
AutoLISP

YUYAN CHENGXU SHEJI

AutoLISP

语言程序设计

郭秀娟 于全通 范小鸥 主编



化学工业出版社

AutoLISP 语言程序设计

郭秀娟 于全通 范小鸥 主 编



化学工业出版社

· 北 京 ·

北京市丰台区右安门外大街22号

010-63792888 010-63792889

本书主要介绍了 AutoCAD 内部的编程语言 AutoLISP, 系统而详细地介绍了 AutoLISP 的各类函数和语法规则。从 AutoLISP 的数据类型和程序结构入手, 循序渐进地介绍了 AutoLISP 的基本函数、AutoLISP 绘图功能、AutoLISP 建筑设计制图程序范例等内容。

本书列举了丰富的程序实例, 图文并茂, 清晰易懂。

本书可作为高等院校建筑学及相关专业的教材, 也可供其他从事设计绘图工作、学习 AutoLISP 语言的有关人员参考, 还可作为 AutoLISP 培训学习的教材。

主 编 郭小娟 于全通 范小鸥

图书在版编目(CIP)数据

AutoLISP 语言程序设计 / 郭秀娟, 于全通, 范小鸥主编.
北京: 化学工业出版社, 2008.6
ISBN 978-7-122-03236-2

I. A… II. ①郭…②于…③范… III. 计算机辅助设计-应用软件, AutoLISP-程序设计 IV. TP391.72

中国版本图书馆 CIP 数据核字 (2008) 第 099737 号

责任编辑: 李彦玲 鲍晓娟

装帧设计: 韩 飞

责任校对: 王素芹

出版发行: 化学工业出版社 (北京市东城区青年湖南街 13 号 邮政编码 100011)

印 刷: 大厂聚鑫印刷有限责任公司

装 订: 三河市宇新装订厂

787mm×1092mm 1/16 印张 14½ 字数 359 千字 2008 年 8 月北京第 1 版第 1 次印刷

购书咨询: 010-64518888 (传真: 010-64519686) 售后服务: 010-64518899

网 址: <http://www.cip.com.cn>

凡购买本书, 如有缺损质量问题, 本社销售中心负责调换。

定 价: 29.00 元

版权所有 违者必究

前 言

在工程设计领域中, AutoCAD 得到了普遍应用, 有许多人在研究使用 AutoCAD, 并画出了很多工程图纸。然而, 人们总会感到作图效率还不够高, 这是因为 AutoCAD 是一个通用绘图软件, 不具备专业特色。AutoCAD 开放的结构为不同的使用者留出了广阔的空间, 提供了许多二次开发工具, AutoLISP 是最强的一个, 可以利用 AutoLISP 开发出具有专业特点的高效率的应用软件。鉴于此, 人们希望了解 AutoLISP 的程序结构, 掌握 AutoLISP 程序设计方法, 进而开发出适合本专业特点的应用 CAD 系统。一些高等院校的工程设计相关专业也开设了 AutoLISP 语言课程。然而, 各类现行出版的 AutoLISP 参考手册, 除了屈指可数的几个例子之外, 剩下的就是按字典顺序排列起来的函数功能简介, 使人望之却步, 看后却不得要领, 无法实现深度学习及开发设想和愿望。

编者通过查阅有关资料, 阅读现存的参考手册, 在计算机上慢慢摸索研究, 历经多年, 对 AutoLISP 语言有了较为深入的理解, 积累了一些难得的实践资料, 已成功应用于教学实践之中。同时, 和从事 CAD 工作的同志切磋交流, 开发了相关的 CAD 应用软件, 在工程设计领域得到了应用。

本书共 11 章, 前 9 章系统而详细地介绍了 AutoLISP 的各类函数和语法规则, 提供了大量的 AutoLISP 程序, 使读者从例子中深化对概念的理解, 领悟程序设计方法; 第 10 章提供了若干个建筑设计制图 AutoLISP 程序范例, 便于建筑专业学生作课程设计或工程设计人员对 AutoCAD 二次开发。第 11 章介绍了 AutoCAD 的基本命令。书中每章后面附有练习题, 以供读者练习。

