第3章 程序的流程控制和AutoLISP文件

3.1 程序的流程控制

AutoLISP程序的流程通过流程控制函数控制。

3.1.1 分支结构

1.（cond（测试表达式1 结果表达式 …）[（测试表达式2 结果表达式 …）] …）

该函数从第一个子表起，计算每一个子表的测试表达式，直至有一个子表的测试表达式成立为止，然后计算该子表的结果表达式，并返回这个结果表达式的值。

例如，当i小于等于1时，n=1；i小于等于2时，n=4；i小于等于3时，n=10；在其他情况下n=100。用cond函数实现变量n和i之间以上关系的源代码如下：

```
(setq n(cond ((<=  i  1) 1)
             ((<=  i  2) 4)
             ((<=  i  3) 10)
             (t  100)
    )
)
```

说明：该函数类似于C语言的switch语句，最后一个测试表达式“t(或T)”相当于C语言的“default”，指其余的情况。例如i等于5时，n等于100。也可以缺少这个测试表达式。与C语言switch语句不同的是，若某一测试表达式成立，即返回相应结果表达式的值，不再向下测试。例如，上述表达式中，若i等于0，已满足第一个测试表达式(<= i 1)，则返回结果表达式的1，最后结果是n等于1，运算结束。

2.（if 测试条件表达式 表达式1 [表达式2]）

若测试条件表达式的结果为真，则执行表达式1，否则，执行表达式2。例如：

```
(if  (>  a 1)(setq b 2))
```

该表达式的含义是：如果a大于1，则b等于2，否则不进行任何计算，求值结束。

```
(if  (>  a 1)(setq b 2)(setq b 3))
```

该表达式的含义是：如果 a 大于 1，则 b 等于 2，否则 b 等于 3，求值结束。

注意：该函数最多只有 3 个变元，即测试条件表达式、表达式 1 和表达式 2。先分析下列程序代码：

```
(if (>  a  1)
    (setq  b  2)  (print  (+b  a))
    (setq  b  4)  (print  b)
)
```

该程序段的本意是，若条件成立，b 等于 2，然后打印 a 与 b 之和，否则 b 等于 4，然后打印 b。但是在执行该程序段时，首先检查 if 函数变元的数量。第 1 个变元是(> a 1)，第 2 个变元是(setq b 2)，第 3 个变元是(print (+ b a))，这样，后面的两个表达式就是多余的变元，因此将拒绝执行并显示“; 错误：参数太多：”的出错信息。

若将该程序段改写为以下代码：

```
(if (>  a  1)
    ((setq  b  2) (print  (+b  a)))
    ((setq  b  4) (print  b))
)
```

用括号将条件成立或不成立的多个表达式括起，此时变元的数量改为 3 个，但会出现“;错误：函数错误：2”的出错信息。原因是在条件成立的情况下执行对应的表达式时，首先计算内层表(setq b 2)，将其返回值 2 作为外层表的函数名，显然这是一个不合理的函数名。

利用 progn 函数可以很好地解决了本程序段存在的问题。

3.（prong 表达式…）

该函数将 *n* 个表达式组合起来，作为 if 函数的一个表达式。在执行时，按顺序计算 *n* 个表达式，返回最后一个表达式的计算结果。例如：

```
(if (>a  1)
    (progn  (setq  b  2)
            (print  (+b  a))
    )
    (progn  (setq  b  4)
            (print  b)
    )
)
```

该程序段的执行过程是，若条件成立，则 b 等于 2，然后打印 a 与 b 之和，返回 a 与 b 之和；若条件不成立，则 b 等于 4，然后打印 4，返回 4。

3.1.2 循环结构

1.（repeat 整数 n 表达式 …）

重复执行 n 次，对所有的表达式求值，返回最后一个表达式的计算结果。例如：

```
(setq  a  1  b  100)
(repeat  10
  (setq   a(1+  a))
  ⋮
  (setq  b(+  10  b))
)
```

执行结果：a 等于 11，b 等于 200，返回值为 200。

2.（while 测试式 表达式 …）

若测试结果不为 nil，则执行各表达式，直至测试结果为 nil。例如：

```
(setq  i  1  a  10)
(while  (<=  i  10)
  (setq  a(+  a  10))
  ⋮
  (setq  i (1+  i ))
)
```

