

目 录

前言

| | |
|------------------------------|-------|
| 一、软件加密技巧 | (1) |
| 1. 问题的提出 | (1) |
| 2. 从救回 BASIC 程序谈起 | (1) |
| 3. 几种简单的加密措施 | (4) |
| 4. 几种解密方法 | (5) |
| 5. 软件加密保护实例 | (8) |
| 6. 文件名加控制符的加密和解密 | (13) |
| 7. 同时采用几种加密措施 | (14) |
| 8. 磁盘加密保护 | (16) |
| 二、机器语言使用技巧 | (20) |
| 1. 子程序的建立 | (20) |
| 2. 子程序的存贮 | (25) |
| 3. 子程序的调用 | (26) |
| 4. 子程序使用技巧 | (32) |
| 5. 实用经验和技巧 | (45) |
| 三、实用管理技巧 | (58) |
| 1. 中华学习机的自动操作 | (58) |
| 2. 菜单式的DOS管理程序 | (62) |
| 3. 自动换页打印程序清单 | (66) |
| 4. 接受任意字符的输入 | (68) |
| 5. DOS的阻截失误 | (70) |
| 6. 内存程序的截断与混乱 | (73) |
| 7. 机器语言子程序转换成 BASIC 程序 | (7 |
| 8. 显示 T 文件内容的方法 | |

| | |
|----------------------|--------|
| 9. 同时打开多个文件的方法 | (82) |
| 10. N组实验数据的合并..... | (85) |
| 11. 定义更多的功能键..... | (91) |
| 12. 自动行号编排..... | (95) |
| 四、信息处理技巧..... | (100) |
| 1. 数据存贮 | (100) |
| 2. 数据交换 | (101) |
| 3. 数据删除 | (103) |
| 4. 数据插入 | (110) |
| 5. 数据修改 | (120) |
| 6. 数据合并 | (124) |
| 7. 数据分类 | (132) |
| 8. 数据检索 | (156) |
| 9. 堆栈技术 | (164) |
| 10. 链接技术..... | (173) |
| 五、图形数据处理 | (199) |
| 1. 一个表演程序 | (199) |
| 2. 不同画面的显示 | (201) |
| 3. 图形的快速合并 | (203) |
| 4. 图形的转移 | (206) |
| 5. 机器语言绘图子程序搬家 | (208) |
| 6. 一个简单的绘图程序 | (210) |
| 7. 图形自动显示和打印 | (213) |
| 8. 图形的对称处理 | (215) |
| 9. 按键控制运动方向 | (216) |
| 10. 全屏幕作图..... | (217) |
| 11. ASCII字符造型程序..... | (223) |
| 12. 电路图绘制..... | (229) |
| 13. 回归曲线拟合程序..... | (240) |

| | |
|--------------------------------|-------|
| 14. 机器语言编制的造型表..... | (246) |
| 六、软件功能扩展技术 | (256) |
| 1. APPLESOFT 中的ERASE 语句模拟..... | (256) |
| 2. 程序运行中的变量表列印... .. | (260) |
| 3. 磁盘文本文件的列印 | (262) |
| 4. 磁盘信息的显示和修改 | (269) |
| 5. 数据管理功能之扩充 | (279) |
| 6. 全屏幕程序编辑系统 | (284) |
| 七、文件管理 | (297) |
| 1. 随机处理文件 | (297) |
| 2. 顺序处理文件 | (309) |
| 八、实用 汉字程序 | (319) |
| 1. 中华学习机 LOGO 语言引导程序 | (321) |
| 2. 评委亮分 | (322) |
| 3. 预测身高 | (323) |
| 4. 中华学习机多功能引导程序 | (325) |
| 5. PC-1500袖珍机语句及键名称一览表 | (326) |
| 6. 中华学习机区位码..... | (331) |
| 7. 图书资料管理程序 | (334) |
| 8. 计算机辅助教学 | (340) |

一、软件加密技巧

1. 问题的提出

随着计算机在我国的日益普及和广泛使用，特别是在经济、铁路、电力、银行、公安、气象、民航、军事等重要部门和全国性信息系统中的应用，对软件和程序采取加密措施，已是程序设计中不可缺少的一个环节。对初学计算机语言并有一定编程能力的人来说，为使自己的程序有一定的权威，了解和掌握简单的保密措施及解密方法，也是必要的。

下面介绍的一些保密方法比较简单，但有一定的实用价值，如果同时采取几种加密措施，可能收到较为满意的保密效果。当然，限于篇幅，我们不可能也没有必要讲述高难度的保密方法，因为，对于具有专门知识而又存心解密的行家来说，总有解开的办法。

2. 从救回 BASIC 程序谈起

假设有一个程序 PRO-1:

PRO-1

```
5  REM PRO-1
10 A = 10
20 B = 5
30 C = 4
40  PRINT A + B - C
50  PRINT A - B + C
60  END
```

这是一个十分简单的BASIC程序，无需多加解释。我们想说明的是，它在内存程序区中是如何存放的。

用监控命令，可以看到程序区的内容如下，见PRO-2：

PRO-2

*800.841

```
0800- 00 0B 0B 05 41 00 B4 20
0808- 00 00 00 42 00 B3 20 00
0810- 00 00 43 00 B3 00 00 00
0818- 00 D0 35 00 24 0B 1E 00
0820- 43 D0 34 00 2F 0B 2B 00
0828- BA 41 C8 42 C9 43 00 3A
0830- 0B 32 00 BA 41 C9 42 C8
0838- 43 00 40 0B 3C 00 B0 00
0840- 00 00
```

有人会提出疑问，你如何知道上述BASIC程序，在内存中是从\$0800开始，到\$0841结束的？\$0800是用户区的首地址（指文本状态），或者说是存放程序的起始地址；而在监控状态下往后扫描程序区，连续出现三个全0字节，就是程序尾。所以，我们在用CALL-151↵后出现*时，键入0800↵，然后继续扫描（即不断按RETURN键），看到连续出现三个全0字节时停止，就找到了上述BASIC程序对应的机器语言程序的程序尾。

现在，我们假定用了NEW命令，再用LIST命令时，显然，原程序的清单什么也列不出来。

我们再回到监控状态，键入800·841↵，再察看程序区，发现\$801和\$802地址中的内容改变了，分别由0B 08改为00 00了，但其它地址中的内容，并没有任何变动。这就给我们一个启发，恢复\$801和\$802中的值，就恢复了

链头，这样程序就连接上了，可用 LIST 命令打出清单。

具体的做法是，跳过 \$801 至 \$804（它们是链指针和行号），从 \$805 开始往后查找第一次出现的 00 字节，这是程序行的终止标志。下一个字节为另一个程序行的头（链指针），本例中，它的位置是 \$080B。这样，只要把 \$801 中改为 \$0B，把 \$802 中改为 \$08，程序就连接上了。用 LIST 命令，就可以看到上述被 NEW 掉的 BASIC 程序。

即在监控状态下，键入 801: 0B 08↵即可，返回 BASIC 状态，LIST 就起作用。

此时应注意，不要 RUN 上述 BASIC 程序。

如果想运行上述 BASIC 程序，应在监控状态下，扫描查找连续三个全 0 字节，它就是程序尾，而三个全 0 字节后的下一个地址，本例是 \$0842，必须把这个值放入变量表的首址 \$69 和 \$6A 中去，即把 \$69 号单元改为 \$42，而把 \$6A 号单元改为 \$08。具体做法是：]CALL-151↵

*69: 42 08↵

返回 BASIC 状态，这样，恢复了的程序即可运行。

RUN↵

11

9

]

综上所述，只要把程序区的链头接上（本例是 \$801 单元改写为 \$0B，\$802 单元改写为 \$08），被 NEW 掉的程序即可再现。把变量表首指针改过来（本例是把 \$69 单元改为 \$42，\$6A 单元改为 \$08），即可重新运行程序，否则会把程序冲掉。

若要键入新的程序行（包括修改），或将程序存盘，程序

的尾指针也应改过来。程序尾指针地址是\$AF和\$BO, 即AF: 42 08↙。

上述BASIC程序, 没有开辟数组, 若有数组, 则数组首、尾指针也应作相应改动, 他们的指针地址分别是\$6B, \$6C和\$6D, \$6E。

上面介绍了如何恢复一个被破坏了的程序的方法, 通俗的说, 就是救回BASIC, 从这个例子, 给我们一个有益的启示, 人为地破坏(或修改)链指针(链头), 可以用于程序的加密保护。

事实上, 将链值作任何修改, 都可以从表面上破坏一个程序。下面介绍的几种保密措施, 都是基于上述思想。

3. 几种简单的加密措施

前面已经指出, 人为地(有意识地)破坏链指针, 可以用于程序的加密。众所周知, 在APPLE-II或紫金-II机上, BASIC程序的存贮是从2049(即16进制的801)这个地址开始的, 因此, 只要设法改动\$801单元的内容, 就可以使LIST命令失效。

(1) POKE 2049, 0

在编制并调试好程序后, 键入POKE 2049, 0。结果程序可以运行, 但用LIST时, 只能在屏幕上显示第一行程序。程序的其它部份什么也看不到, 从而达到程序的保密效果。

(2) POKE 2049, 1

如果在程序调试完毕后, 键入POKE 2049, 1, 那么执行LIST命令, 屏幕上出现连续显示的程序第一行, 而毫无休止的进行, 这也起到了程序的保密作用。执行RUN命令,

程序正常运行。

(3) POKE 214, 128

将要保密的程序运行一次后，键入 POKE 214, 128，再键入任何指令或键盘上的任意符号，结果均变成执行 RUN 命令，而使程序自动运行。当然，程序的清单是列不出来的，并且一行也没有。

(4) POKE 2049, n

为了达到程序加密的目的，把 \$801 单元内容置成 n (n 是一个小于 255 的任意正整数)，此时情况又不同，屏幕上可能显示一些乱七八糟的东西。

总之，改动某些地址单元的内容，将使列表指令失效，达到程序加密的效果。不言而喻，这些人为地破坏程序的方法，是可以重新恢复的。

4. 几种解密方法

(1) 0 保密的程序

对用 POKE 2049, 0 保密的程序，解密的方法之一是，键入一个行号（除已经保存的程序的第一个行号外），然后回车，再执行 LIST 命令，即可显示原程序清单，这种方法简单有效。

(2) 1 保密的程序

对用 POKE 2049, 1 保密的程序，用上述简单的方法，键入任意一个行号来解密就失灵了。

破密的方法之一，见 2. 中救回 BASIC 程序的步骤和方法，因为，人为地破坏链指针，只是局部的，而不是大面积的，所以可以用 2. 中类似的方法进行修复。

考虑到用 2. 中的方法比较烦琐，现改用机器语言子程序

的方法。

如前所述, 一个 NEW 命令, 可以清除内存现行程序及其变量和数组, 这个命令的核心是取下程序区的头一个链指针, 并把它置 0 (即作为程序尾标志), 同时, 将程序尾、变量首、数组首, 尾指针都改成 LOMEM+2 的值。而程序区的其它部份以及变量表、数组表中的内容等并没有改变, 只要将程序区的链头接上, 被 NEW 删掉的程序立即可以再现出来。至于要重新运行程序, 还需把变量表首指针改正过来, 其它诸如修改, 增加新的程序行, 存盘等等, 程序的尾指针也应作相应的改动。所有这些, 都要了解 BASIC 程序在内存中的存贮方式, 这里不再赘述。

下面介绍的是一个机器语言子程序, 只要在监控状态下, 键入从 \$ 6000 开始到 \$ 6065 结束的机器码, 并把它以 BSAVE PRO-3, A \$ 6000, L \$ 65 存盘, 就可随时救回被 NEW 删掉的 BASIC 程序。同时, 存盘的 PRO-3 程序, 还可以对某些加密程序, 进行破译。

机器语言子程序 PRO-3:

PRO-3

*6000.6065

| | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|
| 6000- | A9 | 00 | 85 | 07 | 85 | 06 | A9 | 08 |
| 6008- | 85 | 08 | A0 | 05 | B1 | 07 | C9 | 00 |
| 6010- | F0 | 0F | C8 | D0 | F7 | E6 | 08 | D0 |
| 6018- | F3 | B1 | 07 | C9 | 00 | F0 | 18 | D0 |
| 6020- | EB | A5 | 06 | C9 | 01 | F0 | 2D | A5 |
| 6028- | 08 | 8D | 02 | 08 | 98 | 8D | 01 | 08 |
| 6030- | A9 | 01 | 85 | 06 | 4C | 12 | 60 | CB |
| 6038- | C8 | C0 | 01 | F0 | 06 | 30 | 04 | 4C |
| 6040- | 46 | 60 | ED | 08 | D0 | 00 | 98 | 85 |
| 6048- | 69 | 85 | AF | A5 | 08 | 85 | 6A | 85 |

```

6050- B0 4C 5B 60 C8 D0 C2 E6
605B- 0B D0 BE EE 01 08 D0 03
6060- EE 02 0B 4C BF 00

```

假设有一个BASIC程序PRO-4，若对它执行NEW命令，或者进行了加密保护，现在可以利用机器语言子程序PRO-3，对程序PRO-4进行解密，下面是运行的实例。

PRO-4

ULIST

```

5  REM PRO-4
10 A = PEEK (43634) + PEEK (4363
    5) * 256
20 B = PEEK (43616) + PEEK (4361
    7) * 256
30 PRINT "START ADDRESS: ";A
40 PRINT "LENGHT: ";B

```

UNEW

ULIST

UCALL24576

```

5  REM PRO-4
10 A = PEEK (43634) + PEEK (4363
    5) * 256
20 B = PEEK (43616) + PEEK (4361
    7) * 256
30 PRINT "START ADDRESS: ";A
40 PRINT "LENGHT: ";B

```

操作方法说明如下：

- CALL-151↵，键入 6000 · 6065 的机器语言子程序。

- 按 CTRL-RESET，返回 BASIC 状态。

- 存盘：BSAVE PRO-3，A\$ 6000，L\$ 65。

- 清内存：NEW↵

- 调一个 BASIC 程序，如 LOAD PRO-4↵，LIST↵，看到 PRO-4 程序的清单。

- NEW↵，再 LIST↵，则名为 PRO-4 的程序被清除，屏幕上什么也没有。

- BLOAD PRO-3↵

- 键入 CALL 24576↵，再 LIST↵，PRO-4 程序又复现在屏幕上。

注意：在救回 BASIC 程序 PRO-4 前，必须先调 PRO-3 程序进内存，否则 CALL 24576↵不起作用。

上面介绍的方法，无论对 POKE 2049, 0、对 POKE 2049, 1、或者对 POKE 2049, n 保密的程序，都有满意的解密效果。应用本节介绍的方法时，注意在使用 CALL 24576↵后，可能进入监控状态，那不要紧，只要按 RESET 键，返回 BASIC 状态，就可进行 LIST 和 RUN 了。

5. 软件加密保护实例

下面是一个程序加密的实例，其措施是用“暗码锁”锁住，只有知道暗码的人，才能打开该“锁”，这种方法称之为口令保护，它的想法比较自然，也适合人们对“保密”概念的理解。于是计算机技术和常规的保密技术结合，交相辉映。

(1) 编制密码

为了实现口令保护, 首先应确定口令, 而这个口令应该是经常变换的, 并且让计算机自动编制。例如键入一组字符或数字, 计算机自动将它变成一组密码。方法是计算机随机产生一个整数, 用它与输入字符的 ASCII 码运算, 而得到新的 ASCII 码, 这样, 输入的一组字符就变成别人无法理解的一组密码了。

程序可以这样编写: 见 PRO-5:

PRO-5

LIST

```
5  REM PRO-5
10 A = INT ( RND (1) * 100)
20  INPUT M$
30  FOR I = 1 TO LEN (M$)
40  A$ = MID$ (M$, I, 1)
50  E = ASC (A$) + A
60  IF E > 255 THEN E = E - 255
70  N$ = N$ + CHR$ (E)
80  NEXT I
90  PRINT N$
100 END
```

程序 PRO-5 中的90句, 就是输入字符变成的密码。由于 A 值是一个随机整数, 重复运行上述程序, 每次产生的密码均不同。在实际使用上述程序时, 应删去90句。

(2) 破译密码

由于上述编制密码的过程是随机的, 即是输入相同的字符, 每次产生的密码都不一样, 所以还要一个破译密码的程序, 否则连自己也不能轻易改动源程序, 甚至可能无法运行自己的程序。

破译密码的程序，其设计思想同上，只要改动50句的E值为减A，60句的E值为加255，此外，还应保留密码程序中的A值。

(3) 程序实例

例如，我们真正需要保密的程序是：

```
500 FOR I=1 TO 10: PRINT I;  
    " ";: NEXT I
```

为简单起见，我们仅举了一行程序，实际上它可以是一个系统软件。

参照前面介绍的设计思想，可以方便地写出程序 PRO-6，注意 PRINT N\$ 的语句已删除。为说明方便，A 值取一固定常数，如 $A = 20$ 。

PRO-6

LIST

```
5 REM PRO-6  
10 A = 20: INPUT "1 OR 2 OR 3 ";  
    B: PRINT  
20 ON B GOSUB 30,70,28  
25 GOTO 10  
28 PRINT "END": END  
30 INPUT M$:C = LEN (M$): FOR I =  
    1 TO C:A$ = MID$ (M$,I,1)  
40 E = ASC (A$) + A: IF E > 255 THEN  
    E = E - 255  
50 N$ = N$ + CHR$ (E): NEXT  
60 RETURN  
70 INPUT M$: FOR I = 1 TO LEN (M  
    $):A$ = MID$ (M$,I,1)  
80 E = ASC (A$) - A: IF E < 0 THEN  
    E = E + 255  
90 N$ = N$ + CHR$ (E): NEXT  
100 PRINT "INPUT K$"
```

```

105 INPUT K$
110 IF K$ < > N$ THEN 10
120 GOSUB 500
130 RETURN
500 FOR I = 1 TO 10: PRINT I;" ";
    : NEXT I
510 PRINT
520 RETURN

```

下面是几个运行实例：

(a)

```

URUN
1 OR 2 OR 3      1

?ASD
1 OR 2 OR 3      2

?UgX
INPUT K$
?UgXASD
1 2 3 4 5 6 7 8 9 10
1 OR 2 OR 3      3

```

(b)

```

END

URUN
1 OR 2 OR 3      1

?ASD
1 OR 2 OR 3      2

?UgX
INPUT K$
?UgSASD
1 OR 2 OR 3      3

END

```

(c)

```

URUN
1 OR 2 OR 3      1

```



```

(c)      ?135
          1 OR 2 OR 3    2

          ?EGI
          INPUT K$
          ?EGI134
          1 OR 2 OR 3    3

          END

```

```

(d)      URUN
          1 OR 2 OR 3    1

          ?135
          1 OR 2 OR 3    2

          ?EGI
          INPUT K$
          ?EGI135
          1 2 3 4 5 6 7 8 9 10
          1 OR 2 OR 3    3

          END

```

说明:

- 运行上述程序后, 出现 1 OR 2 OR 3 的提示, 键入 1, 出现? 后, 打入任意字符, 如 ASD, 又出现 1 OR 2 OR 3, 键入 2, 再次出现? 后, 键入经编制的密码, 如 ASD 的密码为 U9X, 屏上出现 INPUT K\$ 时, 键入 U9X ASD, 则程序可以正确运行, 本题是打印 1—10 的 10 个数字, 最后又一次出现 1 OR 2 OR 3, 打入 3, 程序结束, 并打印出 END 信息。反之, 在出现 INPUT K\$ 时, 若上例键入 U9S ASD, 则不能执行本程序, 打入 3 结束。

- 如果不了解本程序的原理结构, 别人是无法运行程序

的，因为他们不知道 ASD 的密码是 U9X，同时，也不知道在出现 INPUT K\$ 时，如何送数以及送什么数。

- 程序中的10句 A = 20, 可以改成 INPUT "A = "; A, 这样，即使送 ASD 这三个字符，但由于 A 值变了，ASD 的密码也跟着改变。

- 为使程序有更完善的保密手段，再加上其它措施，使程序清单列不出来。

- 本程序还可以进一步完善，如加上菜单技术，便于操作；整个程序改成循环，使程序自动进行等等。

6. 文件名加控制符的加密和解密

采取在文件名中掺和一些隐含字符的方法，也可以达到保护程序的目的。如用 CTRL 加字符 A、G、D 等等。因为控制字符被用于文件名时，将不被显示。

例如，要把一个程序存入盘中，可键入

] SAVE PROG CTRL-A . BJO ✓

当用 CATALOG 列目录时，控制符不会显示出来，从而达到文件的保密，使得不知道隐含字符的人根本调不出文件。

当用 LOAD PROG. BJO 或 RUN PROG. BJO 时，屏幕上均显示“FILE NOT FOUND”字样。

对于上述在文件名中加夹若干非打印控制码的方法，也极易破解，例如执行程序 PRO-7，再列目录时，会显示出 PROG A. BJO，其中 A 是以反相方式显示的。用 LOAD PROG CTRL-A. BJO 即可列出程序清单。

PRO-7 ULIST

```
10 REM PRO-7
20 POKE 44572,223: POKE 44573,188

30 FOR I = 0 TO 12
40 READ A
50 POKE 48351 + I,A
60 NEXT I
70 PRINT CHR$(4); "CATALOG"
80 DATA 201,160,144,4,32,237,253,
    96,105,128,76,227,188
```

7. 同时采用几种加密措施

(1) 为避免 BASIC 程序清单被列出,设法取消 LIST 功能,方法是把语句 POKE 214,128 写在 HELLO 程序的开头,因为机器在启动后,首先执行的就是 HELLO 程序,所以,在系统刚启动后便取消 LIST 功能。

(2) 为避免程序在运行过程中,被 CTRL-C 中断,可以在加密程序中加入如下语句:

```
5 ONERR GOTO 1000
1000 NEW
```

这样,在程序运行过程中,即使按 CTRT-C 时,则程序会转到 1000 句而执行 NEW 命令,结果清除了原程序。

或者安排以下语句:

```
5 ONERR GOTO 200
200 POKE 214,128
```

这样每按一次 CTRT-C,程序再运行一次。

(3) 为了消除原程序,设法在程序运行结束后,重新启

动DOS，做法是在程序的最后一句，设置 PRINT CHR\$ (4);“PR#6”语句。也可以安排 PRINT CHR\$ (4);“CATALOG”等语句。

(4) 为了防止在程序运行过程中，被RESET后而暂停，可以改变RESET的入口地址\$3F2和\$3F3，这样就破坏了RESET的执行。

例如，将RUN的入口地址\$D566放入\$3F2和\$3F3中，也可以放其它的数在\$3F2和\$3F3中。

下面是两个加密程序实例，为简单计，加密的内容仅仅是40句一句，POKE 214,128已放在招呼程序HELLO中，存盘采用隐含字符的方法。

PRO-8 ULIST

```
0 REM PRO-8
1 PRINT : PRINT
2 D$ = CHR$ (4)
5 ONERR GOTO 1000
10 REM CHINE RESET ADDRESS
20 POKE 1010,66
30 POKE 1011,200
40 FOR I = 1 TO 500: PRINT I;" ";
   : NEXT I
45 PRINT
50 PRINT D$;"PR#6"
1000 NEW
```

在执行程序 PRO-8时，若按 RES TE，又重新启动。在运行过程中若按 CTRL-C时，会暂停，再执行LIST命令或 RUN命令时，什么也没有。而当程序运行结束时，又会重新启动DOS。

PRO-9

ULIST

```
0  REM PRO-9
1  PRINT : PRINT
2  D$ = CHR$ (4)
3  POKE 214,128
5  ONERR GOTO 3
10 REM CHINE RESET ADDRESS
20 POKE 1010,66
30 POKE 1011,200
40 FOR I = 1 TO 500: PRINT I;" ";
   : NEXT I
45 PRINT
50 PRINT D$;"PR#6"
```

执行程序 PRO-9时，和上述情况类似，即按 REST E，重新启动。运行过程中按 CTRL-C，程序继续运行。而当程序运行结束时，又重新启动 DOS。

说明：将 POKE 214, 128或 POKE 214, 255 放置在磁盘的 HELLO 程序中，整个磁盘的所有文件均将列不出清单，从而达到了磁盘加密的目的。同时，若用一般的拷贝程序，例如用 DOS 3.3 中的 COPY A，也难以复制经上述处理的磁盘文件，但是，如果你有 CRAZY COPY 程序，则可以复制上述盘片，在列出 HELLO 程序清单后，删除 POKE 214, 128语句，就可以使盘片解密（仅指可以列出清单）。

8. 磁盘加密保护

磁盘加密保护，方法很多，限于篇幅仅介绍四种，简述如下：

(1) 改变 DOS 磁盘格式参数。标准的 DOS 磁盘在进

行初始化工作时，即已完成轨道格式及参数的设定，显然，改变有关参数，就制作了一个非标准的磁盘，这样，正常的拷贝程序无法进行复制，从而达到加密保护的目。

(2) 移动 VTOC 保护磁盘。标准 DOS 系统的磁盘文件管理，主要由 VTOC 完成，它放在17 轨(\$ 11) 零扇区，因此，移动 VTOC，那么正常的 DOS 将因找不到它而不能正常地读和写。

(3) 备用轨道保护法。正常 DOS 系统使用的磁盘轨道是从零轨到35 轨，而第36 轨称为备用轨，可以想像将某些信息资料放在备用轨上，则正常的拷贝程序难以复制，从而使磁盘得到保护。

(4) 修改 DOS 命令制作非标准磁盘。DOS 命令的键入到执行要经过许多环节，而这些环节中有一处被修改都会造成命令不能正确执行，例如修改命令字符，改变 DOS 命令键或属性，变换 DOS 命令的入口地址等。

下面介绍一个实用程序，可以将35 磁道的磁盘改为40 磁道，并且已将目录轨搬家。本程序有以下几个特点：

- 用该程序修改后的 DOS 格式化盘，目录轨可定在5—40 轨之间。

- 采用40 道格式化后，软盘贮存容量比原 DOS 3.3 增加20 K 字节。

- 具有一定加密功能，因目录轨已不在17 道，原 DOS 无法认清。

注意：若采用36 道以上方式时，最好采用双密度软盘。

程序 PRO-10 清单如下：

PRO-10

ULIST

```
5  REM PRO-10
10  TEXT : HOME : PRINT "DOS PROTE
    CT": PRINT
20  INPUT "THE DIRECTORY TRACK LOC
    AION:";DT
30  IF DT > 39 OR DT < 4 THEN PRINT
    "ERR": GOTO 20
40  INPUT "TOTAL NUMBER OF AVAILAB
    LE TRCKS:";TT
50  IF TT > 40 OR TT < 5 THEN PRINT
    "ERR": GOTO 40
60  POKE 44033,DT: POKE 44703,DT: POKE
    44764,DT: POKE 45715,DT: POKE
    44586,DT: POKE 46012,DT: POKE
    46268,DT
70  POKE 44741,DT * 4: POKE 44745,
    DT * 4 + 4
80  POKE 48894,TT: POKE 46063,TT: POKE
    44725,TT * 4
90  PRINT "INITALIEE DISK(Y/N)";: GET
    A$: PRINT A$: IF A$ < > "Y" THEN
    120
100 PRINT "YOU SURE INIT?(Y/N)";:
    GET A$: PRINT A$: IF A$ < >
    "Y" THEN 120
110 PRINT CHR$(4)"INIT HELLO": END

120 PRINT "DOS CHANGED,NOW YOU CA
    N PUT A NEW DISK,AND THEN PRE
    SS'INIT HELLO' TO INITALIEE."
    : END
```

程序中20句是设置目录轨数DT，由键盘键入DT应在5—39之间，40句是设置有用轨道的总数TT，它必须在5—40之间，90句提示初始化磁盘，用Y和N，110句利用DOS命令初始化磁盘，并以HELLO作为招呼程序，120句提示

原 DOS 已经改变，你可以在驱动器放一个新盘片，并请执行 INIT HELLO 命令，完成初始化。

附：另有一种简便操作，完成41道格式化，步骤如下：

- 加载 DOS 或 PRODOS
- 进入监控，CALL-151 ✓
- 出现 * 后，键入
 - * AEB5: A4
 - * B3FF: 29
 - * BEFF: 29
 - * CTRL-C
- 取出 DOS 或 PRODOS，插入需格式化盘
- 键入 INIT HELLO ✓

经上述步骤格式化后的磁盘，轨道为41道。

二、机器语言使用技巧

机器语言是计算机能够直接识别的唯一语言，它的最大特点是占用内存容量少，执行速度快。在 BASIC 程序中，使用机器语言子程序，可以充分发挥机器语言的特长，实现 BASIC 语言无法实现的功能。事实上，有许多实用软件，都是采取 BASIC 语言和机器语言相结合编制的。

为配合本章及全书其它章节讲授需要，首先介绍一下机器语言子程序的建立和调用基本方法，再介绍一些实用经验和使用技巧。

1. 子程序的建立

在 BASIC 状态下，建立机器语言子程序通常采用下述几种方法。

(1) 用 POKE 指令

利用 POKE 语句，可以将机器语言存贮到相应的单元中去。

例如，为了描绘对称图形，可以采用机器语言子程序 PRO-11，这是一个快速复制左半页图象的程序，它可以使屏幕上显示左右相同的双幅图象，其特点是跳出传统的绘图方式，省去了重编程序计算坐标的困难，并且速度快占用内存少。

PRO-11

*0300.0395

```
0300- 20 7F 03 20 25 03 A9 28
0308- 85 28 85 30 A9 3C 85 2A
0310- 85 31 20 25 03 A9 50 85
0318- 28 85 30 A9 64 85 2A 85
0320- 31 20 25 03 60 A0 00 B1
0328- 28 85 24 A9 7F 25 24 91
0330- 2A A9 08 20 A8 FC C8 C0
0338- 14 D0 EC A5 29 18 69 04
0340- 85 29 85 2B C9 40 90 DD
0348- E6 2C A5 2C C9 02 F0 17
0350- A5 28 18 69 80 85 28 A5
0358- 2A 18 69 80 85 2A A5 2E
0360- 85 29 85 2B 4C 25 03 A0
0368- 00 84 2C A5 30 85 28 A5
0370- 31 85 2A E6 2E A5 2E 85
0378- 29 85 2B C9 24 90 A6 A9
0380- 00 85 2C 85 28 85 30 A9
0388- 20 85 2E 85 29 85 28 A9
0390- 14 85 2A 85 31 60
```

对应的 BASIC 程序如 PRO-12 所示:

PRO-12

```
10 REM PRO-12
20 FOR I = 768 TO 917
30 READ A
40 POKE I, A
50 NEXT I
60 DATA 32, 127, 3, 32, 37, 3, 169, 40, 1
    33, 40, 133, 48, 169, 60, 133, 42, 13
    3, 49, 32, 37, 3, 169, 80, 133, 40
65 DATA 133, 48, 169, 100, 133, 42, 133
    , 49, 32, 37, 3, 96, 160, 0, 177, 40, 1
    33, 36, 169, 127, 37, 36, 145, 42, 16
9
```

```

70 DATA 8,32,168,252,200,192,2
    0,208,236,165,41,24,105,4,133
    ,41,133,43,201,64,144,221,230
    ,44,165
75 DATA 44,201,2,240,23,165,40,24
    ,105,128,133,40,165,42,24,105
    ,128,133,42,165,46,133,41,133
    ,43
80 DATA 76,37,3,160,0,132,44,16
    5,48,133,40,165,49,133,42,230
    ,46,165,46,133,41,133,43,201,
    36
85 DATA 144,166,169,0,133,44,133,
    40,133,48,169,32,133,46,133,4
    1,133,43,169,20,133,42,133,49
    ,96
140 CALL 768
150 END

```

140句CALL768语句的功能是转入从地址768（相应的16进制为\$300）开始的机器语言子程序。由于子程序以返回指令RTS（汇编指令RTS操作码为\$60，相应的10进制数是96）结束，所以执行完后又会返回BASIC状态，继而执行CALL命令的下一句END而结束。

下面是运行机器语言子程序PRO-11或BASIC程序

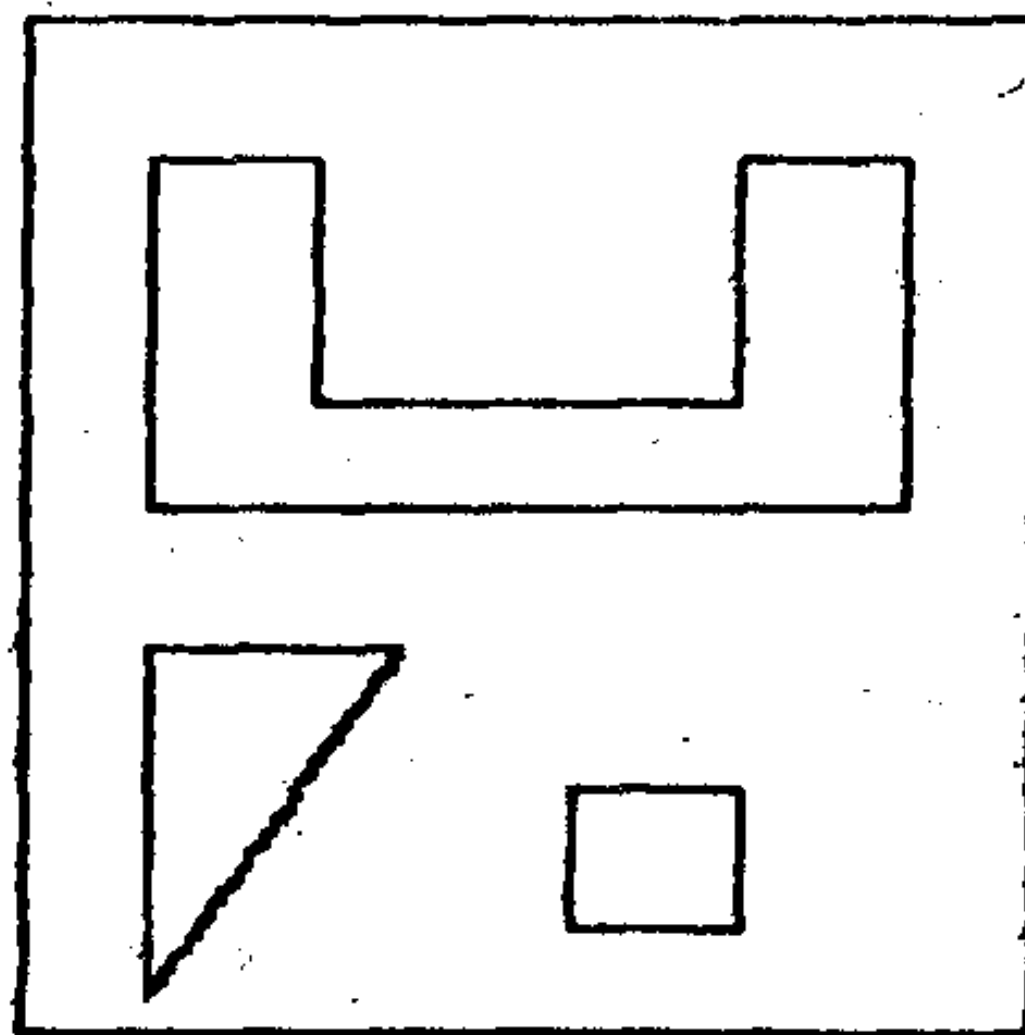


图 1

PRO-12的实例。设要复制的图形为图1所示,相应的BASIC程序见PRO-13。

PRO-13

LIST

```
5 REM PRO-13
10 HGR
20 HCOLOR= 3
30 HPLLOT 30,30 TO 50,30 TO 50,65 TO
    100,65 TO 100,30 TO 120,30 TO
    120,80 TO 30,80 TO 30,30
50 HPLLOT 30,100 TO 60,100 TO 30,1
    50 TO 30,100
60 HPLLOT 80,140 TO 100,140 TO 100
    ,120 TO 80,120 TO 80,140
70 HPLLOT 15,10 TO 135,10 TO 135,1
    55 TO 15,155 TO 15,10
```

用机器语言子程序PRO-11,复制图1的步骤如下:

- BLOAD PRO-11 ✓
- RUN PRO-13 ✓
- CALL 768 ✓

屏幕上显示图2图形。

用BASIC程序PRO-12,复制图1的步骤如下:

- RUN PRO-13 ✓
- RUN PRO-12 ✓

屏幕上也显示图2图形。

可见上述两个程序功能是一样的。

(2) 放入字符串中

用POKE语句的方法,将机器语言存放到适当的存贮单元里,必须首先把机器码(它们都是16进制数)转换成10进制数,这在实用上有许多不便,而当机器语言子程序太长

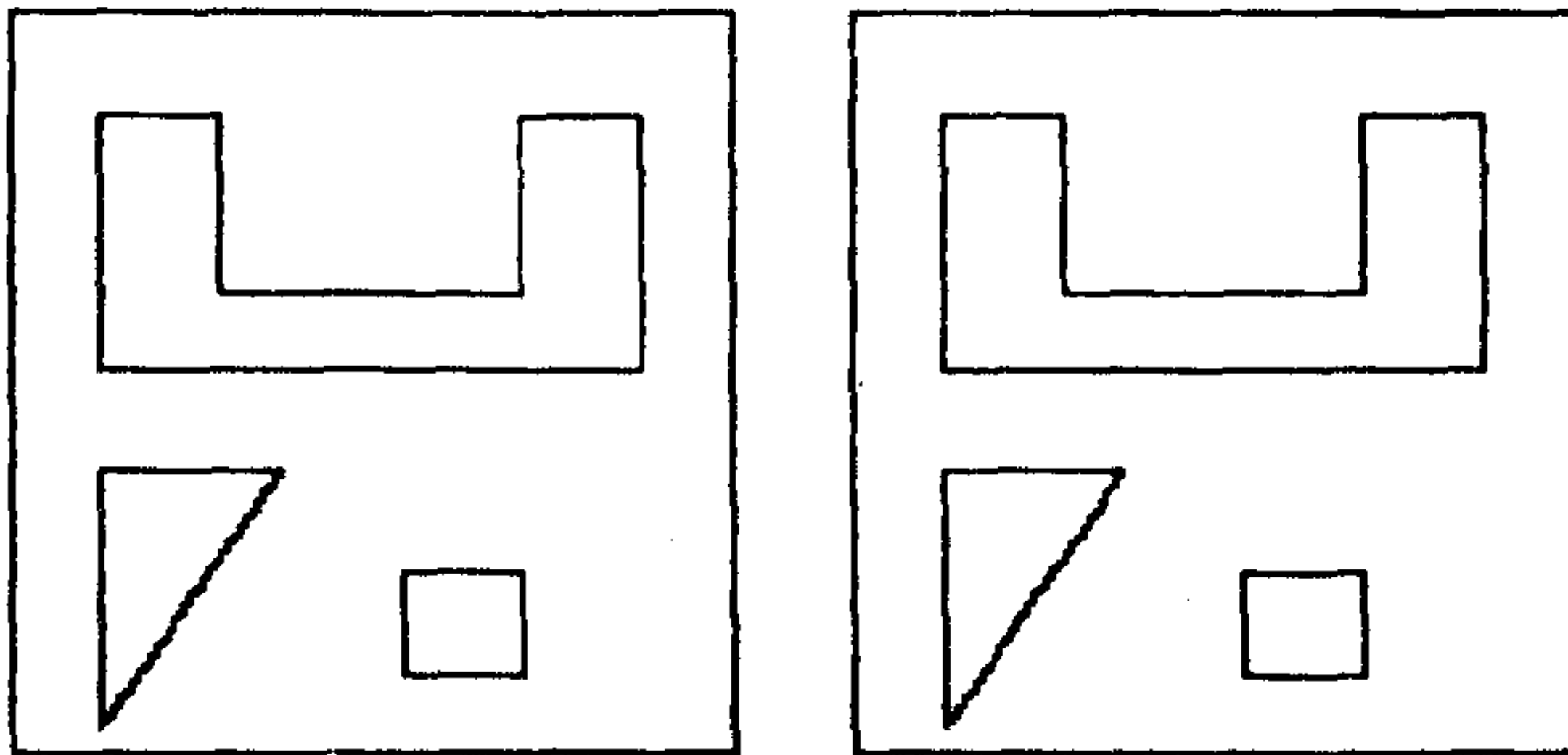


图 2

时，则上述换算会花费很多时间。用字符串的方式读入第二页，可以省去这种麻烦。

例如，显示 A—Z 26 个英文字母，最简单的方法是用 BASIC 语言来编写，见程序 PRO-14，也可以用机器语

PRO-14

ULIST

```

10 REM PRO-14
20 X = 65
30 PRINT CHR$(X); " ";
40 X = X + 1
50 IF X <= 90 THEN 30
60 END

```

言子程序来编写，然后再用 BASIC 语言来调用。

这个程序不难阅读，首先将 65 赋给变量 X（A 的 10 进制的 ASCII 码是 65），经 30 句 ASCII 码数值 65 转换成字符 A，然后计数器加 1，只要 X 值不超过 90（10 进制的 ASCII 码 90

是字符 Z)，再循环过去，从而不断打印出 A、B、……Z 字符。

用字符串的方法，就是将机器码写入字符串中，显示 26 个英文字母的程序见 PRO-15。

PRO-15

OLIST

```
10 REM PRO-15
20 K$ = "300:A9 C1 20 ED FD 18 69
    01 C9 DB D0 F6 60 N DB23G"
30 FOR I = 1 TO LEN (K$)
40 POKE 511 + I, ASC ( MID$ (K$, I
    ,1)) + 128
50 NEXT I
60 CALL - 144
70 CALL 768
```

上述程序中，\$D823 是 BASIC 的入口地址，CALL-144 是调用键盘扫描子程序，显然，用这种方法建立的机器语言子程序，可以不必把机器码转换成 10 进制数。

2. 子程序的存贮

机器语言子程序存贮比较简单，例如，有一段机器语言：

300-A9 0B 8D F6 03 A9 03 8D

可按下述步骤存贮：

在 BASIC 状态下，用 CALL-151↵ 即可进入监控 (MONITOR) 状态，在屏幕上立即看到 *，这是监控状态的提示符，然后敲入 300，再敲入：依次送入 09□ 0B □ 8D□ F6□ 03□ A9□ 03□ 8D↵，则上述机器语言存贮

完毕。

若此时键入300·307↵，则屏幕上显示：0300-A9 0B 8D F6 03 A9 03 8D信息。

退出监控状态可以用如下几个命令：

3D0G（在有DOS时方可使用）；

CTRL-C↵；

CTRL-RESET。

在BASIC状态下，用BSAVE〈文件名〉，A\$300,L\$8↵，即可存入磁盘备用。文件名可以任取，如Q-1，A\$后面300是上述机器语言的首地址，它是16进制的，L\$后面的8是机器语言的长度，这里也是16进制数。当然它们也可以用10进制表示，但参数前面不需冠以“\$”记号。

至此，上述机器语言建立存贮完成。

3. 子程序的调用

调用机器语言子程序，通常有以下几种方法：

(1) 用BLOAD或BRUN命令。

我们曾经介绍用修改链指针的方法，可以修复经NEW指令清除的BASIC程序，这种方法简单直观，容易掌握。但在程序较长时仍然显得比较麻烦，尤其对初学者来说，必须了解有关程序在程序区如何存放等问题。下面提供一个工具软件，这是一个可以修复被NEW指令清除的BASIC程序的机器语言子程序。这个程序具有连接程序区第一个链指针，恢复程序尾、变量表首、数组表首及数组表尾指针的功能。假设我们已将它以文件的形式存放在盘中，文件名为PRO-16，则可采用下述方法调用。

BRUN PRO-16↵

机器语言子程序PRO-16清单如下.

