Autolisp 程序设计与应用

H

中国科学院希望电脑技术公司
一九八八年十一月
北京

目　　录

引　言

为什么学习 Auto LISP

设计Auto LISP的目的是为了帮助使用者充分利用Auto CAD，从而节省时间和提高效率。可能你同大多数Auto CAD的用户一样，既要用Auto LISP解决日常问题，又不愿过多花费时间去成为Auto LISP的专家。

这种要求当然是合理的，其实，学习Auto LISP不必非花很多精力。可以用简明的语言来解释Auto LISP而使普通人都能理解如何使用它。

Auto LISP提供了一种用程序控制图形外观和图形数据库的手段，很多第三厂家开发的软件都依赖它提供了服务程序。更重要的是，它提供了解决特定问题的专用手段，这也就是你首先考虑购买Auto CAD的原因。

内容简介

本书的目的是帮助你建立提高日常工作的效率的简单而有效的程序。例如，你可能要一个正文输入程序，也可以在图形中进行全局替换或者要对特殊图形建立一个简化操作步骤的程序。阅读本书后，你就能很快建立这种程序。

本书假定你对程序设计和Auto LISP没有什么经验，在前九章内容之间给出十多个实例。在每章末尾有一个贯彻全章重点的实用例子。为使你容易了解，样本程序的每一行都进行了详细的分析。

第一章　告诉你如何用最基本的命令启动和运行一个Auto LISP程序。

第二章　解释了如何把表分开和如何构造一个新表。

第三章　介绍很多关于如何从键盘、鼠标或数字化仪上输入和选择数据的语句。

第四章　讨论角，核心程序包括平行线、坐标几何、机械图和辅助制造等。

第五章　讲解如何操纵Auto CAD中最有用的系统变量。

第六章　给出了从设想到程序完成的设计过程，通过建立一个程序，一步一步地体会如何建立、调试和查错。

第七章　介绍了很多有用的Auto LISP方法，包括选择和改变图素，你可以改变Auto CAD数据库并开始使用Auto LISP的高级部分。

第八章　介绍了编程、调试的要点。这些是你节省时间和提高Auto LISP水平的捷径。

第九章　包括十二个典型的Auto LISP程序，它们可以充实你的Auto LISP程序库。每个程序都是实用的，除逐行解释外，还给出改进它们的建议。

三个附录中包括如何使用EDLIN，如何改变PGP文件，以及如何使用选择命令，最后还提供了一个词汇索引，对定义给出了解释。

你需要对 Auto LISP 了解多少

本书始终假定你对Auto CAD有一些了解，但你不必是CAD专家。只要你能启动AutoCAD并出一些线、弧和圆，还能存贮起来，你就具备了学习Auto LISP的条件。

本书要比《 Auto CAD 参考手册 》容易得多，更不必说《 Auto LISP程序员参考》了。除此之外，编写程序可以更好地了解 Auto CAD ，提高学习效率。

软件和硬件的要求

本书要求 Auto CAD 软件是2.18版到第9版的。在2.17版以前，由于没有 Auto LISP解释器，所以不能使用。

每个新版本都给 Auto LISP增加了一些过去没有的新特性。除了介绍命令外，本书还指出这些命令源于哪个版本，没有特别指出的表示它在2.18版中就存在。本书始终使用 EDLIN （在DOS系统盘上的一个编辑程序）编写 Auto LISP程序，如果你用其它编辑程序（如Wordstor或Word Perfect等），也可以用它们写程序。EDLIN能力有限，但提供了编写正文的一般手段。关于更多的 EDLIN内容，可以参考附录 A或者你的DOS手册。

在硬件方面，如果你的设备能运行 Auto CAD ，那就可以运行 Auto LISP。因此，如果 Auto CAD 仍然是用软盘的版本， Auto LISP 也用软盘。如果 Auto CAD 需要硬盘。则 Auto LISP 也要。

尽管与版本有关，但你最少需要512K内存容量。640K可以让 Auto LISP 工作得更好。推荐使用最少20M的硬盘。至少要有一个彩显和彩色图形卡，或者单色显示器加上图形卡。但 8087 和 80287 并不是必需的。