执行结果：i 等于 11，a 等于 110，返回值为 11。

例如，定义求解百钱买百鸡的函数。题目是：若母鸡每只 3 元钱，公鸡每只 2 元钱，小鸡每只 0.5 元钱。用 100 元钱买 100 只鸡，有几个答案，每个答案各有几只母鸡、公鸡和小鸡（不包括 0 只），打印所求的结果。

该例没有合适的计算公式，只能利用枚举，试探出合适的结果。首先粗略地分析母鸡 hen 和公鸡 cock 的数量范围。用 100 元钱全部买母鸡，也不会超过 33 只，全部买公鸡，也不会超过 50 只。小鸡 chick 的数量只能是 100－hen－cock。当钱数 3×hen＋2×cock＋0.5×chick＝100 时，hen、cock、chick 的值即为一组解。

【例 3.1】 解百钱买百鸡程序。

```
(defun chicken( / hen cock chick cost)
  (setq hen 1)
  (while (<hen 33)                                    ;母鸡的数量不超过 33
    (setq cock 1)
    (while (<cock 50)                                 ;公鸡的数量不超过 50
        (setq chick (-100 hen cock))                  ;小鸡的数量
        (setq cost (+(* 3 hen) (* 2 cock) (* 0.5 chick)))   ;3 种鸡的钱数
            (if (=cost 100)
```

```
            (print (list "母鸡=" hen "公鸡=" cock "小鸡=" chick))
        )
        (setq cock (1+cock))                    ;公鸡的数量加 1
    )
    (setq hen (1+hen))                          ;母鸡的数量加 1
)
(princ)                                         ;静默退出
)
```

说明：

① 程序的第 2 行(setq hen 1)不可缺少，否则第 3 行(while (<hen 33)中的 hen 将是无定义的。

② 第 4 行(setq cock 1)若改写在第 2 行(setq hen 1 cock 1)，虽然没有语法错误，但内层循环 cock 的值第一次从 1 增加到 50 后，就固定为 50，出现了算法错误。

③ 程序的第 11 行(setq cock (1+cock))不能只写成(1+cock)，表达式(1+cock)返回 cock+1 的值，但 cock 的值不变，这是 1+函数与 C 语言++ 运算的不同之处。

④ 第 9 行(print (list "母鸡=" hen" 公鸡="cock" 小鸡="chick))若写成(print "母鸡=" hen "公鸡=" cock "小鸡=" chick)是错误的，因为 print 函数只需一个变元。

⑤ 程序结束前的(princ)表达式不打印任何值，因为它是该函数的最后一个表达式，所以该函数将不返回任何值，这样就不会干扰 print 函数的正常打印信息。

加载该程序之后，在“命令：”提示下输入 (chicken)，输出以下结果：

```
("母鸡=" 2 "公鸡=" 30 "小鸡=" 68)
("母鸡=" 5 "公鸡=" 25 "小鸡=" 70)
("母鸡=" 8 "公鸡=" 20 "小鸡=" 72)
("母鸡=" 11 "公鸡=" 15 "小鸡=" 74)
("母鸡=" 14 "公鸡=" 10 "小鸡=" 76)
("母鸡=" 17 "公鸡=" 5 "小鸡=" 78)
```

3.2 AutoLISP 程序文件

3.2.1 AutoLISP 文件的特点

AutoLISP 文件的扩展名为 lsp，是由若干个 AutoLISP 表达式构成的。

虽然在命令提示下，通过输入 AutoLISP 表达式的方法可以定义或调用 LISP 函数，但不便于保留或调试程序，因此，都是以文件形式实现函数的定义或调用的。

一个 LISP 文件可定义多个函数或 AutoCAD 命令。

表达式相当于语句。一个表达式可以分写在若干行上，一行可以写若干个表达式。连续的多个空格相当于一个空格。以下是一个表达式分写在若干行上的实例：

```
(defun plus(x y)
```

```
    (+ ( * x y) x)
)
```