PRO-16

UCALL-151

*300.343

```
0300- A2 04 E8 BD 00 08 D0 FA
0308- 8A 18 69 01 BD 01 08 B5
0310- AF A9 08 BD 02 08 85 B0
0318- A0 01 B1 AF F0 0C AA 88
0320- B1 AF 35 AF 8A 85 B0 4C
0328- 18 03 A5 AF 18 69 02 85
0330- AF 85 69 85 6B 85 6D A5
0338- B0 69 00 85 B0 85 6A 85
0340- 6C 85 6E 60
```

(2) 用CALL语句

假设我们编制了一个机器语言子程序PRO-17, 并把它存放在\$9000开始的地址单元中, 清单如下:

PRO-17

*9000.900A

```
9000- A2 0A A9 AA 9D 36 06 CA
9008- D0 FA 60
```

用BSAVE PRO-17, A\$9000, L\$OB存盘。

使用时可以采取下面任意一个方法调用:

方法 1

PRO-18

ULIST

```
10 REM PRO-18
20 PRINT CHR$(4); "BLOAD PRO-17"

30 CALL 36864
```

方法 2

PRO-19

ULIST

```
5  REM  PRO-19
10 K = 9 * 4096: HIMEM = K
20  FOR I = 0 TO 10
30  READ X: POKE K + I, X
40  NEXT I
50  CALL K
60  DATA 162, 10, 169, 170, 157, 54, 6, 2
      02, 208, 250, 96
```

执行上述调用方法后，可以在屏幕的中心位置上连续显示出10个星号，这就是机器语言子程序PRO-17的功能。

(3) 用DRAW命令

对于用机器语言编制的造型表，要调出其中的某一个造型，常常要用DRAW命令，执行它可以指定在屏幕的什么位置画图，同时配合SCALE命令，以设定图形的大小，再利用造型旋转的命令ROT，使造型旋转，从而使单调、呆板的显示变得饶有情趣，生动活泼。

例如，PRO-20是一个用机器语言编制的小汽车子程序：

PRO-20

*8500.8577

```
8500- 01 00 05 00 00 41 38 20
8508- 25 2C 2D 65 0C 0C 0C 2D
8510- 2D 2D 2D 2D 15 15 AD 2D
8518- 2D 15 2D 2D 2E 2E 3E 3E
8520- BF 37 3F 3C 1C 24 1C 3F
8528- 17 17 36 3F 3F 3F F7 27
```

```

8530- 3F E4 1C 3F 17 3E 3F 2D
8538- 25 6D 15 F6 3F 07 20 64
8540- 2D 04 20 05 28 28 2D 2D
8548- 2D 35 35 35 3F 3F 3F 24
8550- 34 36 3F 3F 3F 4A 49 92
8558- 3A 2D 4E 49 49 20 64 2D
8560- B6 1E 7F 49 09 2C 25 3C
8568- 2C 2D 2D 3C 3F 3F EF 3F
8570- 3F 27 2D 00 00 00 00 00

```

它存贮在\$ 8500开始的地址单元中，为了调用小汽车这个造型，见程序PRO-21和PRO-22：

PRO-21

LIST

```

3  REM RO-21
5  HOME
10 D$ = CHR$ (4)
20 POKE 232,0: POKE 233,133
30 PRINT D$;"BLOAD PRO-20,A$8500"

40 HGR : HCOLOR= 3
50 ROT= 0: SCALE= 1
55 FOR I = 0 TO 240 STEP 80
57 FOR J = 50 TO 160 STEP 50
60 DRAW 1 AT I,J
63 NEXT J
65 NEXT I
70 END

```

RUN ✓



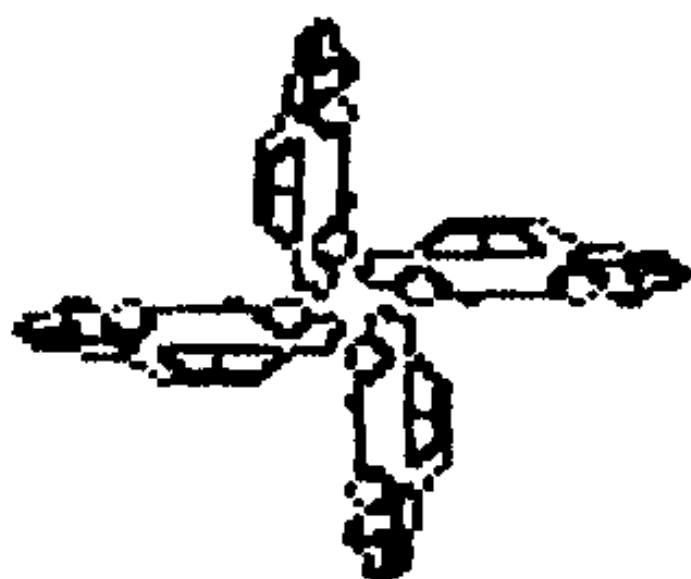
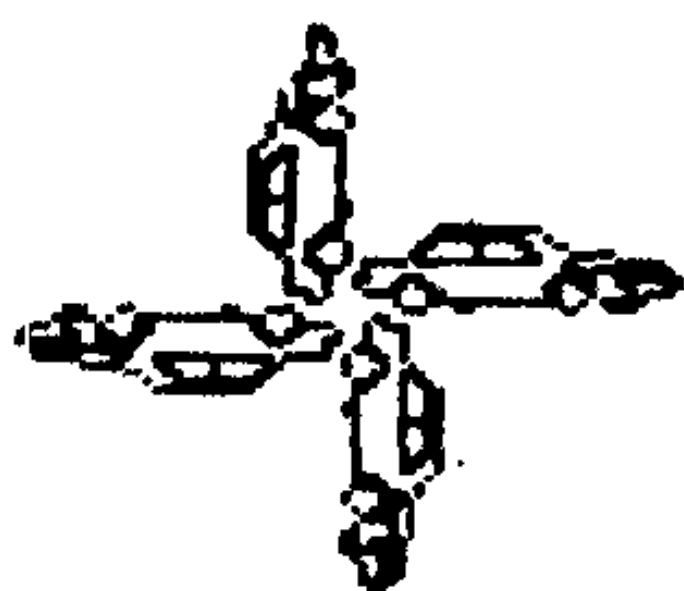
PRO-22

LIST

```
3  REM PRO-22
5  HOME
10 D$ = CHR$ (4)
20 POKE 232,0: POKE 233,133
30 PRINT D$;"BLOAD PRO-20,A$8500"

40 HGR : HCOLOR= 3
42 FOR K = 1 TO 60 STEP 15
50 ROT= K: SCALE= 1
55 FOR I = 50 TO 240 STEP 95
60 DRAW 1 AT I,100
65 NEXT I
67 NEXT K
70 END
```

RUN ↙



上面两个程序中都用了E8(232), E9(233)这两个单元, 它们是专门用来存放图形表起始地址的, 由于小汽车造型的首地址是\$8500, 对应的10进制数是: 低字节为0, 高字节为133, 所以安排了20句, POKE232,0:POKE233,133。

另外, APPLESOFT BASIC 软件都是从E8,E9这两个单元取得造型表的起始地址。

(4) 用&来调用

在APPLESOFT 的解释程序中, 连接符&也是一条指令, 当用APPLESOFT BASIC 语言编制的程

序执行到这条指令时，将会使原程序无条件地调用处于内存\$3F5位置上的机器语言子程序。这样，实质上就使我们多了一条调用机器语言子程序的指令。

\$3F5就是&的入口地址，在监控状态下，若键入3F5·3F7↵，则可以看到：03F5-4C 58 FF的信息，其中4C是6502指令系统中助记符JMP的操作码（绝对寻址方式），这是一条无条件转移指令，也就是说在\$3F5—\$3F7单元中存放了一条JMP \$FF58指令。\$FF58是机器语言子程序的一个地址。

因此，当我们在内存的某一段区域内编制了一段机器语言子程序后，只要修改原内存\$3F5—\$3F7中内容，改成跳转到用户自己编制的机器语言首地址上来，那么，就可以方便地用&来达到调用机器语言子程序的目的。

例如，在监控状态下，键入

3F5:4C 6E A5

回车后按RESET返回BASIC状态。这时，按下“&”键并回车，计算机就自动列出磁盘中所有文件的目录。可见“&”键已被定义为CATALOG了。其中\$A5 6E是CATALOG的入口地址。

需要指出的是，如果用户的机器语言子程序太长，不能直接存放在\$3F5以后的存贮单元，这是因为\$3F5之后没有空出太多的存贮空间（从\$400开始是显示器地址），因此，解决的方法是在\$3F5存入指令JMP××××，使程序转至××××代表的可用的存贮空间。

4. 子程序使用技巧

随着微机应用的不断深入, 用户不仅希望充分利用现有机器的功能, 而且要求扩充机器的功能, 以便有更多的应用范围。下面是几个应用机器语言子程序来达到扩充功能的技巧。

(1) RESTORE带行号

在PC-1500或IBM-PC系列机中, 恢复数据区语句RESTORE后面可以带表达式, 其格式为:

《行号》RESTORE〈表达式〉

其中〈表达式〉可以是数值, 它代表行号; 也可以是字符串, 它代表标号。它们表示指针拨到的位置。

例如, 程序PRO-23, 在PC-1500机中就可以运行:

PRO-23

ULIST

```
10  REM PRO-23
20  DATA 8,5
30  DATA 20,30,"BASIC PROGRAM"
40  READ A,B
50  READ C,D,K$
60  RESTORE 2 * A + 14
70  READ X,Y,S$
80  PRINT "A=";A;" "; "B=";B
90  PRINT "C=";C;" "; "D=";D
100 L PRINT "K$=";K$
110 L PRINT "X=";X;" "; "Y=";Y
120 L PRINT "S$=";S$
130  END
```

U

URUN

A=8 B=5

```
C=20 D=30
K$=BASIC PROGRAM
X=20 Y=30
S$=BASIC PROGRAM
```

70 句中 READ 语句，所以有数可读，是借助于 60 句 RESTORE 的作用，它把数据区的指针拨到 30 行的开始位置。

但是，APPLE 及兼容机（包括中华学习机），没有上述功能，即 RESTORE 后面不允许带表达式。这也是基本 BASIC 语言对 RESTORE 的约定。

能否扩展 RESTORE 语句的功能，使它后面也可以带一个表达式呢？解决的方法是重新设定一个执行语句 &N，其中 N 是要转移的行号，&N 就相当于 RESTORE N 的命令。

键入下述机器语言子程序，并把它以文件名 PRO-24 存入磁盘。

PRO-24

***300.326**

```
0300- A9 0B 8D F6 03 A9 03 8D
0308- F7 03 60 20 7B DD 20 52
0310- E7 20 1A D6 90 0B C6 9B
0318- A4 9B A5 9C 84 7D 85 7E
0320- 60 A2 5A 4C 00 00 00
```

为了检验上述机器语言子程序是否具有 RESTORE N 的功能，可用下述 BASIC 程序 PRO-25 来验证：

PRO-25

ULIST

```
20  REM PRO-25
100 CALL 768: READ A,B: & 400
200 READ C: PRINT "A=";A,"B=";B,"
    C=";C
300 DATA 10,20,30,40,50,60,70,
400 DATA 100
```

URUN

A=10

B=20

C=100

运行结果表明，&400确实起到 RESTORE 400的作用。

注意运行上述 BASIC 程序前，必须先将程序 PRO-24 机器语言子程序调入内存，否则 CALL 768不起作用。当然，也可以直接在 BASIC 程序中利用 DOS 命令，例如加入50句：

```
50 PRINT CHR$(4); "BLOAD PRO-24"
```

上述扩展 RESTORE 功能的做法，不影响标准 RESTORE正常使用，而在需要指向任意数据区时，灵活自如，为我所用。

(2) GOTO带变量

在APPLESOFT BASIC中，GOTO语句使用规则最简单，仅要求在 GOTO 语句之后跟上一个程序中存在的行号，其功能是无条件转移到指定的那个行号去执行。但是不允许在 GOTO 的行号中使用变量，这也是基本 BASIC 的一个规定。但我们可以利用机器语言子程序，改变上述规定，使之 GOTO 语句具有带变量的功能。

首先键入下述机器语言子程序,用BSAVE PRO-26,
A\$300,L\$13存盘备用。

PRO-26

*300.312

```
0300- 20 7B DD 20 52 E7 20 1A
0308- D6 90 03 4C 41 D9 A2 5A
0310- 4C 12 D4
```

其次编一个简单的BASIC程序PRO-27,利用机器语言子程序PRO-26,来达到GOTO A的功能,程序中用“&”的方式调用。这个想法比较自然,正如我们在前节所介绍的,在BASIC程序中,当遇到以“&”开头的命令时,会跳转到\$3F5去执行机器语言子程序,而从\$3F5开始的单元,放进机器语言子程序PRO-26,故可以用“&”来调用。

PRO-27

LIST

```
5  REM PRO-27
10  PRINT CHR$(4);"BLOAD PRO-26"

20  POKE 1013,76: POKE 1014,0: POKE
    1015,3
30  INPUT "A=(50,60,70,80,90,100)?
    ";A
40  & A
50  PRINT 1000: GOTO 30
60  PRINT 2000: GOTO 30
70  PRINT 3000: GOTO 30
80  PRINT 4000: GOTO 30
90  PRINT 5000: GOTO 30
100 END
```

```

URUN
A=(50,60,70,80,90,100)? 50
1000
A=(50,60,70,80,90,100)? 60
2000
A=(50,60,70,80,90,100)? 70
3000
A=(50,60,70,80,90,100)? 80
4000
A=(50,60,70,80,90,100)? 90
5000
A=(50,60,70,80,90,100)? 100

```

程序PRO-27中20句，是在\$3F5处设置指令JMP \$300，其中1013就是16进制\$3F5，而76即\$4C（它是JMP的操作码）。相应的1014，0即\$3F6，0；1015，3即\$3F7，3，所以\$3F6，\$3F7存放的是\$0300的地址（低位址在前，高位址在后），它是上述机器语言子程序的入口地址，因此，20句就是跳转到地址为\$0300去执行从这个地址开始的机器语言子程序。

(3) 高分辨图形的快速清屏

在高分辨率图形下，清屏命令HOME不起作用，这是因为高分辨率图形内存区和文本及低分辨率图形内存区不同。为了能使高分辨图形在需要清屏的时候快速清屏，可用PRO-28机器语言来实现。并用BSAVE PRO-28，A\$6ADO，L\$EF存盘备用。

PRO-28

*6ADO.6AEF

```

6ADO- A0 00 A9 20 85 11 A9 00
6ADB- 85 10 91 10 CB D0 FB A6
6AE0- 11 E0 3F F0 05 E6 11 4C
6AEB- DA 6A 60 00 80 80 00 5E

```


为了演示PRO-28机器语言子程序快速清屏的功能，下面举二个例子并分别用两种方法调用。

例1，程序PRO-29是一个十分简单的高分辨绘图程序，它在屏幕的中央绘制一个矩形框。我们希望在图形显示后快速清屏，而采用CALL命令来执行，显然，CALL后面的数值应为十进制的27344，它对应于快速清屏的机器语言子程序入口地址\$6AD0。计算机遇到CALL27344后，立即转向执行机器语言子程序PRO-28，瞬间即可将屏幕清除，见程序PRO-29。

PRO-29

LIST

```
290 REM PRO-29
295 PRINT CHR$(4); "BLOAD PRO-28"
"
300 HGR
310 HCOLOR= 3
320 HPLLOT 120,20 TO 150,20 TO 150
    ,150 TO 120,150 TO 120,20
330 CALL 27344
340 END
```

例2，PRO-30，是一个稍许复杂一点的高分辨率绘图程序，它在屏幕的左上方显示一个三角形，要求图形显示后立即消失，共五次结束。

我们采用&指令来调用快速清屏机器语言子程序PRO-28，根据前节分析，必须把机器语言子程序入口地址\$6AD0放入\$3F6和\$3F7两个单元中，这可以借助于POKE指令来解决，即POKE1014, 208, POKE1015, 106。因为POKE指令中的地址和数值，都是十进制数，所以1014即\$3F6，1015即\$3F7，而208即\$DO，106即\$6A，见

程序 PRO-30。

PRO-30

LIST

```
10 REM PRO-30
20 POKE 1014,208: POKE 1015,106
25 N = 0
30 HGR
40 HCOLOR= 3
50 HPLOT 0,0 TO 100,0 TO 100,100 TO
   0,0
60 &
70 FOR I = 1 TO 3000: NEXT
75 N = N + 1
80 IF N = 5 THEN 100
90 GOTO 30
100 TEXT : HOME : END
```

运行上述程序，可以演示快速清屏的过程。70句是一个延迟程序，目的是看清图形显示和消失的过程，否则，由于清屏速度太快，而使图形无法看清楚。

(4) 低分辨率图像打印

由于APPLE-II及其兼容机（如紫金II，中华学习机等）没有低分辨率图形打印功能，因此，诸如实验数据分析，学生成绩统计，生产进度报表，产品销售情况等各种分类统计图表无法列印在纸上，这给工作上带来许多不便。

通常解决上述问题的方法有两种：

- 在文本状态下，按一定比例打印一行行“*”号，但图形不够美观，编制程序花时较多。

- 在低分辨率状态下画好图像，再运行程序PRO-31，但不是所有打印机都能奏效。例如用PC-80打印机250句要改为CHR\$(16)，FX-80则用CHR\$(35)。

PRO-31

ULIST

```
180 REM PRO-31
190 POKE 1657,255:D$ = CHR$ (4)
200 PRINT D$;"PR#1"
210 PRINT CHR$ (27); CHR$ (65); CHR$
    (8);
220 FOR I = 0 TO 39
230 FOR J = 0 TO 39
240 IF SCRN(J,I) < 1 THEN PRINT
    " ";: GOTO 260
250 PRINT CHR$ (35);
260 NEXT J
270 PRINT
280 NEXT I
290 PRINT D$;"PR#0": END
```

例如，编制一个工厂8个车间某月产量的直方图程序PRO-32，就比较长，且通用性不好。

PRO-32

ULIST

```
10 REM PRO-32
20 D$ = CHR$ (4)
30 DIM F(8)
40 READ N
50 FOR J = 1 TO N
60 READ F(J)
70 F(J) = F(J) / 50
80 F(J) = INT (F(J))
90 NEXT
100 DATA 8,130,300,600,700,800,40
    0,200,100
110 GR
120 COLOR= 15
130 FOR J = 1 TO N
140 VLIN 39,39 - F(J) AT (10 + 3 *
```

RUN ↘



因此，一个较好的方法是编制一个可以打印低分辨率图像的机器语言子程序，并将该子程序存盘备用，例如取名为PRO-33，并假定存放在\$8 D00开始的单元中，使用步骤如下：

- 装入内存，BLOADPRO-33 ✓
- 设置最高内存地址，HIMEM: 36096 ✓
- 接通打印机电源
- 运行低分辨率作图程序
- 键入CALL 36096 ✓

则计算机自动完成低分辨率图像的硬拷贝。

下面给出机器语言子程序清单和一个运行实例。

PRO-33

*8D00.8D8B

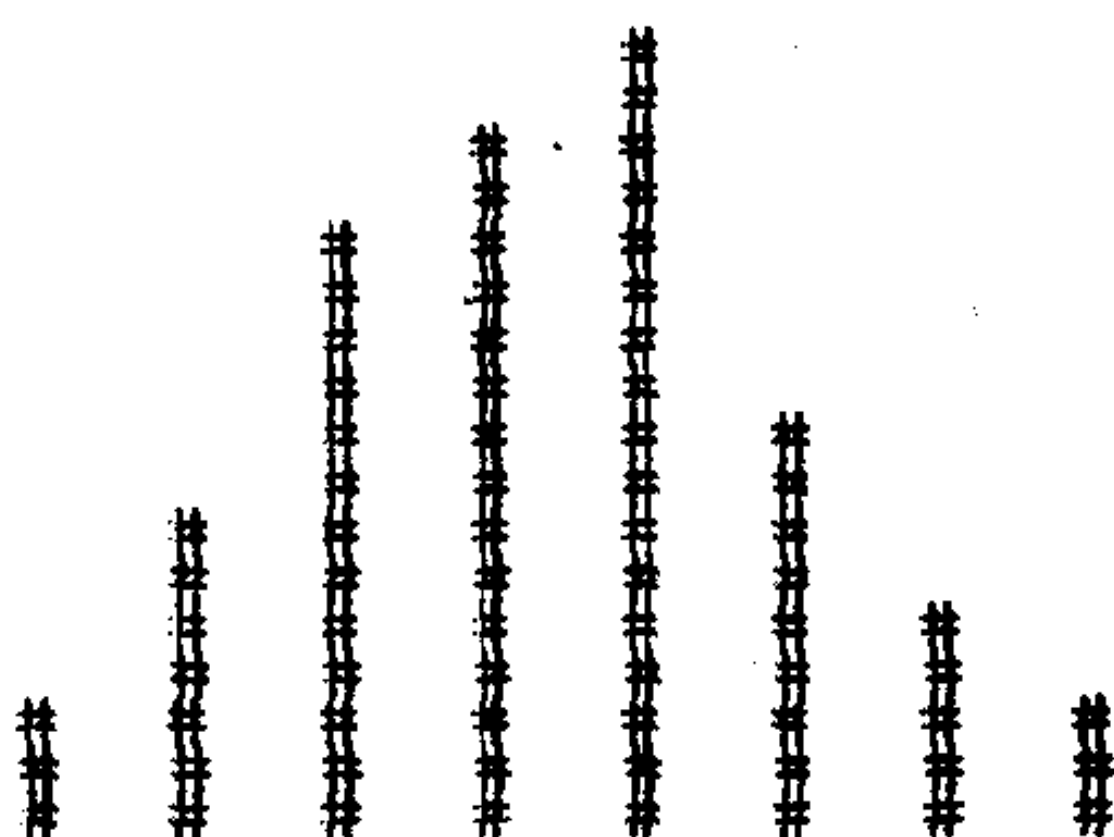
| | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|
| 8D00- | A9 | 0A | 20 | B3 | 8D | A9 | 1B | 20 |
| 8D08- | B3 | 8D | A9 | 33 | 20 | B3 | 8D | A9 |
| 8D10- | 14 | 20 | B3 | 8D | A9 | 00 | 4B | B5 |
| 8D18- | 0A | C9 | 50 | D0 | 05 | A2 | 04 | 4C |
| 8D20- | 24 | 8D | A2 | 0B | A9 | 04 | B5 | 0B |
| 8D28- | A0 | 00 | B1 | 0A | 29 | 0F | F0 | 0B |
| 8D30- | A9 | 23 | 20 | B3 | 8D | 4C | 3D | 8D |
| 8D38- | A9 | 20 | 20 | B3 | 8D | B1 | 0A | 29 |
| 8D40- | F0 | D0 | 0B | A9 | 20 | 99 | D8 | BF |
| 8D48- | 4C | 50 | 8D | A9 | 23 | 99 | D8 | BF |
| 8D50- | C8 | C0 | 2B | D0 | D5 | A9 | 0A | 20 |
| 8D58- | B3 | 8D | A0 | 00 | B9 | D8 | BF | 20 |
| 8D60- | B3 | 8D | C8 | C0 | 2B | D0 | F5 | A9 |
| 8D68- | 0A | 20 | B3 | 8D | A5 | 0A | 1B | 69 |
| 8D70- | 80 | B5 | 0A | 90 | 02 | E6 | 0B | CA |
| 8D78- | D0 | AE | 6B | 1B | 69 | 2B | C9 | 7B |
| 8D80- | D0 | 94 | 60 | 2C | C1 | C1 | 30 | FB |
| 8D88- | 8D | 90 | C0 | 60 | | | | |

PRO-34

ULIST

```
20 REM PRO-34
30 DIM F(8)
40 READ N
50 FOR J = 1 TO N
60 READ F(J)
70 F(J) = F(J) / 50
80 F(J) = INT (F(J))
90 NEXT
100 DATA 8,130,300,600,700,800,400,200,100
110 GR
120 COLOR= 15
130 FOR J = 1 TO N
140 VLIN 39,39 - F(J) AT (10 + 3 * J)
150 NEXT
```

URUN



(5) 电脑演奏乐曲

有许多流行歌曲，深受广大青少年喜爱。如果用微电脑演奏出美妙动听的歌声，这对广大青少年有着很大的吸引力，从而进一步激发他们学习计算机的兴趣。

程序 PRO-35 是一个能发音的机器语言子程序，有了它，

再加上控制音阶和节拍的数值,就可以发出不同频率的声音,从而奏出优美的音乐。机器语言子程序放在\$302(770)—\$316(790)单元中,音阶和节拍数值分别存入768和769两个单元中,使用时用CALL指令调用,它后面的数值是机器语言的入口地址770。

PRO-35

*302.316

```
0302- AD 30 C0 88 D0 05
0308- CE 01 03 F0 09 CA D0 F5
0310- AE 00 03 4C 02 03 60
```

下面是两首歌曲,程序PRO-36演奏“小草”,程序PRO-37演奏“茉莉花”。程序编制方法上稍有不同,程序PRO-36更具有一般性。如想演奏别的歌曲,只需将相应歌曲的音阶和节拍数值改放在90句以后的DATA语句中,而程序的前面部份不要作任何改动。

PRO-36

ULIST

```
10 REM PRO-36
20 PRINT CHR$(4); "BLOAD PRO-35"

30 FOR I = 1 TO 1000
40 READ X,Y
50 IF X = - 1 AND Y = - 1 THEN
    END
60 POKE 768,X: POKE 769,Y
70 CALL 770
80 NEXT
90 DATA 152,70,152,70,128,70,136,
    70,152,160,152,70,152,70,102,
    70,114,70,102,160,102,70,102,
    70,86,70,102,70,114,70,114,70
```




```

110 DATA 30,30,240,120,60,120,240
    ,120,120,60
120 DATA 60,60,60,120,120,120,60,
    60,60,60
130 DATA 30,30,240
140 FOR C = 1 TO 43
150 READ F(C)
160 NEXT C
170 FOR C = 1 TO 43
180 READ D(C)
190 NEXT C
200 FOR C = 1 TO 43
210 POKE 768,F(C)
220 POKE 769,D(C)
230 CALL 770
240 NEXT C

```

运行上面两个程序，所以能唱出歌来，是因为唱歌子程序已经建立，如果没有那个发音子程序，执行CALL 770，不仅唱不出歌来，而且程序也无法运行。

5. 实用经验和技巧

用高级语言编程，面向问题，面向使用者，可以不管计算机的内部操作，因而使用方便，易于学习和掌握，这是高级语言的主要优点。但是机器语言由于是计算机能够直接识别的唯一语言，它不需要解释系统，因而有运行速度快，占有存贮少这样两个突出的优点。从前面几节的实例中，我们已经看到 BASIC 语言无法实现的功能，而用机器语言就能轻而易举的解决。为了更好地使用机器语言，下面提供一些实用经验和使用技巧，这些内容都是笔者使用机器的心得体会。

(1) BASIC 程序和机器语言子程序的连接

通常，机器语言子程序是以独立的程序文件形式存贮在

磁盘中，而由 BASIC 程序调入内存再执行，这就使得程序的执行受到磁盘操作影响而减慢速度，并增加了磁盘工作时间。为此，可以采用特殊技巧将机器语言子程序嵌入 BASIC 程序中，自动随 BASIC 程序一起存贮和调用。

①利用 REM 注释语句

BASIC 中的 REM 语句是一个非执行语句，用来对程序或程序段作一些注释说明，以增强程序的可读性。BASIC 解释程序在碰到 REM 语句时完全不予理会而自动跳过，因此，REM 中的内容可以不符合语法规则。这就给我们一个启发，将机器语言子程序嵌入 REM 语句中，使之成为 REM 语句的内容，就可以将它和主程序完整地统一起来，实现两者的同时存贮和调用。

例如，程序 PRO-38 是一个发音机器语言子程序，程序 PRO-39 是一个 BASIC 程序，用它来调用机器语言子程序以完成键盘弹奏模拟。

PRO-38

*300.317

```
0300- AD 30 C0 88 D0 04 C6 01
0308- F0 08 CA D0 F6 A6 02 4C
0310- 00 03 60 00 00 00 00 00
```

PRO-39

```
1000 REM PRO-39
1010 CALL - 936: PRINT CHR$ (4)
      ; "BLOAD PRO-38"
1020 DIM A(8): FOR I = 1 TO 8: READ
      A(I): NEXT
1030 GET X$: IF X$ < "0" OR X$ >
      "8" THEN 1030
```

```

1040 IF X$ = "0" THEN END
1050 X = VAL (X$): POKE 1,96
1060 POKE 2,A(X)
1070 CALL 768: GOTO 1030
1080 DATA 133,127,121,116,111,107
      ,103,100

```

利用上面介绍的思路，可以将此两个程序合并为一，操作步骤如下：

- 算出机器语言子程序字节数，本例为19个字节；
- 把REM 做为 BASIC 程序的第一个语句，设置REM语句的内容为19个字节，如500 REM□□□……，共19个空格；
- 进入监控状态：CALL -151 ✓；
- 从REM 字节之后连续输入子程序机器码：806: AD
30 CO 88 DO 04 C6 01 FO 08 CA DO F6 A6
02 4C 06 08 60 ✓
- 返回 BASIC 状态；
- 将1070句的CALL 768改为CALL 2054
- 最后删去1010句的PRINT CHR \$(4);“BLOAD
PRO-38”

至此，程序调整已经完成，可以用BSANE 命令存盘，使用时再用BLOAD 命令调入内存，用BRUN 命令即可运行嵌入BASIC 程序中的机器语言子程序，按1—8键发出模拟钢琴弹奏的声音。

采用上面方法时，有几点必须说明一下：

首先，把REM 作为 BASIC 程序的第一个语句，是为了便于计算地址。因为 BASIC 程序区的首地址一般固定为2049 (\$801)，再加上二字节的链指针、二字节的本行行

号及一字节的REM语句内部代码；所以容易求出机器语言子程序的首地址为：

$$2049 + 5 = 2054$$

即16进制的\$806

其次，在REM字节后存贮机器语言子程序，应首先存入它的首地址，即806。而原来子程序是存入\$0300中，故必须将原子程序中\$30F到\$311的4C 00 03改为4C 06 08，即跳转到\$806的新地址上来。

最后，子程序编制时，应防止\$00出现，否则BASIC系统将认为这是一个程序行的结束，而做出错误的处理。例如，子程序的最后一个字节安排\$60（它是6502指令的助记符RTS，作用是从子程序返回，操作码为\$60），而不是\$00，就防止上述错误发生。

②利用系统指针

另一种使BASIC程序携带机器语言子程序的方法是：将子程序存放在BASIC程序结尾之后。

这样做必须修改系统指针，即应将程序尾指针和变量区首指针，指向子程序之后。

BASIC程序的尾指针在零页单元\$AF、\$B0中，变量表首指针由\$69、\$6A两个单元所指示，因此，将机器语言子程序放在程序尾的三个全零字节之后，再利用监控状态修改上述两个指针，即可完成两种不同语言的合并。这样，当执行SAVE命令时，就可以同时存入两个程序，而用LOAD命令时，又可以将两个程序同时装入内存，从而减少了访问磁盘的次数。

例如，程序PRO-40是一个机器语言子程序，它能够将绘制在第一页高分辨率图形画面搬移至第2页高分辨率作图

PRO-40

*300.31F

```
0300- A9 00 85 3C 85 42 A9 20
0308- 85 3D A9 40 85 43 A9 FF
0310- 85 3E A9 3F 85 3F A0 00
0318- 4C 2C FE 00 00 00 00 00
```

PRO-41

ULIST

```
1000 REM PRO-41
1010 HOME : HGR2 : HGR
1020 HCOLOR= 3: FOR I = 8 TO 80 STEP
      4: GOSUB 1070: NEXT
1030 HCOLOR= 0: FOR I = 80 TO 8 STEP
      - 4: GOSUB 1070: NEXT
1040 GOTO 1020
1050 CALL 768
1060 POKE - 16300,0: FOR I = 1 TO
      500: NEXT : POKE - 16299,0: FOR
      I = 1 TO 500: NEXT : GOTO 106
      0
1065 END
1070 HPLOT 140 - I,90 - I TO 140 +
      I,90 - I TO 140 + I,90 + I TO
      140 - I,90 + I TO 140 - I,90 -
      I
1080 IF PEEK ( - 16384) > 128 THEN
      POP : GOTO 1050
1090 RETURN
```

区，程序 PRO-41 是一个简单动画的 BASIC 程序。

组织上述程序的步骤如下：

- 进入监控状态：CALL -151 ✓；
- 查出程序尾地址为 \$ 8 F 9 （在 \$ A F、\$ B 0 指针中）；
- 从 \$ 8 F 9 之后（如 \$ 9 0 0）输入机器语言子程序；

- 将1050句的 CALL 768改为 CALL 2305;
- 检查子程序结束地址为 \$ 91 C ;
- 将 \$ A F 、 \$ B 0 和 \$ 6 9 、 \$ 6 A 改为指向 \$ 91 C , 即在监控状态下打入:

A F : 1 C 09 ✓

69 : 1 C 09 ✓

注意: 上述地址的处理中, 为了应付必要的修改, 在三个全零字节后已空出一些空间, 再存放机器语言子程序的。

经上述重新组织的机器语言子程序, 已被嵌入 BASIC 程序之中, 而成为一个完整程序。

(2) 10进制数和16进制数的互换

在和机器语言打交道的过程中, 经常的和大量的要处理10进制数和16进制数之间的互换问题。

例如, 系统 ROM 中有许多有用的子程序, 它们是用机器语言编写的, 如能正确调用, 这对减少用户编写程序的工作量和节约存贮空间, 都有实际意义。

文本方式下清屏幕语句 HOME, 是大家熟悉的, 它的入口地址为 \$ F C 58, 如用机器语言调用则为 20 58 F C, 改为 BASIC 调用就是 CALL -936。那么它们之间是一个什么样的换算关系呢?

$$\begin{aligned}
 (F C 58)_{16} &= (16^3 \times 15 + 16^2 \times 12 + 16^1 \times 5 + 16^0 \times 8)_{10} \\
 &= (4096 \times 15 + 256 \times 12 + 16 \times 5 + 1 \times 8)_{10} \\
 &= (61440 + 3072 + 80 + 8)_{10} \\
 &= (64600)_{10}
 \end{aligned}$$