本书第 1 章至第 5 章由郭秀娟编写, 第 6 章由王祥瑞编写, 第 7 章、第 11 章和附录部分由范小欧编写, 第 8 章至第 10 章由于全通老师编写, 全书的图形部分由孙东和林娜撰稿, 程序实例验证由崔星华和吉林省经济管理干部学院的张树彬共同完成。最后, 全书由郭秀娟老师统稿, 于全通老师排版设计。

由于编者水平有限, 缺点和不足在所难免, 敬请读者批评指正。

编者
2008 年 6 月

目 录

第 1 章 AutoLISP 语言概述	1
1.1 AutoLISP 语言简介	1
1.1.1 开发 AutoCAD 的重要工具	1
1.1.2 AutoLISP 的特点	1
1.2 AutoLISP 数据类型	2
1.2.1 原子 (ATOM)	2
1.2.2 表和点对	4
1.3 AutoLISP 的程序结构	5
1.4 AutoLISP 的运行环境	7
1.5 AutoLISP 的内存分配	8
1.6 AutoLISP 程序的执行过程	8
练习题	9
第 2 章 数值函数	11
2.1 计算函数	11
2.1.1 (+ <数 1><数 2>...)	11
2.1.2 (- <数 1><数 2>...)	11
2.1.3 (1+ <数>)和(1- <数>)	12
2.1.4 (* <数 1> <数 2>...)	12
2.1.5 (/ <数 1> <数 2>...)	12
2.1.6 (REM <数 1> <数 2>...)	12
2.1.7 (GCD <数 1> <数 2>)	13
2.1.8 (MAX <数 1> <数 2>...)	
(MIN <数 1> <数 2>...)	13
2.1.9 (EXP <数>)	13
2.1.10 (EXPT <底数> <幂>)	13
2.1.11 (LOG <数>)	14
2.1.12 (SQRT <数>)	14
2.1.13 (ABS <数>)	14
2.1.14 (MINUSP <数>)	14
2.1.15 (ZEROP <数>)	14
2.1.16 (NUMBERP <项>)	14
2.1.17 (FLOAT <数>)	15

2.1.18	(FIX <数>)	15
2.2	布尔运算函数	15
2.2.1	(LOGAND <整数> <整数>...)	15
2.2.2	(LOGIOR <整数> <整数>...)	15
2.2.3	(LSH <整数> <次数>)	16
2.3	三角函数	16
2.3.1	(SIN <角度>)	16
2.3.2	(COS <角度>)	16
2.3.3	(ATAN <数 1> [<数 2>])	16
2.4	综合举例	17
	练习题	18

第3章 表处理函数 20

3.1	提取表中数据的函数	20
3.1.1	(CAR <表>)	20
3.1.2	(CDR <表>)	20
3.1.3	(LAST <表>)	21
3.1.4	(NTH <N> <表>) N=0, 1, 2, ...	21
3.1.5	(LENGTH <表>)	22
3.2	构造和修改表的函数	22
3.2.1	(LIST <表达式>...)	22
3.2.2	(APPEND <表>...)	22
3.2.3	(CONS <新的第一个元素> <表>)	23
3.2.4	(REVERSE <表>)	23
3.3	提取并修改表中数据的函数	23
3.3.1	(ASSOC <关键字> <关联表>)	23
3.3.2	(SUBST <新项> <旧项> <表>)	24
3.4	表的循环处理函数	24
3.4.1	(FOREACH <符号原子> <表> <表达式>...)	24
3.4.2	(MAPCAR <函数名> <表 1> <表 2>...<表 n>)	25
3.5	其他表处理函数	26
3.5.1	(MEMBER <表达式> <表>)	26
3.5.2	(LISTP <项>)	26
3.6	综合举例	26
	练习题	27