以上是运行书中例子所需的最低需求，这并不是对专用的 Auto CAD 工作站的要求。

如何使用这本书

本书是教材而非参考手册，所以要在计算机前读它。某些章节一步一步地教你必备概念，并用程序说明命令是如何执行的。

以后你编程时，可能要参考本书中的例子，但它们并不涉及所有 Auto LISP的命令和函数。有关 Auto LISP 命令的完整的表请参考《 Auto LISP 程序员参考 》和《 Auto CAD 参考手册 》。

如果按下面方法阅读本书将使你获得最大的收益。

1．阅读每条命令的解释。

2．逐条输入例子中的命令，并观察计算机的反应，同时参照书中的说法。

3．对每章后的样本程序要逐行阅读其解释，达到清楚每行所完成的功能。

4．输入（或利用 Auto LISP样本程序盘）每章后的样本程序。并在 Auto CAD 中运行程序，观察其效果。

5．按每章后的建议修改程序，注意这是如何影响程序执行的。

6．学习新概念时，要考虑用它编制程序，并利用已学过的内容。

7．重读《 Auto LISP程序员参考》，可以惊奇地发现，你可以看懂它们了。

8．检查你从杂志上，第三厂家软件或用户协会得到的程序，可以学习大量创造性的想法，也可帮助你找到构造自己的程序的捷径。

在开始阅读第一章之前，可能需要浏览一下第六章和第八章，其中包括很多程序设计方面的有用信息，还有一些要点和技巧。尽管你可能不完全理解其中的内容，但你会

发现它们对学习 Auto LISP 是很有用的。

一般规则

前面说过，本书是为了给你一些关于 Auto LISP的实用知识，并教你编写简单的 Auto LISP 程序。因此，你可以发现本书有时并不使用正规的程序设计技术词汇。例如，Auto LISP中的命令一词正规的说法是函数，为此，本书采用了易于理解的通俗语言，而不是死抠技术词汇的准确性。

不过，在以后的学习中，你需要阅读一些关于 LISP和 Auto LISP 的技术书籍。

正如日常的通用说法一样，下面的记号和约定是本书自始至终使用的。

〈RETURN〉表示在键盘上按〈RETURN〉键或者〈ENTER〉键，不是输入该单词。

在每章开始之前，你要处于 Auto CAD 图形编辑状态的Command行。

要把 DOS环境设为 lispheap=39000 或 lispstack=5000，细节问题参考第一章。

输入命令和程序即可用大写，也可用小写。由于LISP程序习惯于小写，本书也采用小写方式。

Type：是指在 Auto LISP 的Command行上用键盘输入一行内容。

关于 LISP和 Auto LISP的注释

LISP 是最早出现的几个高级程序语言之一，它允许程序员使用类似英语的表达方法开发程序。当你第一次使用 LISP时，你也许会怀疑真正的英语是否同 LISP一样，不过，Auto LISP不是 LISP。

Auto LISP所用的 LISP版本只是该语言的一个子集。Autodesk工程师选择了其中合适的部分，抛弃了（或修改了）另外的内容，如果 LISP没有合适的语句，Autodesk干脆就设计一个。因此，Auto LISP实际上是专为 Auto CAD 设计的一种语言。

如果你所读的书中没有明显提及 Auto LISP，你可能是使用了一种不能与 Auto CAD 联用的语言。你可能会说，为什么 Autodesk不选择 BASIC 或其它熟悉的语言作为内部编程工具。当你阅读本书时，你会发现 LISP是唯一具有给表赋值（如 X、Y坐标)能力的语言，所以选择 LISP是明智的。

就要开始了

为了用 Auto LISP很大地提高 Auto CAD 的效率，在后面几页中，你就开始书写第一个 Auto LISP程序了。随着不断努力工作，Auto LISP的能力就开始显示出来，不久就可以编出简单的程序，但却是真正控制你的应用问题的程序。让我们开始吧！