而这个地址也可以用减去65536以后的负数表示, 即 $64600 - 65536 = -936$

这就是 CALL -936 的由来。

由此可见，如果每次都用手算方法进行上述转换，不仅十分麻烦，而且费时甚多，程序 PRO-42，可为你解除这种麻烦。

PRO-42

ULIST

```
5  REM PRO-42
10  HOME :P$ = "0123456789ABCDEF"
20  PRINT "HEXADCIMAL-DECIMAL INTE
    GER CONVERSION": PRINT
30  PRINT : PRINT "PLEASE INPUT NU
    MBERS"
40  INPUT "A$=?";A$: IF A$ = "#" THEN
    END
50  IF LEFT$ (A$,1) = "$" THEN 14
    0
60  B = VAL (A$)
70  PB$ = " ":BB = B
80  IF B < 0 THEN B = 2 ^ 16 + B
90  HD = INT (B / 16):LD = B - HD *
    16:B = HD
100 PB$ = MID$ (P$,LD + 1,1) + PB
    $
110 IF HD > 0 THEN 90
120 PRINT "      ";BB;"=";PB$
130 GOTO 30
140 L = LEN (A$) - 1:B = 0:B$ = RIGHT$ (A$,L)
150 FOR I2 = L TO 1 STEP - 1: FOR
    I1 = 16 TO 1 STEP - 1
160 B1$ = MID$ (B$,I2,1):B2$ = MID$
    (P$,I1,1)
170 IF B1$ = B2$ THEN B = B + (I1
    - 1) * 16 ^ (L - I2)
180 NEXT I1: NEXT I2:A = 2 ^ 16 -
    B
190 PRINT "      $";B$;"=";B;"(- ";
    A;" )"
200 GOTO 40
```


看的某个B文件，要读一读它的内容，或研究一下原程序的结构，那真是望尘莫及，无能为力了。

当然，如果某个B类文件，是你自己编制的，又记得程序的起始地址，用上述方法便可以列出B文件的内容。事实上，还有另一种情况，程序虽是自己编码，但由于时间过长，编的程序又多，也会忘记起始地址，这样自己编写的程序，却不知道程序的内容。

因此，寻找二进制文件的起始地址，是揭开二进制文件内容奥妙的关键。

能否找到一个搜索任意B类文件起始地址的方法呢？

可以回顾，APPLE-II机的DOS操作系统，它在执行装载命令时，当然十分清楚从磁盘中取出的文件，应该放在内存什么地方，从什么地址起始，又是到什么地址结束。也就是说，在DOS操作系统的某几个地址的暂存器中肯定记录了这几个数（地址）。因此，问题已经清楚，只要知道存放最近一次装入内存的B文件起始地址和该文件的长度，就可以查看B文件的内容。

\$AA72(43634)和\$AA73(43635)两个单元分别存放了最近一次装入内存的机器语言程序起始地址低八位和高八位，而在\$AA60(43616)和\$AA61(43637)两个单元中，分别存放该程序的长度。

因此，用PEEK指令，就可以在屏幕上显示待查的B文件起始地址和长度。为实际使用方便，可以编制一段小程序PRO-43:

PRO-43

```
5  REM PRO-43
10 A = PEEK (43634) + PEEK (4363
    5) * 256
20 B = PEEK (43616) + PEEK (4361
    7) * 256
30 PRINT "START ADDRESS: ";A
40 PRINT "LENGHT: ";B
45 P$ = "0123456789ABCDEF"
50 INPUT A$
55 IF A$ = "#" THEN END
60 K = VAL (A$)
70 PB$ = " ";BB = K
80 IF K < 0 THEN K = 2 ^ 16 + K
90 HD = INT (K / 16):LD = K - HD *
    16:K = HD
100 PB$ = MID$ (P$,LD + 1,1) + PB
    $
110 IF HD > 0 THEN 90
120 PRINT " ";BB; "=$";PB$
130 GOTO 50
```

现在，我们可以写出查找任意一个B类文件的步骤：

- 键入上述程序，取名为PRO-43并存盘
- NEW✓
- 装入待查找的B文件，如RESTORE LINE
- BRUN RESTORE LINE✓
- RUN PRO -43✓，屏上出现：

ADDRESS: 768

LINGHT: 39

出现？键入768✓

768 = \$ 300

又出现？键入39✓

39 = \$ 27

再出现? 键入 #

□

768就是B文件的起始地址,39为该文件的长度,经转换后\$300是B文件的十六进制表示的起始地址,\$27是对应的十六进制表示的长度。

• 执行CALL-151进入监控,出现*后键入300•327✓
屏幕上显示如下机器码:

*300.327

```
0300- A9 0B 8D F6 03 A9 03 8D
0308- F7 03 60 20 7B DD 20 52
0310- E7 20 1A D6 90 0B C6 9B
0318- A4 9B A5 9C 84 7D 85 7E
0320- 60 A2 5A 4C 00 00 00 00
```

这就是搜索到的B文件RESTORE LINE的内容。

(4) 扩大二进制文件的存贮范围

二进制文件的存贮格式一般为:

BSAVE <文件名>, A <起始地址>, L <长度>, S
<槽号>, D <驱动器号>, V <磁盘册数>

在DOS手册中,对存入磁盘的文件字节长度L有一个规定,其值应在1—32767之间。否则将显示“RANGE ERROR”错误(L=0或L在32768—65535之间),或者显示“SYNTAX ERROR”错误(L>65535时)而拒绝执行。因此,当二进制文件字节长度超过32767时,应进行特殊处理。

常用的方法是,采用程序覆盖技术,把一个长的程序分成二个或二个以上的相对独立的程序段,分别取不同的文件名存入磁盘,运行时再把它们动态链接起来,这样做无疑给

文件管理带来不便。

下面介绍一个简便实用的处理技巧。

通过对 BSAVE 执行程序的分析,发现字节长度是由控制指针 \$A963、\$A964 决定的,一般情况下这两个单元的值分别为 \$FF、\$7F,这正好是十进制的 32767。因此,修改后一单元的值,就能使二进制文件的存贮长度扩大到 65535。

具体方法是:

在监控状态下 (CALL-151),键入: A964: FF ✓

或者在 BASIC 状态下,键入 POKE 43364,255 ✓

由此可知,采用上述方法处理后,二进制文件的存贮长度正好扩大了一倍。

(5) 用机器语言作招呼程序

初始化一个磁盘时,通常编写一个简短的 BASIC 小程序,然后执行 INIT HELLO ✓,作为招呼程序。引导 DOS 时就会自动启动运行这个程序。有些应用软件为简化操作手续,常常用上述方法作为招呼程序,但这仅限于 BASIC 程序。

事实上,也可以用机器语言程序作为招呼程序。

在 DOS 区的 \$9E41 单元有一个 A9 06 的立即取数指令,其中 \$06 的作用是以以后取数的一个脚注,它直接影响到 INIT 指令的执行结果,将这一单元的值由 \$06 改为 \$34,就能使机器语言作为招呼程序。

具体操作如下:

- 在监控状态 (CALL-151) 下输入: 9E 42: 34 ✓,或 BASIC 状态下输入 POKE 40514,52 ✓

- 在 BASIC 状态下输入: INIT HELLO ✓

- 输入: DELETE HELLO ✓

• 输入: BSAVE HELLO, A <起始地址>, L <长度> ✓

其中 A、L 后面为十进制数, 如用十六进制数表示, 则 A、L 改为 A\$, L\$。以后引导这一磁盘即自动运行 HELLO 机器语言程序。

三、实用管理技巧

在《中华学习机编程技巧》一书中，根据语言划分比较系统地介绍了BASIC程序设计和技巧，但在实际使用机器、编制软件中，还有许多并非纯属于BASIC语言方面的行之有效的技巧。如文件管理、机器使用、图形显示、数据维护、功能扩展、磁盘操作、机器语言等等。本章及以后各章就向读者介绍这些方面的一些实用管理技巧。

1. 中华学习机的自动操作

把一些命令或BASIC程序建立一个文本文件，用EXEC命令来执行它，就可以实现由文本文件向内存输入，这就好像从键盘上输入一样：如果是立即执行命令，就执行它；若是带行号的程序行，就把它送入内存的程序区。文本文件中的命令可以顺序地一个一个地进入内存并执行它们。因此，文件的运行不需要人的干预，也不需要从键盘送入命令，从而变成了一部自动中华学习机。

先看一个例子。

例如，要将磁盘中已有的名为PRO-5的程序调入内存，列出程序中20—40行程序段，然后完成下述操作：

接通打印机

列出全部程序清单

运行该程序

将上述程序存盘（取名KKK）

清内存

将KKK文件加锁
再调入内存
关闭打印机
列出KKK清单
列出磁盘文件目录
停机

对于上述操作，通常的方法是依次敲入下面若干命令，来分别实现的。

LOAD PRO-5 ✓
LIST 20, 40 ✓
PR #1 ✓
LIST ✓
SAVE KKK ✓
NEW ✓
LOCK KKK ✓
LOAD KKK ✓
PR #0 ✓
LIST ✓
CATALOG ✓

显然，上述各种操作是完全正确的，但键盘动作过于频繁。

如果把上述命令作为程序输出的数据，存入到一个顺序文件中（取名为ITT），则上述手工进行的操作，将由机器自动完成，并且执行速度很快。

方法如下：

（1）将下述程序（取名为PRO-44）敲入内存：

PRO-44

```
5  REM  SSDD
10 D$ =  CHR$ (4)
20  PRINT D$;"OPEN ITT"
30  PRINT D$;"WRITE ITT"
40  PRINT "LOAD PRO-5"
50  PRINT "LIST20,40"
60  PRINT "PR#1"
70  PRINT "LIST"
80  PRINT "RUN"
90  PRINT "SAVE KKK"
100 PRINT "NEW"
110 PRINT "LOCK KKK"
115 PRINT "LOAD KKK"
120 PRINT "PR#0"
128 PRINT "LIST"
130 PRINT "CATALOG"
140 END
```

(2) 运行上述程序，即可产生一个名为ITT的T文件。

(3) 执行EXEC命令：EXEC ITT ✓

这样，EXEC命令（它也是一个DOS命令）会自动打开名为ITT的文件，并从文件中按顺序将一个个记录送入内存，好像从键盘上输入一样，实现了程序的自动控制。

运行结果：

调程序PRO-5进内存

屏幕显示：

```
20  PRINT "THIS IS A BASIC PROGRA
    M"
30  PRINT "DO YOU WANT PRINT(Y/N)
    ?"
40  PRINT "THIS IS A AUTO APPLE!"
```

看到打印机启动。

打印纸上输出程序清单：

LIST

```
10  REM  PRD-5
20  PRINT "THIS IS A BASIC PROGRA
    M"
30  PRINT "DO YOU WANT PRINT(Y/N)
    ?"
40  PRINT "THIS IS A AUTO APPLE!"

50  PRINT "USE EXEC INSTRUCTION"
60  END
```

打印纸上输出程序运行结果：

```

THIS IS A BASIC PROGRAM
DO YOU WANT PRINT(Y/N)?
THIS IS A AUTO APPLE!
USE EXEC INSTRUCTION
```

```


```

```


```

重新以文件名KKK存入磁盘。

清内存

加锁KKK文件

又调进KKK文件

自动关闭打印机

屏幕上又列出清单（和原来程序 PRO-5 完全一样）：

```
10  REM  PRO-5
20  PRINT "THIS IS A BASIC PROGRA
    M"
30  PRINT "DO YOU WANT PRINT(Y/N)
    ?"
40  PRINT "THIS IS A AUTO APPLE!"

50  PRINT "USE EXEC INSTRUCTION"
60  END
```

屏幕上显示所有文件目录（省略）

从中可以看到：

```
A 002  PRO-44
T 002  ITT
A 002  PRO-5
* A 002  KKK
```

等内容。

注意再一次执行 EXECITT 命令时，除有上述结果外，还会显示“FILE LOCKED”的信息。这是因为前次运行 KKK 文件后已经加锁。

2. 菜单式的DOS管理程序

下面这个程序是一个菜单式的引导程序，它可以用于DOS管理。

（1）程序特点

- 运行该程序，可以列出磁盘目录，并自动给出文件编

号 A, B, C, ……等标记。

- 按任意一个字母 (编号中有的), 即可运行对应该字母的文件, 而无需从键盘中进行诸如 LOAD, BLOAD, RUN, BRUN 等操作, 也不要按回车。

- 可以根据提示, 方便地进行 RUN, BRUN, LOAD, BLOAD, LOCK, UNLOCK, DELETE 等 DOS 操作。

- 减少键盘操作次数, 加快操作速度, 菜单引导方便, 使用简便灵活。

(2) 程序清单 (见程序 PRO-45)

PRO-45

LIST

```
10 REM PRO-45
20 TEXT : HOME : D$ = CHR$ (4): PRINT
   D$"CATALOG": B = PEEK (37) -
   2: IF B > 22 THEN B = 22
30 T = 0: CH = 4: FOR CV = 0 TO 23:
   GOSUB 160: IF C < > 160 THEN
   POKE P - 1, 219: POKE P, T + 1
   93: POKE P + 1, 221: T = T + 1:
   S = CV
40 NEXT CV: VTAB 24: A$ = "TYPE LE
   TTER TO RUN, OR LOAD=1 LOCK=2
   UNLOCK=3 DELETE=4 EXIT=5....
   "
50 B$ = "RUN": HTAB 1: PRINT LEFT$
   (A$, 39): A$ = MID$ (A$, 2) +
   LEFT$ (A$, 1): K = PEEK ( - 1
   6384): IF K < 128 THEN FOR K
   = 1 TO 75: NEXT K: K = FRE (
   0): GOTO 50
60 POKE - 16368, 0: K = K - 176: IF
   K < 1 OR K > 5 THEN 130
70 HTAB 1: CALL - 868: IF K = 5 THEN
   END
80 PRINT "PRESS 'LETTER' YOU WISH
```

```

      TO "": IF K = 1 THEN B$ = "L
      OAD"
90  IF K = 2 THEN B$ = "LOCK"
100 IF K = 3 THEN B$ = "UNLOCK"
110 IF K = 4 THEN B$ = "DELETE": FLASH
120 PRINT B$;: CALL - 198: NORMAL
      : GET K$:K = ASC (K$) - 48
130 IF K < 17 OR K > T + 16 THEN
      50
140 CH = 1:CV = S - T + K - 16: GOSUB
      160: IF C = 194 AND (B$ = "RU
      N" OR B$ = "LOAD") THEN B$ =
      "B" + B$
150 FOR CH = 6 TO 39: GOSUB 160:B
      $ = B$ + CHR$ (C): NEXT CH: HTAB
      1: CALL - 868: PRINT B$: PRINT
      D$;B$: GOTO 20
160 C1 = INT (CV / 8):C2 = CV - C
      1 * 8:P = 1024 + 128 * C2 + 4
      0 * C1 + CH:C = PEEK (P): RETURN

```

(3) 制作过程

① 新盘

- 引导 DOS 3.3进内存
- 清内存 NEW ✓
- 输入上述程序
- 在驱动器里装上一张新的空磁盘
- 键入 INIT HELLO ✓
- 再清内存
- 输入任一程序，并在程序最后部份加上以下程序段：

```

900 INPUT "Q$ = "; Q$
910 IF Q$ = "END" THEN END
920 PRINT CHR $(4); "RUN HELLO"

```


这段小程序行号要放的大一些，和自编程序行号有一段间隔。本程序的目的是若不要运行自编程序，可键入END结束；若要再运行其它文件，机器自动运行引导程序HELLO，并重新列出清单，供使用者选择其它程序。这样既减少了每次清除旧程序的操作，也不要再去手控列目录。整个操作是一个自动进行的过程。

② 旧盘

- NEW✓
- 键入引导程序：PRO-45
- SAVE HELLO✓ 这样就把原来旧盘中的HELLO程序清除，而改用现在名为PRO-45的HELLO招呼程序。
- 开机后运行HELLO程序，即可自动列清单，给编号，并根据提示运行磁盘中的文件
- 为了使旧盘操作自动化，可将旧盘中已存的每一个程序分别调入内存，并在程序的最后加上900—920句的一段程序，重新存入原磁盘。

(4) 使用方法

运行HELLO程序后，屏幕上出现磁盘目录，并自动在每个文件前加有[A]，[B]，[C]，……等编号。而在屏幕底部出现循环左移的活动字样，“TYPE LETTER TO RUN, OR LOAD=1 LOCK=2 UNLOCK=3 DELETE=4 EXTT=5……”

操作者可以根据上述提示，键入“1”，“2”，“3”，“4”等任一数字，屏幕上分别出现PRESS ‘LETTER’ YOU WISH TO LOAD（或LOCK，或UNLOCK，或DELETE）等字幕，它们的意思是询问你进行何种操作，你可以根据目录中文件前的编号，决定操作的对象文件。

例如，键入1，屏幕出现PRESS 'LETTER' YOU WISH TO LOAD，你再按A、B、C、D……等中任一字母，机器自动运行对应该字母的文件。而按5，则返回BASIC状态。

3. 自动换页打印程序清单

APPLE-II 机与打印机配合使用过程中，当程序清单较长时（如每页超过66行），常常发生程序清单打在打印纸换页缝处。这是因为APPLE SOFT 在执行LIST命令打印程序清单时，不具备自动换页功能，从而使打印出来的资料连在一起。这样，不仅影响了程序清单的美观，而且给保存和阅读带来不便。

一般打印机开机后设定，每页打印纸可以打印66行，这就是所谓正常打印方式。因此，如果设法在每页的末尾跳过几个空行，则打印出来的程序清单，就可以有几个空头和空尾。而跳空的行数，可以由使用者根据实际情况自由选定。

跳行处理方法一般有两种：键盘控制和程序控制。

(1) 键盘控制

开机后，接通打印机（打入PR #1，一般打印机接口插件插在1 #槽口上），然后键入：

```
PRINT CHR$(27);CHR$(78);CLR$(N)↵
```

再执行LIST命令，即可达到跳过打印纸之间的换页隙缝，确保程序清单完整和美观。

其中N系根据需要设定的跳空行数之值。注意不同打印机有所区别，应根据实际使用的打印机操作手册修改有关参数。

(2) 程序控制

即编制一个简短的BASIC程序，如程序PRO-46。

PRO-46

LIST

```
10 REM PRO-46
20 D$ = CHR$ (4)
30 PRINT D$; "PR#1"
40 PRINT CHR$ (27); CHR$ (78); CHR$
   (10)
50 PRINT D$; "PR#0"
60 END
```

其中CHR\$(10)中的10，为设定的跳空行数。

这样，在运行上述程序过程中，就可以列自己的程序清单了。

清除跳行的方法，或者打入PRINT CHR\$(27); CHR\$(79)↵；或者关闭打印机，再次启动即可。

下面再介绍一个控制程序，它同样具备打印指定的BASIC程序清单，也有自动换页的功能。程序针对EPSON打印机编制，见程序PRO-47。

PRO-47

LIST

```
10 REM PRO-47
20 INPUT "NAME OF FILE :"; A$
30 D$ = CHR$ (4); K$ = CHR$ (27)
40 PRINT D$; "PR#1": PRINT
50 PRINT K$; "C"; CHR$ (66);
60 PRINT K$; "N"; CHR$ (8)
70 PRINT TAB(5) "---- "; A$ " ----"
80 PRINT D$; "PR#0"
90 PRINT D$; "OPEN LIST"
100 PRINT D$; "WRITE LIST"
```

```

110 PRINT "LOAD";A$
120 PRINT "PR#1": PRINT "LIST": PRINT
    "PR#0"
130 PRINT D$;"CLOSE LIST"
140 POKE 1657,80
150 PRINT D$;"EXEC LIST"

```

程序中40语句设置打印机每页打印 $66 - 8 = 58$ 行,50行语句设置打印机页间隔为8行,60行系打印一个题头,130行设置每行打印80个字符,若只打印40个字符,可改为 POKE 1657, 40。

最后再强调一点,对不同的打印机则需要改动以上有关参数,并要重复试验几次才行。

4. 接受任意字符的输入

INPUT 语句是一个变量输入语句,它能中止程序的执行,等待操作者从键盘输入必要的信息。但是,INPUT语句并不能接受所有的字符,如逗号和冒号等就不能为INPUT所接受,因而造成一些数据信息处理软件编写和操作的不便。为此,可使用下面的这个小程序段来改进INPUT输入,使其接受禁止字符。

这一段小程序主要利用位于系统\$FD6F(64879)地址开始的GETIN程序来工作的。该程序的作用是等待操作者从键盘输入一行信息(可以包含禁止字符),将它显示在屏幕上,并存入键盘缓冲区中。但它并不能对变量进行赋值,因此,1020句呼叫子程序在完成一行信息的输入后,1030句先将X\$置为空,1040句至1070句利用PEEK指令循环读取键盘缓冲区的字符。在读取过程中,假如字符的ASC码不

等于141（即回车键），则加到X\$上去，一直等读取到回车符时循环结束，此时X\$中的值即为键盘输入的信息，见程序PRO-48。

PRO-48

```
1000 REM SAMPLE-8
1010 PRINT "INPUT THE STRING"
1020 CALL - 657
1030 X$ = ""
1040 FOR I = 512 TO 767
1050 IF PEEK (I) = 141 THEN 108
      0
1060 X$ = X$ + CHR$ ( PEEK (I))
1070 NEXT
1080 PRINT X$: END
```

此外，我们再介绍一个程序，作为语句功能扩展的另一个实例，即INPUT语句可以接受分数输入并给出结果，见程序PRO-49。

PRO-49

LIST

```
5 REM 'PRO-49
10 INPUT T$
20 IF T$ = "END" THEN END
30 Y = VAL (T$)
40 IF STR$ (Y) = T$ THEN 70
45 GOSUB 100
50 PRINT "YOUR INPUT IS: "; A1; "/" A
  2; "="; A1 / A2
60 GOTO 80
70 PRINT "YOUR INPUT IS: "; Y
80 GOTO 10
100 L = LEN (T$)
110 K = 0: A1 = 0: A2 = 0
120 FOR I = 1 TO L
```

```

130 IF MID$ (T$,I,1) = "/" THEN
140 K = K + .1
150 NEXT I
160 T1$ = LEFT$ (T$,K)
170 T2$ = RIGHT$ (T$,L - K - 1)
180 A1 = VAL (T1$):A2 = VAL (T2$)
190 RETURN

```

```

URUN
?1/2
YOUR INPUT IS:1/2=.5
?3/4
YOUR INPUT IS:3/4=.75
?23/5
YOUR INPUT IS:23/5=4.6
?END

```

5. DOS 的阻截失误

在包含有 DOS 操作的 BASIC 程序中，有时会碰到这样一种情况：位于 PRINT CHR \$ (4) 之后的 DOS 命令操作不能被正常执行，而作为一般的打印语句在屏幕或打印机上显示或输出。这就是 DOS 的阻截失误现象。

先看一个简单实例见程序 PRO-50。

PRO-50

LIST

```

5 REM PRO-50
10 D$ = CHR$ (4)
20 FOR K = 1 TO 10
30 PRINT K,
40 NEXT
50 PRINT D$;"RUN USING 2"

```

这个程序表面上看没有语法错误，但运行后，却得到下面结果：

```
URUN
1          2          3
4          5          6
7          8          9
10         RUN USING 2
```

说明50句的连接运行命令未被执行，使它变成了打印输出的对象，从而无法实现连接运行的功能。分析原程序可以看出，第30行规定了按标准格式输出，逗号起到了预留显示空白位置的作用，结果影响了第50句的正确执行。

这个实例告诉我们，在程序中使用DOS命令能使主机和外部设备方便地交换信息，但如果使用不当会造成失误。即使标点符号也会影响DOS命令的正确执行，所以一定要严格按照编写格式正确使用这些命令。

对于上述程序，若加一行45 PRINT就可结束标准输出格式，顺利执行连接运行的DOS命令。

同样，程序PRO-51中1020句删除文件的操作也不被执行，而在屏幕上或打印机上输出字符串“DELETE ABC”。

PRO-51

ULIST

```
1000 REM PRO-51
1010 PR# 1
1020 PRINT CHR$(4); "DELETE ABC"

1030 END
```


URUN
DELETE ABC

现在，我们较为深入地讨论DOS阻截失误的原因。

众所周知，DOS命令作为BASIC语句必须放在PRINT CHR\$(4)后面的引号之中，系统通过这一特殊格式将它与一般输入输出语区相区别。在DOS有效时，所有的输入输出都首先由它阻截下来进行判断，如是DOS命令则由DOS自行完成，否则交BASIC解释程序执行，这就是DOS的阻截作用。

所有的输入输出信息，最终被传送到哪里由零页输入输出指针（输入\$38，\$39，输出\$36，\$37）来控制，或者指向DOS，或者指向外设输入输出程式。而由DOS阻截下来DOS指令在执行时，首先将系统输入输出暂存区的内容传送至DOS的输入及输出暂存区，然后指令字符经暂存区传送到某种设备上。之后，这一零页指针又自动与DOS相连接，指向DOS的处理程序。

那么程序PRO-51中第1020句的DOS指令，为什么没有被阻截下来呢？

由以上分析可知，任何一个输入输出指令，在到达DOS时，即可改变零页输入输出指针，使其不指向DOS，因而切断了DOS的联系。造成阻截失误。

例如，上述程序中的PR#1命令，使输出指针指向\$C100的印字程式，而不指向DOS，这就是其后的1020句不执行阻截的原因。同样，PR#0输出指针指向\$FDF0的显示程式，也使其后的DOS操作不被执行。

上述失误一直到执行一个输入操作时才能消除。原因是PR#虽然改变了输出指针，但没有改变输入指针。一旦有

输入操作,DOS 仍然进行阻截作用,然后自动将输入、输出指针一起恢复,从而中止阻截失误现象的继续。

这样,上述程序中的问题可用以下方法之一来解决:

(1) 在1020句前插入一程序行

```
1015 POKE 54, 189 : POKE 55, 158: POKE  
56, 129: POKE 57, 158
```

这样做就是硬性恢复输入和输出指针,使其指向DOS。

(2) 增加一个语句

```
1015 CALL 43089
```

这是调用恢复指针的系统子程序。

(3) 将1010句改写为

```
1010 PRINT CHR$(4); "PR#1"
```

这样,受DOS阻截执行后,指针自动恢复而转向DOS。

(4) 在DOS操作前加输入语句

```
1015 INPUT A$
```

这样处理后,由于零页输入指针仍指向DOS,INPUT指令为DOS所阻截,执行后零页二个指针恢复,使其后的删除文件指令也为DOS所阻截。

读者可以根据上述方法,逐一投试程序,检验其方法的正确性。

其中文件ABC应为磁盘中已经存放的文件。但当删除的文件ABC根本不存在时,上述删除命令将执行不下去。

6. 内存程序的截断与混乱

在内存BASIC程序的运行和调试过程中,有时会出现全部或部分程序的丢失,或程序发生混乱的现象,从而使程序或图形被破坏掉,造成不应有的损失和麻烦。

这类现象常常由于程序较长，或变量较多，或数组较大，延伸到不是它们应该工作的存贮区造成的。例如：

(1) 用户程序及各类变量应存贮在\$ 800至\$ 1 FFF区域内，如果程序尾超出程序区（\$ 800—\$ 1 FFF，即2048—8191），而侵入高分率图形第一页屏幕存贮区（\$ 2000—\$ 3 FFF，相应的十进制为8192—16383），同时程序中使用了HGR作图语句，则运行该指令时自动截去超出程序区的部份。

(2) 程序尾超出第一页高分辨率作图区，而侵入第2页高分辨图形存贮区（\$ 4000—\$ 5 FFF，即16384—24575），同时程序中使用了HGR2作图指令，则运行时该指令自动截去超出第一页高分辨率区域区的部份。

(3) 使用第2页文字页，而没有设置程序区位置(地址)，或程序中涉及的机器语言内存地址误入程序区，造成程序混乱或丢失。

因此，必须采取必要的保护措施，以保证程序或图形的安全。

了解程序存贮是否超过了不应该存贮的区域，就应该了解存放程序的尾地址。由于程序的尾指针是在零页单元（\$ 0000—\$ 00FF）中它为系统所用，主要用来设置一些指针和一些工作单元。其中\$ AF（175）、\$ BO（176）就存放了尾指针地址，因此，程序调入内存后，用：

PRINT PEEK(175) + PEEK(176) × 256 ✓

即可得到该程序的十进制数的存放尾地址。

因此，根据程序尾的地址情况，可以分别采取相应措施进行保护。

• 如程序尾足够小于8191，程序变量可以存放在程序

区，则高分辨作图不会造成程序的截断，也不会影响程序的运行。

- 如程序尾在程序区内，但程序区已没有足够的剩余空间来存放变量，则必须指定 LOMEM 的值（LOMEM 命令常用来保护高清晰度图形），使变量存放在作图页的上方。这一设置必须在程序运行之前或开始。即：

LOMEM: 16384

或 LOMEM: 24576

- 如程序尾已超出程序区，而侵入相应的高分辨率作图区，则必须在程序调入内存之前，指定程序存放的起始地址，使程序及其变量均存放在相应作图页的上方。指令：

POKE 103,1:POKE 104,64:POKE 16384,0

(以上指存放在第一页高分辨图页上方)

POKE 103,1:POKE 104,96:POKE 24576,0

(以上系存放在第二页高分辨作图页上方)

由内存内配知道，从第2页高分辨作图区至DOS区间 \$ 6000 — \$ 9600 (24576 — 38400) 共有13K以上的存贮空间，这一空间对一般程序来说已足够存贮。例如，我们常常将一些比较冗长的机器语言子程序安全地存贮在这一区域，使用时用POKE指令设置，用CALL指令调用。

7. 机器语言子程序转换成BASIC程序

假定内存中已有一段机器语言子程序PRO-52，它存贮在\$ 0300 — \$ 0319的单元中，现在设法将它转换成BASIC程序。

PRO-52

*300.319

```
0300- 98 46 AD 30 C0 88 D0 05
0308- CE 01 03 F0 09 CA D0 F5
0310- AE 01 03 4C 02 03 60 00
0318- 00 9B
```

方法是借助于辅助程序PRO-53来达到转换目的。

PRO-53

```
5  REM  CCCC
10  D$ =  CHR$ (4)
20  PRINT D$; "OPEN CODE-POKES"
30  PRINT D$; "DELETE CODE-POKES"
40  PRINT D$; "OPEN CODE-POKES"
50  PRINT D$; "WRITE CODE-POKES"
70  FOR PLACE = 770 TO 790
80  COUNTER = COUNTER + 1
90  IF COUNTER = 10 THEN COUNTER =
    1
100  IF COUNTER <  > 1 THEN 140
110  PRINT
120  PRINT LINENUMBER;
130  LINENUMBER = LINENUMBER + 10
140  PRINT "POKE"; PLACE; ", "; PEEK
    (PLACE); ": ";
150  NEXT PLACE
160  PRINT
170  PRINT D$; "CLOSE CODE-POKES"
180  END
```

当上述程序检查无误后，立即运行一次，这样，就建立了名为CODE-POKES的文本文件。

NEW ↙

执行EXEC命令：

EXEC CODE-POKES ↙

屏幕上立即连续显示四个提示符，再执行LIST命令，

屏幕上显示以下结果:

UEXEC CODE-POKES

U

U

U

U

ULIST

```
0  POKE 770,173: POKE 771,48: POKE
    772,192: POKE 773,136: POKE
    774,208: POKE 775,5: POKE 77
    6,206: POKE 777,1: POKE 778,
    3:
10  POKE 779,240: POKE 780,9: POKE
    781,202: POKE 782,208: POKE
    783,245: POKE 784,174: POKE
    785,1: POKE 786,3: POKE 787,
    76:
20  POKE 788,2: POKE 789,3: POKE
    790,96:
```

这样,我们就顺利地完成了将汇编语言(机器码)程序,迅速转换成BASIC程序了。或者说将机器语言子程序(其实不只是程序)变成了用POKE语言组成的BASIC程序。

为了说明这种转换的正确性,不妨剖析一下原机器语言程序。

\$302相当于十进制的770,这个地址中的内容\$AD,相当于十进制数173。

同样从\$303开始到\$316,对应的十进制数为771到790,而各地址中对应的十进制数分别为48(\$30)到96

(\$ 60)。

换句话说,原机器语言子程序,相当于下面这个BASIC程序PRO-54。

PRO-54

```
10 REM PRO-54
20 FOR I = 0 TO 20
30 READ N
40 POKE I + 770, N
50 NEXT I
60 DATA 173, 48, 192, 136, 208, 5, 206
    , 1, 3, 240, 9, 202, 208, 245, 174, 1,
    3, 76, 2, 3, 96
```

因此,执行 EXEC CODE-POKES 命令后,所得结果,和上述机器语言子程序以及上述BASIC程序,是完全等效的。

8. 显示T文件内容的方法

在磁盘文件中, A类或B类文件,一般可用LOAD或BLOAD命令,就能察看其中内容,而要将T类文件在屏幕上显示出来,同时也能在打印机上输出文件内容,就不能用上述命令。

显示或运行T类文件的方法有好几种。

一种是用EXEC命令运行。

例如,已存入盘中的T类文件,包括原有DOS系统盘中带有T标记的文件,都不能用RUN命令运行,而必须用EXEC命令。其格式为:

EXEC (文件名)✓

但是,如果原来建立的T类文件是为了保存一些文章、

信件、资料内容，那么每一条语句用EXEC运行，不能得到结果，只能在屏幕上不断显示

? SYNTAX ERROR (语法错误)

解决的方法是预先键入“MON I, C, O↵”，就可以显示每一语句的内容。我们可用这种粗糙的方法作为检索T类文件内容的一个手段。

如果原来建立的T类文件，是一种游戏或表演程序，那么T类文件中每个语句都对应于一条可以执行的DOS命令。用EXEC运行这种文件时，就依次执行DOS命令，在屏幕上显示各种花样。而且，一条命令执行后，可以从T类文件中读取下一条命令，再不断执行……。这样，就好像代替人工从键盘上不断输入DOS命令一样。

另一种方法是用程序控制。

例如，程序PRO-55，就可以方便地察看T文件内容，并能打印输出。

程序PRO-55，最好先存盘中，使用时再调入内存，因为它可以作为一个检索T类文件的工具软件。

清单如下：

PRO-55

```
10 REM READ T TEXT
25 D$ = CHR$(4):P = 0: ONERR GOTO
   35
30 HOME : VTAB 3: HTAB 12: PRINT
   "READ TEXT PROGRAM
35 VTAB 5: HTAB 5: PRINT "DO YOU
   WANT CATALOG ";: INPUT A$
40 IF A$ < > "Y" AND A$ < > "N
   " THEN 35
45 IF A$ = "N" THEN 55
50 POKE 34,7: PRINT D$;"CATALOG"
```

```

      : PRINT : PRINT "PRESS ANY K
      EY";: GET A$
55  VTAB 7: HTAB 5: INPUT "FILE N
      AME ";C$: IF ASC (C$) < 64 THEN
      55
60  POKE 34,7: HOME
65  PRINT D$;"OPEN";C$
70  PRINT D$;"READ";C$
75  GET A$: IF A$ = CHR$ (13) THEN
      PR# 1: PRINT CHR$ (0);B$:B
      $ = " ";P = 1: GOTO 75
80  B$ = B$ + A$: GOTO 75
85  IF P = 1 THEN PRINT D$;"CLOS
      E";C$: END
90  IF PEEK (222) = 13 THEN PRINT
      "FILE TYPE MISMATCH"; CHR$ (
      7): VTAB 22: PRINT "PRESS AN
      Y KEY TO INPUT ASAIN ";: GET
      A$: POKE 54,0: HOME : RUN
95  IF PEEK (222) = 3 THEN PRINT
      "NO DISK IN DRIVE";:CHR$(7):G
      ET A$:POKE34,0:HOME:RUN
100 PRINT "FILE NOT FOUND "; CHR$
      (7): PRINT D$;"DELETE";C$: POKE
      34,0: HOME : RUN

```

使用方法比较简单。调入内存后运行之，则屏幕出现：

```

      READ TEXT PROGRAM
      DO YOUWANT CATALOG ?

```

意思是询问你是否需要查看磁盘目录，若要，则按 Y↵
列出磁盘目录，如：

```

DISK VOLUME 254
A 002 HELLO
A 002 CAP
A 004 READ TEXT PROGRAM
T 002 LISTING

```

然后屏幕出现:

PRESS ANY KEY

则按任意键后出现:

FILE NAME

此时键入文件名, 如名为LISTINGT类文件, 即可显示T文件的内容。

如LISTING文件的内容为:

ULIST

```
10 N = 3: A$ = "BBB"
20 DIM A(10,N)
30 FOR I = 1 TO 10
40 S = 0
50 FOR J = 1 TO N
60 INPUT A(I,J)
70 S = S + A(I,J)
80 NEXT J
90 A(I,0) = S / N
100 NEXT I
200 D$ = CHR$(4)
210 PRINT D$;"OPEN";A$
220 PRINT D$;"WRITE";A$
230 FOR I = 1 TO 10
240 FOR J = 0 TO N
250 PRINT A(I,J);", ";
260 NEXT J
270 PRINT
280 NEXT I
290 PRINT D$;"CLOSE";A$
300 END
```

若不要列磁盘目录, 则按N↵, 屏幕上也会出现FILE NAMN, 再键入需要显示的T类文件名。

若要在打印机上输出, 可事先打开打印机电源。

中断可按RESET键。

9. 同时打开多个文件的方法

在数据处理的过程中，有时候需要将一个程序的结果暂时保存起来，以备为另一个程序所使用。用同时打开几个文件的方法，可以实现程序之间的数据传递，并且由于数据存于盘中，不仅容量大，而且能脱机保存，减少了多次操作键盘的次数。

例如，有10个学生，第一学期每人考5门功课，第二学期每人考3门功课，试编制一个程序，要求：

- 记录第一学期每人5门功课的成绩和平均分；
- 记录第二学期每人3门功课的成绩和平均分；
- 求出一、二两学期平均分的和。

第一个问题可通过下面程序，从键盘上输入每个学生的考分，运行后即产生名为AAA的T文件。其考分及平均成绩存到这个磁盘文件上，第0号字段放置平均分，其它字段放这个学生的各门考分。

LIST

```
10 N = 5:A$ = "AAA"
20 DIM A(10,N)
30 FOR I = 1 TO 10
40 S = 0
50 FOR J = 1 TO N
60 INPUT A(I,J)
70 S = S + A(I,J)
80 NEXT J
90 A(I,0) = S / N
100 NEXT I
200 D$ = CHR$(4)
210 PRINT D$;"OPEN";A$
220 PRINT D$;"WRITE";A$
230 FOR I = 1 TO 10
```

```

240 FOR J = 0 TO N
250 PRINT A(I,J);", ";
260 NEXT J
270 PRINT
280 NEXT I
290 PRINT D$; "CLOSE"; A$
300 END

```

第二个学期每人只考 3 门课，其它问题同第一学期处理方法，故只需要将上述程序的第 10 句改为：

```

10          N = 3: A $ = "B B B"

```

运行时，同样从键盘上依次输入每个学生的 3 个考分，运行结束后将建立一个名为 BBB 的 T 类文件。

用 CATALOG 命令，可以检查出磁盘目录中多了两个 T 文件，一个是 T 001 A A A；另一个是 T 001 BBB。

为实现本题中提出的第三个要求，可以通过下列程序，再建立一个名为 G G G 的 T 文件，在这个文件上将记载这些学生一学年两个学期平均分的总和：

ULIST

```

10 D$ = CHR$ (4)
20 PRINT D$; "OPEN AAA"
30 PRINT D$; "OPEN BBB"
40 PRINT D$; "OPEN GGG"
50 FOR I = 1 TO 10
60 PRINT D$; "READ AAA"
70 INPUT A
80 PRINT D$; "READ BBB"
90 INPUT B
100 PRINT D$; "WRITE GGG"
110 PRINT A + B
120 NEXT I
130 PRINT D$; "CLOSE"
140 END

```

在这个程序中，80句的 READ命令，会使60句的 READ命令失效；同样，100句的 WRITE命令又会使80句的 READ失效。NEXT回来后，60句的 READ命令重新下达，发生作用，而它却使100句的 WRITE失效。这正是我们所需要的。

为察看上述程序的正确性，了解每个T文件的存贮情况，我们可以用前节介绍的读T类文件的方法。运行结果如下：

(1) 对文件AAA

```
DO YOUWANT CATALOG ?N
FILE NAME AAA
80.8,98,78,76,65,87,
66.8,65,45,78,79,67,
79.6,58,98,87,90,65,
73.8,91,52,65,74,87,
69.6,98,87,65,54,44,
66.6,29,76,65,76,87,
89,98,89,87,86,85,
65.2,54,39,48,98,87,
76.6,70,78,79,80,76,
83.8,89,87,66,88,89,
```

(2) 对文件BBB

```
DO YOUWANT CATALOG ?N
FILE NAME BBB
78,66,78,90,
76.6666667,87,56,87,
81.3333334,90,87,67,
74,56,73,93,
92,94,95,87,
76.6666667,65,77,88,
89.6666667,99,90,80,
77,78,67,86,
65,75,65,55,
90,76,99,95,
```

(3) 对文件GGG

```
DO YOU WANT CATALOG ?N
FILE NAME GGG
158.8
143.466667
160.933333
147.8
161.6
143.266667
180.666667
142.2
141.6
175.8
```

10. N组实验数据的合并

设某实验要求在同一时刻不同地点取样，各单个测点的实验数据分别存放在各自的盘中，程序名分别为LL-1-1, LL-2, LL-3。实验结束后要求按同一程序LL-1处理各采样的数据。这就要解决N组试验数据的合并问题，并需要解决程序LL-1与各组数据的连接问题。

例如，处理程序LL-1如下：

```
ULIST

2  REM  LL-1
5  REM  Z-I-W-1987.7.1-7.14
10 DIM X(150),Y(150)
40 REM  Z-R-W
50 N = 101
60 FOR I = 1 TO N: READ X(I): NEXT
   I
70 FOR I = 1 TO N: READ Y(I): NEXT
   I
80 FOR I = 1 TO N: Y = LOG (Y(I))
   :X = LOG (X(I))
```



```

90 S1 = S1 + X
100 S2 = S2 + Y
110 S3 = S3 + X * Y
120 S4 = S4 + X * X
130 S5 = S5 + Y * Y
140 NEXT I
150 L1 = S3 - S1 * S2 / N
160 L2 = S4 - S1 * S1 / N
170 L3 = S5 - S2 * S2 / N
180 R = L1 / SQR (L2 * L3)
190 B = L1 / L2
200 S = SQR ((1 - R ^ 2) * L3 / (
    N - 2))
210 A = EXP ((S2 - B * S1) / N)
220 PRINT "R=";R
230 PRINT "A=";A
240 PRINT "B=";B
250 PRINT "Z=A*I^B=";A; "*I"; "^"; B

```

第一测点数据记录为LL-1-1:

ULOADLL-1-1.