以下是一行写若干个表达式的实例：

```
(setq a 2.0) (setq b 4.0) (+a b)
```

由于在 AutoLISP 程序中含有大量括号，因此使得程序代码不易阅读。解决这个问题的方法就是缩排对齐格式。程序代码行嵌套的层次越深，越向右缩进。例如，定义下例 $f(x)$函数。

$$f(x)=\begin{cases}1 & (x>0)\\ 2 & (x=0)\\ 0 & (x<0)\end{cases}$$

不采用缩进格式书写，形式如下：

```
(defun fun(x)(cond ((>  x  0) 1)((=  x 0) 2)((<  x  0) 0)))
```

采用缩进格式书写，形式如下：

```
(defun fun(x)
    (cond ((>  x  0)  1)
          ((=  x  0)  2)
          ((<  x  0)  0)
    )
)
```

显然后者便于程序的阅读和调试。Visual LISP 提供了文本格式编排器，可以将随意书写的程序更新为缩进格式的程序。

3.2.2 程序中的注释

注释可增加程序的可读性，不仅便于对程序的阅读和调试，而且也便于对程序的维护、移植和扩充。注释的内容可以是：程序的作者、创建日期、功能、使用方法、变量或函数的说明等。从 AutoCAD 2000 开始，AutoLISP 支持全中文环境，可以用中文注释。

注释的形式可以是整行、整段或行间注释。

整行或后半行注释以分号“;”开头，至行尾为注释部分。例如：

```
(setq area (* pi r r))                          ;计算圆的面积
```

整段或行间注释以“;|”开头，以“|;”结尾，之间可以短到几个字符，长到若干行。例如，行间注释：

```
(setq tilemode ;|此处添加注释|; (getvar "tilemode"))
```

下面是整段的注释：

```
(setvar "orthomode" 1) ;|注释起始部分
    连续注释,
    ⋮
    注释到此为止|; (princ "\nORTHOMODE set On.")
```

下面是一个带有注释的完整的程序。

【例 3.2】 定义打印 ASCII 码为 33～90 的字符的命令。

```
;该程序打印 ASCII 码为 33~90 的字符
;在命令提示下,输入 pras
(defun c:pras(/ as )                    ;定义 pras 为 AutoCAD 命令,as 为局部变元
  (setq as 33)                          ;设置 as 为第一个 ASCII 码 33
  (while(<=  as  90)                    ;while 循环开始
    (princ(chr as))                     ;打印 ASCII 码为 as 的字符
    (terpri)                            ;换新行
    (setq as (1+as))                    ;设置 as 为 as 的下一个 ASCII 码
  )                                     ;while 循环结束
)                                       ;命令定义结束
```

3.2.3 在 AutoCAD 环境下加载 AutoLISP 文件

1. 命令行方式

加载 AutoLISP 文件用 load 函数,调用 load 函数的格式如下:

命令:

```
(load  "驱动器:\\路径\\文件名"  ["出错信息"])
```

若加载成功,则返回被加载 lisp 文件的最后一个表达式的结果;若最后一个表达式是函数的定义,则返回该函数名。若加载失败,则返回用户定义的出错信息;若用户没有定义出错信息,则返回加载失败的信息。

例如,文件 file1.lsp 最后一个表达式是定义函数 func1,它的路径是 d:\user1。加载该文件的表达式如下:

命令:

```
(load  "d:\\user1\\file1"  "没有找到这个文件!")
```

若加载成功,则返回函数名 func1,否则返回"没有找到这个文件!"这个信息。

如果调用该函数时省略了"出错信息",例如:

命令:

```
(load  "d:\\user1\\file1")
```

那么,若加载成功,则返回函数名 func1,否则返回";错误: LOAD 失败: "d:\\ user1 \\file1"。

2. 对话框方式

选择菜单“工具”→“加载应用程序”，或者“工具”→AutoLISP→“加载”，或者在“命令：”提示下输入 appload 命令，都可通过随后弹出的“加载/卸载应用程序”对话框加载或卸载 AutoLISP 文件。

3. 自动加载

AutoCAD 在启动时，可以自动加载四个 LISP 文件：acad.lsp、acad2010.lsp、acaddoc.lsp 和 acad2010doc.lsp。用户可以创建和维护这些文件，其中 acad.lsp 和 acaddoc.lsp 只能由用户来创建。AutoCAD 在加载过程中不报告是否找到或是否加载这些文件的信息。AutoCAD 在加载菜单时，自动加载了与菜单文件同名的扩展名为 mnl 的 LISP 文件。

