第36章 AutoLISP

学习目的

在完成本章学习后，读者将掌握以下内容：

- 利用AutoLISP语言进行数据运算。
- 在AutoLISP中进行三角函数的运算。
- 掌握基本的AutoLISP函数及其应用。
- 加载并运行AutoLISP程序。
- 使用Load/Unload Applications（加载/卸载应用程序）对话框。
- 利用流程图分析问题。
- 利用条件函数测试条件。

36.1 关于AutoLISP

AutoLISP是由Autodesk公司开发的一种采用 LISP程序语言的编程工具 (LISP是LISt Processor的缩写)。除了FORTRAN和COBOL语言，大多数在60年代初期开发的编程语言都已经很少使用了，但LISP保留了下来，并演变成了一种在人工智能（ AI）领域居于领先地位的编程语言。在LISP家族系列中包含了：Common LISP、BYSCO LISP、ExperLISP、GCLISP、IQLISP、LISP/80、LISP/88、MuLISP、TLCLISP、UO-LISP、Waltz LISP和XLISP。其中XLISP是一种流行的解释型的LISP，与AutoLISP较为相似的语言是Common LISP。AutoLISP的解释器是内嵌在 AutoCAD软件包中的。但 AutoCAD LT和AutoCAD2.17以下版本的AutoCAD不含AutoLISP，因此，只能够在AutoCAD 2.18以上版本中应用AutoLISP。

AutoCAD的软件包包含了绝大部分的绘图命令，但仍然有一些命令没有提供。例如：AutoCAD没有画矩形的命令，也不能对图形中的文字对象作全局地改动。利用 AutoLISP语言，可以编一个画矩形的程序，也可以对图形中的文字对象进行全局地或选定地操作。事实上你可以利用AutoLISP编写任何程序，并将该程序嵌入菜单中，从而定制自己的系统以获取更高的效率。

本章假定你已熟悉了AutoCAD的命令和系统变量设置，但还没有深入的编程经验。当然，如果你熟悉其他的编程语言，学习 AutoLISP将很容易。本章将讨论最常用的 AutoLISP功能并用它们来写应用程序。 AutoLISP不需要任何特定的硬件设备，只要一个系统能运行 AutoCAD，就能运行AutoLISP。并且可以用任意的文本编辑器来编辑 AutoLISP源程序。

36.2 数学运算

在任何编程语言中，数学运算都是一种很重要的功能。绝大多数在编程和数学运算中经常使用的数学功能在AutoLISP中都可以实现。可以使用 AutoLISP进行加、减、乘、除，还可以找到采用弧度表示的角度的正弦、余弦、反正切等。在 AutoLISP中还有许多其他运算功能，本节我们将详细讨论大部分经常使用的功能。

1. 加运算

语法：(+num1 num2 num3...)

“加”函数(+)**计算加号右边所有数值的和，即**（num1+num2+num3+...)。**数值可为整数或实数，如果所有运算数为整数，则和也是整数；如果运算数都为实数，则和也为实数；如运算数有整数也有实数，则和为实数。在如下的例子中，前两个运算数都为整数，则和为整数；第三个例子运算数中有一个为实数**（50.0)，则和为实数。

Command:(+2 5) **返回：**7

Command:(+2 30 4 50) **返回：**86

Command:(+2 30 4 50.0) **返回：**86.0

2. 减运算

语法：(-num1 num2 num3...)

“减”函数(-)**从第一个数值中减去第二个数值**（num1-num2)。**如果运算数超过两个，则从第一个数中减去后面所有数的和，即** [num1-(num2+num3+...)]。**在下面的例子中，第一个例子从**28**中减去**14，**结果为**14，**由于两个操作数都为整数，则运算结果也为整数；在第三个例子中，从**50**中减去整数**20**和实数**10.0，**得到结果**20.0。

Command:(-28 14) **返回：**14

Command:(-25 7 11) **返回：**7

Command:(-50 20 10.0) **返回：**20.0

Command:(-20 30) **返回：**-10

Command:(-20.0 30.0) **返回：**-10.0

3. 乘运算

语法：(*num1 num2 num3 ...)

“乘”函数求所有在“*”号后运算数的乘积(num1×num2×num3×...)。**如果运算数都为整数，则积为整数；如果有一个运算数为实数，则积为实数。**

Command:(*2 5) **返回：**10

Command:(*2 5 3) **返回：**30

Command:(*2 5 3 2.0) **返回：**60.0

Command:(*2 -5.5) **返回：**-11.0

Command:(*2.0 -5.5 -2) **返回：**22.0

4. 除运算

语法：(/num1 num2 num3 ...)