ULIST

```

399 REM LL-1-1
400 DATA 0.63,0.79,1.12,0.53,0
    .40,0.36,0.16,0.02,0.42,1.54,
    2.31,3.43,5.98,3.48,2.25,1.4
    8,3.04,0.34,0.18,0.24,0.09,0.
    05,0.16,0.07,0.05,0.04
600 DATA 129.70,170.68,196.97,69.
    50,49.78,41.69,13.28,3.42,225
    .27,879.33,1330.53,1794.88,48
    14.57,1979.53,890.70,570.99,7
    08.71,45.23,17.15,71.27,12.28
    ,13.53,66.20,15.27,9.46,6.27

```

第二测点数据记录为LL-2:

ULIST

```
409 REM LL-2
410 DATA 0.05,0.03,0.24,0.02,0.
    06,1.20,1.16,0.84,0.17,0.36,2
    .61,0.77,0.53,3.05
610 DATA 6.66,1.78,88.00,1.04,4.8
    8,440.86,360.43,264.56,18.76,
    131.37,1027.40,223.34,84.92,1
    123.18
```

第三测点数据记录为LL-3:

ULIST

```
419 REM LL-3
420 DATA 5.11,30.72,5.20,23.74,6.
    40,0.39,7.68,34.18,22.99,11.0
    9,9.74,11.64,5.96,1.09,14.12,
    35.92,33.80,15.31,2.86,3.77,3
    .83,6.18,0.79,2.11,1.38,1.51,
    1.91,0.58,3.31,2.17,5.65,3.50
    ,6.01,8.42,5.45,3.56,0.35,2.3
    3,3.58,2.53,1.25,1.52
425 DATA 2.73,26.84
430 DATA 0.88,0.55,0.36,0.32,1.58
    ,1.03,22.98,10.03,3.26,8.03,1
    4.40,13.17,2.34,3.67,1.87,2.5
    5,0.03
620 DATA 1283.65,12346,1150.08,10
    657.06,2929.16,50.12,3313.26,
    19430.25,13744.12,5957.58,455
    4.44,5134.23,1522.14,114.79,5
    036.62,16222.65,39684.12,6352
    .41,739.82,851.83,799.8,2575.
    59,350.11,819.78,725.91,337.9
    5,885.71,247.53,1625.56,1176.
    15
630 DATA 2615,2092.94,3832.80,7
```

```

512.42, 3572.72, 2400.11, 100.16
, 1374.26, 3111.48, 1247.63, 390.
26, 569.34, 1622.84, 28829.61
640 DATA 251.78, 129.59, 169.32, 63.
11, 666.59, 457.78, 25936.28, 102

```

方法是利用DOS 系统盘的RENUMBER程序，严格按照下述步骤操作，即可完成四个文件（一个程序和三组实验资料）的合并工作。

- (1) 引导DOS 3.3
- (2) RUN RENUMBER↵
- (3) 按RETURN
- (4) 取出DOS 3.3
- (5) 分别放、取四个文件的磁盘，按下述次序操作：

```

LOADLL-1
U&H
PROGRAM ON HOLD, USE "&M" TO RECOVER

```

```

LOADLL-1-1
U&M

```

```

U&H
PROGRAM ON HOLD, USE "&M" TO RECOVER

```

```

LOADLL-2
U&M

```

```

U&H
PROGRAM ON HOLD, USE "&M" TO RECOVER

```

```

LOADLL-3
U&M

```

最后按LIST↵，即可完成程序和全部数据的连接工

作，连接完成的程序清单如下：

ULIST

```
2  REM  LL-1
5  REM  Z-I-W-1987.7.1-7.14
10 DIM X(150),Y(150)
40 REM  Z-R-W
50 N = 101
60 FOR I = 1 TO N: READ X(I): NEXT
   I
70 FOR I = 1 TO N: READ Y(I): NEXT
   I
80 FOR I = 1 TO N: Y = LOG (Y(I))
   : X = LOG (X(I))
90 S1 = S1 + X
100 S2 = S2 + Y
110 S3 = S3 + X * Y
120 S4 = S4 + X * X
130 S5 = S5 + Y * Y
140 NEXT I
150 L1 = S3 - S1 * S2 / N
160 L2 = S4 - S1 * S1 / N
170 L3 = S5 - S2 * S2 / N
180 R = L1 / SQR (L2 * L3)
190 B = L1 / L2
200 S = SQR ((1 - R ^ 2) * L3 / (
   N - 2))
210 A = EXP ((S2 - B * S1) / N)
220 PRINT "R=";R
230 PRINT "A=";A
240 PRINT "B=";B
250 PRINT "Z=A*I^B=";A;"*I";"^";B

399 REM  LL-1-1
400 DATA 0.63,0.79,1.12,0.53,0
   .40,0.36,0.16,0.02,0.42,1.54,
   2.31,3.43,5.98,3.48,2.25,1.4
   8,3.04,0.34,0.18,0.24,0.09,0.
   05,0.16,0.07,0.05,0.04
409 REM  LL-2
410 DATA 0.05,0.03,0.24,0.02,0.
```

06,1.20,1.16,0.84,0.17,0.36,2
.61,0.77,0.53,3.05

419 REM LL-3
420 DATA 5.11,30.72,5.20,23.74,6.
40,0.39,7.68,34.18,22.99,11.0
9,9.74,11.64,5.96,1.09,14.12,
35.92,33.80,15.31,2.86,3.77,3
.83,6.18,0.79,2.11,1.38,1.51,
1.91,0.58,3.31,2.17,5.65,3.50
,6.01,8.42,5.45,3.56,0.35,2.3
3,3.58,2.53,1.25,1.52
425 DATA 2.73,26.84
430 DATA 0.88,0.55,0.36,0.32,1.58
,1.03,22.98,10.03,3.26,8.03,1
4.40,13.17,2.34,3.67,1.87,2.5
5,0.03
600 DATA 129.70,170.68,196.97,69.
50,49.78,41.69,13.28,3.42,225
.27,879.33,1330.53,1794.88,48
14.57,1979.53,890.70,570.99,7
08.71,45.23,17.15,71.27,12.28
,13.53,66.20,15.27,9.46,6.27
610 DATA 6.66,1.78,88.00,1.04,4.8
8,440.86,360.43,264.56,18.76,
131.37,1027.40,223.34,84.92,1
123.18
620 DATA 1283.65,12346,1150.08,10
657.06,2929.16,50.12,3313.26,
19430.25,13744.12,5957.58,455
4.44,5134.23,1522.14,114.79,5
036.62,16222.65,39684.12,6352
.41,739.82,851.83,799.8,2575.
59,350.11,819.78,725.91,337.9
5,885.71,247.53,1625.56,1176.
15
630 DATA 2615,2092.94,3832.80,7
512.42,3572.72,2400.11,100.16
,1374.26,3111.48,1247.63,390.
26,569.34,1622.84,28829.61
640 DATA 251.78,129.59,169.32,63.

11,666.59,457.78,25936.28,102
 49.55,1366.66,2865.90,6621.20
 ,7700.94,844.39,4892.55,833.6
 ,1692.33,11.49

11. 定义更多的功能键

APPLE-II, 紫金-II, 中华学习机虽有一些功能键, 但为了加快程序的输入速度, 还可以定义更多的功能键, 这对使用者可以带来更多的方便。

具体做法如下:

(1) 键入程序 PRO-56, 运行后打印出一个菜单式的图表:

ORUN

```

      APPLE-KEY
      CALL 37600
1-CATALOG      2-LOAD      3-SAVE
4-D$=          5-PRINTD$;  6-DELETE
7-NORMAL      8-INVERSE   9-FLASH
O-LIST        --RUN       A-ASC$(
B-DEFFN       C-CALL      D-DATA
E-PEEK        F-FORI=1TO  G-GOTO
H-CALL-936    I-INPUT     J-READ
K-TAB         L-LEFT$(    M-MID$(
N-NEXT        O-POKE      P-PDL(
Q-RESTORE     R-RETURN    S-GOSUB
T-THEN        U-USR(      V-VTAB
W-COLOR=      X-XDRAW     Y-DRAW
;-RIGHT$(    ,-CHR$(     .-STR$(
  
```

PRO-56

ULIST

```
10 HOME
20 VTAB 1: HTAB 15: PRINT "APPLE
   -KEY"
30 VTAB 3: HTAB 14: PRINT "CALL
   37600"
40 VTAB 5: HTAB 8: PRINT "1-CATA
   LOG      2-LOAD      3-SAVE"
50 VTAB 6: HTAB 8: PRINT "4-D$=
   5-PRINTD$; 6-DELETE"
60 VTAB 7: HTAB 8: PRINT "7-NORM
   AL      8-INVERSE   9-FLASH"
70 VTAB 8: HTAB 8: PRINT "0-LIST
   --RUN      A-ASC$("
80 VTAB 9: HTAB 8: PRINT "B-DEFF
   N      C-CALL      D-DATA"
90 VTAB 10: HTAB 8: PRINT "E-PEE
   K      F-FORI=1TO G-GOTO"
100 VTAB 11: HTAB 8: PRINT "H-CA
   LL-936 I-INPUT      J-READ"
110 VTAB 12: HTAB 8: PRINT "K-TA
   B      L-LEFT$(      M-MID$("

120 VTAB 13: HTAB 8: PRINT "N-NE
   XT      O-POKE      P-PDL("
130 VTAB 14: HTAB 8: PRINT "Q-RE
   STORE R-RETURN      S-GOSUB"

140 VTAB 15: HTAB 8: PRINT "T-TH
   EN      U-USR(      V-VTAB"
150 VTAB 16: HTAB 8: PRINT "W-CO
   LOR=      X-XDRAW      Y-DRAW"
160 VTAB 17: HTAB 8: PRINT ";-RI
   GHT$(      ,-CHR$(      .-STR$("

170 PRINT CHR$(4); "BLOAD APPLE
   -KEY": GET A$: PRINT : PRINT
   CHR$(4); "CATALOG": NEW
```


(2) 插入一张新磁盘，键入INIT HELLO↵，用上面的程序作为HELLO招呼程序，对新磁盘进行初始化。

(3) 键入CALL-151↵

(4) 输入下面机器语言程序PRO-57

(5) 将机器语言程序录入初始化的新磁盘，即执行BSAVE APPLE-KEY, A\$ 92E0, L\$ 1C2↵

以后每次用上述磁盘开机时，屏幕上首先出现前面那张提示表，告诉我们每个键所对应的功能。

然后执行BRUN PRO-57↵，就具备功能键了。

例如，键入下述程序：

ULIST

```
10 FOR I = 1 TO 5
20 READ X
26 PRINT X
30 NEXT
40 DATA 10,20,30,40,50
```

可用下述方法处理：

键入10，按ESC键，再按F，则相当于：

10 FOR I= 1 TO, 再键入5↵

键入20，按ESC键，再按J，相当于逐个敲入R、E、A、D，再键入X↵

对于40句，只要键入40，按ESC，再按N↵即可。

对于50句中的DATA，按ESC，再按D↵，顺序键入各数字即可。

按RESET键，计算机退出功能键状态。

键入CALL37600又重新恢复功能键。

PRO-57

UCALL-151

*92E0.94A2

| | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|
| 92E0- | A9 | 06 | 85 | 3B | A9 | 93 | 85 | 39 |
| 92E8- | 4C | EA | 03 | A4 | 24 | B1 | 2B | 4B |
| 92F0- | 29 | 3F | 09 | 40 | 91 | 2B | 6B | 60 |
| 92F8- | E6 | 4E | D0 | 02 | E6 | 4F | 2C | 00 |
| 9300- | C0 | 10 | F5 | 91 | 2B | 60 | 20 | FB |
| 9308- | 92 | BE | 98 | 93 | AD | 96 | 93 | D0 |
| 9310- | 63 | AD | 00 | C0 | 2C | 10 | C0 | C9 |
| 9318- | 9B | F0 | 01 | 60 | A9 | A0 | 20 | EB |
| 9320- | 92 | 20 | F8 | 92 | AD | 00 | C0 | C9 |
| 9328- | BE | 90 | 11 | C9 | AC | 90 | 04 | C9 |
| 9330- | DB | 90 | 2A | 2C | 10 | C0 | 20 | EB |
| 9338- | 92 | 4C | 06 | 93 | C9 | 85 | B0 | 0C |
| 9340- | 2C | 10 | C0 | 69 | 40 | 3B | 20 | 2C |
| 9348- | FC | 4C | 1C | 93 | C9 | 89 | 90 | E3 |
| 9350- | C9 | 8C | F0 | DF | 1B | 69 | 40 | 20 |
| 9358- | 9B | FB | 4C | 1C | 93 | E9 | AA | 8D |
| 9360- | 97 | 93 | A2 | FF | A9 | C0 | EB | DD |
| 9368- | 99 | 93 | D0 | FA | CE | 97 | 93 | D0 |
| 9370- | F5 | BE | 96 | 93 | AE | 96 | 93 | EB |
| 9378- | BD | 99 | 93 | BE | 96 | 93 | AE | 98 |
| 9380- | 93 | C9 | C0 | D0 | 10 | A9 | 00 | BD |
| 9388- | 96 | 93 | A9 | A0 | 2C | 10 | C0 | 20 |
| 9390- | EB | 92 | 4C | 06 | 93 | 60 | 00 | 00 |
| 9398- | 09 | C0 | C3 | CB | D2 | A4 | AB | C0 |
| 93A0- | D2 | D5 | CE | C0 | D3 | D4 | D2 | A4 |
| 93AB- | AB | C0 | C0 | CC | C9 | D3 | D4 | C0 |
| 93B0- | C3 | C1 | D4 | C1 | CC | CF | C7 | 8D |
| 93B8- | C0 | CC | CF | C1 | C4 | C0 | D3 | C1 |
| 93C0- | D6 | C5 | C0 | C4 | A4 | BD | A2 | 84 |
| 93C8- | A2 | C0 | D0 | D2 | C9 | CE | D4 | C4 |
| 93D0- | A4 | BB | A2 | C0 | C4 | C5 | CC | C5 |
| 93D8- | D4 | C5 | C0 | CE | CF | D2 | CD | C1 |
| 93E0- | CC | C0 | C9 | CE | D6 | C5 | D2 | D3 |
| 93E8- | C5 | C0 | C6 | CC | C1 | D3 | CB | C0 |
| 93F0- | C0 | D2 | C9 | C7 | CB | D4 | A4 | AB |
| 93F8- | C0 | C0 | C0 | C0 | D0 | D2 | C9 | CE |

```

9400- D4 C0 C0 C1 D3 C3 A4 A8
9408- C0 C4 C5 C6 C6 CE C0 C3
9410- C1 CC CC C0 C4 C1 D4 C1
9418- C0 D0 C5 C5 CB C0 C6 CF
9420- D2 C9 BD B1 D4 CF C0 C7
9428- CF D4 CF C0 C3 C1 CC CC
9430- AD B9 B3 B6 C0 C9 CE D0
9438- D5 D4 C0 D2 C5 C1 C4 C0
9440- D4 C1 C2 A8 C0 CC C5 C6
9448- D4 A4 A8 C0 CD C9 C4 A4
9450- A8 C0 CE C5 D8 D4 C0 D0
9458- CF CB C5 C0 D0 C4 CC A8
9460- C0 D2 C5 D3 D4 CF D2 C5
9468- BA C0 D2 C5 D4 D5 D2 CE
9470- C0 C7 CF D3 D5 C2 C0 D4
9478- C8 C5 CE C0 D5 D3 D2 A8
9480- C0 D6 D4 C1 C2 C0 C3 CF
9488- CC CF D2 BD C0 D8 C4 D2
9490- C1 D7 C0 C4 D2 C1 D7 C0
9498- CF CE C5 D2 D2 C7 CF D4
94A0- CF C0 AE

```

12. 自动行号编排

在DOS 33系统盘中有一个RENUMBER程序，它在行号的自动编排方面有较强的功能，是BASIC程序调试的常用工具。但在实际工作中我们发现这一软件有这样几个问题：

一是有将运算数误作行号进行调整的错误，当程序行符合一定的结构时这种错误就会发生。

例如有这样一程序：

```
100 A = A * 100
```

运行RENUMBER程序，调入这一程序行，并对之进行行号调整：& ✓

这时该程序行变为：

$10 A = A \times 10$

RENUMBER 误将运算数 100 当成行号调整为 10。在 BASIC 程序较长时，这一问题会使程序混乱而降低其可靠性。

二是当 BASIC 程序较长时 (但并不侵占 RENUMBER 存放空间), RENUMBER 往往会出现中断而达不到调整的目的, 甚至使内存 BASIC 程序混乱。

三是在使用 RENUMBER 之前, BASIC 程序必须存盘, 待运行 RENUMBER 之后再调入内存进行处理, 因而显得不大方便。

为解决上述问题, 我们对 BASIC 程序的内存存贮及各种转向语句进行了分析, 并使用 6502 机器语言编写了一段程序来实现行号调整的功能, 从而解决了 RENUMBER 程序中存在的问题。

前面我们曾介绍过 BASIC 程序在内存中是以一种链式结构按行号从小到大存放的。每一程序行的头两个字节存放链指针, 指向下一程序行的起始地址; 第三、四个字节存放本行行号, 按十六进制方式, 低位在前高位在后。每一程序行以 00 字节为结束标志。

另外, 转向语句如 GOTO、GOSUB、RUN、IF... THEN 等的后面也紧接行号, 如 GOTO 100。这个行号在内存中不是以十六进制方式存放, 而是用各字符的 ASCII 码减 \$ 80 为代码的。如语句 GOTO 1234 在内存中表示为:
AB 31 32 33 34

其中 AB 为 GOTO 的代码, 后面四个数分别是字符 1、2、3 和 4 的 ASCII 码减 \$ 80 后的数字。

因此, 行号调整包括两个方面: 一是每行行号的调整, 二

是转向行号的调整，见程序 PRO-58。

PRO-58

]CALL-151

*9000.92C3

| | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|
| 9000- | A9 | 90 | 85 | 74 | 8D | F7 | 03 | A9 |
| 9008- | 4C | 8D | F5 | 03 | A9 | 19 | 8D | F6 |
| 9010- | 03 | 60 | E6 | B8 | D0 | 02 | E6 | B9 |
| 9018- | 60 | A0 | 00 | B1 | B8 | C9 | 52 | F0 |
| 9020- | 05 | A2 | 10 | 4C | 12 | D4 | 20 | 12 |
| 9028- | 90 | B1 | B8 | D0 | 0F | A9 | 0A | 85 |
| 9030- | FC | 85 | FE | A9 | 00 | 85 | FD | 85 |
| 9038- | FF | 4C | 82 | 90 | C9 | 2C | D0 | 0E |
| 9040- | A9 | 0A | 85 | FC | A9 | 00 | 85 | FD |
| 9048- | 20 | 12 | 90 | 4C | 74 | 90 | 20 | 67 |
| 9050- | DD | 20 | 52 | E7 | A5 | 50 | 85 | FC |
| 9058- | A5 | 51 | 85 | FD | A0 | 00 | B1 | B8 |
| 9060- | D0 | 0B | A9 | 0A | 85 | FE | A9 | 00 |
| 9068- | 85 | FF | 4C | 82 | 90 | C9 | 2C | D0 |
| 9070- | B0 | 20 | 12 | 90 | 20 | 67 | DD | 20 |
| 9078- | 52 | E7 | A5 | 50 | 85 | FE | A5 | 51 |
| 9080- | 85 | FF | A0 | 01 | B1 | 67 | D0 | 01 |
| 9088- | 60 | A5 | 67 | 85 | B8 | A5 | 68 | 85 |
| 9090- | B9 | A0 | 00 | A5 | FC | 91 | B8 | A5 |
| 9098- | FD | 20 | 12 | 90 | 91 | B8 | A5 | FE |
| 90A0- | 18 | 65 | FC | 85 | FC | A5 | FF | 65 |
| 90A8- | FD | 85 | FD | A2 | 03 | 20 | 12 | 90 |
| 90B0- | CA | D0 | FA | B1 | B8 | F0 | 06 | 20 |
| 90B8- | 12 | 90 | 4C | B3 | 90 | 20 | 12 | 90 |
| 90C0- | A0 | 01 | B1 | B8 | F0 | 03 | 4C | 91 |
| 90C8- | 90 | A5 | 67 | 85 | B8 | A5 | 68 | 85 |
| 90D0- | B9 | A0 | 00 | B1 | B8 | D0 | 06 | 4C |
| 90D8- | 93 | 92 | EA | EA | EA | A2 | 04 | 20 |
| 90E0- | 12 | 90 | CA | D0 | FA | B1 | B8 | C9 |
| 90E8- | AC | D0 | 13 | 20 | 12 | 90 | B1 | B8 |
| 90F0- | C9 | 30 | B0 | 03 | 4C | 0A | 91 | C9 |
| 90F8- | 3A | B0 | F9 | 4C | 16 | 91 | C9 | C4 |
| 9100- | F0 | E9 | C9 | AB | F0 | E5 | C9 | B0 |
| 9108- | F0 | E1 | A2 | 01 | C9 | 00 | D0 | CF |

| | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|
| 9110- | 20 | 12 | 90 | 4C | D1 | 90 | A0 | 00 |
| 9118- | B1 | B8 | C9 | 30 | 90 | 07 | C9 | 3A |
| 9120- | B0 | 03 | C8 | D0 | F3 | 98 | 48 | 20 |
| 9128- | AC | 92 | EA | EA | EA | A5 | 67 | 85 |
| 9130- | A0 | A5 | 68 | 85 | A1 | A0 | 02 | B1 |
| 9138- | A0 | C5 | 50 | D0 | 0A | C8 | B1 | A0 |
| 9140- | C5 | 51 | D0 | 03 | 4C | 66 | 91 | A0 |
| 9148- | 03 | C8 | B1 | A0 | D0 | FB | C8 | B1 |
| 9150- | A0 | F0 | 0F | 84 | 00 | A5 | A0 | 18 |
| 9158- | 65 | 00 | 85 | A0 | 90 | 02 | E6 | A1 |
| 9160- | 4C | 35 | 91 | 4C | A0 | 92 | A0 | 00 |
| 9168- | B1 | A0 | 85 | 9F | C8 | B1 | A0 | 85 |
| 9170- | 9E | A2 | 90 | 38 | 20 | A0 | EB | 20 |
| 9178- | 34 | ED | A2 | 00 | BD | 00 | 01 | F0 |
| 9180- | 03 | E8 | D0 | F8 | 86 | A0 | 68 | C5 |
| 9188- | A0 | D0 | 03 | 4C | 05 | 92 | B0 | 65 |
| 9190- | 85 | A1 | A5 | A0 | 38 | E5 | A1 | 85 |
| 9198- | A5 | A5 | AF | 85 | A0 | A5 | B0 | 85 |
| 91A0- | A1 | A5 | AF | 18 | 65 | A5 | 85 | A2 |
| 91A8- | A5 | B0 | 69 | 00 | 85 | A3 | A0 | 00 |
| 91B0- | B1 | A0 | 91 | A2 | A5 | A0 | D0 | 09 |
| 91B8- | A9 | FF | 85 | A0 | C6 | A1 | 4C | C3 |
| 91C0- | 91 | C6 | A0 | A5 | A2 | D0 | 09 | A9 |
| 91C8- | FF | 85 | A2 | C6 | A3 | 4C | D2 | 91 |
| 91D0- | C6 | A2 | A5 | A1 | C5 | B9 | 90 | 0D |
| 91D8- | F0 | 03 | 4C | B0 | 91 | A5 | A0 | C5 |
| 91E0- | B8 | F0 | 02 | B0 | F5 | A5 | AF | 18 |
| 91E8- | 65 | A5 | 85 | AF | A5 | B0 | 69 | 00 |
| 91F0- | 85 | B0 | 4C | 05 | 92 | 38 | E5 | A0 |
| 91F8- | AA | A0 | 00 | A9 | 20 | 91 | B8 | 20 |
| 9200- | 12 | 90 | CA | D0 | F8 | A2 | 00 | A0 |
| 9208- | 00 | BD | 00 | 01 | F0 | 08 | 91 | B8 |
| 9210- | 20 | 12 | 90 | E8 | D0 | F3 | EA | 20 |
| 9218- | 12 | 90 | 4C | E5 | 90 | 68 | 4C | 17 |
| 9220- | 92 | A5 | 67 | 85 | B8 | 85 | A0 | A5 |
| 9228- | 68 | 85 | B9 | 85 | A1 | 20 | 6F | 92 |
| 9230- | EA | EA | EA | EA | EA | B1 | B8 | F0 |
| 9238- | 06 | 20 | 12 | 90 | 4C | 35 | 92 | 20 |
| 9240- | 12 | 90 | A5 | B8 | 91 | A0 | C8 | A5 |
| 9248- | B9 | 91 | A0 | A5 | B8 | 85 | A0 | A5 |
| 9250- | B9 | 85 | A1 | A0 | 01 | B1 | B8 | F0 |
| 9258- | 2F | 4C | 2D | 92 | A5 | AF | 85 | 69 |
| 9260- | 85 | 6B | 85 | 6D | A5 | B0 | 85 | 6A |


```

9268- 85 6C 85 6E 4C 03 E0 A0
9270- 00 B1 B8 A0 02 91 B8 A0
9278- 01 B1 B8 A0 03 91 B8 A0
9280- 04 20 12 90 88 D0 FA 60
9288- 88 B1 B8 F0 03 4C 2D 92
9290- 4C 5C 92 20 12 90 B1 B8
9298- D0 03 4C 21 92 4C DD 90
92A0- C8 B1 A0 D0 03 4C 1D 92
92A8- 88 4C 53 91 A5 B8 85 00
92B0- A5 B9 85 01 20 67 DD 20
92B8- 52 E7 A5 00 85 B8 A5 01
92C0- 85 B9 60 00

```

按其不同的作用分为三小块:

\$ 9000—\$ 9088: 设置&指令入口; 分析输入指令, 并从中取得起始行号和增量两个参数; 判断内存中是否有 BASIC 程序, 如无则返回。

\$ 9089—\$ 9220: 按起始行号和增量调整各程序行行号, 并改变各转向行号的指向。

\$ 9221—\$ 92C2: 调整链指针和有关的程序参数指针; 返回 BASIC 状态。

使用时先运行一次 (不破坏内存 BASIC 程序):

BRUN RENUM↵

之后, 就可随时使用下列指令对当前内存中的 BASIC 程序进行行号调整:

&R<起始行号><, 增量>

其中起始行号和增量两个参数可省略任一个或全部省略。省略时约定值为10。如

&R↵: 起始行号为10, 增量为10;

&R, 20: 起始行号为10, 增量为20。

四、信息处理技巧

随着计算机的日益发展和广泛使用,各种信息处理应运而生。人们不仅要求计算机从事数值计算,而且也能对非数值量进行加工和处理。因此,自动控制、企业管理、情报检索、过程控制及办公室自动化等各个方面,日益引起人们的兴趣和关注。

本章主要介绍信息处理方面的入门知识和技巧,包括数据存贮、数据交换、数据删除、数据插入、数据修改、数据合并、数据分类、数据检索、堆栈技术以及链接表技术等内容。

这类问题名目繁多,方法不少,结构复杂,构思巧妙。因此,学习和掌握这些技术,将有助于进一步提高编制程序的能力和技巧。

1. 数据存贮

引入下标变量和数组,是 BASIC 语言为描述有序变量提供的有力工具。在处理不少问题上,它可以简化程序设计,而对大量数据的存贮、比较、运算、检索、输出的程序,其优点尤为突出。

对数据存贮,常用以下形式,见程序 PRO-59:

```
PRO-59  
ULIST
```

```
5 REM PRO-59
```

```

10 DIM A(5),B$(5)
20 FOR I = 1 TO 5
30 READ A(I),B$(I)
40 PRINT A(I),B$(I)
50 NEXT I
60 DATA 1,BOOK,2,INK,3,PEN,4,WATCH,
    5,MAP

```

```

URUN
1          BOOK
2          INK
3          PEN
4          WATCH
5          MAP

```

这段程序共有开辟内存，读取记录，存贮信息，打印输出四个功能。

当上述程序运行一次后，任何时刻从键盘敲入PRINT A(N)，或PRINT B\$(N)，(N=1,2,...,5)，都可以得到正确存贮内容。因此，本章中各类问题的存贮，基本上都采用上述形式。

关于 DATA 区中字符串的书写，不同机器是有差别的。对 PC-1500 机，字符串必须加引号“”，这和基本 BASIC 规定一致，但对 APPLE-II 及其兼容机如中华学习机，则引号可以省略。

2. 数据交换

数据交换，实际上是变量取值对调。只有正确地进行数据交换，才能保证我们将要介绍的有关数据处理的各项工作顺利进行。

例如，希望变量 A（假定其值为 3）与变量 B（假定其值为 4）对调，有人这样编写程序 PRO-60。

PRO-60

ULIST

```
5  REM PRO-60
10 A = 3
20 B = 4
30 A = B
40 B = A
50 PRINT A,B
```

URUN

4

4

结果说明该程序并没有实现数据交换，这是因为计算机不能和人一样，将两个盒子里的东西同时交换。

这里有一个冲掉内存的概念，执行上述程序30句时， $A = 4$ ，因而原来存贮在变量A里的值3被冲掉了，执行40句时， $B = 4$ ，实际上应理解原来放在B中的4被冲掉了，又换进了新的4。所以，两个数据的交换不能用两个赋值 $A = B$ 和 $B = A$ 来达到目的。

数据交换常用的方法是设置中间变量，或者叫安排辅助存贮单元。对上述问题处理可以用程序PRO-61实现。

PRO-61

ULIST

```
5  REM PRO-61
10 A = 3
20 B = 4
30 T = A
40 A = B
50 B = T
60 PRINT A,B
```

URUN

4

3

这里的T就是一个中间变量，它是一个起暂存作用的辅助存储单元。

当然，也不一定非用第三个变量，例如我们仍用A、B两个变量，同样可以实现两个数据交换，请看以下实例：

令 $A = A + B$, $B = A - B$, $A = A - B$

故有程序PRO-62。

PRO-62

ULIST

```
5  REM  PRO-62
10  A = 3
20  B = 4
30  A = A + B
40  B = A - B
50  A = A - B
60  PRINT A, B
```

URUN

4

3

3. 数据删除

删除数据（数值或字符）的概念是不难理解的。如删除下列一组数据中的一个记录66

98 97 66 53 42

则首先将66删去，然后让53填入66的位置，最后将42再填补到原来53的地方。

换句话说，上述思想是将后面的数据（或信息）向前移，从而删除了原有的信息。

但是，数据删除必须考虑：

(1) 除要删除的数据以外，所有原来在被删除信息后面

的全体数据，应该依次向前顺移；

(2) 删除一个数据后，原数据组的最后一个记录也应删去。

总之，删除数据（或信息）的思路，可以概括为一个语句：即

$$D(I) = D(I + 1)$$

现给出满足上述设计思想的程序 PRO - 63 读者不难明了程序的构思。

PRO-63

```
10  REM  DELETE DATA PROGRAM
20  DIM D(11)
30  FOR I = 1 TO 10
40  READ D(I)
50  PRINT D(I); " ";
60  NEXT I
70  PRINT
80  K = 0
90  INPUT "N="; N: PRINT "N="; N
100 IF N = - 1 THEN END
110 FOR I = N TO 9
120 D(I) = D(I + 1)
130 NEXT I
140 K = K + 1
150 PRINT
160 FOR I = 1 TO 10 - K
170 PRINT D(I); " ";
180 NEXT I
190 PRINT
200 GOTO 90
210 DATA 12,47,35,26,61,83,90,55
    ,27,88
```

URUN

12 47 35 26 61 83 90 55 27 88

N=10

N=10

12 47 35 26 61 83 90 55 27

N=5

N=5

12 47 35 26 83 90 55 27

N=-1

N=-1

上述程序以10个数据为例，20—60句是读存信息，安排50句的目的是跟踪程序的执行过程，它打印原数据组的全部记录。

90—130句，是本程序的核心程序段，N指输入要删除的数据中那一个位置的信息，120句是实现删除（实际上向前移）的关键语句。为了实现后面的数据顺次向前移位，所以安排了110—130句的一个循环。

巧妙的是K计数指针的安排，删去一个记录，就计数一次。目的是删除一个信息后，原数据集合的最后一个记录必须舍去。

160—180句的循环，是打印出删除某些记录后剩下的数据情况，请注意循环终值是10—K。

为了检查本程序的正确性，同时也是为了继续删除其它信息的需要，特别安排了200句。若要结束程序，只要键入-1即可。

程序的正确性，请读者自行上机检验。

本程序同样适用字符串删除。但对某些变量、语句要作适当更改。例如对货物的删除：

令D\$(I)—I组货物名称，如INK

A\$(I)—I组货物代号，如A—7

B(I)—I组货物数量，如340

C(I)—I组货物价格，如0.47

程序设计的主要思想，用下列语句来实现：

$D(I) = D(I + 1)$

$A(I) = A(I + 1)$

$B(I) = B(I + 1)$

$C(I) = C(I + 1)$

这一工作留给有兴趣的读者完成。

下面给出另外一个删除信息的程序 (E - 3), 主要设计思想和上面介绍的程序十分相近, 框图如图 3 所示。

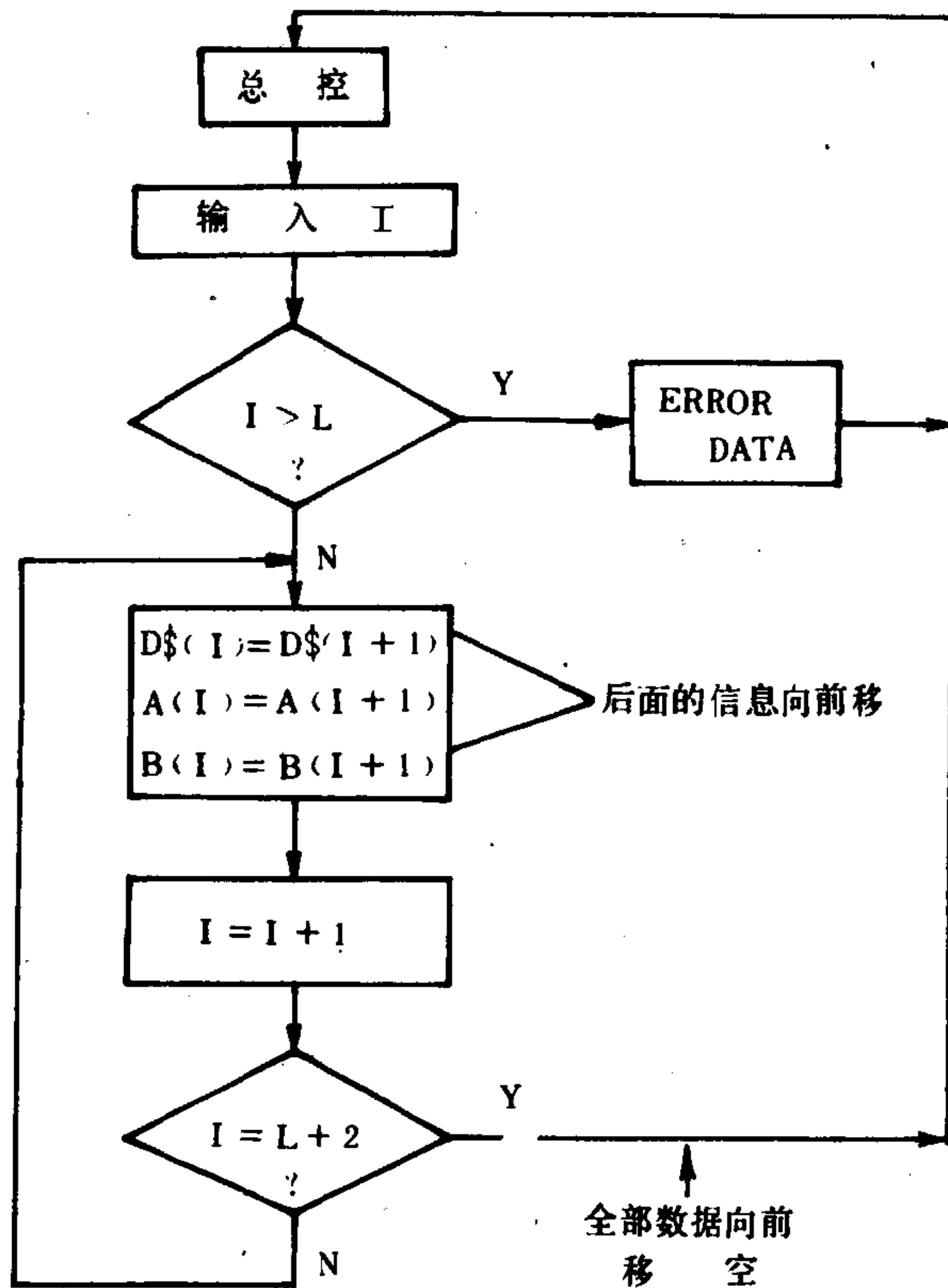


图 3

程序清单:

```

)LIST
5  REM (E-3)
10  DIM D$(20),A(20),B(20)
20  FOR I = 1 TO 10: READ D$(I),A(I),B(I): NEXT I
23  PRINT
25  L = 10
30  INPUT "N=";N: ON N GOTO 40,70
35  END
40  INPUT "I=";I: IF I < 1 OR I > L THEN 60
50  PRINT D$(I),A(I),B(I): PRINT : GOTO 30
60  PRINT "ERROR DATA": GOTO 30
70  INPUT "I=";I: IF I > L THEN 60
80  D$(I) = D$(I + 1):A(I) = A(I + 1):B(I) = B(I + 1)
85  I = I + 1
90  IF I = L + 2 THEN 30: GOTO 80
95  FOR J = 1 TO 10
100  IF J = I THEN 120
110  PRINT D$(J),A(J),B(J)
120  NEXT J
220 DATA "A",40,101,"B",57,102,"C",70,103,"D",99,104
230 DATA "E",55,105,"F",67,106,"G",22,107,"H",11,108
240 DATA "I",33,109,"J",88,110