第一章　开始工作

1.1　基本设置

在开始编写和运行 Auto LISP 程序之前，有些DOS上的工作要事先完成。首先要设置一下 lispheap和lispstack，这要用DOS的 SET 命令来完成。这些设置只被 Auto CAD 和 Auto LISP所识别，用来通知 Auto CAD 为 Auto LISP保留足够的内存空间。

lispheap是主要存贮空间，而 lispstack是 Auto LISP的工作空间，后者用于嵌套IF语句等的处理。 lispheap和 lispstack的总和不能超过45000。

如果没能设置 lispheap和 lispstack，在执行时就会出现 insufficient node space（结点空间不足）的错误信息，表示没有足够空间运行 Auto LISP，不幸的是，在运行另一些程序时，也会看到这条信息，这表明，你运行的程序太多，超出了最大空间，因此要清除一部分程序或使用（ vmon ）命令。有关内容见第四章。

另外一个 SET命令是acadfreeram，它也要在 Auto CAD执行前设定，Acadfreeram允许你为 Auto CAD执行最大可用 RAM空间。对于 AT机且有 640K内存的环境，24 是比较合适的值。这可以测试一下。如果在 Auto CAD中看到了Out of RAM错误，就提高设置的值，最大不要超过 82。如果不修改Acadfreeram值， Auto CAD的缺省值为14。

在执行 Auto CAD之前，有三种方法可用来设置 SET值，如果没有编程经验，也没用过批文件，可选第三种方法。

1．把下列内容加到根目录下的 Autoexec．bat文件中去。

```
set  lispheap= 39000
set  lispstack= 5000
set  acadfreeram  = 24
```

2．建立一个每次启动 Auto CAD的特殊批处理文件，下面给出一个样本批文件的实际样子。

```
cd  \ acad
set  lisphean= 39000
set  lispstack= 5000
set  acadfreeram = 24
acad
```

你可以把很多内容加入这个文件，把目录改到 \ ACAD表示在进入 Auto CAD之前先置缺省目录。要保证在执行 ACAD命令前先进行过三个值的设置。

3．如果没用过，也不懂如何使用批文件。 SET命令可以用下面步骤下 DOS提示符下输入，在提示符C: > 下。

```
set  lispheap= 39000 〈RETURN〉
set  lispstack= 5000 〈RETURN〉
set  acadfreeram= 24  〈RETURN〉
```

记住，如果选择方法 3，则每次开机后都要做这件事。

1.2 设置单位和界限

本书中的大部分样本程序都按下面设置了单位（units）和界限（limits）

结构制（Architectual）或工程制（Engineering）。

角度为度分秒制（4位小数）。

East或三点钟时针方向为0度基准。

角度按逆时针方向度量。

界限为宽144寸，高为96英寸。

为了按这些要求建立样本图形，要按以下步骤操作：

1．启动 Auto CAD。

2．在主菜单中选#1启动新图形

3．在Command行上，输入units

4．选择 Architectual(3)或Engineering（4）

5．选择转度 。

6．对角度选择°′″（Degrees minutes seconds）

7．选择4位精度。

8．选择 East为0度。

9．对 Do you want angles measured clockwise 回答 N。

10．回到 Command 行后，再输入 limits。

11．用0，0回答左下角位置。

12．用144′，96′回答右上角。

回到Command行后，输入ZOOM A，再输入GRID 5′。现在可以看到屏幕栅格，如果没有，则输入ZOOM A后输入 ZOOM E 。

13．保存样本图形待以后使用，输入 SAVE SAMPLE。

14．用 END或 QUIT退出图形。以后每次用主菜单中#2调入样本。以后每章用时，先清除图形，再调出Command行。

随赠盘上提供了三个图形文件，它们都为你设置好了参数。

1.3 建立 Auto LISP程序

可以用正文编辑程序来书写 Auto LISP程序文件。正文编辑程序可以让你输入正文并把它们存到某个文件中。这时不能使用字处理中的控制码，如Wordstar中要用 N方，在Wordperfect中用 Dos text方式。