第4章 符号和函数处理函数 29

4.1	符号处理函数	29
4.1.1	赋值函数	29
4.1.2	禁止求值函数与再求值函数	30
4.1.3	其他符号处理函数	31

4.2 函数处理函数	34
4.3 综合举例	40
练习题	43

第5章 AutoCAD 函数

5.1 GET 类输入函数	45
5.2 图形处理函数	50
5.2.1 (POLAR <点> <方向角> <距离>)	50
5.2.2 (DISTANCE <点 1> <点 2>)	51
5.2.3 (ANGLE <点 1> <点 2>)	51
5.2.4 (INTERS <端点 1> <端点 2> <端点 3> <端点 4> [<任选项>])	51
5.2.5 目标捕捉函数 OSNAP	52
5.3 COMMAND 函数	53
5.4 坐标系转换及单位变换函数	56
5.4.1 (TRANS <点> FROM TO [位移量])	56
5.4.2 (CVUNIT VALUE FROM—UNIT TO—UNIT)	57
5.5 查询和设置系统状态的函数	58
5.5.1 (SETVAR <变量名> <值>)	58
5.5.2 (GETVAR <变量名>)	58
5.5.3 (GETENV <变量名>)	58
5.5.4 (SETCFG CFGNAME EFGVALUE)	59
5.5.5 (GETCFG CFGNAME)	59
5.5.6 (ACAD_COLORDLG colornum [flag])	59
5.5.7 (VER)	59
5.6 显示控制函数	60
5.6.1 (GRAPHSCR)	60
5.6.2 (TEXTSER)	60
5.6.3 (TEXTPAGE)	60
5.6.4 (PRIN1 [<表达式> [<文件标识符>]])	60
5.6.5 (PRINT [<表达式> [<文件标识符>]])	60
5.6.6 (PRINC [<表达式> [<文件标识符>]])	60
5.6.7 (TERPRI)	61
5.6.8 (MENUCMD <字符串>)	61
5.6.9 (GRVECS VLIST [TRANS])	62
5.6.10 (VPORIS)	63
5.7 综合举例	63
练习题	71

第6章 字符串处理函数

6.1 字符与 ASCII 码互换函数	73
6.1.1 (ASCII <字符串>)	73

6.1.2 (CHR <整数>)	73
6.2 数串互换函数	74
6.2.1 (ITOA <整数>)	74
6.2.2 (ATOI <数字串>)	74
6.2.3 (ATOF <数字串>)	74
6.2.4 (RTOS <数> [<模式> [<精度>]])	75
6.2.5 (DISTOF <字符串> [模式])	76
6.2.6 (ANGTOS <弧度表示的角度值> [模式 [精度]])	76
6.2.7 (ANGTOF <字符串> [模式])	77
6.3 字符串处理函数	77
6.3.1 (STRCAT <字符串 1> <字符串 2>...)	77
6.3.2 (SUBSTR <字符串> <开始> <长度>)	79
6.3.3 (STRCASE <字符串> [<模式>])	79
6.3.4 (STRLEN [字符串]...)	79
6.3.5 (WCMATCH <字符串> <样本>)	80
6.4 尺寸标注方式	80
6.5 综合举例	82
练习题	83

第7章 判断函数 85

7.1 相等验证函数	85
7.1.1 关系运算函数	85
7.1.2 逻辑运算函数	85
7.2 条件函数	86
7.2.1 (IF<条件>是——<表达式 1> [<否——表达式 2>])	86
7.2.2 (PROGN<标准表>...)(该函数也称顺序执行函数)	87
7.2.3 (COND(<条件 1> <表达式 1>)	87
7.3 循环函数	88
7.3.1 (WHILE<条件> [<标准表>...])	88
7.3.2 (REPEAT <数> <表达式>...)	90
7.4 函数递归定义	91
7.4.1 递归的概念	91
7.4.2 递归模型	91
7.4.3 递归算法的设计	92
7.5 综合举例	94
练习题	109

第8章 文件处理函数 112

8.1 搜索、获得文件的函数	112
8.1.1 (FINDFILE <文件名>)	112
8.1.2 (GETFILED "对话框标题" "文件名" "扩展名" 标志值)	112