“除”函数以第一个运算数为被除数，第二个数为除数（num1/num2)。**如果运算数超过两个，则第一个数除以其他所有运算数的积，即** [num1/(num2*num3*...)]。**在下面第四个例子中，**200**除以**5.0**和**4,**运算式为**[200/(5.0*4)]。

Command:(/30) **返回：**30

Command:(/3 2) **返回：**1

Command:(/3.0 2) **返回：**1.5

Command:(/200.0 5.0 4) **返回：**10.0

Command:(/200 -5) **返回：**-40

Command:(/-200-5.0) **返回：**40.0

36.3 增量、减量和绝对值

1. 增量

语法：(1+number)

该函数（1+）对运算数加1（整数）后返回，且返回数的数值加1。在下面第二个例子中，1加上-10.5，返回-9.5。

(1+20) **返回：**21

(1+-10.5) **返回：**-9.5

2. 减量

语法：(1-number)

减量函数（1-）对运算数减1并返回结果，返回值在数值上减1。在下面第二个例子中-10.5减1返回结果-11.5。

(1-10) **返回：**9

(1- -10.5) **返回：**-11.5

3. 绝对值

语法：(abs number)

绝对值函数返回一个运算数的绝对值。运算数可以是整数或实数。在下面第二个例子中，返回-20的绝对值20。

(abs 20) **返回：**20

(abs -20) **返回：**20

(abs -20.5) **返回：**20.5

36.4 三角函数

1. 正弦（sin）

语法：(sin angle)

正弦函数计算以弧度表示的角度的正弦值。在下面第二个例子中，正弦函数计算π（180）°的正弦值，返回为0.0。

Command:(sin 0) **返回：**0.0

Command:(sin pi) **返回：**0.0

Command:(sin 1.0472) **返回：**0.866027

2. 余弦（cos）

语法：(cos angle)

余弦函数计算以弧度表示的角度的余弦值。在下面第三个例子中，余弦函数计算π（180）°的余弦值，返回为-1.0。

Command:(cos 0) **返回：**1.0

Command:(cos 0.0) **返回：**1.0

Command:(cos pi) **返回：**-1.0

Command:(cos 1.0) **返回：**0.540302

3. 反正切（atan）

语法：（atan num1）

反正切函数计算运算数 num1**的反正切值，返回值为以弧度表示的角度值。下面第二个例子中，反正切函数计算** 1.0**的反正切值，返回弧度数** 0.785398**。**

Command:(atan 0.5) **返回：**0.463648

Command:(atan 1.0) **返回：**0.785398

Command:(atan -1.0) **返回：**-0.785398

在反正切函数中，可设定第二个运算数：

语法：(atan num1 num2)

如果设定了第二个运算数，则反正切函数计算（ num1/num2**）的反正切值，以弧度表示。在下面第一个例子中，第一个运算数** 0.5**除以第二个运算数** 1.0**，**atan**函数计算（**0.5/1.0=0.5**）的反正切值。**

Command:(atan 0.5 1.0) **返回：**0.463648 **弧度**

Comamnd:(atan 2.0 3.0) **返回：**0.588003 **弧度**

Command:(atan 2.0 -3.0) **返回：**2.55359 **弧度**

Command:(atan -2.0 3.00) **返回：**-0.588003 **弧度**

Command:(atan -2.0 -3.0) **返回：**-2.55359**弧度**

Command:(1.0 0.0) **返回：**1.5708 **弧度**

Command:(-0.5 0.0) **返回：**-1.5708 **弧度**

4. 角度表示（angtos）

语法：（angtos angle[mode [precision]]）

angtos**函数返回表示角度弧度值的字符串。字符串的格式由** mode**（模式）和**precision**（精度）参数控制。例如**:

(angtos 0.588003 0 4) **返回：**“ 33.6901 ”

(angtos 2.55359 0 4) **返回：**“ 146.3099 ”

(angtos 1.5708 0 4) **返回：**“ 90.0000 ”

(angtos -1.5708 0 2) **返回：**“ 270.00 ”

注意 在（angtos angle[mode [percision]]）**中，**angle**是以弧度表示的角度值**; mode**是指角度表示模式**, **与**AutoCAD**系统变量**AUNITS**一致。**

表36-1**是角度表示模式在**AutoCAD**中的定义：**

表 36-1

角度表示模式	编 辑 格 式
0	**十进制度数**(Decimal degrees)
1	**度/分/秒**（Degrees/minutes/seconds）
2	**百分度**（Grads）
3	**弧度**（Radians）
4	**勘探测量单位**（Surveyor’s units）

precision**是一个整数，用来控制小数点的位数，其值与** AutoCAD**的系统变量**

AUPREC一致。 precision的最小值为0，最大值为4。

上面第一个例子中，角度为0.58803(弧度)，mode为0（十进制度数），precision为4（小数点后四位），函数返回33.6901

36.5 关系运算

程序通常都包含测试一个特定条件的功能。如果条件为真，则程序执行某些操作，如果关系为假，则程序执行另一些操作。例如：关系运算式 (if(< x 5)) 比较x与5的大小，如果x小于5，则返回值为真。这样的比较运算经常在程序中使用。接下来我们将讨论在 AutoLISP编程中使用的几种关系运算。

1. 等于

语法：(= atom1 atom2...)

等于函数(=)判断两个数是否相等。如果相等则条件为真，返回值为 T（真值）。同样，如果不相等，则条件为假，返回值为 nil（假值）。例如:

(= 5 5) **返回：**T

(= 5 4.9) **返回：**nil

(= 5.5 5.5 5.5.) **返回：**T

(= “ yes ” “ yes ”) **返回：**T

(= “ yes ” “ yes ” “ no ”) **返回：**nil

2. 不等于

语法：(/= atom1 atom2...)