本书一直用 EDLIN书写 Auto LISP程序，除了容易找到以外，用 EDLIN的优点之一是容易在 Auto CAD中执行。存把 Auto LISP程序存回磁盘之后，EDLIN 返回到 Auto CAD图形编辑态，附录 A中有介绍 EDLIN的内容。

不过，可以通过设置 ACAD.PGP来让其它正文编辑程序代替 EDLIN 工作。如果有别的正文编辑软件，可以不用比较笨拙的 EDLIN。

重要的是记住，要用 EDLIN或其它程序中非文本状态来编辑 Auto LISP 程序。

现在，让我们建立一个简单的 Auto LISP 程序，进入 Auto CAD 图形编辑，在 Command 下。

输入： EDIT〈RETURN〉

回应： File to edit

输入： TEST1.LSP

回应： New File

输入： I〈RETURN〉

回应： 1

输入： (defun myprog) ( ) 〈RETURN〉

(prompt" My program") 〈RETURN〉

) 〈RETURN〉

CTRL + C

回应： *

输入： e〈RETURN〉

上面存贮的程序，叫做 TEST1. LSP，在返回Command行后，可以执行这个程序。

(load " test1") 〈RETURN〉

回应： myprog1

Autt LISP 确认调入程序为 myprog

输入： (myprog1) 〈KETURN〉

回应： " My program "

祝贺你完成了第一个 Auto LISP 程序。现在解释一下。

Auto LISP 程序必须有 LSP后缀，否则 Auto CAD在调入时找不到它。如本章末的 Auto LISP program lesson# 1 中，例子程序为 LES1. LSP。

如果是新建程序，EDLIN回答 Newfile，否则将回答 End of Input File。这不是错误，而是说，文件已全读入内存。这时就可以用 EDLIN的命令编写程序了。

1.4 用分号注释程序

当程序中一行以分号开头时，本行后面内容不被当作程序。这同 BASIC 中的 REM 语句一样，用来注解程序。程序调入时，分号后的内容被忽略而不调入，不过，可以用正文编辑改动这些内容。

最好养成注解程序的习惯。记下你所做的所用的变量，以及今后要如何修改它。这些注解在调试和修改程序时会很有用。由于注释不占用 Auto CAD内存，所以可以随便使用。

分号不必总在一行开始，例如：

(prompt " This will print") ; This is a comment

从分号起，后面的内容是注释结束于第一个回车符。如果下行仍是注释，要用分号开头。

1.5 括号( )

括号是 Auto LISP程序中关键的符号，所有命令都要用括号括住。Auto LISP 用括号表示嵌套，允许用一个命令执行另一个命令，也就是说，命令可以嵌套，当你不断加入括号时，命令嵌套就不断加深。

记住，退出嵌套时要用对应的括号。简单的括号配对方法是左括号加1，右括号减1。

例如：

```
( × × × ( × × × ( × × × ) ( × × × )  )  )
1       2       3       2  3      2  1  0
```

开始，第一（为1，然后为2和3，逢）减到2，然后又是3和2，最后是1和0。在计数过程中，最后总应该是0，表示括号已经匹配。虽然这不保证命令或程序是正确的，但括号不配对的程序肯定是错误的！

最后，括号不必在同一行。例如：

```
( × × ×   ( × × × )
( × × ×   ( × × × )   ( × × × )
 )
```

上面的写法同前例一样。在写 Auto LISP程序时，缩进书写便于阅读，可以清楚地理解和记录所做的工作。

如果括号没正确结束，Auto CAD在Command行提示 1 >，要改正它，只须再用一个括号结束 LISP参数，Auto LISP这时开始执行。如：

输入：(load " test1 ")　(RETURN)

回应：1 >

这是提示输入另一半括号。

输入：)

回应：myprog1

在缺失的括号补上后，Auto LISP文件被正确地调入了。

有三种情况可以出现1 >，缺少括号，缺少双引号和缺少单引号，缺少括号最常见。如果补上括号无效，就试着补双引号或单引号。

出现2 >表示缺两重括号，数字表示 Auto LISP目前结束语句所需要的括号数。

注意，有时尽管你修改了括号，程序仍然没有正确执行，不过，至少你能够返回到Auto CAD的 Command行上了。

1.6 怎样调入一个程序

在执行程序之前，包括该程序的文件必须首先被调入内存。在Command行调入程序的方法是：

(load " program name ")　(RETURN)