8.2	打开、关闭文件的函数	113
8.2.1	(OPEN <文件名> <方式>)	113
8.2.2	(CLOSE <文描述符>)	114
8.3	用于文件的输入输出函数	114
8.3.1	(READ [字符串])	114
8.3.2	(READ-LINE <文件描述符>)	115
8.3.3	(WRITE-LINE <字符串> <文件描述符>)	116
8.3.4	(READ-CHAR <文件描述符>)	116
8.3.5	(WRITE-CHAR <数> <文件描述符>)	117
8.4	综合举例	117
	练习题	117
第9章	实体和设备访问函数	119
9.1	基本概念	119
9.1.1	实体	119
9.1.2	实体名	119
9.1.3	选择集	119
9.2	选择集操作函数	120
9.2.1	(SSGET str Ptl Pt2)	120
9.2.2	(SSGET "x" [<过滤表>])	121
9.2.3	(SSLENGTH s)	122
9.2.4	(SSNAME <选择集> n) n=0, 1, 2, ..	122
9.2.5	(SSADD <实体名> <选择集>)	123
9.2.6	(SSDEL <实体名> <选择集>)	124
9.3	实体名操作函数	124
9.3.1	(ENTNEXT <实体名>)	124
9.3.2	(ENTLAST)	124
9.3.3	(SSNAME <选择集> <序号>)	125
9.3.4	(NAMED OBJECT)	125
9.3.5	(HANDENT HANDLE)	126
9.3.6	(ENTSEL [<提示>])	126
9.4	实体数据函数	126
9.4.1	获得实体定义数据函数 ENTGET	126
9.4.2	(ENTMOD <实体数据表>)	130
9.4.3	(ENTUPD <实体名>)	130
9.4.4	(ENTMAKE <实体数据表>)	131
9.5	对符号表的访问	135
9.5.1	(TBLNEXT <符号表名> [<第一>])	135
9.5.2	(TBLSEARCH <符号表名> <符号>)	136
9.6	对图形屏幕和输入设备的访问	136
9.6.1	(GRCLEAR)	137

9.6.2 (GRTEXT [<框区> <文本字符串> <加亮>])	137
9.6.3 (GRREAD [track] [allkeys [curtype]])	137
9.7 综合举例	139
9.7.1 实体名和选择集在开发 CAD 程序中的实际应用	139
9.7.2 生成局部放大视图的简便方法	140
9.7.3 求圆或圆弧中心线	143
练习题	147

第10章 建筑设计制图的 AutoLISP 程序范例 149

10.1 常用建筑设计 AutoLISP 程序	149
10.1.1 建筑符号设计 AutoLISP 程序	149
10.1.2 双线墙插门窗 AutoLISP 程序	154
10.2 墙体设计 AutoLISP 程序	161
10.2.1 单线墙 AutoLISP 程序	161
10.2.2 双线墙 AutoLISP 程序	168
10.3 楼梯设计 AutoLISP 程序	181
10.3.1 绘制平面楼梯 AutoLISP 程序	181
10.3.2 剖面楼梯 AutoLISP 程序	187
练习题	190

第11章 AutoCAD 基本命令 192

11.1 基本图形的绘制	192
11.1.1 绘制直线	192
11.1.2 构造线	192
11.1.3 多线	193
11.1.4 多段线	193
11.1.5 矩形	193
11.1.6 正多边形	194
11.1.7 圆	195
11.1.8 弧线	195
11.1.9 椭圆	195
11.1.10 圆环	196
11.2 基本图形的编辑	196
11.2.1 复制图形	196
11.2.2 移动图形	197
11.2.3 偏移图形	197
11.2.4 旋转图形	198
11.2.5 比例缩放图样	199
11.2.6 图形镜像	199
11.2.7 阵列图形	200
11.2.8 修剪图形	201

11.2.9 延伸和拉伸	202
11.2.10 倒角和圆角	203
11.2.11 分解图块	204
附录 A 图形绘制与编辑各命令中英文及简捷命令一览表	206
附录 B 联机程序错误代码	207
附录 C 标准 ASCII 码表	209
附录 D AutoLISP 函数概要	211
参考文献	219