4. 间接自动加载

如果把调用 autoload 函数的表达式写在自动加载的 acad2010doc.lsp 等文件内，那么在 AutoCAD 启动时，随着 acad2010doc.lsp 等文件的自动加载，被调用的 autoload 函数还可以加载一些 LISP 文件。调用 autoload 函数的格式如下：

命令：

```
(autoload  LISP文件名  命令表)
```

该函数要求 LISP 文件必须在 AutoCAD 支持的文件搜索路径下，命令表列出了在该 LISP 文件中定义的部分的 AutoCAD 命令名。无论是否加载成功，该函数均返回 nil。

autoload 函数与 load 函数的不同之处是：执行完调用该函数的表达式之后，AutoCAD 只是记录了被加载的 LISP 文件名和相应的命令表，而 LISP 文件本身并没有被加载。只有等到命令表内的任意一个命令被调用之后，这个 LISP 文件才被真正加载，这时该 LISP 文件所定义的全部 AutoCAD 命令才处于可以被调用状态。也就是说，如果某个 LISP 文件所定义的命令没有被调用，那么这个 LISP 文件就暂时不被加载到内存，从而节省了内存空间。

例如，文件 file1.lsp 定义了 cmd1、cmd2 等多个 AutoCAD 命令，该文件存放在 AutoCAD 的 support 目录下。在命令行用 autoload 加载 file1.lsp 的表达式如下：

命令：

```
(autoload  "file1.lsp"  '("cmd1"  "cmd2"))
```

此后如果调用 file1.lsp 文件定义的命令表之外的命令，会显示“未知命令"×××"。按 F1 查看帮助。”的出错信息，因为该 LISP 文件尚未被真正地加载。如果调用命令表内的任意一个命令，例如调用 cmd1 命令，file1.lsp 文件才被 AutoCAD 真正地加载，随后该文件定义的所有 AutoCAD 命令均可被调用。

如果将表达式(autoload "file1.lsp" '("cmd1" "cmd2"))，追加到 acad2010doc.lsp 等文件的后面，即可在启动 AutoCAD 时，间接自动地加载了 file1.lsp。

习　题

3.1　加载以下 5 个表达式之后，a、b、c、d、e 的值分别是多少？

```
(setq a 1)
(if (/=a 0) (setq b 2))
(if (=a 0) (setq c 3) (setq c 4))
(if (>a 0) (setq d (* b c)) (setq d (* a c)))
(if (and (>b 2) (<c 3)) (setq e(+a b))(setq e(-c d)))
```

3.2　下列程序能否正常执行？为什么？

```
(setq a 1 b 100)
(while (<a 10)
       (setq b (* b 10))
       (1+a)
)
```

3.3　执行下面表达式能否获得正常结果？为什么？

```
(repeat 10) (setq m(+n 10))
```

3.4　下面表达式能否执行？若不能，请改正。

```
(setq x 1 y 2)
(if (>x 0) (setq y (+x-y)) (setq y(+x y )))
```

3.5　下面是函数 fun 的定义。若 x 的值依次是－5、5、15、25，先预测 n 的值，然后通过程序验证预测的结果。

```
(defun fun(x)
(setq a 1 b 2)
(setq n(cond((<x 0) (+a b))
      ((<x 10) (-a b))
      ((<x 20) (* a b))
      (t (/ a b))
    )
    )
)
```

3.6　编写将自然数从 1～n 累加的程序。

3.7　编写程序，求解用壹圆的人民币兑换壹分、贰分、伍分硬币的所有兑换方案（不包括某种硬币为 0 的解）。

3.8　一球从 100 米的高度自由落下，每次落地后反跳回原高度的一半。编写程序，求解

该球在第10次落地时共经过多少米，并求其第10次反弹的高度。

3.9 某猴第一天摘了若干桃子，当即吃了一半，还不过瘾，又吃了一个。第二天早上它又吃了剩下桃子的一半零一个，以后每天早上都吃前一天剩下的一半零一个。第10天早上想再吃时，只剩下一个桃子了。编写程序，求解该猴第一天摘了多少桃子。