```

程序简要说明:

I —— 为要删去的第几组值, L —— 为总共多少组,
D\$ (I) —— I 组货物名称, A (I) —— I 组货物数量,
B (I) —— I 组货物代号。

10—23 句: 读入并存贮信息。

30 句为总控, N = 1, 打印查找的某组信息; N = 2, 删除某组信息。

40句为查找，仅为一句，方法巧妙。

70—90句为删除信息的程序段。

95—120句检查是否删除某组信息。

运行实例：

```

]RUN
N=1
I=5
E          55          105

N=2
I=7
A          40          101
B          57          102
C          70          103
D          99          104
E          55          105
F          67          106
H          11          108
I          33          109
J          88          110

```

如果删除的是 $I + 1$ 组的信息（在数据结构的链接技术中，称结点 I 的后继结点〈地址为 I 〉）则用下述 E—4 程序，原理及程序基本上和 E—3 程序相同，但指针安排上应有所调整，请读者自己分析。

```

]LIST
5 REM (E-4)
10 DIM D$(20),A(20),B(20)
20 FOR I = 1 TO 10: READ D$(I),A(I),B(I): NEXT I

```

```

23 PRINT
25 L = 10
30 INPUT "N=";N: ON N GOTO 40,70
35 END
40 INPUT "I=";I: IF I < 1 OR I > L THEN 60
50 PRINT D$(I),A(I),B(I): PRINT : GOTO 30
60 PRINT "ERROR DATA": GOTO 30
70 INPUT "I=";I: IF I > L THEN 60
72 J = I
75 J = J + 1
80 D$(J) = D$(J + 1):A(J) = A(J + 1):B(J) = B(J + 1)
85 I = I + 1
90 IF J = L + 1 THEN 30: GOTO 72
95 FOR J = 1 TO 10
100 IF J = I + 1 THEN 120
110 PRINT D$(J),A(J),B(J)
120 NEXT J
220 DATA "A",40,101,"B",57,102,"C",70,103,"D",99,104
230 DATA "E",55,105,"F",67,106,"G",22,107,"H",11,108
240 DATA "I",33,109,"J",88,110

```

IRUN

N=1

I=5

| | | |
|---|----|-----|
| E | 55 | 105 |
|---|----|-----|

N=2

I=7

| | | |
|---|----|-----|
| A | 40 | 101 |
| B | 57 | 102 |
| C | 70 | 103 |
| D | 99 | 104 |

| | | |
|---|----|-----|
| E | 55 | 105 |
| F | 67 | 106 |
| G | 22 | 107 |
| I | 33 | 109 |
| J | 88 | 110 |

请注意上述运行结果， $N = 1$ 时，进入检索模块，此时输入5，是指查找第五组的信息； $N = 2$ 时，进入删除模块，此时输入7，删除的不是第7组信息，而是第8组信息。这是本程序和E-3程序的主要差别。

读者不难分析，不论是E-3还是E-4程序，结构比较复杂，这是因为安排了指针的关系。从结构、原理以及程序的易读性来看，显然开始提出的第一个程序较好。

4. 数据插入

插入一个信息的方法也是比较简单的。例如，有下列数组：

33 44 55 61 29

现在，要在55和61之间插入99这一个数。方法是首先设法把55和61之间空出一个空位来，那么数据99就有插入进去的位置，为此，必须把55后面的两个数相应地向后移一个存储单元。

但是这种移动，又必须是最后面一个数据先向右移，然后倒数第二个数据再向右移，这样，就自然地空出了55以后的位置，并保留了55以后的其它数据。

反之，如果55以后的第一个数据先向右移动一个存储单元，则最后一个数据（本例是29）将被冲掉，而新的存储单

元存放的值将不正确。

上述设想，用 BASIC 语句来处理，简单归结为：
 $D(I+1)=D(I)$

这个语句是将第 I 个存贮单元之值，移动到第 $I+1$ 个存贮空间去。

为了保存其它数据，并依次向右移，则用循环语句来控制将是十分方便的。

至此，整个程序的设计思想，已经叙述清楚。

由于本程序是插入新的数据，或者说加入新的信息，这样，原有数组开辟的内存就不够用了。故 DIM 说明语句可用 $DIM D(20)$ 来解决（设原有 10 个数，增加到最多 20 个数）。

现在，可以仿照删除数据程序的模式，书写程序 PRO-64。

PRO-64

```
10  REM  ENTER DATA PROGRAM
20  DIM D(20)
30  FOR I = 1 TO 10
40  READ D(I)
50  PRINT D(I); " ";
60  NEXT I
70  PRINT
80  K = 0
90  INPUT "N="; N: PRINT "N="; N
100 IF N = - 1 THEN 240
110 FOR I = N TO 10
120 D(I + 1) = D(I)
130 NEXT I
140 K = K + 1
150 INPUT "A="; A: PRINT "A="; A
160 D(N) = A
170 PRINT
```

```

180 FOR I = 1 TO 10 + K
190 PRINT D(I); " ";
200 NEXT I
210 PRINT
220 GOTO 90
230 DATA 12, 47, 35, 26, 61, 83, 90, 55
    , 27, 88
240 PRINT "END": END

```

URUN

12 47 35 26 61 83 90 55 27 88

$N = 4$ (在第四个位置插入新的数据)

$A = 3333$ (在第四个位置插入3333)

12 47 35 3333 26 26 26 26 26 26 26

结果表明3333是插入到第四个位置了, 但后面的数据出现了问题, 7个26中只有最左边一个26是正确的, 其余均错, 看来程序有问题。

先停机检查:

$N = -1$

END

分析原程序, 不难发现问题出在110句。这是因为110—130句的循环是从左 (即第 N 个位置开始) 向右移位的, 所以出现7个26的情况。它显然不符合我们前面分析关于数据移动, 应从数据尾 (最后一个) 向右移动的约定。

故改动110句为:

110 FOR I = 10 TO N STEP - 1

再次运行程序:

12 47 35 26 61 83 90 55 27 88

N = 4

A = 3333

12 47 35 3333 26 61 83 90 55 27 88 (好!

这一次对了)

N = 8

A = 2222

12 47 35 3333 26 61 83 2222 90 55 27 0

又出现错误, 2222是插进去了, 最后一个数据88怎么变成 0?

再次停机检查。

N = - 1

END

分析: 开始插入3333对, 因为第一次插入加了一个数据变成11个, 所以执行 110 句以及 180 句、190 句都不会有问题。而当再次插入第二个数2222时, 应变成12个数, 可是 110 句的初值仍为 10, 执行 120 句: $D(I+1)=D(10+1)=D(11)$, 因此, $D(12)$ 中没有数据进入, 故执行180句、190句、200句时, $D(12)$ 中的值为 0。

问题找到了, 只要改动110句:

110 FOR I=10+K TO N STEP - 1

现在, 可以满怀信心地运行改动后的程序:

RUN ↙

12 47 35 26 61 83 90 55 27 88

N = 4

A = 3333

12 47 35 3333 26 61 83 90 55 27 88

N = 8

A = 2222

12 47 35 3333 26 61 83 2222 90 55 27 88

好! 成功了。

N = 12

A = 7777

12 47 53 3333 26 61 83 2222 90 55 27
7777 88

妙极了! 完全成功!

N = - 1

END

完整的程序, 见PRO-65。

PRO-65

```
10  REM  ENTER DATA PROGRAM
20  DIM D(20)
30  FOR I = 1 TO 10
40  READ D(I)
50  PRINT D(I); " ";
60  NEXT I
70  PRINT
80  K = 0
90  INPUT "N="; N: PRINT "N="; N
100 IF N = - 1 THEN 240
110 FOR I = 10 + K TO N STEP -
    1
120 D(I + 1) = D(I)
130 NEXT I
140 K = K + 1
150 INPUT "A="; A: PRINT "A="; A
160 D(N) = A
170 PRINT
180 FOR I = 1 TO 10 + K
190 PRINT D(I); " ";
200 NEXT I
```

```

210 PRINT
220 GOTO 90
230 DATA 12,47,35,26,61,83,90,55
    ,27,88
240 PRINT "END": END

```

运行实例为:

```

URUN
12 47 35 26 61 83 90 55 27 88
N=4
N=4
A=3333
A=3333

12 47 35 3333 26 61 83 90 55 27 88
N=8
N=8
A=2222
A=2222

12 47 35 3333 26 61 83 2222 90 55 27 88
N=12
N=12
A=7777
A=7777

12 47 35 3333 26 61 83 2222 90 55 27 7777 88
N=-1
N=-1
END

```

最后说明一下，N是代表插入数据的位置，或叫地址。变量A代表插入的数值，它应该放入N地址的存贮单元D(N)中。K为计数器的指针，每加入一个数据，指针K自动加1。请注意体会110句循环初值及180句循环终值的安排。

同样，本程序也适用包括字符串信息的插入处理，而对某些数据、变量、数组以及个别语句都应作相应的修改。

下面再介绍一种数据插入的程序 E-5, 主要设计思想和上面相近, 但程序结构安排上有新的特点, 即安排了几个指针。

为叙述方便, 我们以 10 种货物 (其实不一定是货物情况, 任何资料文件、人事档案等都可以) 为例。货物名称用 A, B, ..., J 表示存入 D(I)$ 中, $I = 1, 2, \dots, 10$, 货物数量存入 $A(I)$ 中, 货物代号存入 $B(I)$ 中。

设指针 L, 初值为 10, 指向最后一组信息。M 也是指针, 定为 20, 作为 10 组信息全部移出的存贮空间的指针。

数入插入的数据 N, N_1 , N_2 , 为相应于原信息的新数据。I 也是插入的数据, 但它作为地址来考虑的。输入 I 值, 是指插入 I 这一结点的后继结点 (即 $I+1$ 的地址)。$

L 初值为 10, 当 $L=0$ 时, 说明插入信息的工作完成。由此可知, L 指针的移动是从信息的最后一组, 逐渐移至最前一组的。

将 L 值赋给 J, J 作为信息数组的下标, 当 $J=10$ 时, D(J+1) = D(J) 的意思是第 10 组的某一信息移至第 11 组的存贮单元中去。

当 $J=I$ 时, 说明找到了需要插入的地址, 否则再循环过去, 把信息向后移。框图见图 4。

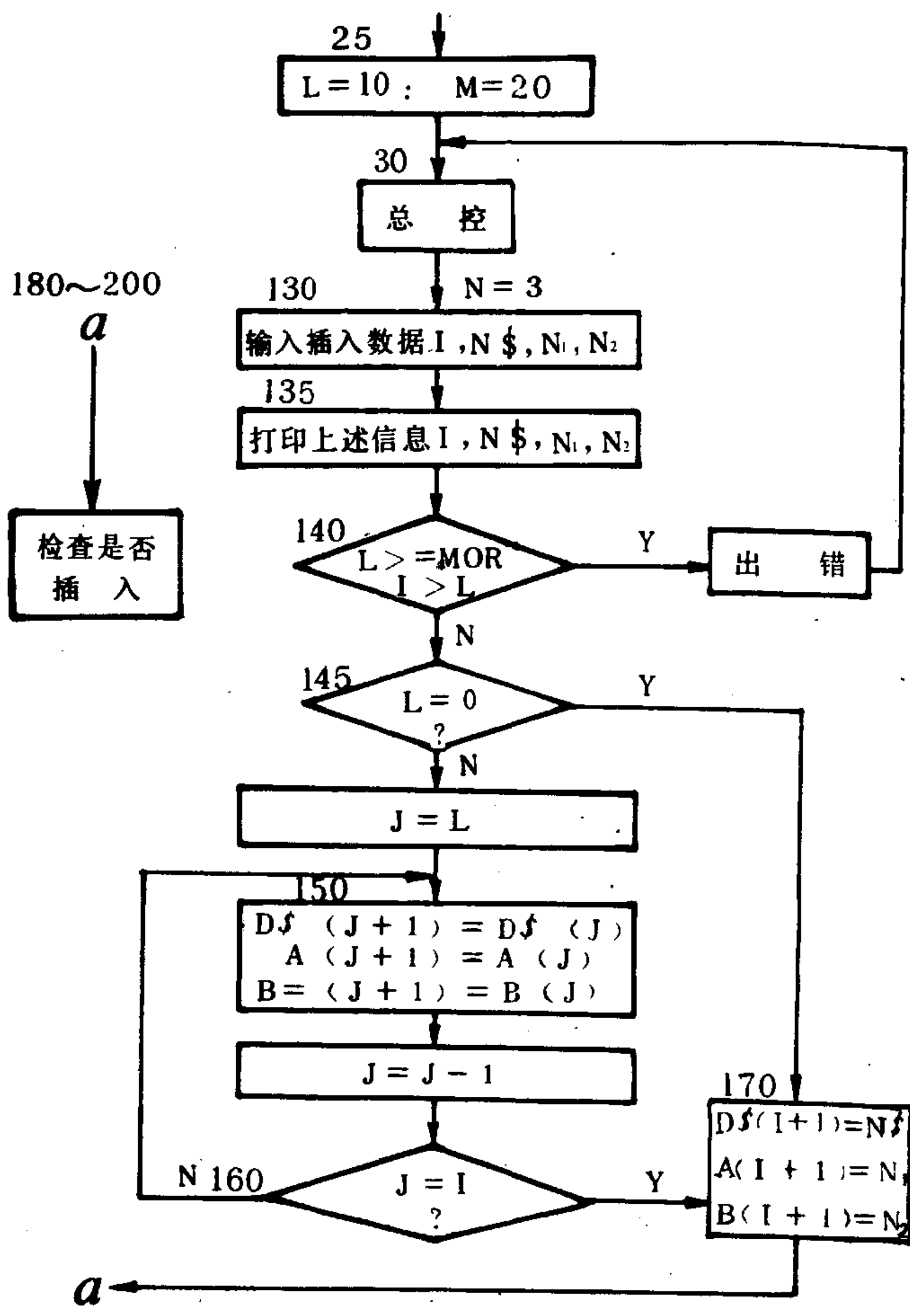


图 4

程序清单及运行实例:

LIST

```
5 REM (E-5)
10 DIM D$(20),A(20),B(20)
20 FOR I = 1 TO 10: READ D$(I),A(I),B(I): NEXT I
23 PRINT
25 L = 10:M = 20
30 INPUT "N=";N: ON N GOTO 40,70,130
35 END
40 INPUT "I=";I: IF I < 1 OR I > L THEN 60
50 PRINT D$(I),A(I),B(I): PRINT : GOTO 30
60 PRINT "ERROR DATA": GOTO 30
70 INPUT "I=";I: IF I > = L THEN 60
72 J = I
75 J = J + 1
80 D$(J) = D$(J + 1):A(J) = A(J + 1):B(J) = B(J + 1)
90 IF J = L + 1 THEN 30: GOTO 72
95 FOR J = 1 TO 10
100 IF J = I + 1 THEN 120
110 PRINT D$(J),A(J),B(J)
120 NEXT J
125 PRINT
130 INPUT "I,N$,N1,N2=";I,N$,N1,N2
135 PRINT I,N$,N1,N2
140 IF L > = M OR I > L THEN 60
145 IF L = 0 THEN 170:J = L
150 D$(J + 1) = D$(J):A(J + 1) = A(J):B(J + 1) = B(J):J = J - 1
160 IF J = I THEN 170: GOTO 150
170 D$(I + 1) = N$:A(I + 1) = N1:B(I + 1) = N2
180 FOR I = 1 TO 10
185 IF I = J + 1 THEN 200
190 PRINT D$(I),A(I),B(I)
```

```

200 NEXT I
220 DATA "A",40,101,"B",57,102,"C",70,103,"D",99,104
230 DATA "E",55,105,"F",67,106,"G",22,107,"H",11,108
240 DATA "I",33,109,"J",88,110

```

```

3RUN

```

```

N=1
I=7
G          22          107

```

```

N=2
I=5
A          40          101
B          57          102
C          70          103
D          99          104
E          55          105
G          22          107
H          11          108
I          33          109
J          88          110

```

```

I,N$,N1,N2:5,F,99,106
5          F          99          106
A          40          101
B          57          102
C          70          103
D          99          104
E          55          105
F          99          106
G          22          107
H          11          108
I          33          109
J          88          110

```

注：20句为存贮原数组信息。70—120句为删去后继结点的程序段。180—200句为检查是否插入了信息。

5. 数据修改

数据修改就是更改数据（或者字符）。这个问题相对于信息的插入、删除要简单的多。而利用下标变量或者字符串变量更为方便。例如，要更改字符串，只要设法指出存贮在那一个单元的信息要改，然后重新赋给修改的内容即可。更简单地说是这样的：

若 D(S)$ 中存有 PPI，今要改成 RHI，只要键入 $N = 5$ 和 N = RHI$ ，那么，

D(N) = N$$

即可完成信息更改的任务。

下面，以更改10个国家的名称为例，具体看一下程序是如何实现的。

10个国家为：

CHINA, JAPAN, KOREA, FRANCE,
INDIA, CANADA, AMERTCA, TAPAN,
ENGLAND, SOUIET UNION.

把它们放在DATA区中，见程序PRO-66。

PRO-66

```
10 REM  RENAME STRING PROGRAM
20 DIM D$(10)
30 FOR I = 1 TO 10
40 READ D$(I)
50 PRINT D$(I); " ";
60 NEXT I
70 PRINT
80 FOR I = 1 TO 10
```



```

90  INPUT "N=";N: PRINT "N=";N
100 IF N = - 1 THEN 140
110 INPUT "N$=";N$: PRINT "N$=";
    N$
120 D$(N) = N$
130 NEXT I
140 PRINT
150 FOR I = 1 TO 10
160 PRINT D$(I); " ";
170 NEXT I
180 DATA CHINA, JAPAN, KOREA, FRA
    NCE, INDIA, CANADA
190 DATA AMERTCA, ENGLAND, TAPAN, S
    OUIET UNION
200 PRINT : PRINT "END": END

```

```

URUN
CHINA JAPAN KOREA FRANCE INDIA CANADA
AMERTCA ENGLAND TAPAN SOUIET UNION

```

现在，来看修改方法。

N = 3

(准备修改第三组记录)

N \$ =ITAIY

(即把朝鲜改为意大利)

N = 5

(准备修改第五组印度的国名)

N \$ =SRI LANKA

(换上斯里兰卡)

N = 8

(修改第八组记录)

N \$ =PANAMA

(用巴拿马代替英格兰)

N = -1

(修改完成)

CHINA JAPAN ITALY FRANCE
SRI LANKA CANADA AMERTCA
PANAMA TAPAN SOUJET UNION

(修改后的正确记录)

END

(结束)

以上程序虽稍长一点，但很好懂，读者可以自行分析。如有困难，可按程序及运行的部份结果，上机亲自试一下，你会得到满意的结果。

类似其它问题的处理，仍可以本程序为模式，而只要改动DATA区中的信息。

如果修改的内容是N组，每组有K个记录，今要修改任意一组的任一记录，程序又是如何编制呢？请读者试编一个程序，看看能否得到满意的结果。然后，再参阅程序“数据修改一”。

LIST

```
800 REM 数据修改一
810 DIM D$(4,3)
820 FOR I = 1 TO 4
825 FOR J = 1 TO 3
830 READ D$(I,J)
840 PRINT D$(I,J); " ";
850 NEXT J
```

```

855 PRINT
860 NEXT I
862 PRINT
865 REM 870-920连续修改
870 FOR I = 1 TO 4
875 FOR J = 1 TO 3
880 INPUT "N,M=";N,M: REM 修改的位置
890 IF N = - 1 AND M = - 1 THEN 930: REM 修改结束, 显示, 返回
900 INPUT "N$=";N$: REM 修改的数据
910 D$(N,M) = N$
915 NEXT J
920 NEXT I
930 PRINT
940 FOR I = 1 TO 4
950 FOR J = 1 TO 3
955 PRINT D$(I,J);" ";
960 NEXT J
962 PRINT
964 NEXT I
966 PRINT
970 DATA CHINA,111,222,KOREA,333,444,JAPAN,555,666,INDIA,777,888
990 END

```

```

JRUN
CHINA 111 222
KOREA 333 444
JAPAN 555 666
INDIA 777 888

```

```

N,M=3,2
N$=999
N,M=-1,-1

```

```

CHINA 111 222
KOREA 333 444
JAPAN 999 666
INDIA 777 888

```

```
IRUN  
CHINA 111 222  
KOREA 333 444  
JAPAN 555 666  
INDIA 777 888
```

```
N,M=3,1
```

```
N$=INDIA
```

```
N,M=3,3
```

```
N$=555
```

```
N,M=4,1
```

```
N$=JAPAN
```

```
N,M=4,3
```

```
N$=666
```

```
N,M=-1,-1
```

```
CHINA 111 222
```

```
KOREA 333 444
```

```
INDIA 555 555
```

```
JAPAN 777 666
```

6. 数据合并

设有两组数据已经顺序排好，要求把这两组数据合并在一起，并按从大到小的顺序重新存贮。

对于两组数据，如果不要按顺序大小合并，问题比较简单。而本例中要求合并好的两组数据有序排列，在程序上就应仔细考虑。对这类问题的处理方法较多，今介绍一种比较通俗易懂的方法。

编制程序的思路。

为叙述方便，暂定一组数为 8 个，另一组数为 10 个。

用数组 A (9)、B (11) 作为这两组数的存贮单元，每

个数组的最后一个存储单元作为附加单元备用。

用数组 $X(19)$ 作为合并后全部数据记录的存放单元。

设置三个指针: I 、 J 、 N , 分别表示指向数组 $A(9)$ 、 $B(11)$ 、 $X(19)$ 的位置。例如, 开始时令 $I=1$, $J=1$, $N=0$, 这就表示两组数据尚未合并时, I 指向 A 数组的首地址的数据 $A(1)$, J 指向 B 数组的首地址的数据 $B(1)$, 而 $N=0$, 则表示指向 X 数组的地址下标为零的位置, 此时 $X(0)=0$, 设有数据存入。

以上就是关于变量设置和指针安排。

程序设计的主要思路:

开始, 每个数组中各自最大的一个值进行比较 (原来它们各自都已按从大到小排好), 那一个值大, 就把最大值存入 X 变量的数组, 并把原来最大值的指针加 1。

例如, 若 $A(I) > B(J)$, 则大数 $A(I)$ 的值存入 $X(N)$ 中, 并使指针 $I = I + 1$; 反之, 则表示 $A(I) < B(J)$, 大数是 $B(J)$ 里的值, 大数 $B(J)$ 的值送入 $X(N)$ 中, 同时使指针 $J = J + 1$ 。

如此不断循环, 从而实现两组数据的合并, 同时, 合并后的数据正好是顺序排列的。

为判断合并是否完成, 可以安排 N 指针作计数次数处理, $N=18$, 表示合并完成。

现在, 可以按照上述思路, 来编写程序, 见 PRO-67。

PRO-67

ULIST

```
5  REM PRO-67
10 REM READ DATA PROGRAM
20 DIM A(9), B(11), X(19)
30 FOR I = 1 TO 8: READ A(I): NEXT
```

```

40  FOR I = 1 TO 10: READ B(I): NEXT

50  REM  MEGRE DATA PROGRAM
60  I = 1: J = 1: N = 0
70  N = N + 1
80  IF A(I) > B(J) THEN LET X(N) =
      A(I): I = I + 1: GOTO 100
90  X(N) = B(J): J = J + 1
100 IF N < 18 THEN 70
110 FOR I = 1 TO 18
120 PRINT X(I); ", ";
130 NEXT
140 PRINT
150 DATA 98, 97, 89, 85, 84, 71, 69, 63
160 DATA 96, 94, 92, 90, 87, 83, 72, 69,
      60, 55

RUN
98, 97, 96, 94, 92, 90, 89, 87, 85, 84, 83, 72, 71, 69, 69, 63,
60, 55,

```

说明：20句定义数组，比原数据多开一个内存单元，否则程序会发生下标越界的错误。30句，40句分别读存原来两组数据(合并前)。60—100句完成合并数据工作。110—130句打印输出合并完成的数据。注意80句扩展BASIC语句的用法，THEN语句后面如果是赋值语句，则一定要加LET (APPLE-Ⅱ，紫金-Ⅱ均是如此)，而在PC-1500机上运行，则可以采用两种形式之一：…… THEN LET……；或者……LET……，而省去THEN语句。

对于有N组数据的合并情况，原则上照此办理。但每一组数据均应设置一个指针。下面给出三组数据合并的程序及结果，见PRO-68。

PRO-68

ULIST

```
5  REM PRO-68
10  REM  READ DATA PROGRAM
20  DIM A(9),B(11),C(12),X(29)
30  FOR I = 1 TO 8: READ A(I): NEXT

40  FOR I = 1 TO 10: READ B(I): NEXT

50  FOR I = 1 TO 11: READ C(I): NEXT

60  REM  MEGRE DATA PROGRAM
70  I = 1:J = 1:K = 1:N = 0
80  N = N + 1
90  IF A(I) > B(J) AND A(I) > C(K)
    THEN X(N) = A(I):I = I + 1: GOTO
    120
100  IF B(J) > C(K) THEN LET X(N)
    = B(J):J = J + 1: GOTO 120
110  X(N) = C(K):K = K + 1
120  IF N < 29 THEN 80
130  FOR I = 1 TO 29
140  PRINT X(I);", ";
150  NEXT
160  PRINT
170  DATA 97,85,83,79,78,71,69,65

180  DATA 94,83,82,77,75,68,64,59
    ,58,57
190  DATA 92,82,79,76,68,66,62,6
    0,58,51,40
```

URUN

```
97,94,92,85,83,83,82,82,79,79,78,77,76,75,71,
69,68,68,66,65,64,62,60,59,58,58,57,51,40,
```

还有一种方法，就是设置特征变量。

如有两组数据已经顺序排好，要求把这两组数据合并并在

一起，并按从大到小的顺序存贮。

为叙述方便，暂定一组数为10个。另一组为9个。合并好的数据存入C (19) 数组中。

用数组A (11)、B (10) 存贮这两组数，每个数组的最后一个单元作为附加单元。

设置三个指针：I、J、K，它们分别表示A (11)、B (10) 和C (19) 的指针。开始时 $I = 1$ 、 $J = 1$ 、 $K = 0$ ，表示未合并时I指向A (11) 的首数据A (1)，J指向B (10) 的首数据B (1)，而 $K = 0$ 没有开始时，还未合并数据。

设一个辅助存贮单元M，开始时让A (1) 的数值存入M中，即 $M = A (1)$ 。

再设一个特征变量P (特征即标志)。开始时把P置于1，即 $P = 1$ ，例如定义 $A (I) < B (J)$ ， $P = 2$ ， $A (I) > B (J)$ ， $P = 1$ 。

K要自动计数，即 $K = K + 1$ ，相当于一个计数器，表示是否合并完成。例如19个数据全部合并好，则 $K = 19$ ，打印输出并结束， $K < 19$ 则再做合并的工作。

每次总是把大数并入C (19) 的数组中，同时大数对应的指针也要加1，如P标志为1，表示： $A (I) > B (J)$ ，则让大数A (I) 存入C (K) 中，并使指针 $I = I + 1$ ；如P标志为2，表示： $A (I) < B (J)$ ，则让大数B (J) 送入C (K) 中，并使指针 $J = J + 1$ 。

至于附加单元，有这样几点说明：

I指向附加单元，则I指针不再变化。A (I) 中的数据均存入C (K) 中了；

J指向附加单元，则J指针也不再变化。B (J) 中的数据也均存入C (K) 中了；

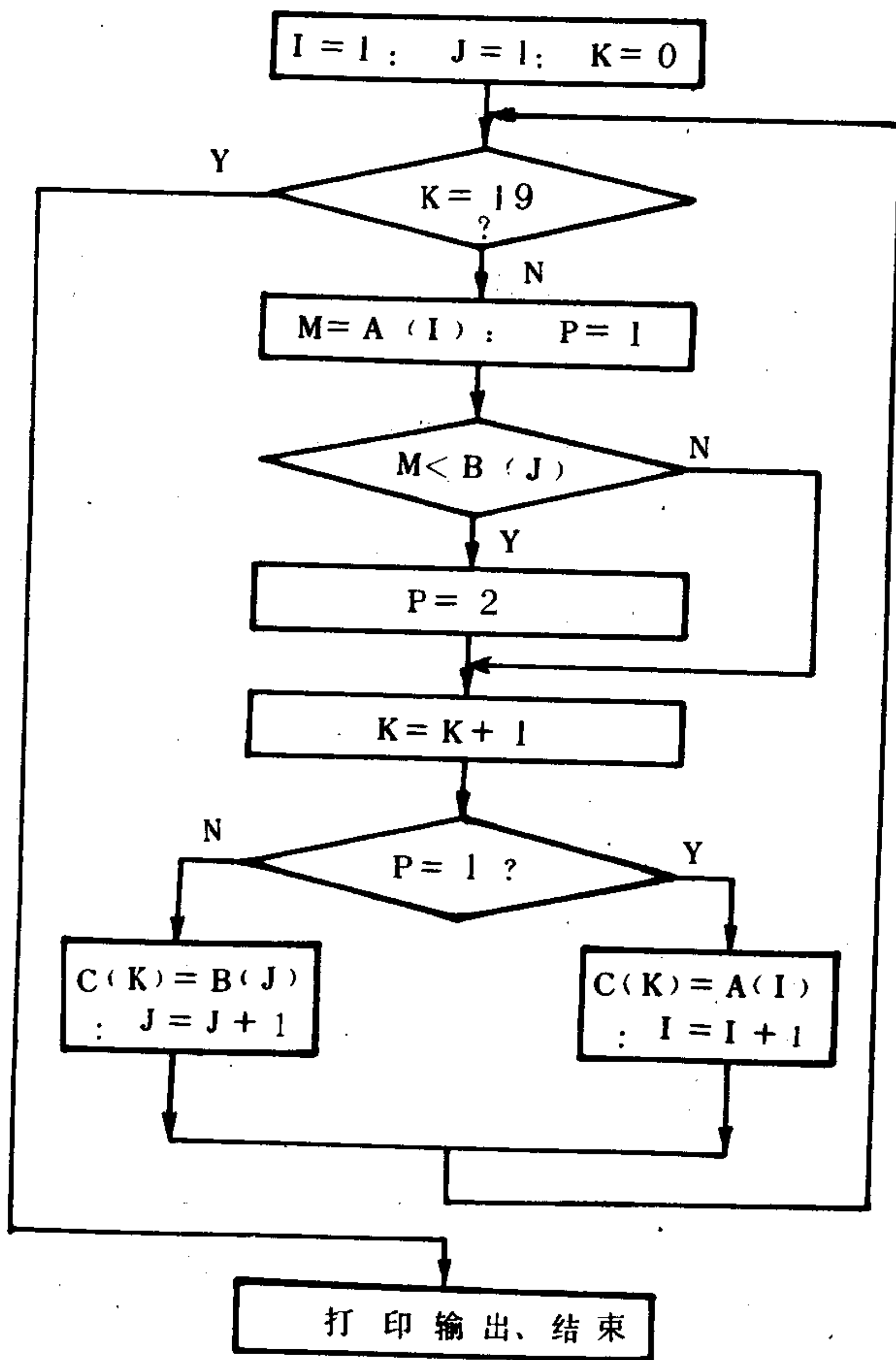


图 5

所以当I、J都指向各自的附加单元时，合并结束。

上述设计思想反映的框图，见图5。

对照框图，不难写出（E-1）程序：

LIST

```
10 REM (E-1)
20 DIM A(11), B(10), C(19)
30 FOR I = 1 TO 10: READ A(I): NEXT
40 FOR I = 1 TO 9: READ B(I): NEXT
50 REM MERGE DATA PROGRAM
60 I = 1: J = 1: K = 0
70 M = A(I): P = 1
80 IF M < B(J) THEN P = 2
90 K = K + 1
100 IF P = 1 THEN C(K) = A(I): I = I + 1: GOTO 110
105 C(K) = B(J): J = J + 1
110 IF K < 19 THEN 70
120 FOR I = 1 TO 19
130 PRINT C(I); ", ";
140 NEXT I
150 DATA 98, 97, 89, 85, 84, 84, 71, 69, 63, 58
160 DATA 96, 94, 92, 90, 87, 83, 72, 69, 60
170 END
```

IRUN

98, 97, 96, 94, 92, 90, 89, 87, 85, 84, 84, 83, 72, 71, 69, 69, 63, 60, 58,

程序中10—30句读入并存贮数据。

110—130打印输出合并好的数据。

说明：

10到30句是将150句中的10个数存入A(I)数组中（I =

1, 2, ..., 10), 同时将160句中的9个数存入 $B(I)$ 数组中 ($I = 1, 2, \dots, 9$)。

40句是将 I 、 J 、 K 三个指针定初值, 分别为1, 1, 0。

60句先是特征变量 $P = 1$, 同时把 $A(I)$ 的值送入辅助存储单元 M 中, 开始 $I = 1 \therefore A(I) = A(1) = 98 \rightarrow M$ 。

70句, $B(J) = B(1) = 96$, $M < B(J)$? 不是 $\therefore \rightarrow 75$ 句。

75句, 计数器 K 加1 \therefore 第一组执行75句时 $K = 0 + 1 = 1, \rightarrow 80$ 句。

80句, $P = 1$, 满足条件, 设 $A(I) = A(1) = 98 \rightarrow C(K) = C(1)$, 即两数组中最大的一个数98存入 $C(1)$ 中, 同时让 I 加1, 即此时 $I = 2$, 要表示指向 A 数组中的第二个元素 $A(2) = 97 \rightarrow$ 指向100句。

100句, $K = 1 < 19$ 指向60句。

60句, $M = A(I) = A(2) = 97$, 此时 A 数组中的第二个元素97取到辅助存储单元中, 为与 $B(J)$ 比较作准备, P 仍等于1, $\rightarrow 70$ 句。

70句, $M < B(J)$? $97 < 96$? 不满足 $\rightarrow 75$ 句。

75句, 计数器加1, 即 $K = 1 + 1 = 2, \rightarrow 80$ 句。

80句, $P = 1$ 满足条件 $A(I) = A(2) = 97 \rightarrow C(K) = C(2)$ 中, 即两数组中次大的一个数97存入 $C(2)$ 中, 同时 I 指针改变, 即原来比较大的数对应的指针加1, $I = 2 + 1 = 3, \rightarrow 100$ 句。

100句, $K < 19$ 满足 $\rightarrow 60$ 。

60句, $A(I) = A(3) = 89 \rightarrow M$, 且 $P = 1 \rightarrow 70$ 。

70句, $M < B(J)$? $89 < 96$? 满足条件 \rightarrow 让 $P = 2 \rightarrow 75$ 。

75句, $K = K + 1 = 2 + 1 = 3 \rightarrow 80$ 。

80句, $P = 1$? 不满足 $\rightarrow 90$ 。

90句, $B(J) = B(1) = 96 \rightarrow C(K) = C(3)$, 这就是剩下的两组数中最大的一个数, 存入C数组中, 此时存入 $C(3)$ 中, $\therefore C(3) = 96$ 。较大的一个数存入 $C(1)$ 中, 对应大数的那一个数组指针要加1, $\therefore J = J + 1 = 2$, 即指向94。

如此循环下去, 只要 $K < 19$, A数组与B数组均要再比较, 那一个数组元素大, 就再存入 $C(1)$ 中, 并把相应数组的指针加1……

到此为止, $C(1) = 98$; $C(2) = 97$; $C(3) = 96$ 可以看出, 它们也是按顺序排列的。

以后的程序, 就是重复执行上述类似过程。

7. 数据分类

对一组数或一组字符串, 按从大到小或从小到大的次序进行排列, 称为分类, 又称排序 (SORT)。它是数据处理中最常用、最基本的工作之一, 又是对分搜索的基础和先决条件。

分类方法名目繁多, 结构多样, 如枚举分类法、插入分类法、气泡分类法、SHELL分类法、快速分类法等十余种。

这些分类方法, 往往构思巧妙, 技巧很强。熟悉几种分类方法, 对进一步提高程序设计技巧和编程能力大有帮助。

在各种各样的分类中, 有些流程比较简单, 程序短少, 易于初学者接受, 但占用机时较长, 例如, 枚举分类法, 就是最简单、最基本、速度最慢的分类方法; 有些程序, 结构复杂、迂回曲折、程序长占用内存多, 但执行速度却很快, 例如, 快速分类法, 对这两种分类方法, 我们将作比较仔细

的介绍，以飨读者。同时介绍一些新的处理技术，以达到开阔视野、扩大思维的目的。

(1) 枚举分类法

枚举分类法的主要思想是逐一比较，反复交换。

为简单起见，先讨论 N 个数值而不是字符的从大到小排序问题。

程序设计主要思想如下：

在数组 $R(N)$ 中，存有 N 个数，先以 $R(I)$ 与后面一个存贮单元内容 $R(J)$ 相比较。

如果 $R(I) > R(J)$ ，说明 $R(I)$ 内的值大，两者不交换，将 $R(J+1)$ 与 $R(I)$ 比较；如果 $R(I) < R(J)$ ，说明 $R(J)$ 内的值大，应将两者进行交换，交换后的 $R(I)$ 中的值仍为最大，然后再将 $R(J+1)$ 与 $R(I)$ 比较。

重复上述工作， $R(I)$ 与 $(N-1)$ 个数一一比较，直到所有 N 个数都比较完毕，那么比较的结果就是求出 N 个数中最大的一个数，并把它存入 $R(I)$ 中，例如 $R(1)$ 中，这里的 1 也可理解为上述全部工作的第一轮比较。