第 1 章 AutoLISP 语言概述

1.1 AutoLISP 语言简介

1.1.1 开发 AutoCAD 的重要工具

LISP (List Processing Language) 是一种计算机的表处理语言, 是在人工智能学科领域广泛应用的一种程序设计语言。AutoLISP 语言是嵌于 AutoCAD 内部的计算机语言, 它是 AutoCAD 开放式体系结构的具体表现, 它是 LISP 语言和 AutoCAD 有机结合的产物。使用 AutoLISP 可直接调用几乎全部的 AutoCAD 命令。AutoLISP 语言既具备一般高级语言的基本结构和功能, 又具有一般高级语言所没有的强大图形处理功能, 是当今世界上 CAD 软件中被广泛采用的语言之一。

美国 AutoDesk 公司在 AutoCAD 内部嵌入 AutoLISP 的目的是让用户充分利用 AutoCAD 进行二次开发, 实现直接增加和修改 AutoCAD 命令, 扩大图形编辑功能, 建立图形库和数据库, 并对当前图形进行直接访问和修改, 开发 CAD 软件包等。在 AutoCAD 为用户提供的 AutoLISP、ARX、VBA 等开发工具中, AutoLISP 是一种简便易学的解释性语言, 具有很强的数据表处理功能, 是开发 AutoCAD 的一种重要手段。

AutoLISP 语言最典型的应用之一是实现参数化绘图程序设计, 包括尺寸驱动程序, 鼠标拖动程序等。尺寸驱动是指通过改变实体标注的尺寸值来实现图形的自动修改; 鼠标拖动即利用 AutoLISP 语言提供的 (GRREAD[<track>]) 函数, 让用户直接读取 AutoCAD 的输入设备 (如鼠标), 任选项追踪光标移动存在且为真时, 通过鼠标移动光标, 调整所需的参数值而达到自动改变屏幕图形大小和形状。

到目前为止, 大多数参数化程序都是针对二维平面图编制的。实际上, 立体图同样可以实现参数化绘图, 在 AutoCAD 中编制实体的立体图参数化程序比其平面三视图程序更简单, 而且立体图生成后, 可以很方便地生成三视图、剖面图和轴侧图等。

AutoLISP 语言还能够利用 PDB 函数驱动 DCL (Dialog Control Language) 文件创建自己的对话框。

自从在 AutoCAD 中嵌入 AutoLISP 以后, 使仅仅作为交互式图形编辑软件的 AutoCAD 变成能真正进行计算机辅助设计、绘图的 CAD 软件。由于 LISP 灵活多样, 又易于学习和使用, 因此使 AutoCAD 成为功能很强的工具性软件。

1.1.2 AutoLISP 的特点

AutoLISP 具有如下特点:

① AutoLISP 语言是在普通 LISP 语言基础上, 扩充了许多适用于 AutoCAD 应用的特殊功能而形成的计算机语言, 是一种仅能以解释方式运行于 AutoCAD 内部的程序设计语言。

② AutoLISP 语言表达式形式为前缀式表达式。

③ AutoLISP 语言中的一切成分都是以函数的形式给出的，它没有语句概念或其他语法结构。执行 AutoLISP 程序就是执行一些函数，再调用其他函数。

④ AutoLISP 语言把数据和程序统一表达为表结构，即 S-表达式，故可把程序当作数据来处理，也可把数据当作程序来执行。

⑤ AutoLISP 语言中的程序运行过程就是对函数求值的过程，是在对函数求值的过程中实现函数的功能。

⑥ AutoLISP 语言比较典型的程序结构是递归方式。由于递归方式的使用，使得程序设计简单易懂。

1.2 AutoLISP 数据类型

AutoLISP 语言主要用到如下数据类型：

整型数	(INT)
实型数	(REAL)
符号	(SYM)
字符串	(STR)
表（及用户定义的函数）	(LIST)
文件描述符	(FILE)
AutoLISP 的内部函数	(SUBR)
AutoCAD 的选择集	(PICKSET)
AutoCAD 的实体名	(ENAME)
函数分页表	(PAGETB)

本节只介绍前五种数据类型，其他类型将在后面相应的章节中介绍。

在上述数据类型中，前四种称为原子（ATOM），原子中包括数值原子（整型数和实型数）、符号原子和字符串原子。

所以 AutoLISP 语言最基本的数据类型是原子和表，它们又总称为符号表达式（Symbolic-Expression），也称为 S-表达式见图 1.1。其中原子（ATOM）是表的最基本元素，它本身只是一个值或符号。