重要的是要用括号括住 load命令。如果不这样，Auto CAD就试图去调入一个shape文件或 text文件（因为没有括号的 load命令也是合法的）。

几乎所有 Auto LISP命令都能直接在Command 行上输入，只是用括号括起来。还是

注意，要把 Auto LISP 文件名用引号括起来。

尽管文件必须用 LSP作为扩展名，可你不用加上 LSP。load 命令会为调入文件先加上这个名字的。

一旦调入了程序，Auto CAD会把最后一个函数名在屏幕上显示出来。

1.7 （defun 定义函数

在 Auto LISP 中，一个函数就是一个程序。同其它语言不同，文件名不是程序。实际上，一个 Auto LISP 文件中可以有多个程序。一旦文件调入，所有程序（函数）就可以使用了。在本书中，程序和函数不加以区分。

程序或函数的名字必须在第一个语句中被定义，这需要用（defun语句。

（defun是程序中的第一个命令，程序名后面并不立即跟着闭括号，实际上，程序的最后一个括号才与之配对。

（defun跟随一个函数名：

```
( defun  drawline
```

其中 drawline 是函数名。给出名字后，可以有三种选择，首先可以只给一对括号。

```
( defun    drawline    ()
```

这等于让程序中所有变量都是全局的。全局变量在程序结束后不失去原来的值。如：变量 R在一个函数中定义为 17.52，在下个函数开始时仍然是 17.52，我们就说它是全局的。

其次，也可以使所有变量都是局部的。例如，R在一个函数中定义为 17.52，但只在该函数中才有定义，这就是局部于该程序。如果让变量是局部的，就在程序各后用／符号引导。

例如：

```
( defun  drawline ( /  pnt1 pnt2 )
```

记住，drawline是函数名，上例中，后面跟随着程序中的局部变量，局部变量只在程序运行时才有值，无局部变量的函数在函数名后必须跟随（）。

还要记住，用闭括号结束函数。由于一个文件中可能有多个函数，每个函数都要这样开始，并用闭括号结束。

第三种选择是没有／符号的变量表。这意味着变量由程序外部赋给值。如：

```
( defun  dtr ( a)
```

该函数用于把度数转化为弧度，使用函数时，要首先知道度数，你可以在 Command 行上输入（dtr 32.234）来把32.284转化为弧度。32.284被赋与 defun 后面没有／符号的第一个变量。这就叫传变量给程序。

最后，可以结合以上选择。如：

```
( defun   drawline   ( a / b r d) ,
```

drawline 是函数名，a接收外部传递的第一个变量值，b、r和 d是局部变量，其它未提到的变量是全局变量

另外，对（defun还有一种选择，如果在函数名前加上 C:，就不用在调用函数时加上括号。Auto CAD这时把这个函数当作是（Command）命令，也就是说，可以把

函数象 Auto CAD 的 LINE， ARC 或 CIRCLE 命令一样使用。

也许一开始你会认为C：驱动器，实际上它与C 驱动器无关。在 Auto LISP 中，C：仅是通知 Auto CAD这相当于普通命令。如：

```
( defun C:drawline  ( )
```

定义了函数 drawline，并表明 drawline是一个普通 Auto CAD命令，后面的（）表示无局部变量，所有变量都是全局的。

1.8 使用变量

变量就象代数中的 X和 Y，表示某些值，把它们存贮起来以便使用。

在 Auto LISP中，变量可以是以字母开头的任何字母数字串，一般地说，AutoLISP 变量与很多语言（如 BASIC ）相同，可以使用 A— E及 0 — 9 。合法变量 的例子 如下：

```
a  arc print1  pnt1  d8
```

非法变量例子如下：

87	要以字母开头
1point	要以字母开头
pnt•8	出现非法符号•
a $	仅在 BASIC 语言中有效

Auto LISP 变量与其它语言中的一个主要不同之处是一个变量中可以存贮多个值。值可以是任何内容，如：

实数	Real	（如 184.2089 ）
串	String	（如 John ）
整数	Integer	（如 3， 5 或 9 ）
选择集	Pickset	（后面将介绍）