不等于函数（/=）判断两个操作数是否不相等。如果不相等，则条件为真，返回值为 T。同样，如果指定操作数相等则条件为假，返回值为 nil。 例如:

(/= 50 4) **返回：**T

(/= 50 50) **返回：**nil

(/= 50 -50) **返回：**T

(/= “ yes ”“ no ”) **返回：**T

3. 小于

语法：(< atom1 atom2 ...)

小于函数(<)判断第一个操作数（ atom1）是否小于第二个操作数（ atom2），如果小于，则条件为真，返回值为T，否则返回nil。例如:

(< 3 5) **返回：**T

(< 5 3 4 2) **返回：**nil

(< “ x ” “ y ”) **返回：**T

4. 小于等于

语法：(<= atom1 atom2....)

小于等于函数（<=）判断第一个操作数(atom1)是否小于或等于第二个操作数(atom2)，如果小于或等于，则条件为真，返回值为T，否则返回nil。例如:

(<= 10 15) **返回：**T

(<= “ c ” “ b ”) **返回：**nil

(<= -2.0 0)　　返回：T

5. 大于

语法：(> atom1 atom2...)

大于函数判断第一个操作数（atom1）是否大于第二个操作数（atom2），如果大于，则条件为真，返回值为T，否则返回nil。例如:

(>15 10)　　返回：T

(> 10 9 9)　　返回：nil

(>“c”“b”)　　返回：T

6. 大于等于

语法：(>= atom1 atom2...)

大于等于函数（>=）判断第一个操作数（atom1）是否大于或等于第二个操作数（atom2），如果大于或等于，则条件为真，返回值为T，否则返回nil。例如:

(>= 78 50)　　返回：T

(>=“x”“y”)　　返回：nil

36.6 defun、setq、getpoint和Command函数

1. defun

defun函数用于在AutoLISP程序中定义一个函数。其语法如下：

(defun name [argument])

其中：Name----------函数名。

Argument---------参数表。

例如:

```
(defun ADNUM()
```

定义一个没有参数和位置标记的函数ADNUM。这表示该函数使用的所有变量都是全局变量，在该程序运行结束后，全局变量不会失去其数值。

```
(defun ADNUM(a b c)
```

定义一个有三个参数a、b和c的ADNUM函数。变量a、b和c可调用该程序以外的值。

```
(defun ADNUM(/a b)
```

定义一个有两个局部变量a 和b的ADNUM函数。局部变量指变量值仅在本程序执行过程中保持其值，且只能在该程序中使用的变量。

```
(defun C:ADNUM()
```

在函数名称前的C：指出该函数可以通过在AutoCAD的Command:提示下输入函数名称来调用，此时函数名必须用括号括起来。

注意　AutoLISP中包含某些内置函数。不可以用这些名称为函数或变量命名。以下是为AutoLISP的一些内置函数保留的名称的列表（可查阅 AutoLISP编程手册，以获得AutoLISP内置函数的完整列表)。

abs	ads	alloc
and	angle	angtos
append	apply	atom

ascii	assoc	atan
atof	atoi	distance
equal	fix	float
if	length	list
load	member	nil
not	nth	null
open	or	pi
read	repeat	reverse
set	type	while

2.setq

setq函数用于为一个变量赋值。其语法如下：

(setq Name Value[Name Value]----)

其中：Name----------变量名。

Value----------赋给变量的值。

赋给变量的值可以是任何表达式（数值、字符串、字母）。如果设定值是字符串，则其长度不能超过100个字符。

```
Command:(setq X 12)
Command:(setq X 6.5)
Command:(setq X 8.5 Y 12)
```

在最后一个表达式中，8.5被赋值给变量X，12被赋值给变量Y。

```
Command:(setq answer "YES")
```

在这个表达式中，变量answer被赋为字符串“ YES ”。

setq函数也可以与另一个函数连接来为变量赋值。下面的例子中， setq函数用于为不同的变量赋值。

```
(setq pt1(getpoint "Enter start point:"))
(setq ang1(getangle "Enter included angle:"))
(setq answer(getstring "Enter YES or No:"))
```

注意 AutoLISP使用某些内置的函数名和符号。不要为这些函数名或符号赋任何值。下面的例子在语法上是成立的，但不能使用，因为pi和angle是AutoLISP的保留字。

```
(setq pi 3.0)
(setq angle(...))
```

3.getpoint

getpoint函数等待用户为一个点输坐标值 X、Y或X、Y、Z。坐标可以用键盘或是屏幕光标输入。其语法如下：

(getpoint[point][prompt])

其中：point.----------输入点或选择点。

prompt----------在屏幕上的提示文字。

例如:

```
(setq pt1(getpoint))
(setq pt1(getpoint"Enter starting point"))
```

注意 在响应getpoint函数时，不能输入任何其他AutoLISP程序的名称。

一个点是二维还是三维是由当前的用户坐标系统（UCS）定义的。

4. Command

使用Command函数可在AutoLISP程序中执行标准的AutoCAD命令。AutoCAD命令名称和选项必需放在双引号中。其语法如下：

(Command “ commandname ”)

其中：Command----------AutoLISP函数。

commandname----------AutoCAD 命令。

例如:

(Command “ line ” pt1 pt2 “ ”)

其中：“ Line ” ----------AutoCAD LINE 命令。

pt1----------第一点。

pt2----------第二点。

“ ----------“ ”返回。

注意 在AutoCAD R12以前的版本中，Command函数不能执行AutoCAD的PLOT命令，即(Command “ plot ”...)为不合法命令。在AutoCAD 2000 、AutoCAD R14 和AutoCAD R13版本中，可以使用Command “ plot ”...。

Command函数不能使用 AutoCAD 的DTEXT和TEXT命令来输入数据（可以用Command命令执行DTEXT和TEXT命令，也可以输入文字的高度和文字的方向，但当DTEXT和TEXT命令提示输入文字时，则无法输入)。

不能使用Command函数执行AutoLISP的输入函数，如getpoint、getangle、getstring、getint等。因此像(Command“getpoint”...) 或者(Command“getangle”...) 都是不合法的。如果程序中包含这样的语句，则在程序被调入时会显示错误信息。

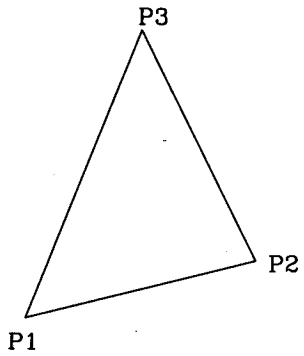


图36-1 三角形(P1,P2,P3)

例1

编写一个程序，该程序可提示输入一个三角形的三个点，并可通过这三个点画线以形成一个三角形，如图 36-1 所示。

大多数程序都由三个基本部分组成：输入、输出和处理过程。处理过程是指由已知输入经过运算得到所需输出的过程（如图36-2所示）。在开始编写一个程序之前，必须先明确这三个部分。在本程序中，输入到程序中的是三个点的坐标值，而要求的输出是一个三角形。处理过程则是要从点P1到P2、P2到P3、P3到P1画三条线来组成一个三角形。明确了这三部分后，可以在编程过程中减少错误。

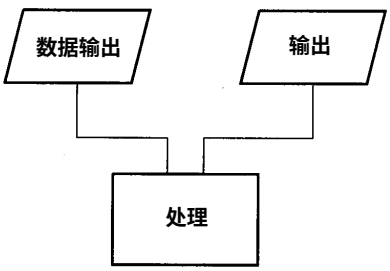


图36-2 一段程序的三个基本部分

处理过程对于程序的成功至关重要。有时处理过程是简单的，但有时却包含了复杂的计算过程。如果程序包含较多的计算，则可以将其按逻辑和系统性分成几个部分

(和子部分)。

另外，由于程序需要一遍又一遍地修改，而且可能还有其他程序员参与，因此应当为程序写一份尽可能清晰明了、没有歧异的文档，这样别的程序员在不同的执行环境下都可理解程序是干什么的。在可能的情况下给出流程图和标识点。

输入	输出
点1的坐标	
点2的坐标	三角形（P1、P2、P3）
点3的坐标	

处理

从P1到P2画线

从P2到P3画线

从P3到P1画线

以下文件是例1的AutoLISP程序清单。注意，右边的数值并不是程序的组成部分，它们在这里仅作参考。

```
;This program will prompt you to enter three points                    1
;of a triangle from the keyboard, or select three points                2
;by using the screen cursor. P1, P2, P3 are triangle corners.           3
                                                                        4
(defun c:TRIANG1()                                                      5
   (setq P1 (getpoint "\n Enter first point of Triangle: "))            6
   (setq P2 (getpoint "\n Enter second point of Triangle: "))           7
   (setq P3 (getpoint "\n Enter third point of Triangle: "))            8
   (Command "LINE" P1 P2 P3 "C")                                        9
)                                                                       10
```

说明:

1~3行：前3行是注释行，用于描述该程序的功能。这是很重要的一些行，这可以使编程更容易进行。在所需处，必需使用注释。所有的注释行必需以“；”号开始，这些行在程序加载时忽略。

第4行：该行是一个空行，用于将注释区与程序分开。空行或用于将程序中的不同模块分开，这有助于明确构成一个程序的不同部分。空行并不影响程序。

第5行：(defun c:TRIANG1()

在该行中，defun是一个AutoLISP函数，用于定义函数TRIANG1，TRIANG1是函数名。在函数名TRIANG1前的c:使得该函数可以像一个AutoCAD命令一样调用。如果没有c:，则命令TRIANG1只有括在括号中如（TRIANG1）才能运行。TRIANG1函数有三个全局变量（P1、P2和P3）。在刚开始编写AutoLISP程序时，使变量成为全局变量是一个好的方法，因为在加载或运行程序后，可以在AutoCAD命令提示下，通过在变量名前加感叹号“！”(Command: !P1）的方法，检查这些变量的值。一旦程序被测试通过后，则应该使这些变量变为局部变量(defun c: TRIANG1(/P1 P2 P3))。

第6行：(setq P1 (getpoint “\n Enter first point of Triangle:”))

在该行中，getpoint函数可等待用户输入三角形的第一个点。提示 Enter first point of Triangle显示在屏幕的提示区域。可以由键盘输入该点的坐标或用屏幕光标选择一个点， setq函数将这些坐标值分配给变量P1。\n用于回车，它使得在\n后的语句打印在下一行（“n”表示