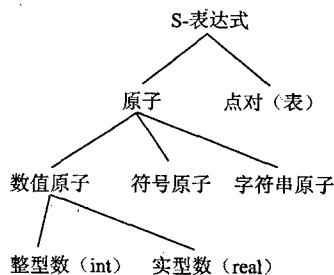


图 1.1 S-表达式

1.2.1 原子（ATOM）

（1）整型数（INT）

整数是由 0、1、2、…、9、+、- 等字符组成，整数的大小与所使用的计算机系统有关。AutoLISP 支持 32 位有符号整数，范围从 -2 147 483 648 到 +2 147 483 647，如整数大小超出此范围，计算机将提示出错信息。但是（GETINT）函数只能取得 16 位有符号整数，范围从 -32 728 到 +32 767。

注意：在实际应用中，若设定和计算结果超出 AutoLISP 语言整数范围时，可改用实型数。

（2）实型数（REAL）

AutoLISP 支持双精度实数，并且至少有 14 位的精度，即整数后跟小数。如果实数的绝对值小于 1，小数点前必须加 0，不能直接以小数点开头，否则被误认为“点对”而出错。

双精度实数 (DOUBLE)，是以 8 个字节存储的实数，共有 64 个位。实数在计算机内是以指数形式存储的。实型数的数值范围比整型数大得多，实型数范围约为 $-1.797\ 693 \times 10^{308}$ 到 $+1.797\ 93 \times 10^{308}$ 它不易超界，故用户可以尽量采用实型数。

数据在内存中的存储方式见表 1.1。

表 1.1 数据在内存中的存储方式

位	63	62	61~52	51~0
用途	符号位 (+/-)	指数符号位 (+/-)	指数位	基数位

其中：指数部分有 10 位，即 $2^{10}=1024$ ，因此，最大的指数值是 2^{1024} 约等于 $10^{308.55}$ ，所以实数的范围约在 $\pm 1.79 \times 10^{308}$ 之间。而基数用 52 位存储一个 0 到 1 之间的纯小数，即 2^{52} 约等于 $10^{15.65}$ ，所以有效位数大约为 16 位，这就是实数运算产生误差的主要原因。

如果运算后产生实数的绝对值大于 1.79×10^{308} 时，AutoLISP 会返回 $\pm 1.\#INF$ (正或负无限大)，这种情况称为 Overflow (高溢)。

AutoLISP 最小的正实数约为 2.228×10^{-308} ，如果运算后产生实数的绝对值小于这个数，便会出现错误，这种情况称为 Underflow (低溢)。

测试实数范围，可用 (EXPT) 函数实现。

例如：

命令: (EXPT 10.0 308) 返回 1.0e+308

命令: (EXPT 10.0 308.55) 返回 1.#INF; Overflow (高溢)

命令: (EXPT 10.0 -308) 返回 错误: 出现异常: 0xC0000093

警告: 忽略展开异常

实型数也可采用科学记数法表示，如 0.12×10^{19} 可表示为 0.12E+19。

注意：双精度实数的有效位可达 16 位，但实际中用 14 位，这里是考虑误差的因素，而 AutoLISP 指令行响应的一般为 6 位有效数字。

(3) 符号(SYMBOL)

① 符号(SYMBOL)包括除左右圆括号 “()”、小数点 “.”、单引号 “'”、双引号 “””、分号 “;” 及全部由数字组成的字符之外的任何可打印字符。

② 符号原子的长度没有限制，命名时要以能够表达清楚变量的含义为主，但尽量不要超过 6 个，否则要占用额外的内存，降低运行速度。

③ 在 AutoLISP 中符号的大小写是等效的，如以下的符号原子都是合法的。

A A12 PC X-38-6 *A

④ AutoLISP 中的任何符号都是有值的，即符号都要赋以一定的数值，或者说符号总是约束在一定值上。一般用赋值函数 SETQ 进行赋值。

例如: (SETQ x 25.0)