在 Auto LISP 中，这些变量叫做符号. 本书中变量和符号为了方便都称为变量。

局部变量仅用于函数内，换句话说，如果在（ defun 命令中使用了／，这些变量只在函数中保持其值，如果在其它程序中定义了同名局部变量，它们只在自己的函数内有效，不会与其它同名变量冲突。

有两个理由来使用全局变量代替局部变量，首先，你可能要在其它函数中使用某函数中建立的变量。其次，在调试时，当程序出错时，你可以查看全局变量的值。

局部变量建立后，它的值在程序出错时为 Nil（表示没有值）。不过，全局变量直到其它程序改变它之前都保持值不变。所以，除非彻底调试程序，变量不必被说明。详细内容见后几章。

同局部变量不同，参数是程序开始前赋值的变量。如：

```
( defun  drawline  ( d1 /  pnt1  pnt2 )
```

d1 可能在程序调用时（开始时）表示起始距离。

```
( drawline 8 )
```

d1 开始的值为 8， pnt1 和 pnt2 是局部变量，它们跟在／后面。

如果没有局部变量，也不要参数。则

(defun drawline ( )

( )表示不建立局部变量，程序开始时也没有参数传递。

1.9 原子和表

表可以认为是包括多个值的变量，每个值叫做元素。如果变量只有一个值，就叫做原子。Auto LISP 处理表的能力使它适合于 Auto CAD。

如，图形中的一个点被指述为X坐标值和Y坐标值。在 Auto LISP中，一个点可以被写作有两个元素的表，前面为X值，后面是Y值。

Auto LISP 中的表用括号标记。如：

(7 9)

(3 5 7 9 2)

(3.2039 6.9029 8.2039)

若变量只有一个值，就无需括号，如6表示原子而不是表。

稍复杂一些的表是用来描述一条直线图素的。在直线中，要有始点X、Y坐标，终点坐标，层名，线型，和线比例，最后还用图素名（LINE）与其它图素（CIRCLE 等）相区分。

所有这些信息都存贮在一个变量中。该变量包括所有元素，如下所示

```
( (-1 . <Entity name: 60000014>)
  ( 0 . "LINE")    (8 . "LAYER1")
  (10 . 563.000000  484.000000)
  (11 . 1162.000000 745.000000) )
```

上面是 Auto CAD 数据库中的一条简单直线的存贮值（很简单！），细节在第七章中讨论。由于单个变量可以存贮图素所有信息，使 Auto LISP 成为最理想的语言。

1.10 启动程序

在编好程序文件，并调入内存以后，余下的事情就简单了。

(drawline)

在Command行用括号括住函数名，如果需要参数，则

(drawline 3)

程序可以从键盘调用，也可以从屏幕或图形板菜单上调用，只要保证程序已被调入，程序名要用括号括住。可以参见 Auto LISP样本程序1。

1.11 (graphscr) 图形屏幕

这个命令用来把图形切换回图形屏幕（如果正处于单屏系统的正文屏幕）。如果已经在图形屏幕，则不起变化。可以把该命令放在程序开头以保证一执行程序就进入图形状态。

在单屏系统中，用（F1）切换到正文状态。当处于正文状态时，在Command 行输入。

(graphscr) (RETURN)

观察如何到图形屏幕，再试一次，不过可以先处在图形屏幕，如果没变化，就成功了。

1.12 （prompt）通用打印语句

该命令把正文显示在Command行上。它并不换行，所以两次连续打印将处于同一行上。所以，一般打印都在末尾加上（terpri）命令来保证分行显示。在Command 行上输入。