“newline（新行）”)。

7~8行：(setq P2 (getpoint “\n Enter second point of Triangle: ”))和(setq P3 (getpoint “\n Enter third point of Triangle: ”))

这两行提示输入三角形的第二点和第三点，其坐标值分配给变量 P2和P3。\n引起回车操作，使得输入提示出现在下一行。

第9行：(Command “LINE” P1 P2 P3 “C”)

在该行中，Command函数用于输入AutoCAD的LINE命令，然后可从点P1到P2、P2到P3绘制直线。“C”(表示“闭合”选项）可使最后一点P3与第一点P1连接。当用于AutoLISP程序中时，所有的AutoCAD命令和选项都必须处于双引号中，变量P1、P2和P3则用空格分开。

第10行: 该行是由一个闭合括号组成，它表示函数 TRIANG1的定义结束。该括号可以写在上一行中。但是把它放在单独一行是一个好习惯，这可以使其他编程者很容易确认一个定义的结尾。在本程序中，因为只有一个函数定义，所以很容易确定定义的结尾。但是在某些程序中，在一个相同的程序中可能会有多个函数或模块，这就需要明确地表示。括号和空行有助于确定程序中一个定义或一部分的开始与结束位置。

36.7 装载AutoLISP程序

一个AutoLISP程序通常有两个相关的名称：程序文件名和函数名。例如： TRIANG.LSP是一个文件名，而不是函数名。所有的 AutoLISP文件都以.LSP为文件扩展名。在一个相同的AutoLISP程序文件中可以有一个或多个函数定义。例如：在上述例 1中的TRIANG1就是一个函数名。要运行一个函数，定义该函数的 AutoLISP程序文件必须被装载。当正在图形编辑器中工作时，可用下面的命令装载一个 AutoLISP文件：

Command:(load “[path]file name”)

其中：Command----------AutoCAD命令提示符。

load----------调入一个AutoLISP程序文件。

file name----------AutoLISP程序文件的路径和文件名。

AutoLISP文件名和选项的路径名必须包含在双引号中， load命令和file name参数必须包含在圆括号中。如果不加圆括号，则 AutoCAD将试图装载一个图形或一个字体文件，而不是装载一个AutoLISP程序文件。在load和file name之间不需要空格。 AutoCAD成功装载文件后，将在屏幕的命令提示区显示函数名。

要运行程序，只要在 AutoCAD命令提示行键入该函数名，并按 Enter即可(Command: TRIANG1)。如果编程时函数名中没有包含“ C：”，则可以通过带圆括号的函数名来运行程序：

```
Command:TRIANG1   或Command:(TRIANG1)
```

注意 为加载一个AutoLISP程序，在定义其路径时采用正斜杠。例如，如果AutoLISP文件TRIANG存在于C:盘的LISP子目录中，则可使用如下命令加载文件，也可以使用双反斜杠代替正斜杠。

```
Command:(load:"c:/lisp/triang"或者Command:(load"c:\\lisp\\triang")
```

你也可以使用标准的 Windows的拖放功能来装载应用程序。装载一个 LISP程序，可以从Windows资源管理器中把一个文件拖进 AutoCAD的图形窗口中，这样被选中的程序将自动被

加载。另一个装载程序的方法是：使用 Load/Unload Application对话框（如图36-3所示）。这个对话框可以从 Tools菜单中选择 Load Application命令或在 AutoCAD命令提示行中输入 APPLOAD命令来调用。