第二轮工作，是对剩下下来的 $(N-1)$ 个数，进行同上的工作，而每次都是把剩余下来的 $(N-1)$ 个数中的最大值，存入 $R(I)$ ，实际上是存入 $R(2)$ 中，这就是第二轮比较，一共要进行 $(N-2)$ 次比较。

重复上述过程，直到所有的数都相互比较完毕，并按从大到小的次序顺序排列为止。

若 $N = 10$ (10个数)，则要进行 $(N-1)$ 轮比较(9轮)，每次比较次数如下：

第一轮比较：比较 9 次

第二轮比较：比较 8 次

第三轮比较：比较 7 次

:
:
:

第八轮比较：比较 2 次

第九轮比较：比较 1 次

由此可见，枚举分类法的比较次数为 M：

$$M = N(N - 1) / 2。$$

以上面 10 个数比较为例，则完成排序工作要进行 45 次比较。

值得提出的是，上述设计思想决定了比较次数无法减少。因为它们是一一比较，即使数据已经分类好，仍要逐个访问，一一比较，所以，枚举法排序速度很慢。

对 20 个凌乱无序的数，要求从大到小顺序排列，程序 PRO-69 及运行结果如下。

PRO-69

ULIST

```
5  REM PRO-69
10  DIM R(20)
20  FOR I = 1 TO 20
30  READ R(I)
40  NEXT I
50  FOR I = 1 TO 19
60  FOR J = I + 1 TO 20
70  IF R(I) > R(J) THEN 90
80  C = R(I):R(I) = R(J):R(J) = C
90  NEXT J
100 PRINT R(I); " ";
110 NEXT I
120 PRINT R(20)
130 END
160 DATA 37,24,53,0,-1,97,21,34,5
      7,87,-64,473,-250,48,23,77,7,
      8,9,44
```



```
ORUN  
N=380
```

PRO-72

```
ULIST
```

```
40 REM PRO-72  
50 FOR I = 1 TO 19  
60 FOR J = 2 TO 20  
65 N = N + 1  
90 NEXT J  
110 NEXT I  
115 PRINT "N=";N
```

```
ORUN  
N=361
```

PRO-73

```
ULIST
```

```
40 REM PRO-73  
50 FOR I = 1 TO 19  
60 FOR J = I + 1 TO 20  
65 N = N + 1  
90 NEXT J  
110 NEXT I  
115 PRINT "N=";N
```

```
ORUN  
N=190
```

此外，本程序在打印输出语句的安排上，也有一个小技巧，见100句和120句。这里没有另设新的循环，而在每次内循环后，打印每一轮比较后的最大值，在全部两重循环结束后，打印最后一个（一定是N个数中的最小值）数据。

对上述程序稍加改动，即可处理字符串排序的问题。下面给出计算机自动排国家英语名的程序（共8个国家）和结果。两个程序模式几乎一样，差别在PRO-74采用字符串比

PRO-74

ULIST

```
5  REM PRO-74
10  DIM A$(8)
20  FOR I = 1 TO 8
30  READ A$(I)
40  NEXT I
50  FOR I = 1 TO 7
60  FOR J = I + 1 TO 8
70  IF A$(I) < A$(J) THEN 90
80  T$ = A$(I):A$(I) = A$(J):A$(J) =
    T$
90  NEXT J
100 PRINT A$(I); " ";
110 NEXT I
120 PRINT A$(8)
130 END
140 DATA CHINA,JAPAN,CANADA,KOREA
    ,INDIA
150 DATA CHILE,GREECE,ITALY,NEPAI
    ,PANAMA
```

URUN

```
CANADA CHILE CHINA GREECE INDIA ITALY
JAPAN KOREA
```

较的方法。

(2) 两种排名新法程序

传统的排序，通常都是采用比较的方法，改变和交换数组中的内容。这样做缺点比较明显，并列名次的情况不能区分开来。

下面介绍的排序方法，能够克服上述缺点。主要思路是不采用改变和交换数组内容，而是另设名次计数器。

例如，有20个学生，已知他们的学号和成绩，要求能排出学号、名次（包括并列者）和成绩。

首先，安排一个名次计数器K，并令 $K = 1$ 。这实际上是假设第一个学生的成绩为全班之首。

建立一个数组 $R(I)$ ，存放各人成绩。

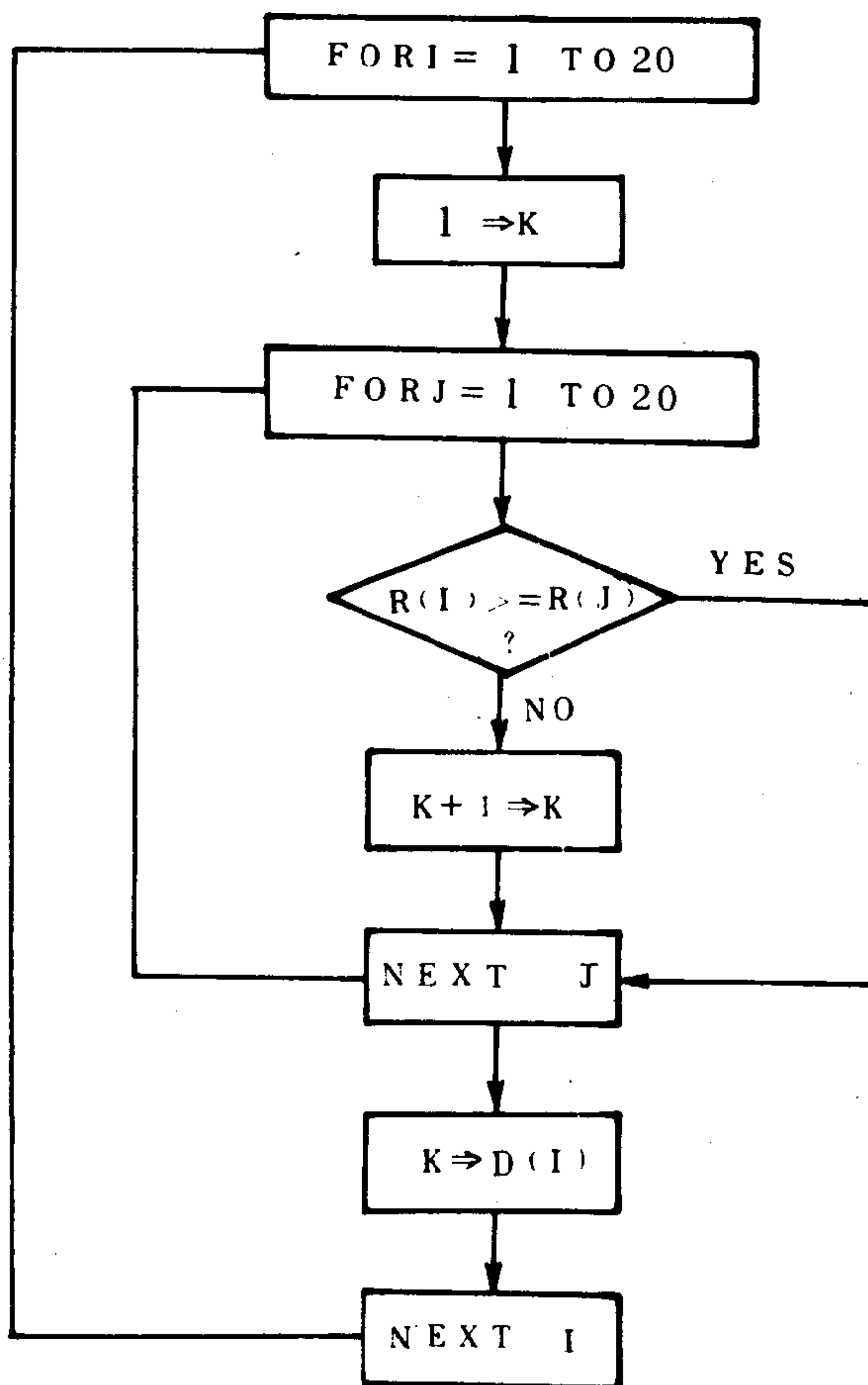


图 6

若 $R(I) \geq R(J)$, 则再比较, 如将 $R(I)$ 与 $R(J+1)$ 比较;

反之, 则表示 $R(I)$ 的成绩低于 $R(J)$ 的成绩, 此时令 $K = K + 1$, 这表明 $R(I)$ 的成绩名次退后一名。

如此重复往返, 当所有学生成绩都比较完毕, 则令 $D(I) = K$, 这表示在第一轮比较中, 全班成绩名次最低者, 名次放在 $D(I)$ 中。

重复上述工作, 进行 $N-1$ 轮比较, 依次确定各名次, 直到第一名。

主要程序段框图见图 6。

程序 PRO-75 及执行结果如下:

PRO-75

```
10  REM PRO-75
20  DIM R(20), D(20)
30  FOR I = 1 TO 20
40  READ R(I)
50  NEXT I
60  FOR I = 1 TO 20
70  K = 1
80  FOR J = 1 TO 20
90  IF R(I) >= R(J) THEN 110
100 K = K + 1
110 NEXT J
120 D(I) = K
130 NEXT I
140 FOR I = 1 TO 20
150 PRINT I; TAB(5); D(I); TAB(10); R(I)
160 NEXT I
170 DATA 65, 98, 75, 72, 69, 54, 91, 79,
        100, 86, 84, 98, 74, 62, 45, 96, 87,
        79, 85, 53
```

| URUN | | |
|------|----|-----|
| 1 | 16 | 65 |
| 2 | 2 | 98 |
| 3 | 12 | 75 |
| 4 | 14 | 72 |
| 5 | 15 | 69 |
| 6 | 18 | 54 |
| 7 | 5 | 91 |
| 8 | 10 | 79 |
| 9 | 1 | 100 |
| 10 | 7 | 86 |
| 11 | 9 | 84 |
| 12 | 2 | 98 |
| 13 | 13 | 74 |
| 14 | 17 | 62 |
| 15 | 20 | 45 |
| 16 | 4 | 96 |
| 17 | 6 | 87 |
| 18 | 10 | 79 |
| 19 | 8 | 85 |
| 20 | 19 | 53 |

注：I 起控制学号作用，J 控制成绩比较，K 是名次计数器，比较成绩决定名次，R(I) 存贮成绩，D(I) 存贮名次。

另一种排序程序，设计思想和上述程序相近，只是开始时，都把各人成绩假定为第一名，然后用两重循环来判别成绩名次。方法是：

若 $R(J) > R(K)$ ，则将 $D(K) + 1 \Rightarrow D(K)$ ，成绩低的退居一位；若 $R(J) < R(K)$ ，则让 $D(J) + 1 \Rightarrow D(J)$ ，也是成绩低的退居后位。见程序 PRO-76。

PRO-76

```

10  REM PRO-76
20  DIM R(20), D(20)
30  FOR I = 1 TO 20: READ R(I): D(I)
    ) = 1: NEXT I

```

```

40  FOR J = 1 TO 19: FOR K = J + 1
      TO 20
50  IF R(J) > R(K) THEN LET D(K) =
      D(K) + 1
60  IF R(J) < R(K) THEN LET D(J) =
      D(J) + 1
70  NEXT K
80  PRINT J; TAB( 5);D(J); TAB( 10
      );R(J)
90  NEXT J
100 PRINT 20; TAB( 5);D(20); TAB(
      10);R(20): END
110 DATA 65,98,75,72,69,54,71,79,
      100,86,84,98,74,62,45,96,87,7
      9,85,53

```

运行结果和前面程序一致。

这两个程序最大特点是：打破常规的设计思路，跳出传统的设计框框，有效地解决了名次并列的排序问题。

(3) 气泡分类法

实际分类工作中，常常有这样的情况，N个数据大部份已经排序好，但尚需调整个别数据。显然，不宜采用枚举分类法。正如前面指出的，枚举分类法的思想是逐一比较，逐个访问，即使对已经分类好的数据也不例外，因此，必须另求它法。

气泡分类法就是一个能适用于数据已经粗略分类，但尚需调整个别数据的分类方法。

在具体讲述气泡分类法之前，不仿先看一下程序PRO-77。

PRO-77

```

5  REM PRO-77
10  DIM R(5)
20  FOR I = 1 TO 5
30  READ R(I)

```



```

40 NEXT I
50 FOR I = 4 TO 1 STEP - 1
60 A = 1
70 FOR J = 1 TO I
80 IF R(J) > R(J + 1) THEN 110

90 T = R(J):R(J) = R(J + 1):R(J +
    1) = T
100 A = 0
110 NEXT J
120 IF A = 1 THEN 140
130 NEXT I
140 FOR I = 1 TO 5
150 PRINT R(I);", ";
160 NEXT I
170 DATA 4,7,6,5,1
180 END

```

URUN
 7,6,5,4,1,

程序第一段10—40句，开辟内存，读5个数并存在数组R(I)中。

程序第三段140—160句，打印R(I)的值， $I = 1, 2, 3, 4, 5$ 。

程序第二段50—130句，也是一种分类方法，它是气泡分类法的核心语句段。为了理解这一段内容，我们来代替计算机执行这段程序。当然，要考虑第一段的存贮内容，即 $R(1) = 4$ ， $R(2) = 7$ ， $R(3) = 6$ ， $R(4) = 5$ ， $R(5) = 1$ 。

上述存贮情况告诉我们，只要将R(1)中的值4插入R(4)和R(5)之间，那么这样排列就是一个有序序列。

$I = 4$ 时， $A = 1$ 。

$J = 1$: $R(J) = R(1) = 4$ ， $R(J + 1) = R(2) = 7$ ， $R(1) < R(2)$ ，交换，现在 $R(1) = 7$ ， $R(2) = 4$ ，且 $A = 0$ 。

$J = 2: R(J) = R(2) = 4, R(J+1) = R(3) = 6, R(2) < R(3)$, 交换, 现在 $R(2) = 6, R(3) = 4$, A 仍为 0。
 $J = 3: R(J) = R(3) = 4, R(J+1) = R(4) = 5, R(3) < R(4)$; 交换, 现在 $R(3) = 5, R(4) = 4$, A 仍为 0。
 $J = 4: R(J) = R(4) = 4, R(J+1) = R(5) = 1, R(4) > R(5)$, 不交换, 跳出内循环。

注意上述交换 $A = 0$ 。

这里跳出内循环第一轮比较结束。而且可以看出数据已经排好, 即:

$R(1) = 7, R(2) = 6, R(3) = 5, R(4) = 4, R(5) = 1$ 。
因为 $A = 0$, 不满足 $A = 1$ 的条件而转向 130 句, I 加步长, 此时 $I = 3$, 并重新使 $A = 1$, 进行第二轮循环。由于第一轮循环已经将排好, 没有任何交换了, 即特征值 A 不变, 满足 120 句条件, 转向 140 句, 打印输出结果, 分类结束。可见本程序仅比较了 4 次, 交换速度较快。

若用枚举法比较时, 则要进行 4 轮, 第一轮比较 4 次, 第二轮比较 3 次, 第三轮比较 2 次, 第四轮比较 1 次, 共比较 10 次。因此, 它比气泡分类法慢。

现在, 我们来说明一下气泡分类法的主要设想:

- 用相邻两数进行比较, 每次总是把大数放在前面, 小数放在后面;

- 小数再与下一个邻接的单元比较, 若 $R(J) \geq R(J+1)$ 不交换, $R(J) < R(J+1)$ 交换;

- 设置了一个特征存贮单元 A , 它能标志分类是否结束, 这是和枚举法分类的主要区别, 它不像枚举法一定要进行 $N(N-1)/2$ 次比较, 因而速度较快。

气泡分类法, 见程序 PRO-78。

PRO-78

```
2  REM PRO-78
5  INPUT N
10 DIM R(N)
20 FOR I = 1 TO N
30 READ R(I)
40 NEXT I
50 FOR I = N - 1 TO 1 STEP - 1
60 A = 1
70 FOR J = 1 TO I
80 IF R(J) > R(J + 1) THEN 110

90 T = R(J):R(J) = R(J + 1):R(J +
    1) = T
100 A = 0
110 NEXT J
120 IF A = 1 THEN 140
130 NEXT I
140 FOR I = 1 TO N
150 PRINT R(I); " ";
160 NEXT I
170 PRINT
180 END
190 DATA (N1),.....,(Nn)
```

变量设置: $R(N)$ ——待分类的 N 个数据,

I, J ——循环控制变量,

T ——辅助存贮单元,

A ——特征标志, 数值 1 或 0。

(4) 插入分类法

将 N 个数用插入分类法进行由大到小的排序。

例如, 有 4 个数已存入 $R(I)$ 中, 即 $R(1)=15, R(2)=14, R(3)=9, R(4)=13$ 。

可以看出 15, 14, 9, 13 中, 前面三个数已经排好, 要“插入”, 即将 13 插入到 14 的后面, 而且数值 9 必须右移一个位置。用存贮的观点讲, 就是 $R(4)$ 单元的值要放入 $R(3)$ 中, 而

原来 $R(3)$ 单元的值移至 $R(4)$ 中去。这实际上也是一种交换。

插入分类的思路是：假定 N 个数中前 K 个数已经按从大到小次序分类，现在让 $K+1$ 单元的数 $R(K+1)$ 与前 K 个数进行比较，如果比较到第一个比 $R(K+1)$ 之值大的数是 $R(J)$ ，就让 $R(K+1)$ 存入第 $J+1$ 个单元，而把原来从 $J+1$ 到 K 的存贮单元内容都顺次后移一个单元。这样就把前 $K+1$ 个单元中的数分类完成，再考虑第 $K+2$ 个单元的数，直到最后一个为止。

为实现上述分类设想，需设一个指针 J ，同时，还要设置一个辅助存贮单元 X ，以存放当前处理的数据。

下面给出上述设想的主要程序段，并分析一下执行过程。

```

50  FOR I = 2 TO 4
60  J = I - 1
70  X = R(I)
80  IF X <= R(J) THEN 120
90  R(J + 1) = R(J)
100 J = J - 1
110 IF J > 0 THEN 80
120 R(J + 1) = X
130 NEXT I

```

执行过程如下：

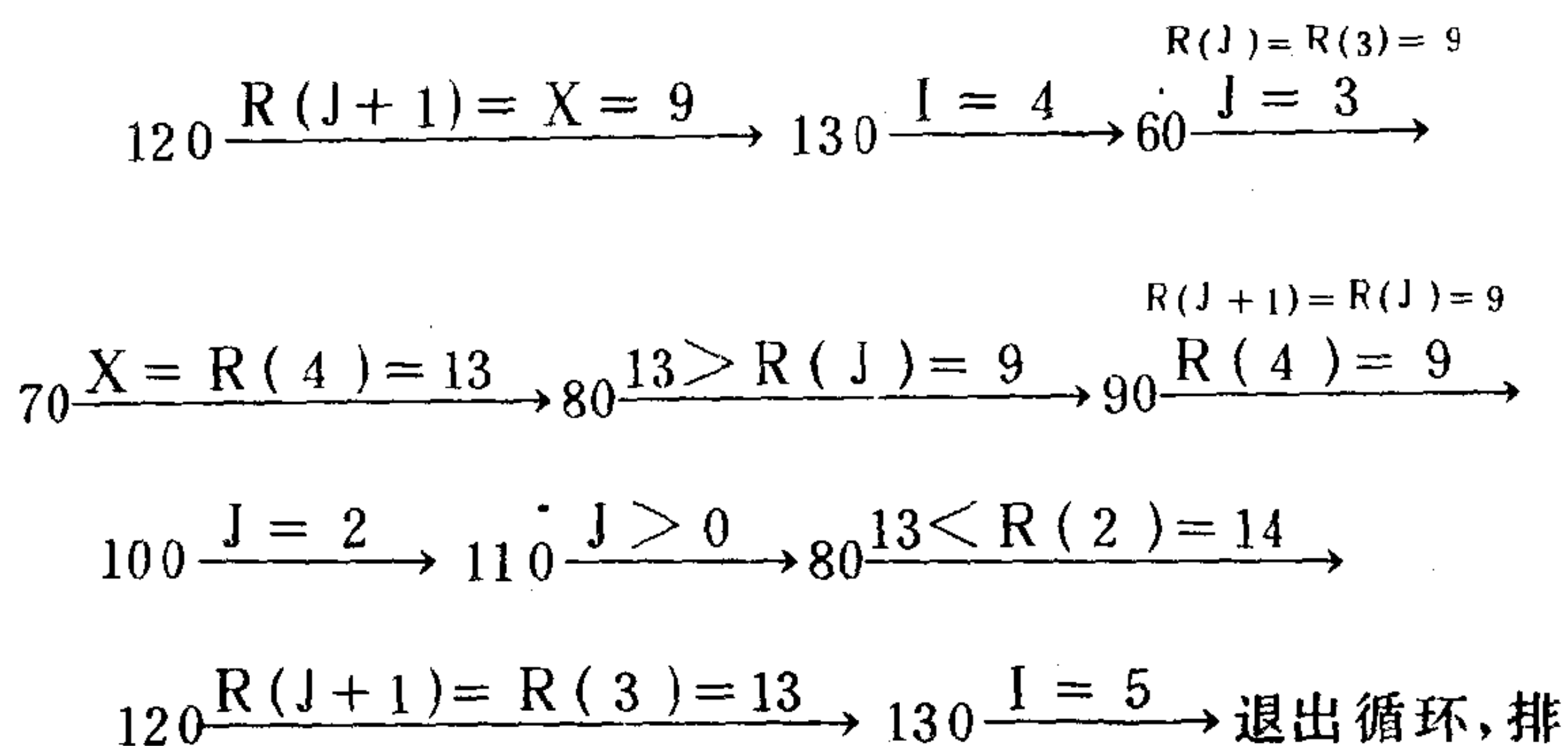
$R(J) = R(1) = 15$

50 $I = 2$ → 60 $J = 1$ → 70 $X = R(I) = R(2) = 14$ →

80 $X < R(J)$ → 120 $R(J+1) = X = 14$ → 130 $I = 3$ →

$R(J) = R(2) = 14$

60 $J = 2$ → 70 $X = R(3) = 9$ → 80 $X < R(J)$ →



序完成。

如果有20个数，用插入分类法从大到小排序，则只要在上述程序基础上加上两段，见程序PRO-79。

PRO-79

```

5  REM PRO-79
10  DIM R(20)
20  FOR I = 1 TO 20
30  READ R(I)
40  NEXT I
50  FOR I = 2 TO 20
60  J = I - 1
70  X = R(I)
80  IF X < R(J) THEN 120
90  R(J + 1) = R(J)
100 J = J - 1
110 IF J > 0 THEN 80
120 R(J + 1) = X
130 NEXT I
140 FOR I = 1 TO 20
150 PRINT R(I); ", ";
160 NEXT I
170 END
180 DATA 12, 5, 3, 9, 7, 6, 0, 90, 176, 4
190 DATA 54, 76, 888, 71, 23, 11, 22, 34
    , 99, 1

```

URUN

888, 176, 99, 90, 76, 71, 54, 34, 23, 22, 12, 11, 9, 7, 6,
5, 4, 3, 1, 0,

(5) 快速分类法

快速分类法是目前名目繁多的各种分类方法中速度最快的一种。这一方法结构比较复杂, 技巧性也很强。

设计思想: 选取一个关键字, 例如, 待分类的第一个记录 $R(1)$, 作为比较标准。开始把 $R(1)$ 存入暂存单元 X 中, 然而让 X 与 $R(J)$ 比较, $R(J)$ 可以是待分类数据中最后一个记录, 如 20 个数据, 使 $R(J) = R(20)$, 若 $X < R(J)$, 则把所有大于 X 值的 $R(J)$ 放在 X 的右边; 若 $X > R(J)$, 则把所有小于 X 值的 $R(J)$ 放在 X 的左边。这样, 就可以把原数据记录分成左、右两个数据集合, 称子集合或子数组。然后, 再对这两个子数组重复上面的工作。

由于该方案是不断地对数据记录进行划分, 又不断地从左、右两部份进行处理, 所以就需要在存贮空间上开辟一定数量的栈区, 用于存放待处理的子集合中有关信息。

为了实现上述设计思想, 首先设置两个指针 I 和 J 。开始 I 取 1, 指向首数据 $R(1)$, J 取 20, 指向最后一个数据(若我们要分类的一共 20 个数据)。 I 叫首指针, J 叫尾指针。

然后, 把 $R(1)$ 存入暂存单元 X 中, 让 X 与 $R(J)$ 比较; 如图 7 所示。

这样, X 与 $R(J)$ 比较后, 总有一根指针移动一个位置。接下来, 让 X 再与移动了位置的指针所指向的单元内容相比较, 过程同上。因而 I 和 J 指针逐渐向中间靠拢。当 $I = J$ 时(两根指针重在一起), 则把暂存单元 X 的内容存入 I 单元。经过上述一轮比较, 则 X 的位置确定了, 见图 8。

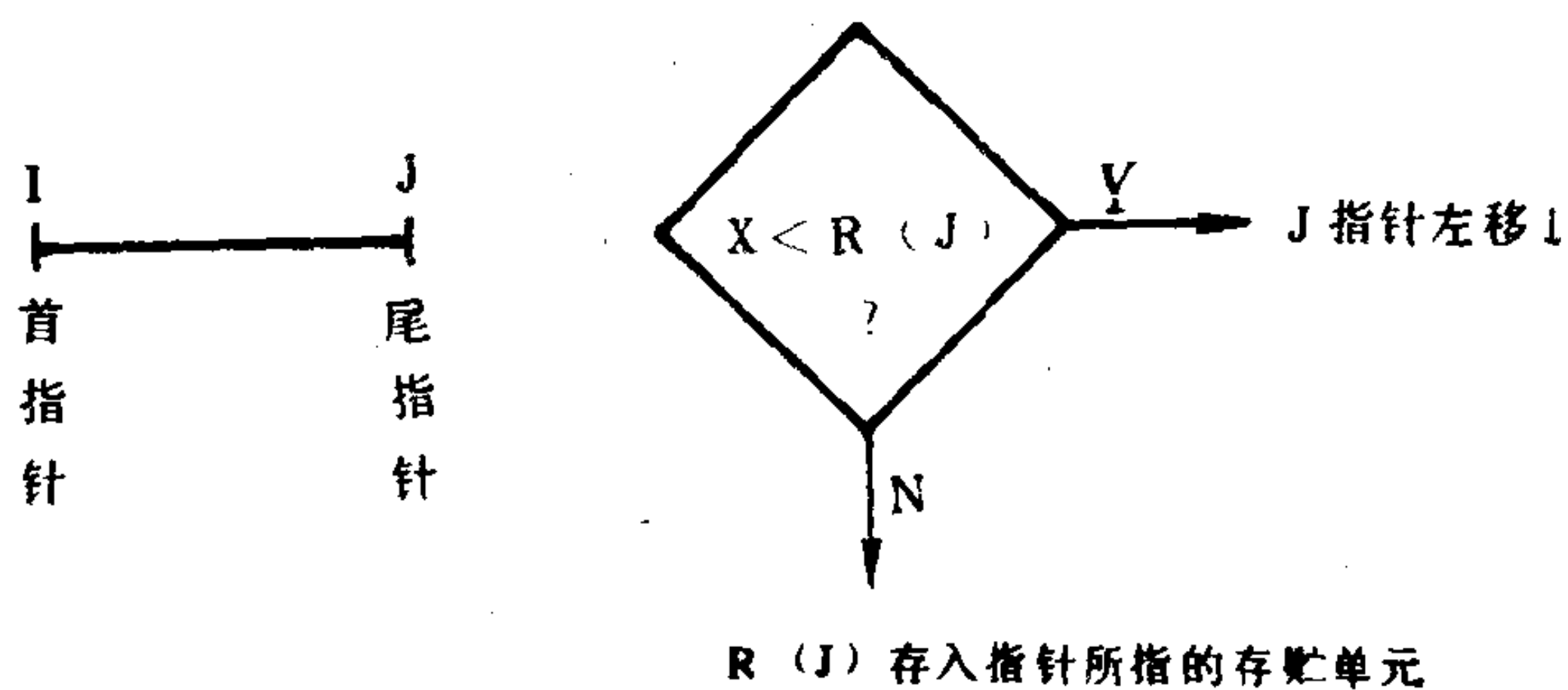


图 7

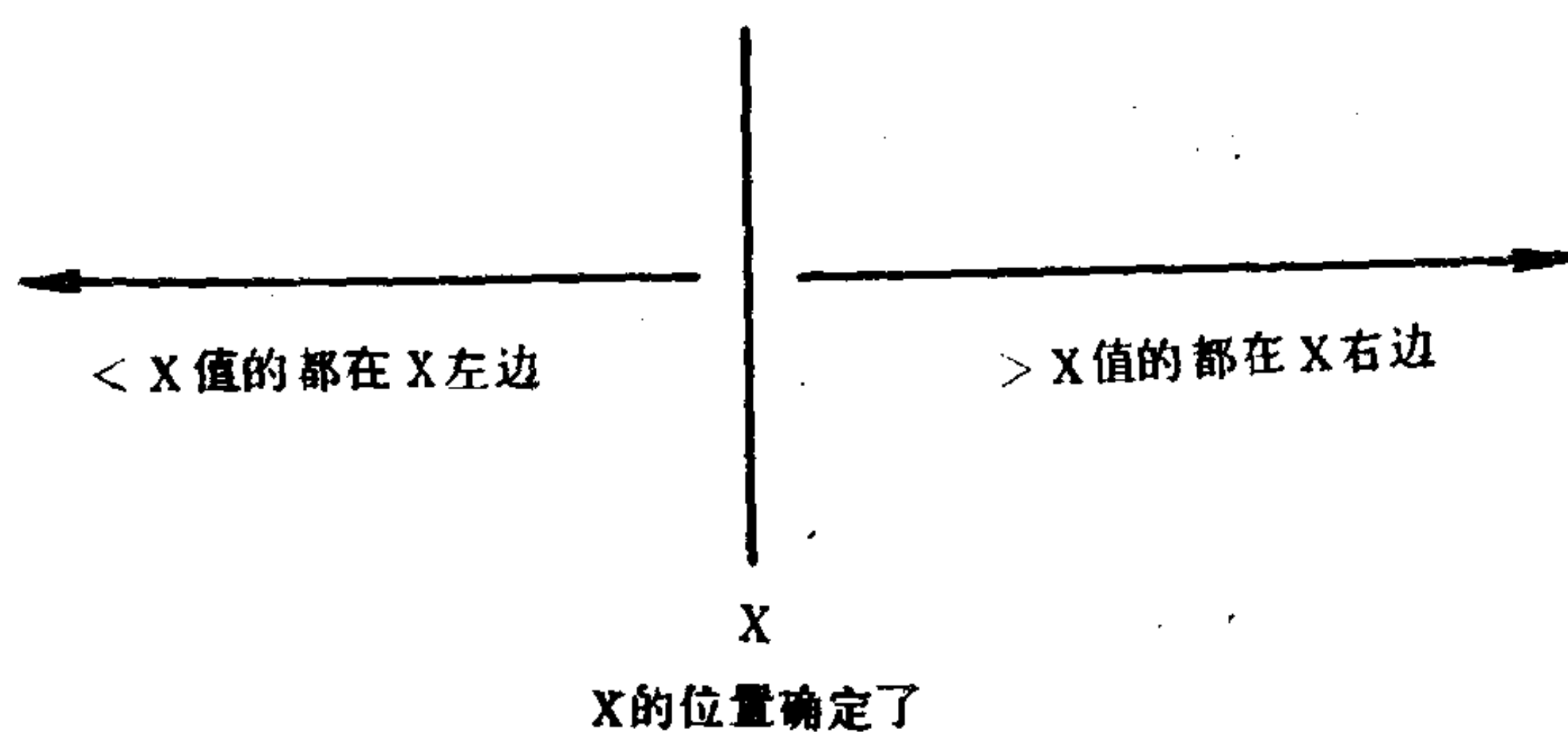
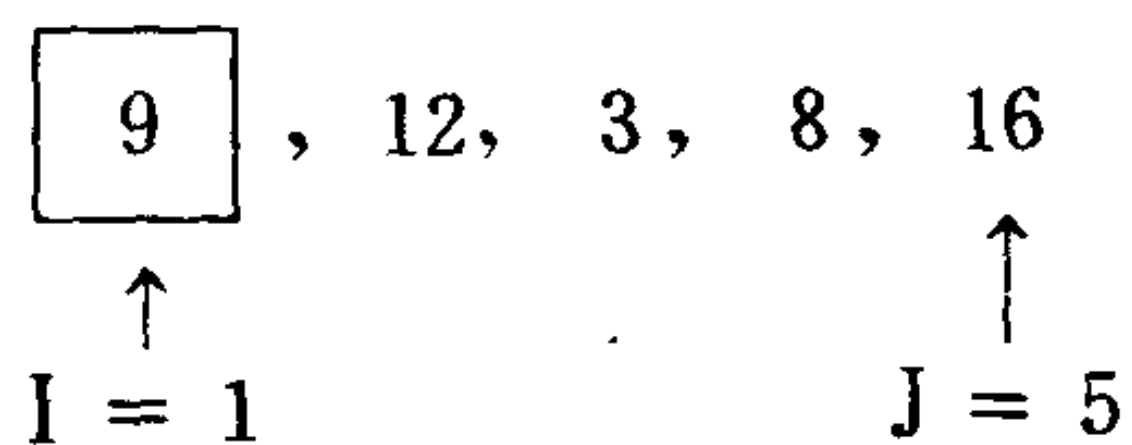


图 8

这样，就把原数组分成两个待分类的新数组。

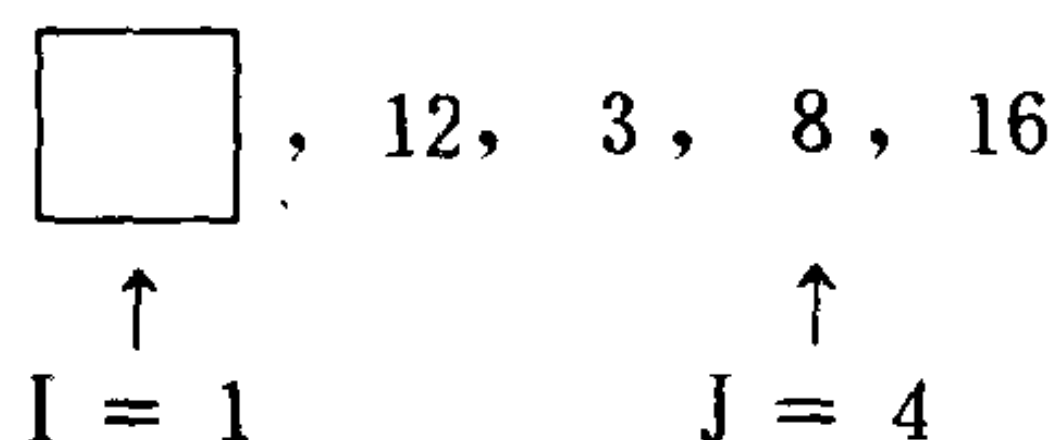
例如，有 5 个数，要求从小到大排列。



起始状态 $X = 9$, X 是暂存单元, 表示该单元已空, 作为以后分类的缓冲单元。

① 第一次比较

$X = 9 < R(J) = R(5) = 16$, J 指针左移 1

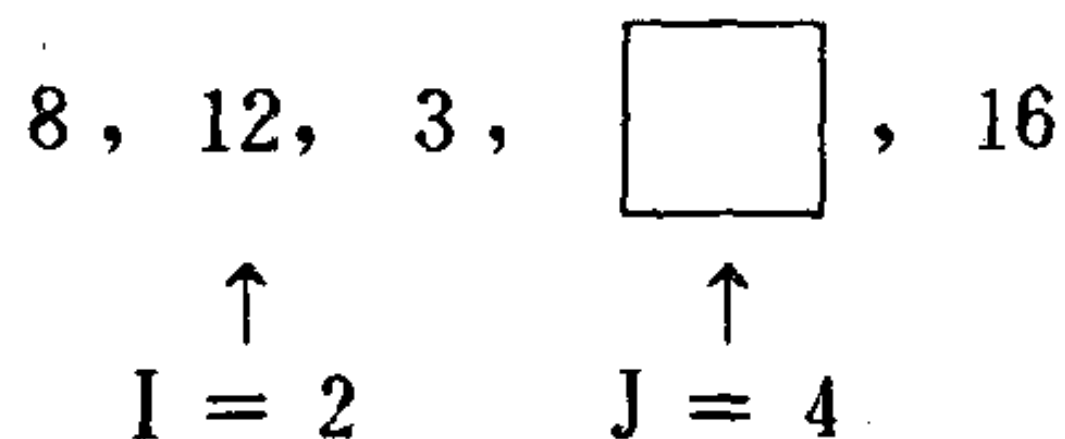


② 第二次比较

X 与移 3 位的指针 J 所指向的内容比较:

$X = 9 > R(J) = R(4) = 8$

$\therefore X > R(J)$, \therefore 把 $R(J)$ 存入 I 指针所指的存贮单元, 并把 I 指针右移 1, 即:

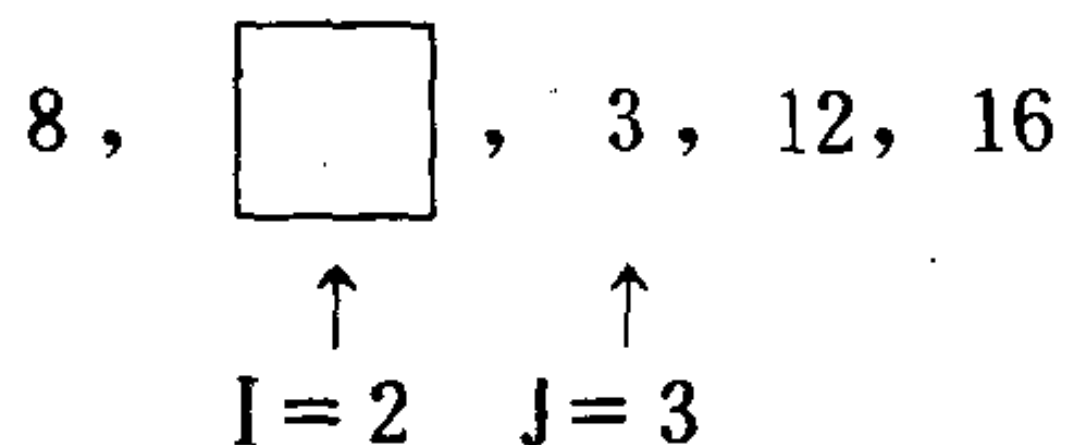


③ 第三次比较

X 与移了位的指针 I 所指单元内容比较:

$X = 9 < R(I) = R(2) = 12$

$\therefore X < R(I)$, \therefore 要把 J 指针左移 1, 同时把 $R(I) = 12$, 存入缓冲单元, 而 $I = 2$ 的单元空缺, 即:



④ 第四次比较

X 与移了位的指针 J 所指单元内容比较:

$X = 9 > R(J) = R(3) = 3$

PRO-80

```
5  REM PRO-80
10  DIM R(20),T(10)
20  T(1) = 1:T(2) = 20
30  Z = 2
40  FOR I = 1 TO 20
50  READ R(I)
60  NEXT I
70  PRINT
75  IF Z = 0 THEN 180
80  P = T(Z):Z = Z - 1
90  L = T(Z):Z = Z - 1
100 GOSUB 300
110 IF P < I + 1 THEN 140
120 Z = Z + 1:T(Z) = I + 1
130 Z = Z + 1:T(Z) = P
140 IF I < = L + 1 THEN 75
150 Z = Z + 1:T(Z) = L
160 Z = Z + 1:T(Z) = I - 1
170 GOTO 75
180 FOR I = 1 TO 20
190 PRINT R(I);", ";
200 NEXT I
205 PRINT
210 GOTO 510
300 I = L:J = P + 1:X = R(I)
310 J = J - 1
320 IF I = J THEN 400
330 IF X < = R(J) THEN 310
340 R(I) = R(J)
350 I = I + 1
360 IF I = J THEN 400
370 IF X > = R(I) THEN 350
380 R(J) = R(I)
390 GOTO 310
400 R(I) = X
410 RETURN
420 DATA 37,49,2,40,100,4,20,38,
      85,939,101,2,30,98,40,63,65,1
      ,9,402
510 RESTORE
520 FOR I = 1 TO 20
```

```

530 READ R(I)
540 NEXT I
550 FOR I = 1 TO 19
560 FOR J = I + 1 TO 20
570 IF R(I) < R(J) THEN 590
580 C = R(I):R(I) = R(J):R(J) = C
590 NEXT J
600 PRINT R(I);", ";
610 NEXT I
615 PRINT R(20)
620 END

```

GRUN

```

1,2,2,4,9,20,30,37,38,40,40,49,63,65,85,98,
100,101,402,939,
1,2,2,4,9,20,30,37,38,40,40,49,63,65,85,98,
100,101,402,939

```

优劣。

注：上述程序中，5—420句为快速分类程序，510—620句为枚举法分类程序。210句的GOTO 510为连接两个程序的语句。510句恢复数据区语句RESTORE为枚举法分类程序的读数做准备。

快速分类法流程较长，结构复杂，迂回曲折，构思巧妙，但省机时，速度快，对大量数据信息的分类，此法甚优。

(6) 新的快速分类法

下面再介绍一个新的快速分类法，这个方法有以下几个特点：

- 结构简单，通俗易懂；
- 节省内存，速度很快；
- 设计新颖，构思巧妙。

首先，我们举一个简单的实例，来剖析一下这种新的快速分类的设计思想。为分析方便计，只安排了五个学生的成绩，要求从大到小按序排列，见程序PRO-81。

PRO-81

```
5  REM PRO-81
10  INPUT N
20  DIM A(100)
30  FOR I = 1 TO N
40  READ X
50  A(X) = A(X) + 1
60  NEXT I
70  FOR I = 100 TO 0 STEP - 1
80  IF A(I) = 0 THEN 130
90  FOR J = 1 TO A(I)
100 C = C + 1
110 PRINT C;" ";I
120 NEXT J
130 NEXT I
140 DATA 95,81,99,87,69
```

```
ORUN
75 .
1 99
2 95
3 87
4 81
5 69
```

变量及存贮说明：

X为成绩，设满分为100。

A(X)考分为X的学生数。

30—60句，读成绩并把它们存贮在A(X)中。A(X)=A(X)+1，是统计成绩为X的学生人数，成绩X兼作数组A的下标。

70—130句，按考分由高到低打印名次和成绩。

N为学生人数，本例 $N=5$ 。

I为循环变量，控制分数。

J为循环变量，控制相同成绩的人数。

程序执行过程如下：

首先从键盘敲入5，第一次读入95，并放在 $A(95)$ 中，经50句处理 $A(95)=1$ ，再循环上去，以后不断读入、存贮并统计，直到所有数据读完、存好，并统计同成绩人的个数。本题没有相同成绩的学生，故执行70句前存贮结果如下：

$A(95)=1, A(81)=1, A(99)=1, A(87)=1, A(69)=1$ 。

70句开始，由I控制分数，从满分100分开始，只要 $A(I)=0$ ，就不要（这个意思是，原存贮区 $A(I)$ 中，没有相应成绩的人数，则循环上去），否则执行J循环，并打印名次和成绩。由于I是从高分（满分100）开始，到0分逐个扫描成绩，并通过计数器C计数，所以能打印出名次C和成绩I，并按高分到低分顺序排列。

可见，本程序十分简单，但也很巧，完全不受考生人数的限制，因而占用内存特别少， $A(100)$ 不变。而传统的排序方法，将考分存入数组，势必增加存贮空间，而在考生人数超过一定限度时，将因内存空间不够而无法执行程序。其次，本程序的最大特点，是高速排序，它比不少书上介绍过的各种不同的“快速”排序方法，还要快一些，其根本原因，是废除了反复比较，不断交换的过程。

若将上述题目，加上处理学号的部份，程序只要作适当改动即可。程序清单及运行实例如P R O-82。

PRO-82

ULIST

```
5  REM PRO-82
10 N = 10
20  DIM A(100),A$(100)
30  FOR I = 1 TO N
40  READ X$,X
50  A(X) = A(X) + 1:A$(X) = A$(X) +
    X$
60  NEXT I
70  FOR I = 100 TO 0 STEP - 1
80  IF A(I) = 0 THEN 130
90  FOR J = 1 TO A(I)
100 C = C + 1
110 PRINT MID$(A$(I),4 * (J - 1
    ) + 1,4),C,I
120 NEXT J
130 NEXT I
140 DATA 8801,91,8802,97,8803,76,
    8804,79,8805,91,8806,73,8807,
    66,8808,94,8809,99,8810,61
150 END
```

URUN

| | | |
|------|----|----|
| 8809 | 1 | 99 |
| 8802 | 2 | 97 |
| 8808 | 3 | 94 |
| 8801 | 4 | 91 |
| 8805 | 5 | 91 |
| 8804 | 6 | 79 |
| 8803 | 7 | 76 |
| 8806 | 8 | 73 |
| 8807 | 9 | 66 |
| 8810 | 10 | 61 |

8. 数据检索

检索是数据处理中另一个功能重要而又最基本的工作。在事务管理、情报检索、文献组织以及办公室自动化各个方面，都有着广泛的应用。

检索方法也是多种多样，最简单的一种是线性检索，又称顺序查找，其思路十分简单，即按照先后次序对记录集中记录（数字或字符），逐一进行访问（对数值或字符进行比较），查看是否与需要查找的信息相同，如果一样，则打印显示此信息，否则继续搜索，直到全部记录查找完毕。

下面是一个实例。

编制一个通用程序，要求能查找书名和作者姓名都符合的 BASIC 程序。

为简单计，DATA 区中只放了 4 个图书信息，它们分别是书号、书名、作者姓名、出版年月。

若找不到，程序给出有关提示信息。

下面是程序 PRO-83 清单及运行结果：

PRO-83

```
2  REM PRO-83
3  REM  SEARCHING PROGRAM
5  REM  ARE YOU TRYING TO FIND A B
   DOK?
10  INPUT A$,D$
12  PRINT A$;" ";D$
15  IF A$ = "NO" AND D$ = "NO" THEN
   END
18  RESTORE
20  FOR I = 1 TO 4
30  READ N,B$,C$,M
40  IF B$ = A$ AND C$ = D$ THEN 80
50  NEXT I
```

```

60 PRINT "----NOT FOUND----"
70 GOTO 10
80 PRINT N; " ";B$; " ";C$; " ";M
85 GOTO 10
100 DATA 3721,BASIC,JOHN SMITH,19
    67
110 DATA 1114,BASIC,GREEN,1980
120 DATA 7701,FORTRAN,BROWN,1983
130 DATA 3340,APPLE-2,BILL,1982

```

```

URUN
?BASIC
??GREEN
BASIC GREEN
1114 BASIC GREEN 1980
?APPLE-2
??BILL
APPLE-2 BILL
3340 APPLE-2 BILL 1982
?FORTRAN
?BROWN
FORTRAN BROWN
7701 FORTRAN BROWN 1983
?BASIC
??BILL
BASIC BILL
----NOT FOUND----
?NO
??NO
NO NO

```

应该指出的是，线性查找的速度是比较慢的。

对于由 N 个记录组成的集合进行一次成功的查找，在最好的情况下只需要一次关键字的比较；而最坏的情况则需要进行 N 次的比较。其平均查找次数为 $K_1 = N/2$ 。所以当 N 足够大时，线性查找的速度是很慢的。如 $N = 64K = 64 \times 1024 = 2^{16} = 65536$ ，用线性查找其平均次数高达 32768 次。减少搜

索次数，提高查找速度，找到一个更有效的检索方法，无论在理论上还是在实践中都是有意义的。

下面介绍的对分搜索法（又称二分查找法），就是一个比较理想的查找方法。它的平均查找次数为 $K_B = \log_2 N - 1$ 。如 $N = 64$ $K = 2^{16}$ ，则对分搜索的平均查找次数仅为15次。

由此可知，对分搜索要比顺序查找优越得多。一个明显的例子是，在一个藏书高达几百万册的图书馆里，要进行现代化管理，对分搜索技术，不是“可用可不用”，而是非用不可了。

所谓二分搜索，就是将查找范围二等分，把中点处记录关键字记为 $R(K)$ ，待查找的记录关键字叫 A ，每次让 $R(K)$ 与 A 进行比较。

(1) 如果 $A = R(K)$ ，则输出结果，第一次就查找成功，查找结束；

(2) 如果 $A > R(K)$ ，则对左半部份的记录实行二分查找；

(3) 如果 $A < R(K)$ ，则对右半部份的信息进行对分搜索。

重复上述步骤，直到找到为止。

应该特别指出的是，应用二分查找的先决条件是，数据区中的数值或字符串必须是有序排列的。

为了实现上述设计思想，在程序中应设置两个指针 I ， J 。 I 为查找区域的尾指针， J 为查找区域的首指针。另外，每次把查找区域二等分，也需要一个指针，定为 K ， K 实际上是一个中间指针，即 $K = \text{INT}((I + J)/2)$ 。如该指针指向中间数 $R(K)$ 时，就与输入的关键字 A 进行比较，相等则输出结果。

这里还有一个问题必须解决,如果第一次没有查找到(这是经常发生的),需要改变中间指针 K 。我们这样来处理。

若 $A > R(K)$, 则 $K \Rightarrow J$;

若 $A < R(K)$, 则 $K \Rightarrow I$ 。

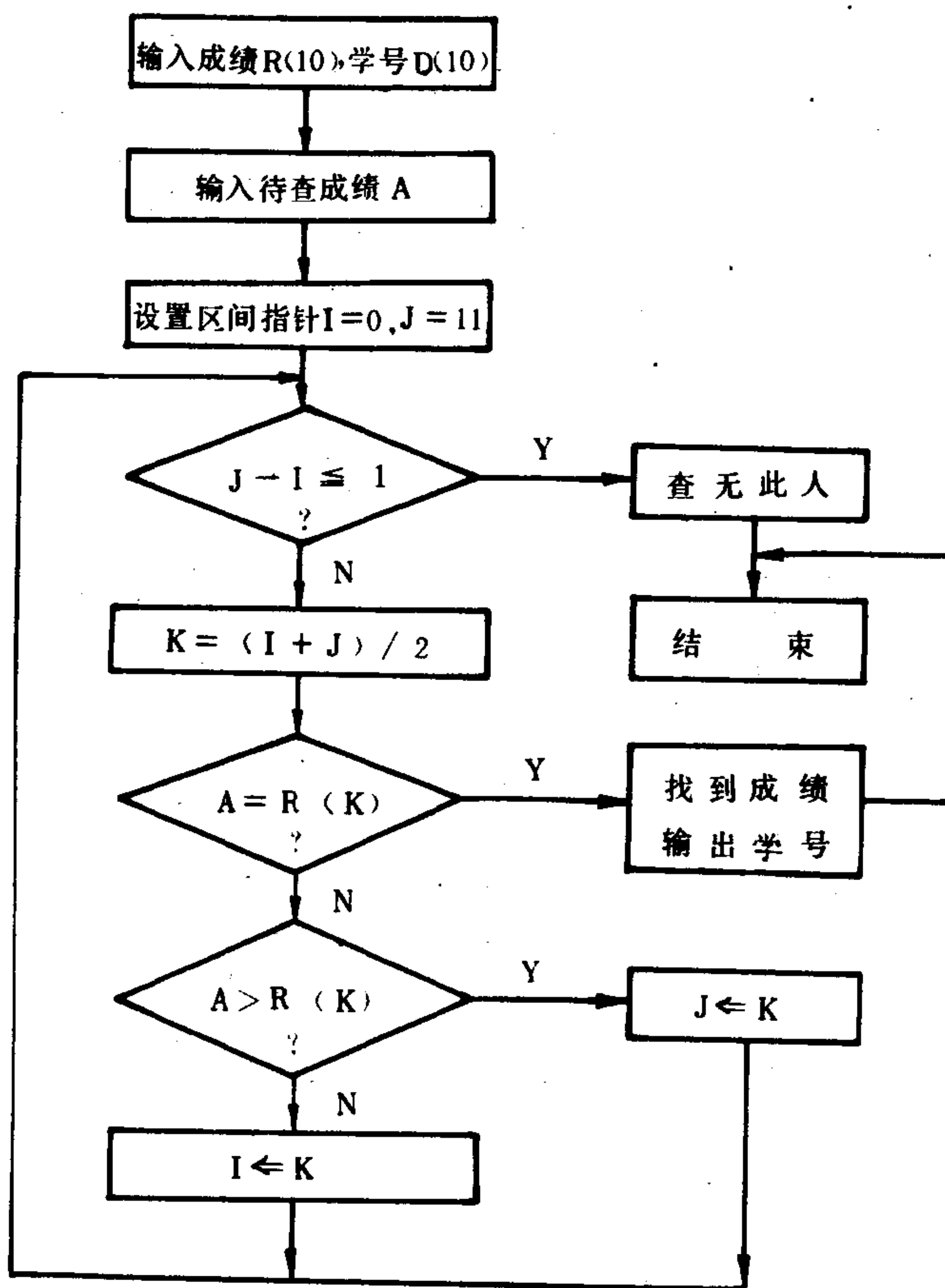


图 10

然后,再用新查找区域的中间数 $R(K)$ 与 A 比较。当 $I = J$ 时,表明查找区域没有了,全部搜索完毕。

为帮助理解上述设计思想,今从一个具体实例出发。已知有10个学生,学号和成绩放在 DATA 区中,成绩互异各不相等,且已按高分到低分顺序排好。今输入一个成绩,要求查找对应这个成绩的学生学号和成绩,如果没有,则显示“NOT FOUND”的信息。

变量设置: $R(10)$ ——成绩,
 $D(10)$ ——学号,
 A ——待查找的成绩,
 I, J ——待查区间的指针。

框图见图10。

下面是程序 PRO-84 清单和结果:

PRO-84

```
5  REM PRO-84
10  DIM R(10),D(10)
20  FOR I = 1 TO 10
30  READ R(I),D(I)
40  NEXT
50  INPUT A
55  IF A = -1 THEN 155
60  I = 0:J = 11
70  IF J - I <= 1 THEN 140
80  K = INT ((J + I) / 2)
90  IF A = R(K) THEN 120
100 IF A > R(K) THEN J = K: GOTO 70
110 I = K: GOTO 70
120 PRINT A; ":"; "FIND"; " "; D(K)
130 GOTO 150
140 PRINT A; ":"; "NOT FIND"
150 RESTORE : GOTO 20
155 END
160 DATA 98,104,94,111,90,103,89,
```

1133,87,121,85,117,80,120,72,
115,66,108,50,110

```
GRUN
?50
50:FIND 110
?94
94:FIND 111
?85
85:FIND 117
?66
66:FIND 108
?79
79:NOT FIND
?-1
```

为加深对二分搜索快速查找特点的了解，从键盘输入一个成绩如50分，看需要查找几次就能得到我们要求的结果。

运行上述程序就可以看到：

第一次：K = 5, 指向 R (5), 找到 87;

第二次：K = 8, 指向 R (8), 找到 72;

第三次：K = 9, 指向 R (9), 找到 66;

第四次：K = 10, 指向 R (10), 找到 50。

因此，只需要查找 4 次，即可查到我们需要检索的關鍵字 50。如果用线性查找方法，则需要 10 次。而当信息量很大时，二分查找的优越性更为显著和突出。

从上述实例，我们还看到：输入 50，每次搜索总是从逐步接近 50 的那一部份区域再找，每搜索一次，查找范围都缩小一半。而所有这一切，都是由于巧妙地安排了指针的移动。

二分查找同样适用于字符串查找。下面给出一个实例，见 PRO-85。注意 DATA 区中字符串 DUA, FUA, ……，

ZCK已经按序排好。340—400句为线性搜索，10—220句为对分搜索。

PRO-85

```
5  REM  PRO-85
10  REM  BINERAY SEARCH PROGRAM
20  DIM A$(30),B$(30),C$(30),D$(30)
   )
30  PRINT "INPUT KEY OF RECORD"
32  FOR I = 1 TO 10
34  READ A$(I),B$(I),C$(I),D$(I)
35  PRINT A$(I),B$(I),C$(I),D$(I)
36  NEXT I
40  INPUT K$
50  IF K$ = "NO" THEN 230
70  I = 0
80  J = 11
85  IF J - I < = 1 THEN 210
90  K = INT ((I + J) / 2)
150 IF K$ = A$(K) THEN 180
160 IF K$ < A$(K) THEN LET J = K
   : GOTO 85
170 I = K: GOTO 85
180 PRINT A$(K),B$(K),C$(K),D$(K)

190 GOTO 40
210 PRINT K$;"NOT FOUND"
220 GOTO 40
230 GOTO 340
240 DATA DUA,M,45,T
250 DATA GUA,W,30,W
260 DATA LAN,W,37,T

270 DATA LIL,W,37,T
280 DATA LIN,M,29,W
290 DATA MID,W,31,T
300 DATA PAN,M,21,M
310 DATA WAR,W,60,T
320 DATA XET,W,50,W
330 DATA ZCK,M,25,T
340 INPUT "#";K$
350 FOR I = 1 TO 10
```

```

360 IF A$(I) = K$ THEN PRINT A$(
      I); " "; B$(I); " "; C$(I); " "; D$
      (I): GOTO 390
370 NEXT I
380 PRINT K$; " NOT FOUND"
390 RESTORE
400 END

```

GRUN

INPUT KEY OF RECORD

| | | |
|--------------|---|----|
| DUA | M | 45 |
| T | | |
| GUA | W | 30 |
| W | | |
| LAN | W | 37 |
| T | | |
| LIL | W | 37 |
| T | | |
| LIN | M | 29 |
| W | | |
| MID | W | 31 |
| T | | |
| PAN | M | 21 |
| M | | |
| WAR | W | 60 |
| T | | |
| XET | W | 50 |
| W | | |
| ZCK | M | 25 |
| T | | |
| ?GUA | | |
| GUA | W | 30 |
| W | | |
| ?LIL | | |
| LIL | W | 37 |
| T | | |
| ?MID | | |
| MID | W | 31 |
| T | | |
| ?ZCJ | | |
| ZCJNOT FOUND | | |
| ?NO | | |
| #ZCK | | |
| ZCK M 25 T | | |

9. 堆栈技术

堆栈 (STACK) 是一种数据结构。它的特点是数据按照后进先出 (LAST IN FIRST OUT) 的原则工作。这是因为堆栈是一个只可以从它的一端加进和取出数据项的部件。既然从一头存取信息, 所以先加进去的数据必然后出来, 而后加进去的数据则先出来。

堆栈有所谓栈顶和栈底之分。堆栈中可以进出数据的一端称之为栈顶, 而另一端则叫做栈底。

在栈顶仅可以执行两种操作: 压入和弹出。压入 (PUSH) 是将一个数据项加到栈顶, 弹出 (POP) 是从栈的顶端移出一个数据项。移出一个数据项后, 原来与栈顶相邻的数据项, 将弹上来变为栈顶。所以栈顶是可以“移动”的, 而栈底则是固定不变的。

在计算机, 把内存的一个区域作为堆栈。所以, 实质上堆栈是一个按照后进先出原则组织的一段内存区域。

为了表明堆栈的位置, 常常用一个“指针”来指示, 指针相当于地址。在 BASIC 语言中, 没有指针这一类变量类型 (在 Z-80 中指针是用 SP 来表示的), 而是用一维数组来实现。元素的下标指示着元素在数组中的位置。代表下标值的变量和指针等效。所以, 堆栈又是一个由指针控制下标的特殊数组。

用 BASIC 语言表示堆栈的办法, 就是把堆栈元素存于数组之中, 相邻的堆栈元素占有相邻的存储单元。

“指针”指向栈顶, 每当在堆栈上进行压入或弹出操作时 (在 Z-80 中就是堆栈操作指令: 压入堆栈指令 PUSH, 弹出堆栈指令 POP), 指针将相应地“移动”。

应该注意的是,堆栈不能超出数组之外,换句话说,堆栈的大小,应受到所说明的数组大小决定。当栈已满仍要向栈中加入新的元素时,则发生上溢错误;同样,当栈已空却要从栈中删除元素,则发生下溢错误。这种情况类似栈房取存货物一样。这两点在栈的操作时应给予充分注意。

为更好理解上述概念,我们用大家比较熟悉的 BASIC 语言,编一个程序。程序结构这样安排:

用一个主程序控制三个子程序。

第一个子程序是建立空堆栈(又称空栈),即让堆栈指针(S T A C K P O I N T E R)具有数组的最大下标值。

第二个子程序是将元素压入堆栈(P U S H E L E M E N T O N T O S T A C K)。

第三个子程序是将元素从堆栈中弹出(P O P E L E M E N T F O R M S T A C K)。

程序中变量设置:

S (5)——5个元素的数组作堆栈用;

P ——堆栈指针,空栈时 $P = 5$;

U ——压入堆栈的变量;

D ——从堆栈弹出的变量;

F ——成功/失败的标志。 $F = 1$ 成功 (S U C C E S S),
 $F = 0$ 失败 (F A I L U R E)。

程序按模块化结构编制。10—60句主程序,作总控用。
100—120句为建立空栈模块。200—290句是将元素压入堆栈的模块。300—380句为弹出栈顶元素模块。

下面分段说明各模块的作用。

(1) 总控:


```

5  DIM S(5)
10 REM  STACK MANIPULATION
20  INPUT "X= ?";X: PRINT
30  IF X = - 1 THEN 60
40  ON X GOSUB 100,200,300
50  GOTO 10
60  END

```

X = -1, 为终止标志。X = 1, 2, 3, 分别转入 100, 200, 300 开始的三个子程序模块。并可自动返回到总控。

(2) 建立空栈

```

100 P = 5
120 RETURN

```

开了 5 个元素的数组, 作堆栈用。指针是 5。

(3) 将元素压入堆栈

```

200 REM PUSH ELEMENT ONTO STACK
210 INPUT "U";U: PRINT "U=";U
220 IF P >= 1 THEN 250
230 F = 0: PRINT "PUSH MANILPULATI
    ON FAILURE"
240 RETURN
250 S(P) = U
260 PRINT "S(";P;")=";S(P)
270 P = P - 1
280 F = 1: PRINT "SUCCESS"
290 RETURN

```

U 为压入堆栈的变量, 即送数至堆栈。

$P < 1$, 则堆栈已满。

第一次压入的元素放在 S (5) 中, 并将堆栈指针减 1。

如果堆栈已满, 则指出操作失败并返回总控。反之, 将给定元素插入堆栈指针 P 所指位置的存贮单元, 将指针减 1, 指出压入操作成功并返回总控。

(4) 弹出栈顶元素

```

300 REM POP ELEMENT FROM STACK
310 IF P < 5 THEN 340
320 F = 0
325 PRINT "POP MANIPULATION FAILURE"
330 RETURN
340 P = P + 1
350 D = S(P)
360 PRINT "S(";P;")=";D
370 F = 1: PRINT "SUCCESS"
380 RETURN

```

如果堆栈是空的，即 $P > 5$ ，则指出操作失败，并返回总控。

反之，将堆栈指针加 1。将堆栈指针所指元素送入 D 中暂存后打印输出。指出弹出操作成功并返回总控。

下面给出完整的程序 PRO-86 清单及运行结果：

PRO-86

ULIST

```

2 REM PRO-86
5 DIM S(5)
10 REM STACK MANIPULATION
20 INPUT "X= ?";X: PRINT
30 IF X = - 1 THEN 60
40 ON X GOSUB 100,200,300
50 GOTO 10
60 END
100 P = 5
120 RETURN
200 REM PUSH ELEMENT ONTO STACK
210 INPUT "U";U: PRINT "U=";U
220 IF P > = 5 THEN 250
230 F = 0: PRINT "PUSH MANIPULATION FAILURE"
240 RETURN
250 S(P) = U
260 PRINT "S(";P;")=";S(P)
270 P = P + 1
280 F = 1: PRINT "SUCCESS"

```

```

290 RETURN
300 REM POP ELEMENT FROM STACK
310 IF P < 5 THEN 340
320 F = 0
325 PRINT "POP MANIPULATION FAILURE"
330 RETURN
340 P = P + 1
350 D = S(P)
360 PRINT "S(";P;")=";D
370 F = 1: PRINT "SUCCESS"
380 RETURN

```

```

URUN
X= ?1

```

```

X= ?2

```

```

U100
U=100
S(5)=100
SUCCESS
X= ?2

```

```

U200
U=200
S(4)=200
SUCCESS
X= ?2

```

```

U300
U=300
S(3)=300
SUCCESS
X= ?2

```

```

U400
U=400
S(2)=400
SUCCESS
X= ?2

```

```

U500
U=500

```

```
S(1)=500
SUCCESS
X= 72
U999
U=999
PUSH MANIPULATION FAILURE
X= 73

S(1)=500
SUCCESS
X= 73

S(2)=400
SUCCESS
X= 73

S(3)=300
SUCCESS
X= 73

S(4)=200
SUCCESS
X= 73

S(5)=100
SUCCESS
X= 73

POP MANIPULATION FAILURE
X= 7-1
```

说明：在压入操作时，第一次数100存入S (5)中，第二次数200存入S (4)中，……，第五次数500存入S (1)中，栈顶在上移。第六次数1000，无法压入堆栈，因为只有五个元素的存贮空间，堆栈已满，压入操作失败。

在弹出操作时，第一次弹出堆栈的是S (1)=500，第二次弹出是S (2)=400，……，第五次弹出的是S (5)=100，充分说明先进后出。当堆栈已空时，再要弹出元素（数据），将失败。

以上程序可以帮助我们了解堆栈结构和堆栈操作。了解这些知识，将有助于我们理解堆栈在数据处理中的应用。

顺及说明的是，BASIC 程序中对于循环语句的处理，是通过循环栈的辅助作用来实现的。同样，对子程序的处理，也是通过子程序栈的作用控制的。可见“栈”是计算机软件中一个非常重要的概念。它是深入学习循环语句和子程序结构的重要基础。

下面给出汉字注释的“堆栈技术”程序 PRO-86-1 和运行结果的详细说明。

PRO-86-1

```
3 REM PRO-86-1
5 DIM S(5)
10 REM 堆栈技术
15 REM 20-60总控
20 INPUT "请输入1-建立空栈,2-压入堆栈,3-弹出堆栈,-1-结束";X
25 PRINT
30 IF X = - 1 THEN 60
40 ON X GOSUB 100,200,300
50 GOTO 10
60 END
100 P = 5
110 PRINT "已建立5个空栈"
115 PRINT
120 RETURN
200 REM 将元素压入堆栈
210 INPUT "输入元素U:";U: PRINT "U=";U
220 IF P >= 5 THEN 250
230 F = 0: PRINT "堆栈已满"
235 PRINT
```

```

240 RETURN
250 S(P) = U
260 PRINT "S(";P;")=";S(P)
270 P = P - 1
280 F = 1: PRINT "成功"
285 PRINT
290 RETURN
300 REM 将元素弹出堆栈
310 IF P < 5 THEN 340
320 F = 0
325 PRINT "堆栈已空"
328 PRINT
330 RETURN
340 P = P + 1
350 D = S(P)
360 PRINT "S(";P;")=";D
370 F = 1: PRINT "成功"
375 PRINT
380 RETURN

```

IRUN

请输入1-建立空栈,2-压入堆栈,3-弹出堆栈,-1-结束1

已建立5个空栈

请输入1-建立空栈,2-压入堆栈,3-弹出堆栈,-1-结束2

输入元素U:500

U=500

S(5)=500

成功

请输入1-建立空栈,2-压入堆栈,3-弹出堆栈,-1-结束2

输入元素U:400

U=400

S(4)=400

成功

请输入1-建立空栈,2-压入堆栈,3-弹出堆栈,-1-结束2

输入元素U:300

U=300

S(3)=300

成功

请输入1-建立空栈,2-压入堆栈,3-弹出堆栈,-1-结束2

输入元素U:200

U=200

S(2)=200

成功

请输入1-建立空栈,2-压入堆栈,3-弹出堆栈,-1-结束2

输入元素U:100

U=100

S(1)=100

成功

请输入1-建立空栈,2-压入堆栈,3-弹出堆栈,-1-结束2

输入元素U:999

U=999

堆栈已满

请输入1-建立空栈,2-压入堆栈,3-弹出堆栈,-1-结束3

S(1)=100

成功

请输入1-建立空栈,2-压入堆栈,3-弹出堆栈,-1-结束3

S(2)=200

成功

请输入1-建立空栈,2-压入堆栈,3-弹出堆栈,-1-结束3

S(3)=300

成功

请输入1-建立空栈,2-压入堆栈,3-弹出堆栈,-1-结束3

S(4)=400

成功

请输入1-建立空栈,2-压入堆栈,3-弹出堆栈,-1-结束3

S(5)=500

成功

请输入1-建立空栈,2-压入堆栈,3-弹出堆栈,-1-结束3

堆栈已空

请输入1-建立空栈,2-压入堆栈,3-弹出堆栈,-1-结束-1

10. 链接技术

信息处理中常常用到数据结构知识。我们在这一节主要介绍数据结构中常用的一种技术——链接表技术。

链接表是信息存贮的一个重要手段。链接表与一般按自然数列的存贮结构不同,它对每一并列的信息存贮单元都附加一个或几个链指针。其存贮结构如图11所示。

如果只有一个链指针,就称为单链接表。单链接表的每

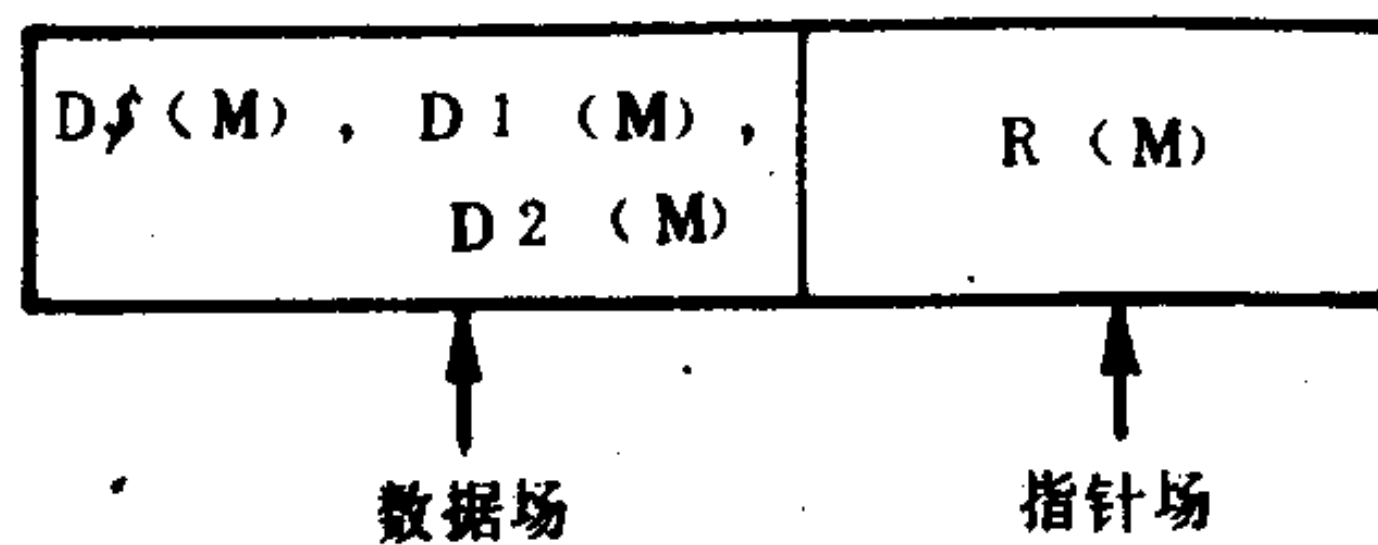


图 11

个结点都有且只有一个后继结点，它的后继结点是由该结点的指针场指出的。单链接表的最后一个指针场为“0”标志表的结束。它需要一个指针指示单链接表的首地址（例如用 A 表示）。

为了理解单链接表（其它还有循环链接表，双重链接表）的概念，结构和使用方法，我们不妨先看一个具体例子。

例1，输入一个整数序列 $b_1, b_2, b_3, \dots, b_n$ ，该序列的数在 1 到 999 之间，而当输入为“0”时，标志结束。试编写一个程序，按递增次序存放成一个线性链接结构。

解：按次序输入一个整数序列 M_i 中的各个数，在输入过程中每当读入下一个数 b_{i+1} 后，通过将 b_{i+1} “插入”到已构成的单链接表的适当位置上，最后生成一个按递增次序存放的线性链接结构。

开始时，设置二个结点：见图12。

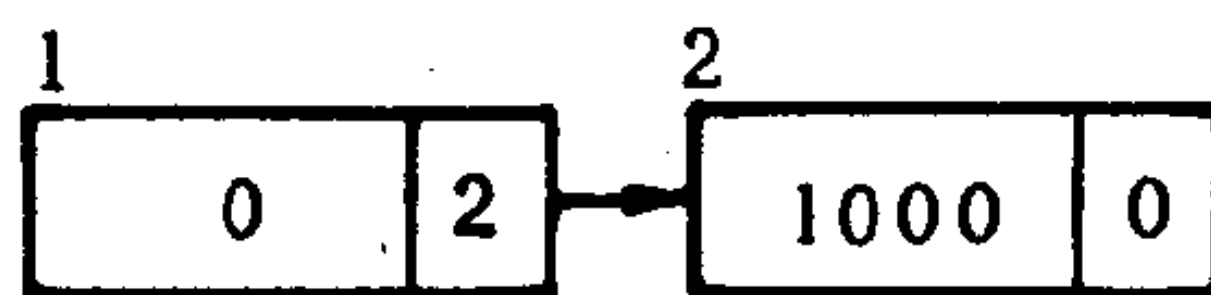


图 12

这个链接表的首指针是 1，指向数据场的 0，所对应的链指针为 2（称为结点 1 的后继结点），同时这个链指针 2 指向下一个存贮单元，即指向 2 号结点的数据场的有关信息（本