含义是将 25.0 赋给 x，这时 x 的当前约束值即为 25.0。一个符号在使用前如没有赋以任何值，则该符号的值为 NIL(空)，它不占用内存空间。

⑤ 符号名避免使用 AutoLISP 的内建函数、常量名称，AutoCAD 的命令、系统变量，acad.pgp 文件内定义的外部命令等。

注意：为区别起见，常用术语“符号”来指存储静态数据的一个符号名，例如内建函数和用户定义函数名是一个符号。

用术语“变量”来指存储程序数据的符号名，如上述(SETQ x 25.0)中的变量名为 x，它

的值为 25.0。AutoLISP 程序中每一个变量都要消耗少量内存，故当变量值不再有用时，重复使用变量名或将变量值设置成 NIL 是良好的程序设计习惯。符号名或变量名不能包含空格字符或分隔符，并总是以字母开头。

⑥ 常量。在程序运行过程中其值保持不变的量称为常量；AutoLISP 有 4 个内建常量，用户在设定变量或自定义函数时，要避免和这 4 个常量同名。

T/t 逻辑真值；
 NIL/nil 逻辑假值，同时也代表空值(或空表)；
 Pi 圆周率 π 值，约等于 3.141592654；
 Pause 双反斜线“\\”字符，用于(COMMAND)函数等待用户输入。

(4) 字符串(STR)

字符串(STR)是由包含在一对双引号内的一组字符组成的。

例如：

"ABC" "135" "Ab C" ""

字符串可以包括任何可打印的字符。字符串中字母的大小写及空格都是有意义的。若字符串中没有任何字符，则为空串""。

当用户在 AutoLISP 表达式中直接使用引号引起来的字符串时，该值被称为字符串常量。

例如："string 1"和"\n Enter first point: "都是有效的字符串。

在用引号引起来的字符串中，用反斜杠“\”字符可以添加控制字符（或换码代码），即反斜杠“\”作为控制字符，与其后的字符组成，控制字符及含义见表 1.2。

表 1.2 控制字符及含义表

控 制 字 符	含 义	用 ASCII 码表示
\\	表示反斜线“\”字符	\\114
\"	表示双引号“”	\\042
\e	表示换码字符 (Esc)	\\033
\n	表示换行	\\012
\r	表示回到行首	\\015
\t	表示移到下一个定位(Tab)	\\011
\nnn	表示八进制码为 nnn 的字符	

注意：“\”后面的字符 e、n、r、t 必须为小写字母。

1.2.2 表和点对

(1) 表(LIST)

在 AutoLISP 语言中，表作为一种基本数据类型，它有如下特点：

- ① 表是指放在一对相匹配的左、右圆括号中的一个或多个元素的有序集合。
- ② 表中的每一个元素可以是任何类型的 S-表达式，既可以是数字、符号、字符串，也可以是表。
- ③ 表中元素与元素之间至少要用一个空格隔开，而元素与括弧之间可不用空格，因为括弧本身就是有效的分隔号。

例如： (15 (a b) c d)

在此例中，表内有 4 个元素，即 15、(a b)、c 和 d，其中第二个元素又是一个表。

④ 表是可以任意嵌套的,上列表中即嵌套了一个表(a b)。表可以嵌套很多层,从外层向里依次称为0层(也称顶层)、1层、2层、…我们所说表中的元素是指表的顶层元素,即0层元素。

⑤ 表中元素是有顺序的,为便于对表中元素进行存取,每个元素都有一个序号。从左向右,第一个元素序号为0,第二个元素序号为1,第*i*个元素序号为*i*-1。

⑥ 表的大小为表的长度,即表中顶层元素的个数。没有任何元素的表称为空表。空表用()或NIL表示。在AutoLISP语言中,NIL是一个特殊的符号原子,它既是原子又是表。

⑦ 表有两种基本类型:标准表和引用表。

a. 标准表:标准表是AutoLISP程序的基本结构形式,AutoLISP程序就是由标准表组成的。标准表是用于函数的调用,其中第一个元素必须是系统内部函数或用户定义的函数,其他的元素为该函数的参数,如上面提到的赋值函数的调用,即采用标准表的形式。

(SETQ x 25.0)