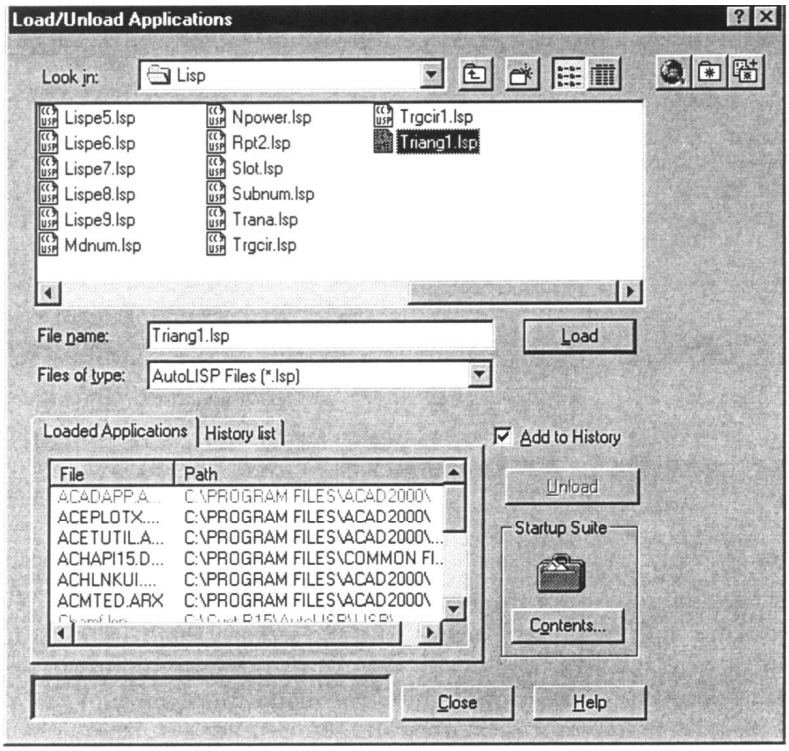


图36-3 用Load/Unload Application对话框加载AutoLISP文件

Load/Unload Application对话框可用于装载LSP、VLX、FAS、VBA、DBX和ObjectARX应用程序。其中 VBA、DBX和ObjectARX程序当被选中时可以立即加载；LSP、VLX和FAS文件在Load/Unload Application对话框关闭时才被加载。对话框上部的列表列出了被选定的目录中的文件。可通过在 Files of type: 编辑框中输入文件类型（*.lsp）修改文件类型，也可以从下拉列表选择文件类型。选中一个文件，然后点击 Load按钮，就可以装载一个文件。下面是 Load/Unload Application对话框一些其他特性的说明：

1) Load: 该按钮用来装载或重复装载所选文件。文件可以从文件列表、 Loaded Application（已装载文件）选项卡或 History list（历史列表）选项卡中选择。 ObjectARX类型的文件不能被重复装载，只能先卸载该 ObjectARX文件，然后再装载它。

2) Load Application 选项卡: 当选择该选项卡后， AutoCAD显示当前加载的程序。从文件列表中选择并拖动文件至 Load Application选项卡的列表中并释放，也可以加载文件。

3) History list选项卡: 当选择该选项卡时， AutoCAD将显示Add to History选中时已被加载文件的列表。如果 Add to History 复选框没有被选中，且拖动一个文件到 History list列表中，文件将被加载，但不会出现在 H istory list中。

4) Add to History: 当Add to History 复选框被选中时，拖动一个文件到有 History list标签的文件列表中，该文件将自动地加入该文件列表中。

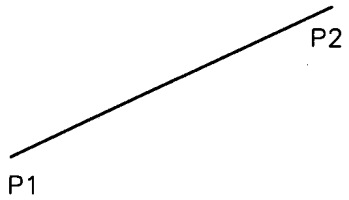
5) Unload: 当选择Loaded Application 选项卡时，Unload按钮将出现。要卸载一个程序时，可以在Loaded Application选项卡的列表中选定文件，然后点击 Unload按钮。其中没有被注册为可卸载的LISP和ObjectARX文件不能被卸载。

6) Remove: 当选择History list选项卡时，Remove按钮出现。在History list中选定一个文件，并点击Remove按钮，可将该文件从History list中删除。

7) Startup Suite（启动组）: Startup Suite组中包含的文件在AutoCAD每次启动时可自动加载。当选择 Startup Suite时，AutoCAD弹出Startup Suite对话框，该对话框内含一个文件列表。可以利用Add按钮将文件加到 Startup Suite列表中，也可以从文件列表中拖动文件放进 Startup Suite组中。要从History list中加入文件，只要在相应的文件上右击即可。

练习1

编一个在两点之间画线的 AutoLISP程序（见图36-4）。要求该程序能够提示用户输入X、Y点的坐标。



P1 (用户定义的点)
P2 (用户定义的点)

图36-4 从点P1到P2画直线

36.8 getcorner、getdist和setvar函数

1. getcorner函数

getcorner函数等待用户输入一个点的坐标值。用户可以用键盘或屏幕上的十字光标输入坐标值。这个函数需要一个基准点，然后当在屏幕周围移动光标时，显示一个相对于基准点的矩形。getcorner函数的语法如下：

（getcorner point[prompt])

其中：point----------基准点。

prompt----------显示在屏幕上的提示信息。

例如:

```
(getcornor pt1)
(setq pt2 (getcornor pt1))
(setq pt2 (getcornor pt1 "Enter second point:"))
```

注意 基准点和所选择的点与getcorner函数有关的点的定位与当前的UCS相关。

如果所选择的点是具有X、Y、Z三维坐标的三维点，则Z坐标将被忽略，该点假定取当前标高为它的Z坐标。

2. getdist函数

getdist函数等待用户输入距离，然后以实数形式返回该距离值。 getdist函数的语法如下：

(getdist [point][prompt])

其中：point----------距离的第一点。

Prompt---------显示在屏幕上的提示信息。

例如 ：

```
(getdist)
(setq dist(getdist))
```

```
(setq dist(getdist pt1))
(setq dist(getdist "Enter distance"))
(setq dist(getdist pt1 "Enter second point for distance))
```

距离可以通过在屏幕上选择两个点来输入。例如，如果用(setq dist(getdist))赋值，则可以输入一个数值或选择两个点；如果用(setq dist(getdist pt1))赋值，因其中第一点pt1已被定义了，所以只需要输入第二点即可。getdist函数总是以实数形式返回距离值。如当前设置为建筑式，且距离值以建筑单位输入，但返回的距离值为实数值。

3. setvar函数

setvar函数可以给一个AutoCAD系统变量赋值，其中系统变量的名称必须包含在双引号内。setvar函数的语法如下：

(setvar “variable-name” value)