（prompt "Hello how are you "）（RETURN）

1.13 （terpri）结束打印行

这是换行命令，导致下次打印内容在下一行上，一般都跟随 prompt命令。如：

（prompt "Hello how are you "）（terpri）

Auto LISP还有其它换行方法（如在下个打印语句中加入\n）。不过（terpri）命令不易出错。

有时，你想两次显示在同一行上，有时又想分在两行，如：

（prompt " pick a point "）（prompt " pick 2nd point "）（RETURN）

这时它们出现在同一行。

（promt " pick a point"）（terpri）（prompt " pick 2nd point "）（RETURN） ET

这时，两个提示在两行上。书写 Auto LISP程序的目的之一是让 Auto CAD尽可能自然，所以，用这些命令可以按需要显示。

1.14 （setq）赋值

这是一条赋值命令，它把变量的值域常数赋给另一变量。（setq）很象 BASIC 中LET语句。

IF A = 5 AND B = 6 THEN LET A = B

上例在 BASIC 中把 A赋为原来 B的值、6。

（setq）是 Auto LISP的基本赋值命令。在 Auto LISP中，"="不是赋值号，它只存在于 if或其它非赋值语句中，无赋值能力。（setq）的赋值顺序看上去有些奇怪，（setq a b）是把 b的值赋给 a。setq的第一个变量接收后面的值，这与 BASIC中的 LET命令相似。

LET A = B （A被赋予 B的值）

（setq A B）（把 B的值赋给 A）

不仅可以把变量赋给变量，也可以用常量，数字或字符串，如：

（setq a "Hello hou are you "）

如果现在打印 a，就可以得到 Hello how are you。

（setq a 7）

现在 a为7。下面试几个 Auto CAD操作：

（setq a 7）（RETURN）

! a （RETURN）

这时屏幕上出现 7。符号! 用来打印变量的值。任何时候要看变量的值，都可以用!

后跟变量名，其值立刻被显示。

（ setq b " John "）〈RETURN〉

! b

单词 Jone出现在屏幕上。最后

（ setq a b）〈RETURN〉

! a

单词 John又出现了。本例把一个变量值赋给另一变量，所以 a 和 b有相同的值。

1.15 （ getpoint ）选择一点

上面命令要求输入一个点。它从屏幕上读取一个点，如果与（ setg）命令联用，被读取的 X和 Y坐标值就被赋给用（ setq）指定的变量。

在（ getpoint）中可以加入提示信息，以便用户知道该输入点了。输入：

（ setpoint）〈RETURN〉

发现了什么？看起来没什么反应，但计算机等待你指定一点输入。现在在屏幕上指定一点，坐标就出现在Command行上的括号中。

（ setq a （ getpoint" pick first point "））〈RETURN〉

（ setq）提通过（ setpoint）命令选择的点的 X和 Y坐标值赋给 a。在（setpoint）后面的引号中的内容出现在Command行上。如果要下次信息显示在下行上，就使用（ terpri）命令。

还要注意， getpoint也要有自己的括号。不管是否跟随其它命令，所有命令都要以括号开始，而且要正确地给出匹配的括号。

输入：! a

作为 a的值，一个包含两个值的表出现了，分别是 X和 Y坐标的实数值。

现在要指出 Auto LISP的一个特性，其它的 Auto LISP手册中用“返回”这个词，如（ setpoint）返回了两个实数的表。用日常说法就是，当你用 Auto LISP的命令时，它通常可以得到一个可以使用的值。如果你不用，它就消失了。

所以，使用返回值的方法是用（ setq）把值保存在一个变量中。所谓（ setpoint ）返回两个实数的表是指，当使用（getpoint）时生成了包含 X和 Y坐标的表。为了保存备用，应该把它们存放在变量中 如：

（ setq a （ getpoint " pick first point "））〈RETURN〉

将给 a赋一对 X和 Y坐标值。

（ setg b （ setpoint " pick another point "））〈RETURN〉

这时，第二对坐标送给了变量 b。不要破坏它们，下节要使用它们。

1.16 （ Command ） Auto CAD命令

这个命令提供了在 Auto LISP中使用 Auto CAD命令的方法，后面的变量可以看成命令，输入：

（ Command " line " a b）〈RETURN〉

由于两对 X和 Y坐标分别被存贮在变量 a和 b中，这个命令在 a和 b两点之间画一