或 "[ABC-]"）时，才按字义读取。否则，在括号中的连字符用于指定某特定字符的取值范围。该范围仅限于单个字符，所以，"STR[1-38]"

与 STR1、STR2、STR3 和 STR8 匹配，而 "[A-Z]" 与任意单个大写字母匹配。

如果闭括号 ("]") 是括号内的首个字符或跟随在波浪号之后（如 "[ ]ABC）" 或 "[~]ABC）"），它将被按字义读取。

while

对测试表达式进行求值，如果它不是 nil，则对其他表达式进行求值，重复这个计算过程，直到测试表达式的求值结果为 nil

(while testexpr expr...)

testexpr 函数重复对表达式进行求值，直到 testexpr 的求值结果为 nil。它返回最后的 expr 表达式最新的值。

下列代码调用了 10 次 SOME-FUNC 函数，其参数 test 的取值从 1 到 10。最后它返回 11，因为这是它最后那个表达式的值。

```
(setq test 1)
(while (<= test 10)
  (some-func test)
  (setq test (1+ test))
)
```

write-char

将一个字符写到屏幕上或一个已打开的文件中

(write-char num [file-desc])

num 参数是要输出的那个字符的十进制 ASCII 码，并且 write-char 函数也返回该值。

(write-char 67) 返回 67

并在屏幕上写上字符 C。假设 f 是一个已打开的文件的文件描述符，那么，

(write-char 67 f) 返回 67

并将字符 C 写入该文件中。

AutoCAD 运行其上的各种操作系统，对文本文件使用了不同的行结束字符。例如，在 UNIX 系统上使用单字符的换行符（换行 [LF]，即 ASCII 码 10），而在 DOS 系统上却使用两个字符（回车换行 [CR]/LF，即 ASCII 码 13 和 10）作为行结束字符序列。为了便于 AutoLISP 应用程序的开发，write-char 函数将换行符（ASCII 码 10）转换成当前使用的操作系统上的行结束字符（或字符序列）。因此，在 DOS 平台上，

(write-char 10 f) 返回 10

但会将字符序列 CR/LF（ASCII 码 13 和 10）写入文件中。该函数不能往文件中写入 NUL 字符（ASCII 码 0）。

请参见 关于 ASCII 码列表，请参见附录 A“ASCII 码”

write-line

将一个字符串写到屏幕上或一个已打开的文件中

(write-line string [file-desc])

本函数返回的字符串 string 带有双引号，但将字符串写入到文件中时会省略双引号。例如，假设 f 是一个已打开的文件的文件描述符，那么，

(write-line "Test" f)　　写入 Test　　并返回 "Test"

X

xdroom

返回一个对象（图元）剩下的可供使用的扩展数据 (Xdata) 空间的数量

(xdroom ename)

如果不成功， xdroom 函数返回 nil。由于可分配给一个图元定义的扩展数据量是有限制的（目前是 16KB），而且允许多个应用程序往同一个图元上附加扩展数据，所以 AutoLISP 提供了这个函数，这样，应用程序就可以检查是否还有足够的空间来附加扩展数据。它经常与 xdsize 函数连用，xdsize 函数用于返回扩展数据表的大小。

下面的例子可找出一个视口对象还有多少可供使用的扩展数据空间。假设变量 vpname 包含的是一个视口对象的图元名，那么，

(xdroom vpname)　　返回 16162

在本例中，原来的 16,383 字节的扩展数据空间还剩 16,162 字节可供使用，也就是说，已经用了 221 字节的空间。可以调用 xdsize 函数来直接求出一个图元所剩的可供使用的数据空间。

xdsize

返回一个表作为扩展数据连接到对象（图元）上时所占用的空间大小（以字节表示）

(xdsize lst)

如果不成功，xdsize 函数返回 nil。lst 参数必须是一个有效的扩展数据表，它必须包含先前用 regapp

函数注册的一个应用程序名。花括号（组码 1002）必须平衡。无效的 lst 参数会引发一个错误，并将系统变量 ERRNO 设为相应的错误代码。如果扩展数据中包含了一个未注册的应用程序名，会给出如下错误信息（假设系统变量 CMDECHO 处于打开状态）：

1001 组中的应用程序名无效

lst 可以用 -3 组码（扩展数据标志）打头，但并非必须。由于扩展数据中可能包含了来自多个应用程序的信息，所以表中必须有一组闭合括号。

```
(-3 ("MYAPP" (1000 . "SUITOFARMOR")
            (1002 . "{")
            (1040 . 0.0)
            (1040 . 1.0)
            (1002 . "}")
    )
)
```

下面是相同的例子，但它不带 -3 组码。该表是前面那个表 cdr 的结果，但重要的是必须有一对闭合括号。

```
( ("MYAPP" (1000 . "SUITOFARMOR")
            (1002 . "{")
            (1040 . 0.0)
            (1040 . 1.0)
            (1002 . "}")
    )
)
```

xload

加载一个 ADS 应用程序

(xload application [onfailure])

application 参数既可以是一个由双引号引起来的可执行文件名，又可以是一个包含了可执行文件名的变量。在加载文件时，会检查该 ADS 应用程序的有效性。另外，还会对 ADS 程序的版本、ADS 本身和正在运行的 AutoLISP 版本作兼容性检查。

如果 xload 操作失败，它通常会引发一个 AutoLISP 错误。然而，如果提供了 onfailure 参数，在操作失败时 xload 函数会返回该参数的值，而不会发出一条错误信息。

如果成功加载指定的应用程序，本函数返回应用程序名。

(xload "/myapps/xapp")　　如果成功，返回 "/myapps/xapp"

如果试图加载一个已经加载的应用程序，xload 函数会发出如下信息：

已加载应用程序“application”

并返回应用程序名。在调用 xload 函数之前，可以调用 ads 函数来检查当前已加载的 ADS 应用程序。

xunload

卸载一个 ADS 应用程序

(xunload application [onfailure])

如果成功卸载指定的应用程序，本函数返回该应用程序名；否则它发出一个错误信息。

application 参数既可以是一个由双引号引起来的应用程序名，又可以是一个包含了应用程序名的变量，该应用程序必须已用 xload 函数加载。必须准确输入应用程序名，就像在调用 xload 函数加载应用程序时指定的那样。如果在调用 xload 函数时在应用程序名前指定了路径（目录名），在调用 xunload 函数时可以省去该路径。

例如，以下函数将卸载先前用 xload 函数加载的应用程序：

(xunload "ame")　　如果成功，返回 "ame"

如果 xunload 操作失败，它通常会引发一个 AutoLISP 错误。然而，如果提供了 onfailure 参数，在操作失败时 xunload 函数会返回该参数的值，而不会发出一条错误信息。xunload 函数的这一特性与 xload

函数类似。

Z

zerop

检查一个数的求值结果是否为 0

(zerop number)

如果 number 的求值结果为 0，本函数返回 T；否则它返回 nil。

(zerop 0) 返回 T

(zerop 0.0)　返回 T

(zerop 0.0001)返回 nil