其中：variable-name----------AutoCAD 系统变量名。

value----------需赋给系统变量的数值。

例如:

```
(setvar "cmdecho" 0)
(setvar "dimscale" 1.5)
(setvar "ltscale" 0.5)
(setvar "dimcen" -0.25)
```

例2

编写一个AutoLISP程序，其功能是：在两条指定的直线间建立一个倒角，可输入倒角的角度和倒角距离。要形成一个倒角，AutoCAD需使用赋给系统变量CHAMFERA和CHAMFERB的值。当在AutoCAD中选定CHAMFER命令后，第一个和第二个倒角距离将自动赋给变量CHAMFERA 和CHAMFERB， CHAMFER命令然后利用这些值来产生一个倒角。然而在大多数工程制图中，较好的方法是如图36-5那样输入倒角的长度和角度。

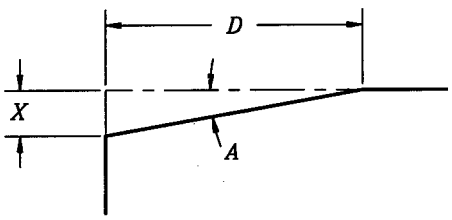


图36-5 具有角度A和距离D的倒角

输入 输出

第一倒角距离（D） 在任何两条所选直线间的倒角

倒角角度（A）

处理过程

1) 计算倒角的第二个距离

2) 将两个距离值赋给系统变量CHAMFERA 和CHAMFERB

3) 利用AutoCAD的CHAMFER命令生成倒角。

计算

x/d = tan a

x = d*(tan a)= d*[(sin a)/(cos a)]

```
;This program generates a chamfer by entering                    1
;the chamfer angle and the chamfer distance                       2
;                                                                3
(defun c:chamf (/ d a)                                           4
```

```
(setvar "cmdecho" 0)                                          5
  (graphscr)                                                  6
  (setq d (getdist "\n Enter chamfer distance: "))            7
  (setq a (getangle "\n Enter chamfer angle: "))              8
  (setvar "chamfera" d)                                       9
  (setvar "chamferb" (* d (/ (sin a) (cos a))))               10
  (command "chamfer")                                         11
  (setvar "cmdecho" 1)                                        12
  (princ)                                                     13
)                                                             14
```

下面是例2的程序代码。右边的行号不是程序的组成部分，在此只是用来参考。

说明:

第7行:（setq d (getdist " \n Enter chamfer distance: "))

getdist函数提示并等待输入倒角的距离值，然后 setq函数将该值赋给变量d。

第8行:（setq a (getangle " \n Enter chamfer angle: "))

getangle函数提示并等待输入倒角的角度，然后 setq函数将该值赋给变量a。

第9行:（setvar " chamfera " d)

setvar函数将变量d的值赋给AutoCAD系统变量chamfera。

第10行:（setvar " chamferb " (*d (/ (sin a)(cos a))))

setvar函数将表达式(*d (/ (sin a)(cos a)))的计算结果赋给AutoCAD系统变量chamferb。

第11行:（command " chamfer ")

command函数调用 AutoCAD命令CHAMFER生成一个倒角。

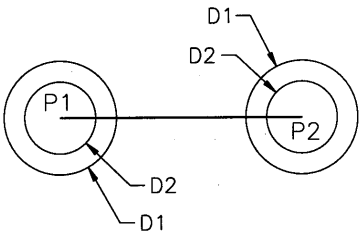


图36-6 具有连线的同心圆

练习2

编写一个 AutoLISP程序以生成如图 36-6所示的图形。程序应能够提示用户输入点 P1和P2的坐标以及圆的直径D1和D2。

36.9 list函数

list函数在AutoLISP中用来定义一个 2D或3D点。如果表达式中没有任何变量或未定义的项，可以用单引号（‘ ）来代替list函数。

例如:

(Setq x (list 2.5 3.56)) 返回2.5, 3.56

(Setq x ‘ (2.5,3.56)) 返回2.5, 3.56

36.10 car、cdr和cadr函数

1. car函数

car函数返回一个列表中的第一个元素的数值。如果该列表中没有任何元素，则返回空值nil。car函数的语法如下：

(car list)

其中： car----------返回第一个元素。

list----------元素列表。

例如:

(car ‘ (2.5 3.56)) **返回**2.5

(car ‘ (x y z)) **返回** x

(car ‘ ((15,20) 56)) **返回** (15 20)

(car ‘ ()) **返回** nil

例中的单引号代表list**函数。**

2. cdr**函数**

cdr**函数将一个列表的第一项删除后返回该列表。** cdr**函数的语法如下：**

(cdr list)

其中：cdr----------**返回一个删除了第一项的列表。**

list----------**元素列表。**

例如:

(cdr ‘ (2.5 3 56)) **返回** (3 56)

(cdr ‘ (x y z)) **返回** (y z)

(cdr ‘ ((15 20) 56)) **返回** (56)

(cdr ‘ ()) **返回** nil

3. cadr**函数**

cadr**函数相当于联合使用**cdr**和**car**函数，它返回一个列表的第二项。先使用** cdr**函数删除列表的第一项，然后再使用** car**函数返回这个新列表的第一项，** cadr**函数的语法如下：**

(cadr list)