表中第一个元素SETQ为系统内部定义的赋值函数,x和25.0均为SETQ的参数。

b. 引用表:这种表第一个元素不是函数,即不作为函数调用,常作为数据处理,在程序中以如下两种形式存在:

'(a d b)或(QUOTE (a d b))

引用表的一个重要应用是表示图中点的坐标。当表示点的坐标时,表中的元素是用实型数构成的。

表示二维点的坐标是用两个实型数构成的表,如(20.0 30.5),其中第一个元素表示点的X轴坐标,第二个元素表示点的Y轴坐标。三维点的坐标表示,是用三个实型数构成的表,如(20.0 82.5 1.0),其中三个元素依次表示点的X轴坐标、Y轴坐标和Z轴坐标。

(2) 点对(DOTTED PAIR)

点对也是一种表,该表中只有两个元素,两元素中间为一圆点“.”,且圆点与元素之间必须用空格分开。

例如:(A . B)就是一个点对,A、B与圆点均用空格分开,其中第一个元素A为该点对的左元素,第二个元素B为点对的右元素。点对亦可任意嵌套。当使用点对时,切记要注意它的书写格式。

例如:(X . (B . (Y . Z)))为合法点对,而(X .(B . Y). Z)即为非法的。

点对常用于构造关联表。

1.3 AutoLISP 的程序结构

AutoLISP语言没有“语句”这一术语,AutoLISP程序一般是由一个或一系列按顺序排列的标准表所组成。

例如:(SETQ x 25.0)

是上面提到的标准表,又可以看作是一个AutoLISP程序。

建议按照如下原则书写AutoLISP源程序:

① 先写出对称的括号,再填入其内的函数和参数。

② 适当的缩排,使程序各标准表间的主从关系明确化。

③ 做必要的空行,以区分程序中的各个单元。

④ 尽量加上注释,最好连程序流程、变量的类型与用途、用到的AutoCAD系统变量与

命令都加以说明, 方便程序维护。程序注释可以使用任何语言, 但不影响程序的执行。

首先给出几个基本问题实例。

【例 1】 在图形屏幕上, 画一个圆心在 (5 5), 半径为 8 的圆。

```
(DEFUN mm()
  (SETQ r (GETREAL "\n 半径:"))
  (SETQ p (GETPOINT "\n 中心点:"))
  (COMMAND "circle" p r)
)
```

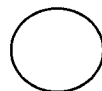


图 1.2 圆

程序执行结果见图 1.2。

【例 2】 编程: 随机输入两点坐标 p1、p2, 求两点距离及两点连线的方位角, 并画出此线段见图 1.3, 程序文件名为 PROG1.lsp。

```
(DEFUN mm1()
  (SETQ p1 (GETPOINT "\n p1:"))
  (SETQ p2 (GETPOINT "\n p2:"))
  )
  (SETQ d (DISTANCE p1 p2))
  (SETQ ang (ANGLE p1 p2))
  (PRINT d)
  (PRINT ang)
  (COMMAND "line" p1 p2 "") ; 在 p1 与 p2 两点间画一条直线
)
```

图 1.3 线段

执行结果:

命令: (LOAD "C:PROG1.LSP") 返回 mm1

命令: (mm1)

命令: p1:30,45

命令: p2:123,89

139.461

0.439063 nil

【例 3】 文件名为 PQ.lsp 的 AutoLISP 文件是由以下程序组成的。

```
(DEFUN mm2()
  (SETQ x 25.0)
  (SETQ y 12.2)
  (+ (* x y) x) ; 表示 x*y+x
)
```

以上是由三个标准表组成的程序, 每个标准表的第一个元素均为系统提供的函数 (如: SETQ, +, *) 称为系统的内部函数。SETQ 为赋值函数, + 为加函数, * 为乘积函数。标准表中的其他元素为相应函数的参数。这个程序是将 25.0 赋给变量 x, 将 12.2 赋给变量 y, 求变量 x 和 y 的值的乘积, 再求此乘积与 x 的和。

AutoLISP 语言源程序的书写格式有如下特点:

① 由于 AutoLISP 语言的一切成分都是函数, 而所有函数又以表结构形式存在, 所以 AutoLISP 程序的所有括号都需要左右匹配。