其中：cadr----------**相当于**(car (cdr ‘ (x y z))**。**

list----------**元素列表。**

例如：

(cadr ‘ (2,3)) **返回** 3

(cadr ‘ (2 3 56)) **返回** 3

(cadr ‘ (x y z)) **返回** y

(cadr ‘ ((15 20) 56 24)) **返回** 56

在这些例子中，cadr**函数相当于执行了两个操作：**

(cadr ‘ (x y z)) =(car (cdr ‘ (x y z))

=(car (y z)) **返回** y

注意 **除函数**car**、**cdr**和**cadr**外，另一些函数也能从列表中提取元素项。下面是这些函数的一个列表，其中函数**f**由一个列表**’** ((x y) z w))**构成。**

(setq f ‘ ((x y) z w))

(caar f) = (car (car f)) **返回** x

(cdar f) = (cdr (car f)) **返回** y

(cadar f) = (car (cdr(car f))) **返回** y

(cddr f) = (cdr (cdr f)) **返回** w

(caddr f) = (car (cdr (cdr f))) **返回** w

36.11 graphscr、textscr、princ和terpri 函数

1. graphscr函数

假设系统只有一个屏幕，graphscr函数可将文本窗口切换为图形窗口，如果系统有两个窗口，则该函数无效。

2. textscr函数

假设系统只有一个屏幕，textscr函数可将图形窗口切换为文本窗口，如果系统有两个窗口，则该函数无效。

3. princ函数

princ函数可打印（或显示）变量的值。如果变量是用双引号括起的表达式，则打印（或显示）该表达式。princ函数的语法如下：

```
(princ [variable or expression])
```

例如:

(princ) 在屏幕上显示一个空格

(princ a) 在屏幕上显示变量a的值

(princ “Welcome”) 在屏幕上显示字符串Welcome

4. terpri函数

terpri函数在屏幕上另起一新行，就像\n一样。该函数可用于打印在terpri函数后的行。

例如:

```
(setq p1(getpoint"tEnter first point:"))(terpri)
(setq p2(getpoin"tEnter second point:"))
```

第一行（Enter first point:）出现在屏幕提示行上，然后terpri函数进行回车换行，第二行（Enter second point:）出现在紧接第一行的新行上。如果没有terpri函数，则上面的两行将显示在同一行上：(Enter first point: Enter second point:)。

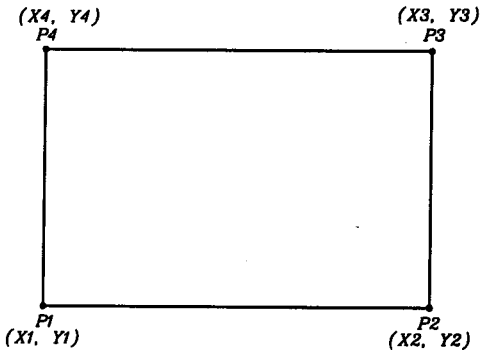


图36-7 矩形(P1,P2,P3,P4)

例3

编写一个程序，该程序可以提示用户输入一个矩形的两对角点的坐标，然后画出该矩形。如图36-7所示。

输入

点P1的坐标

点P3的坐标

输出

矩形

处理过程

1) 计算出点P2和P4的坐标

2) 画出下列直线

P1到P2的直线

P2到P3的直线

P3到P4的直线

P4到P1的直线

P2和P4点的X、Y坐标值可以用car和cadr函数来计算。其中car函数从一个给定的列表中提取X坐标，而cadr函数则提取Y坐标。

点P2的X坐标值

x2 = x3

x2 = car (x3 y3)

x2 = car p3

点P2的Y坐标值

y2 = y1

y2 = cadr(x1 y1)

y2 = cadr p1

点P4的X坐标值

x4 = x1

x4 = car (x1 y1)

x4 = car p1

点P4的Y坐标值

y4 = y3

y4 = cadr(x3 y3)

y4 = cadrp3

这样可以得到P2和P4点：

p2 = (list(car p3)(cadr p1))

p4 = (list(car p1)(cadr p3))

下面是例3的程序代码，其中右边的行号不是程序的组成部分，它们只起参考作用。

```
;This program will draw a rectangle. User will                    1
;be prompted to enter the two opposite corners                     2
;                                                                 3
(defun c:RECT1(/ p1 p2 p3 p4)                                      4
(graphscr)                                                         5
(setvar "cmdecho" 0)                                               6
(prompt "RECT1  command  draws  a rectangle")(terpri)              7
(setq p1 (getpoint "Enter first corner"))(terpri)                  8
(setq p3 (getpoint "Enter opposite corner"))(terpri)               9
(setq p2 (list (car p3) (cadr p1)))                                10
(setq p4 (list (car p1) (cadr p3)))                                11
(command "line" p1 p2 p3 p4 "c")                                   12
(setvar "cmdecho" 1)                                               13
(princ)                                                            14
)                                                                  15
```

说明:

第1~3行: 前3行是本程序的注释，用于描述程序的功能，所有注释行都要以分号（;）开头，在程序调入后，这些注释行将被忽略。

第4行: (defun c:RECT1(/ p1 p2 p3 p4)

用defun函数定义函数RECT1。