例是1000)。

也就是说，通过链指针把若干结点的数据（或信息），像链条一样一节一节地串起来。

我们题目的意思是，输入若干个数，使它们插入 0 和 1000 之间。

例如，假设给出整数数列为11, 3, 9, 0, 则插入过程中链接分配情况见下面图例：

存贮设置：

$D(M)$ ——存放整数序列中的数据；

$R(M)$ ——存放后继结点地址的指针；

N ——新结点的下标地址；

I ——新结点的值；

M ——最多可存放结点的个数；

P ——辅助变量。

初态见图13： $D(1) = 0, R(1) = 2, D(2) = 1000, R(2) = 0, N = 3$ （下次结点的地址）

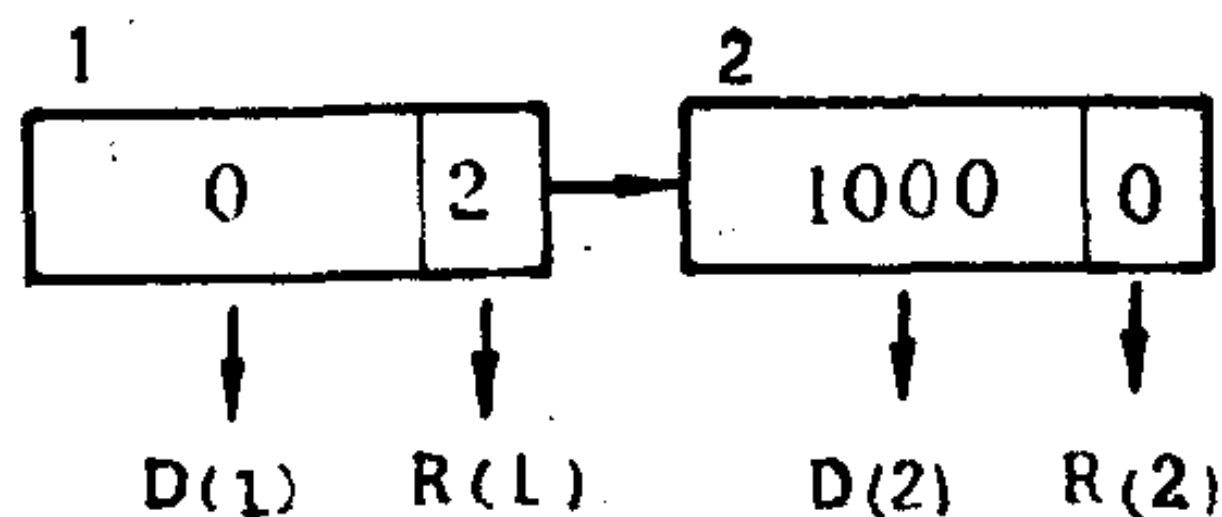


图 13

输入 11 后，见图 14。

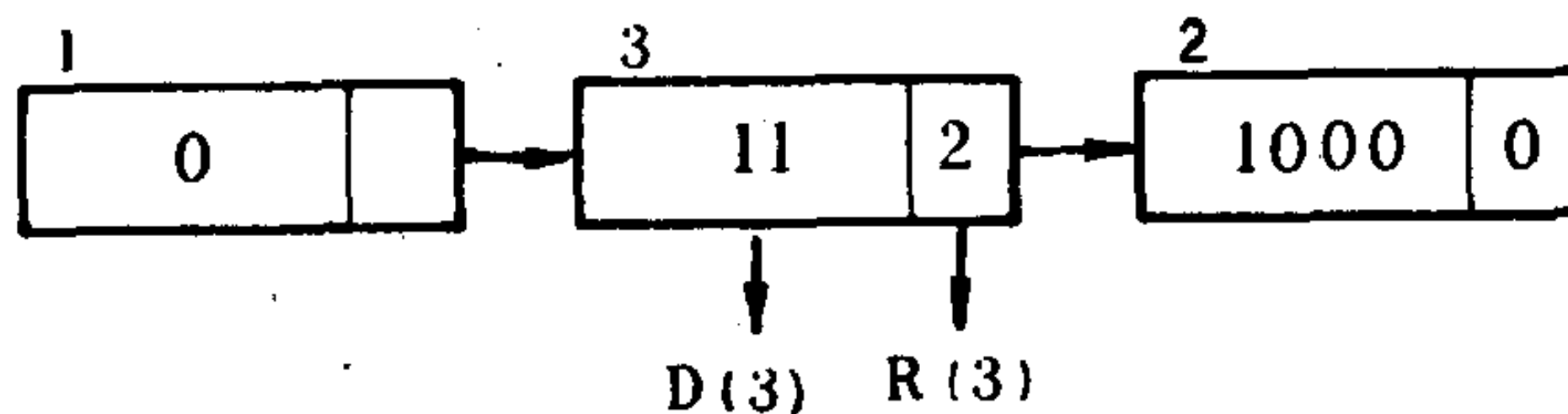


图 14

输入 3 后, 见图15。

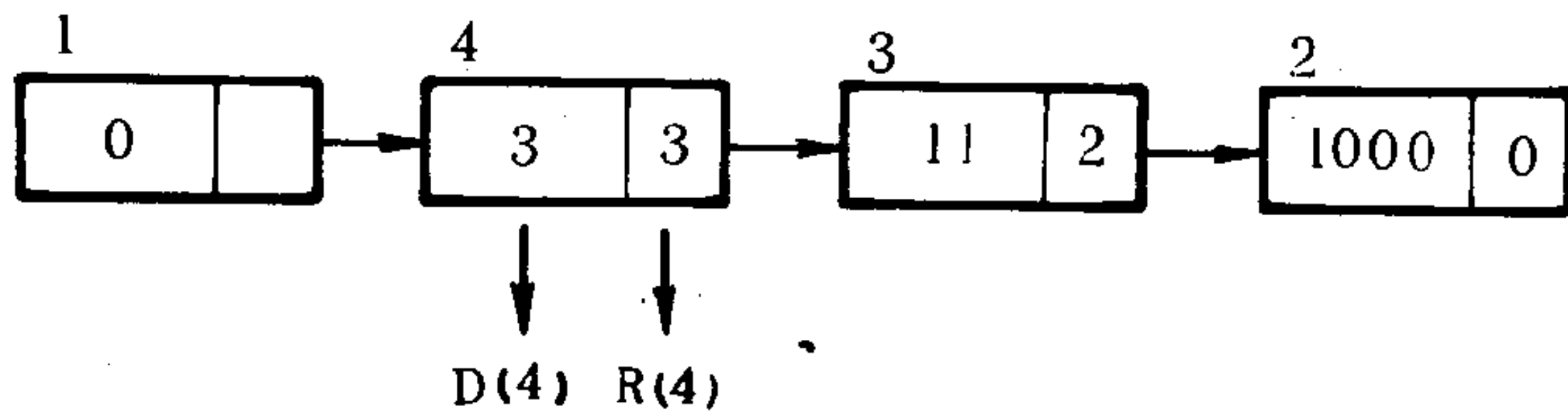


图 15

输入 19 后, 见图16。

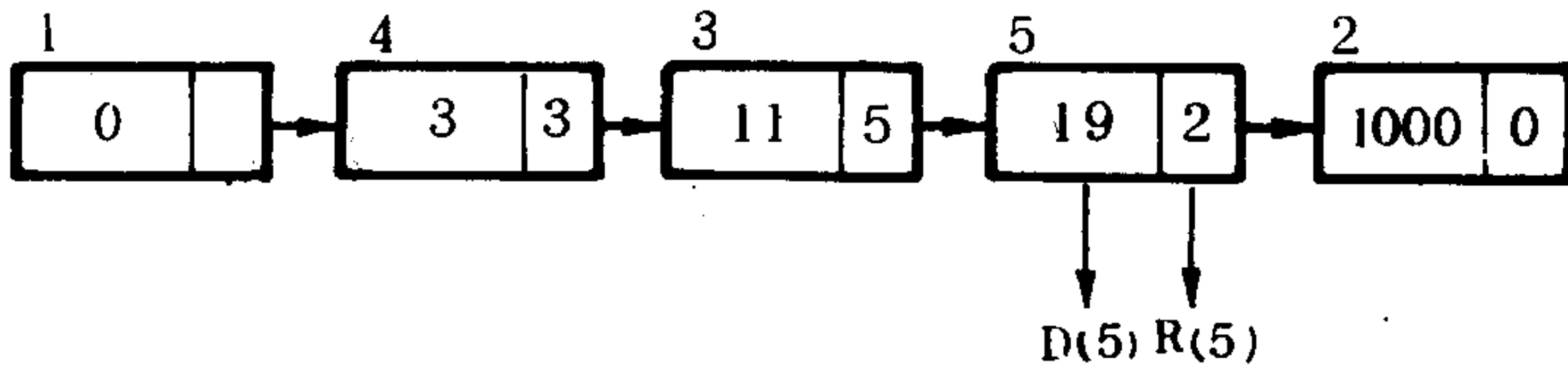


图 16

输入 0 后, 结束。

我们要编制的程序, 必须像上述图例顺次工作, 程序的流程图也几乎和上述图例的流程一致。

为此, 先画出程序的流程图 (见图17), 然后对照框图写出程度, 并逐个输入数据, 看程序的执行过程。从而加深对链接表结构、使用方法、概念和实用性的理解。并逐步体会它在信息处理中的重要性, 以便解决更为复杂的问题。

相应的程序PRO-87及运行结果如下。

PRO-87

ULIST

```
5  REM PRO-87
10  DIM D(10),R(10)
15  D(1) = 0:R(1) = 2:D(2) = 1000:R
    (2) = 0:N = 3:M = 10
20  INPUT "ENTER THE VALUE OF NEW
```

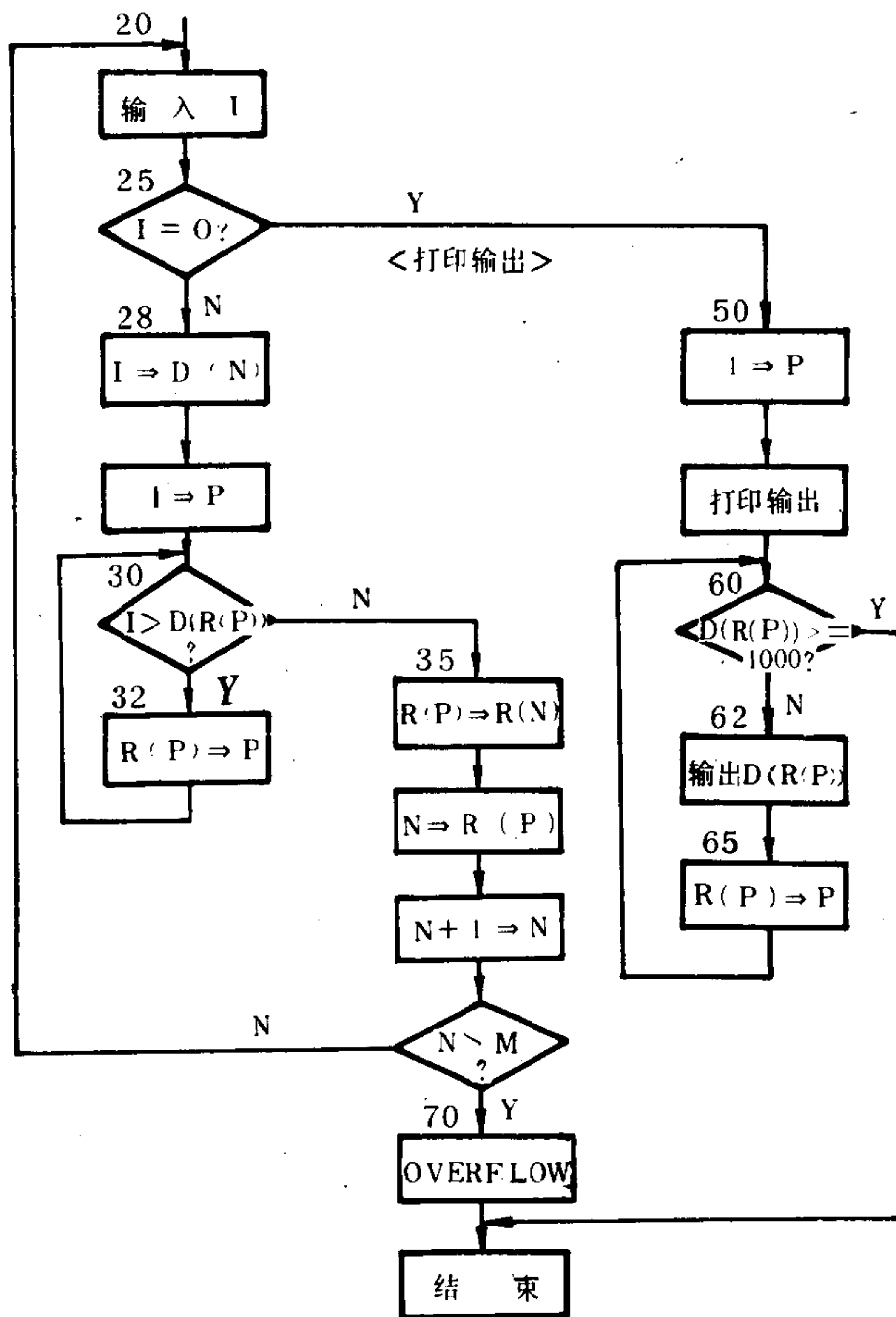


图 17

```

      /
      ";I: PRINT
25  IF I = 0 THEN 50
28  D(N) = I:P = 1
30  IF I < = D(R(P)) THEN 35
32  P = R(P): GOTO 30
35  R(N) = R(P):R(P) = N:N = N + 1:
    IF N > M THEN 70
40  GOTO 20
50  P = 1: PRINT : PRINT "OUTPUT:";

60  IF D(R(P)) = 1000 THEN 80
62  PRINT D(R(P));" ";
65  P = R(P)
68  GOTO 60
70  PRINT "OVERFLOW"
80  PRINT : END

```

```

ORUN
ENTER THE VALUE OF NEW          11

ENTER THE VALUE OF NEW          3

ENTER THE VALUE OF NEW          19

ENTER THE VALUE OF NEW          0

```

```

OUTPUT:3 11 19

```

现在，我们来剖析一下程序中主要语句段的安排：

开始时，结点1, 2的存贮情况如图18所示。

即结点1数据场中的值为0，它存放在D(1)中，指针场的链指针为2，存贮在R(1)中；结点2的数据场中存贮的数值是1000，它存贮在D(2)中，指针场的链指针是0，它存在R(2)中。

为了插入一个数据I，例如I = 11，它应在数值0和1000之间，即比0大而比1000小（这是因为题意要求按递增排列存放），所以I必须安排在结点1和2之间。为此，必须插入

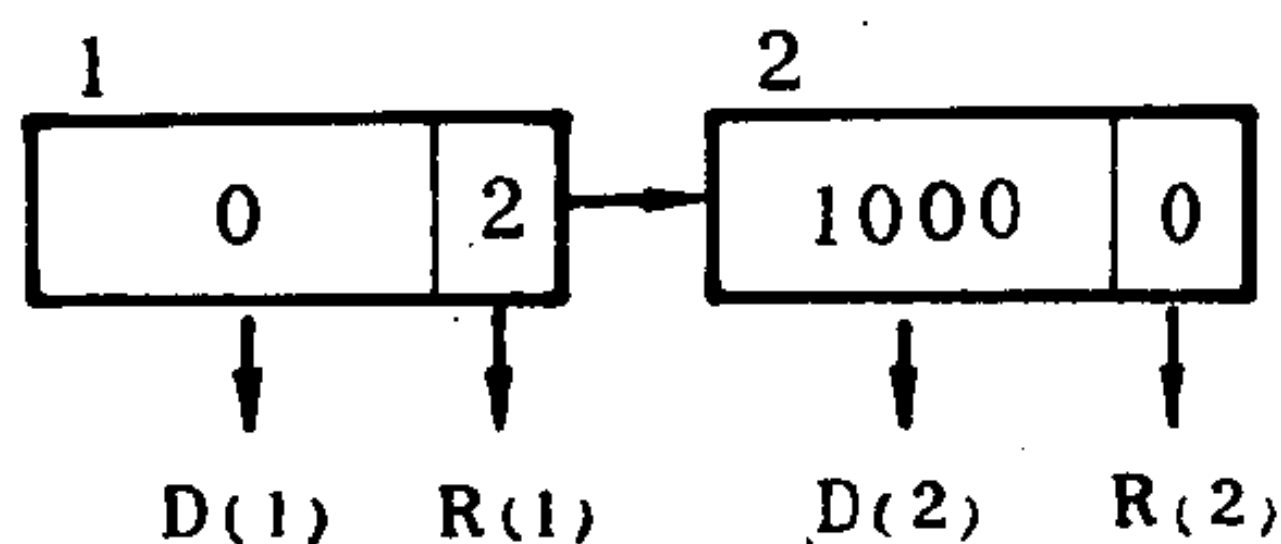


图 18

一个新结点，其地址为 N （如我们在初态中已经安排的， $N=3$ ），新结点的值就是 I ，它等于 11。上述想法，在程序上如何处理呢？

15句，定新结点的下标地址 $N=3$ ；

20句，输入新结点的值 $I=11$ ；

25句，因为 $I \neq 0$ （ $I=0$ ，是结束标志），所以转向 28 句。

执行 28 句，把新结点的 $I=11$ ，赋给 $D(N)=D(3)$ ，所以 $D(3)$ 中存贮了数值 11，这就是新结点 3 的数据场中的数值。

让 1 赋给 P 。

经 30 句判断，由于 $D(R(P))=D(R(1))=D(2)=1000$ ，所以 $11 < 1000$ ，而转向 35 句。30 句的作用是判别输入数 I 是在那二个结点的数据场的数据之间，以便正确按递增数列存放。

执行 35 句，让 $R(P)=R(1)$ 的值赋给 $R(N)$ ，因为 $R(1)=2$ （原来第一个结点的链指针号），所以 $R(N)=R(3)=2$ 。这就是说，新结点的链指针为 2。然后又让 N 赋给 $R(P)$ ，即 $R(1)=3$ 。

经过上面各语句处理后，存贮情况如下： $D(1)=0$ ， $R(1)=3$ ， $D(2)=1000$ ， $R(2)=0$ ， $D(3)=11$ ， $R(3)=2$ 。这样，新结点 3 和原结点 1, 2 就通过链指针把三个数据连

在一起了。

最后 $N = N + 1 = 4$, 作用是为下一个数据输入准备好新结点的地址。而条件判断 N 是否大于 M , 是为了不超过 $M = 10$ 的最多插入 8 个数的约定 (若要插入更多的数, 则 M 可以加大, 且应多开存贮单元)。

至此, 1, 2, 3 三个结点的存贮形式如图19所示。

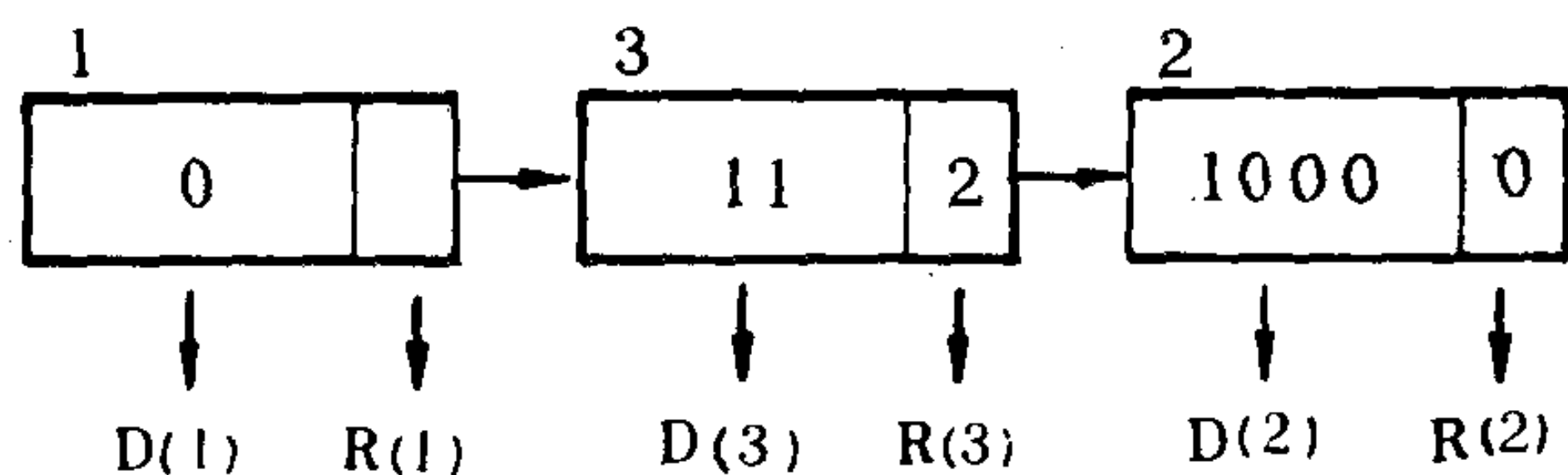


图 19

下面继续分析。

又执行20句, 输入新结点值 I , 如 $I = 3$, 此时, 它应在 0 和 11 之间, 而不是在数据 11 和 1000 之间。那么, 程序又是如何处理呢?

执行28句, $D(N) = D(4) = 3$, 即新结点的数值 3 放入了新结点 4 中的数据场, 让 $1 \Rightarrow P$ 。

执行30句, 此时 $D(R(P)) = D(R(1)) = D(3) = 11$, 所以 $I = 3 < D(R(P)) = 11$, 而转向35句。

执行35句, $R(P) = R(1) = 3 \Rightarrow R(N) = R(4)$, 所以 $R(4) = 3$, 这就是新结点 4 的链指针为 3, 然后把 $N = 4 \Rightarrow R(P) = R(1) = 4$ 。而原来 $D(3) = 11$, $R(3) = 2$, 不变。至此, 结点 1, 4, 3 之间的连接完成。它们由链指针 $R(1) = 4$, $R(4) = 3$, $R(3) = 2$ 象链条一样一节一节地串了起来。

结点 4 的确插入结点 1 和 3 之间, 而数据 3 也插入 0 和 11 之间。其存贮方式见图20。

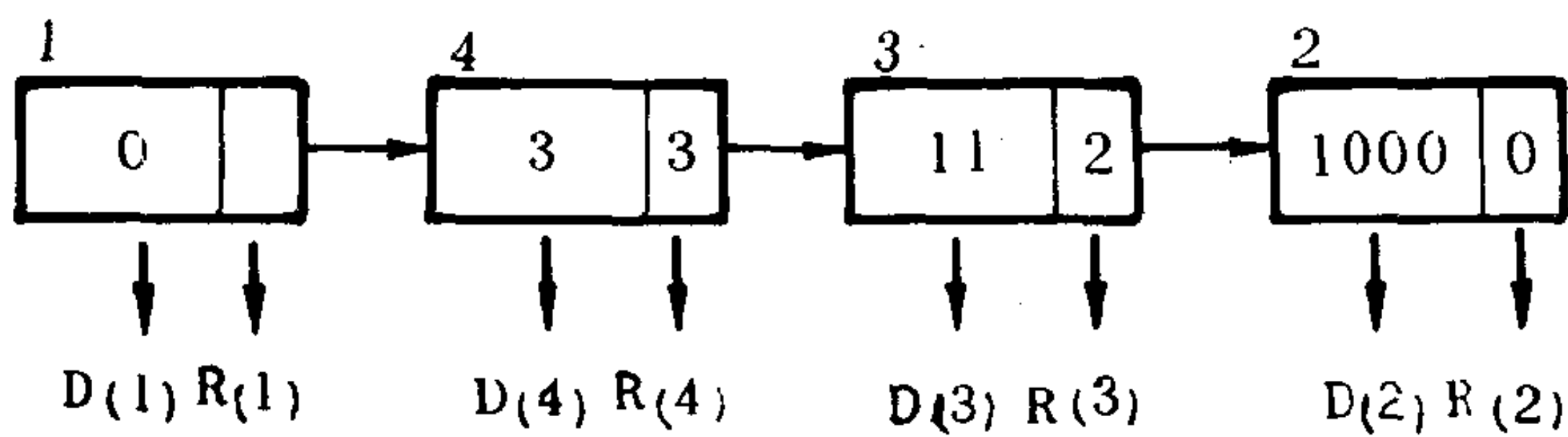


图 20

以后的分析同上过程。输入 $I = 0$ ，则打印排好序的数并结束。

上面这个简单的例子，说明了链表的概念和简单应用。事实上，链表的作用不仅如此，它可以处理多种信息的加工，在程序设计和数据处理中有着重要的应用。下面，我们再举一个稍为复杂的例子。

例 2，现有 10 种货物的名称、数量、编号的明细表一张（见表 1），要求对表中数据完成如下 4 种操作：

表 4.1 明 细 表

| 名 称 | 数 量 | 编 号 |
|----------|-----|-----|
| MACHINE | 98 | 4 |
| MOTOK | 96 | 7 |
| BLOCK | 94 | 13 |
| PLATE | 89 | 10 |
| FRAME | 84 | 6 |
| FITTER | 82 | 14 |
| TANK | 75 | 1 |
| BRAKES | 70 | 5 |
| IGNITION | 63 | 9 |
| HOUSING | 58 | 0 |

① 以编号为链指针，按数量多少从大到小用链表存储；

- ② 打印这一链接表;
- ③ 根据编号查找相应货物名称和数量;
- ④ 修改某一编号的货物名称及数量。

解: 由于题意要求以编号为链指针, 按货物数量多少从大到小排列, 则链接后的形式如图21所示。

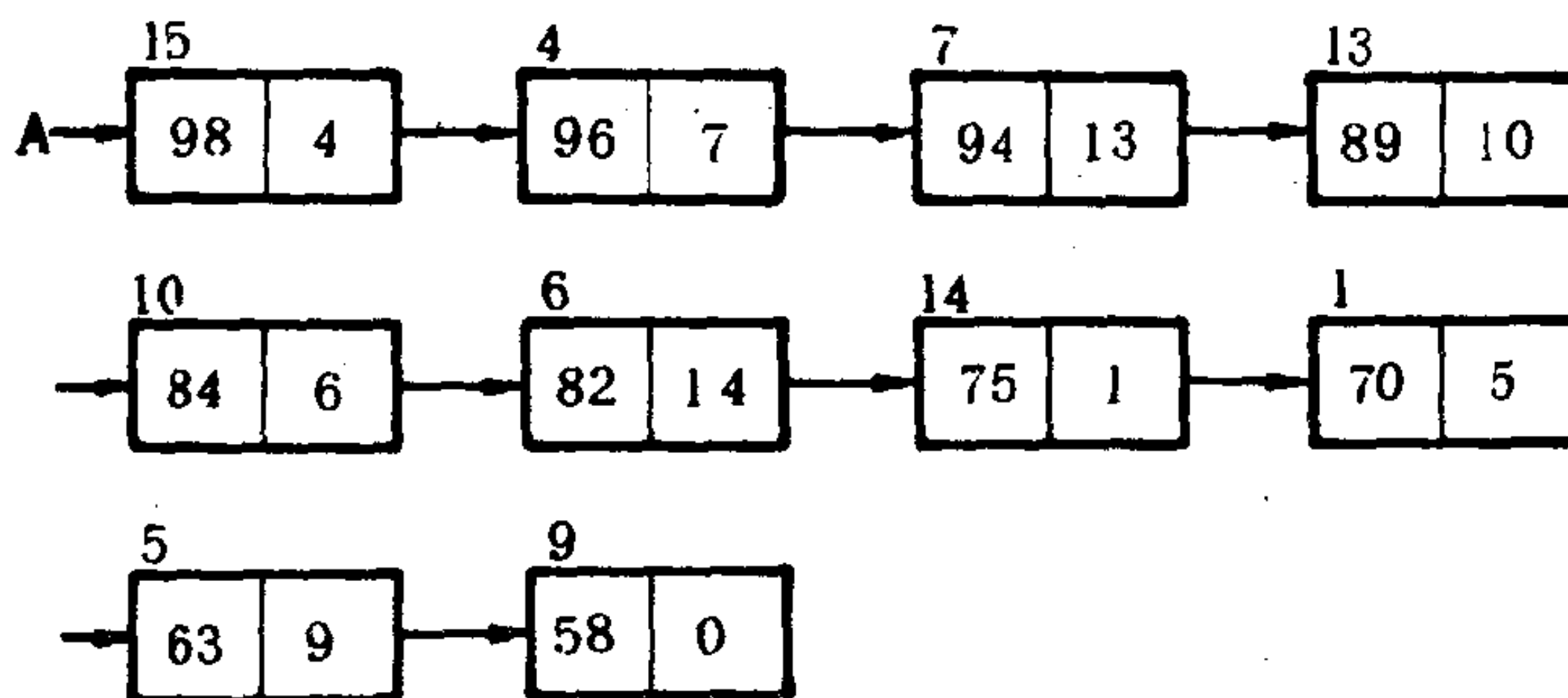


图 21

这个链接表的首指针是 A, 它指示线性表的首地址为 15。链接表中的数据场是货物的数量, 而链接表中的指针场是货物的编号。

存贮说明:

$B(M)$ ——存贮货物数量;

A(M)$ ——存贮货物名称;

$R(M)$ ——存贮货物编号 (即存放链指针)。

例如, 首指针是 15 时, $B(15) = 98$, A(15) = MACHINE$, $R(15) = 4$ 。所以编号 (即链指针) 兼作数组 B 和 A(M)$ 的下标。 $R(15) = 4$, 则指向下一个存贮单元, 即编号为 4 的货物有关信息; $R(4) = 7$, 再指向编号为 7 的货物有关信息。通过链指针 $R(M)$, 就把货物的信息 (数量、货名、编号) 串接起来。这就是我们上面介绍的单链接表的概念。注意链接表的首指针是给定的, 而链指针为 0 时, 则表示链结束。

题目要求完成四种操作（实际上可以更多），第一种操作是至关重要的，它是完成其它几种操作的基础和前提。为此，先编制第一种操作的程序EXERCISES-(2)。

```

5  REM  EXERCISES-(2)
10  DIM B(20),A$(20),R(20)
20  A = 15:P = A
30  READ B(P),A$(P),R(P)
35  PRINT "B(";P;")=";B(P);"
      ;"A$(";P;")=";A$(P);"
      R(";P;")=";R(P)
40  P = R(P)
50  IF P < > 0 THEN 30
60  STOP

```

对应的框图如图22。

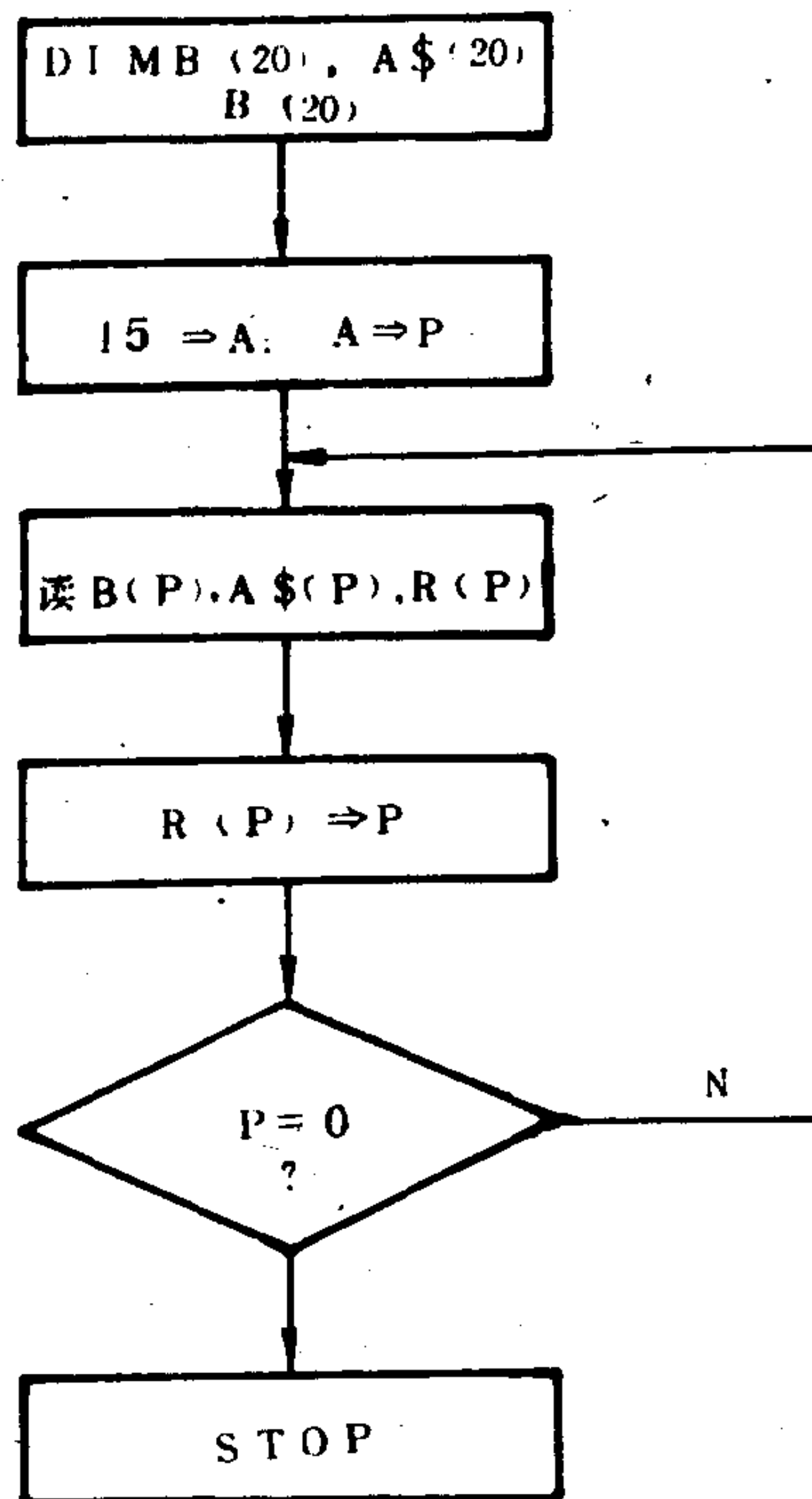


图 22

为了说明程序EXERCISES-(2) 以及将要编制的其它几个功能程序的方便, 不妨先将数据区中的信息, 一并抄录下来。

```

600 DATA 98,MACHINE,4,96,MOTOR,7
    ,94,BLOCK,13,89,PLATE,10,84,F
    FRAME,6
610 DATA 82,FITTEE,14,75,TANK,1,7
    0,BRAKES,5,63,IGNITION,9,58,H
    OUSING,0

```

程序EXERCISES-(2)是这样安排的, 首先建立数组 $B(20)$, $A\$(20)$, $R(20)$, 让它们分别存贮货物的数量、名称和编号。给链接表以首指针 A , 其值为15。再安排一个辅助变量 P , 它实际上指编号, 是当前处理的结点。然后读 $DATA$ 区中的信息 (暂时不看35句), 读好链指针为15的信息 $B(15)$, $A\$(15)$, $R(15)$ 后, 让 $R(15) \Rightarrow P$, 即 $R(15) = 4 \Rightarrow P$, 所以 $P = 4$, 这样就给出了下一个存贮单元的指针 (或者说结点4)。只要 $P \neq 0$, 就再读, 一直循环下去, 直到 $P = 0$ 时为止 (这里用 $STOP$ 暂停语句)。这样, 就完成了链接表形式的存贮。现在, 再看35句, 它的任务是打印出这一链接表, 加35句的作用就在于此, 实际使用时可以删去35句。下面是执行程序 EXERCISES-(2) 的结果:

| | | |
|------------|-------------------|------------|
| $B(15)=98$ | $A\$(15)=MACHINE$ | $R(15)=4$ |
| $B(4)=96$ | $A\$(4)=MOTOR$ | $R(4)=7$ |
| $B(7)=94$ | $A\$(7)=BLOCK$ | $R(7)=13$ |
| $B(13)=89$ | $A\$(13)=PLATE$ | $R(13)=10$ |
| $B(10)=84$ | $A\$(10)=FRAME$ | $R(10)=6$ |
| $B(6)=82$ | $A\$(6)=FITTEE$ | $R(6)=14$ |
| $B(14)=75$ | $A\$(14)=TANK$ | $R(14)=1$ |
| $B(1)=70$ | $A\$(1)=BRAKES$ | $R(1)=5$ |
| $B(5)=63$ | $A\$(5)=IGNITION$ | $R(5)=9$ |
| $B(9)=58$ | $A\$(9)=HOUSING$ | $R(9)=0$ |

注:多开存贮单元,是为插入信息而准备的。安排 STOP 语句,是为调试程序而设置的,上一段程序调试完成后,可以删掉60句的STOP 语句。

现在,再看第二种操作,在程序上如何实现。

前面已经指出,程序EXERCISES-(2)35句的安排已能打印以货物编号为指针的链接表。如果去掉35句,则可以重新安排一个打印链接表的程序段400—440句,这段程序用子程序方式写入。

```
400 P = A
410 PRINT P; TAB( 8);A$(P); TAB(
    20);B(P)
420 P = R(P)
430 IF P < > 0 THEN 410
440 RETURN
```

为了和程序 EXERCISES -(2) (第一种操作链接表存贮) 构成一个整体,在删去35句的同时,应加上调用点,即55 GOSUB 400。

在执行55句以后结果为:

| | | |
|----|----------|----|
| 15 | MACHINE | 98 |
| 4 | MOTOR | 96 |
| 7 | BLOCK | 94 |
| 13 | PLATE | 89 |
| 10 | FRAME | 84 |
| 6 | FITTEE | 82 |
| 14 | TANK | 75 |
| 1 | BRAKES | 70 |
| 5 | IGNITION | 63 |
| 9 | HOUSING | 58 |

这也是一张以货物编号为指针的链接表。其执行过程简叙如下:执行55句,转入400句开始的子程序模块。首先给出首指针 $A \Rightarrow P$, 执行410句打印 P 为15, A(P) = A(15)

为MACHINE的货物名称及 $B(P) = B(15)$ 为货物数量98的信息。执行420句,将 $R(P) = P(15) = 4$ 赋给下一个存储单元P(此时 $P = 4$),只要P不为零,再循环,并打印其它信息。最后 $P = 0$,执行440句,自动返回调子程序段的55句的下一句,60句暂停。

关于本题的第三种操作,是根据编号查找相应货物名称和数量,可以用100—160句的程序段来实现:

```
100 INPUT M
110 P = A
120 IF P = M THEN 160
130 P = R(P)
140 IF P < > 0 THEN 120
150 PRINT "NOT FIND ": GOTO 100
160 PRINT M; TAB(4); A$(M); TAB(
    16); B(M)
```

为执行100—160句查找程序段,可先将60句STOP删去。执行100句后:

?1

5 ↙ (键盘输入5)

5 IGNITION 63

最后,我们来安排第四种操作,修改某一编号货物名称和数量。程序如500—590句安排:

```
500 INPUT M
510 P = A
520 IF P = M THEN 560
530 P = R(P)
540 IF P < > 0 THEN 520
550 GOTO 590
560 INPUT A$(M), B(M)
570 A$(P) = A$(M); B(P) = B(M)
580 GOSUB 400
590 END
```

修改操作的程序段也比较清楚，输入要改的编号如14，执行510句后，会找到编号为14的信息，从而转入560句，此时输入需要修改的内容（即货物名称和数量），并经570句重新赋给原存贮货物名称及数量的A\$(P)，B\$(P)。这样即冲掉了原有的信息，达到修改的目的。然后转向400句打印，最后返回到590句结束。

运行500—590句的结果是：

| | | |
|----|----------|----|
| 15 | MACHINE | 98 |
| 7 | BLOCK | 94 |
| 13 | PLATE | 89 |
| 10 | FRAME | 84 |
| 6 | FITTEE | 82 |
| 14 | APPLE-2 | 72 |
| 18 | PC-1500 | 76 |
| 1 | BRAKES | 70 |
| 5 | IGNITION | 63 |
| 9 | HOUSING | 58 |

结果表明，编号14中的信息TANK 75改为APPLE-2 72。

至此，本题四种操作全部设计完成。

如果我们希望本题能够完成更多的操作，也就是希望程序具有更多的功能，可以在原程序的基础上加上适当的语句段。例如，增加两种新的操作：

(1) 在链接表中插入一个货物的编号，数量和名称，插入后的数量仍然满足从多到少的存贮要求；

(2) 删去某一编号的数量、货名，若无此编号，则提示找不到的信息。

则只要加上以下两个程序段(a)和(b)：

(a) 200—280句完成插入信息的操作。

```

200 INPUT M, A$(M), B(M)
210 P = A
220 IF B(M) > B(P) THEN 240
230 K = P: P = R(P)
235 IF P < > 0 THEN 220
240 R(M) = P
250 IF A = P THEN 270
260 R(K) = M: GOSUB 400
265 GOTO 200
270 A = M: GOSUB 400
280 GOTO 200

```

(b) 300—390句完成删除信息的操作。

```

300 INPUT M
310 P = A
320 IF P = M THEN 360
330 K = P: P = R(P)
340 IF P < > 0 THEN 320
350 PRINT "NOT FIND": GOTO 300
360 IF A = P THEN 380
370 R(K) = R(P): GOSUB 400
375 GOTO 300
380 A = R(P): GOSUB 400
390 GOTO 300

```

最后，把本题的全部操作，编制在一个总的程序里，以加深对链接表总体结构的了解。

各主要程序段说明：

12—18句，打印菜单；

60—80句，安排总控；

35句，完成打印链接表的工作；

20—50句，完成链接表的存贮；

100—180句，完成查找功能；

200—280句，完成插入信息工作；

300—390句，完成删除信息工作；

400—440句，打印链接表；

500—590句，完成修改信息工作。

程序清单及部分运行实例，见PRO-88。

PRO-88

```
5  REM  EXERCISES-(2)
10  DIM B(20),A$(20),R(20)
12  PRINT "1--SEARCHING A RECORD"

14  PRINT "2--ADD A RECORD"
16  PRINT "3--DELETE A RECORD"
18  PRINT "4--CHANGE A RECORD"
19  PRINT
20  A = 15:P = A
30  READ B(P),A$(P),R(P)
35  PRINT "B(";P;")=";B(P);"
      "; "A$(";P;")=";A$(P);"
      "; "R(";P;")=";R(P)
40  P = R(P)
50  IF P < > 0 THEN 30
52  PRINT
55  GOSUB 400
60  INPUT N
70  ON N GOTO 100,200,300,500
80  PRINT "END": END
100 INPUT M
110 P = A
120 IF P = M THEN 160
130 P = R(P)
140 IF P < > 0 THEN 120
150 PRINT "NOT FIND ": GOTO 60
160 PRINT "NOW FIND"
170 PRINT M; TAB( 8);A$(M); TAB(
      20);B(M)
175 PRINT : PRINT "*-*-*-*-*-*-*-*
      -*-*-*-*-*"
180 GOTO 60
200 INPUT M,A$(M),B(M)
210 P = A
220 IF B(M) > B(P) THEN 240
230 K = P:P = R(P)
235 IF P < > 0 THEN 220
240 R(M) = P
```



```

250 IF A = P THEN 270
260 R(K) = M: GOSUB 400
265 GOTO 60
270 A = M: GOSUB 400
280 GOTO 60
300 INPUT M
310 P = A
320 IF P = M THEN 360
330 K = P: P = R(P)
340 IF P < > 0 THEN 320
350 PRINT "NOT FIND": GOTO 60
360 IF A = P THEN 380
370 R(K) = R(P): GOSUB 400
375 GOTO 60
380 A = R(P): GOSUB 400
390 GOTO 60
400 P = A
410 PRINT P: TAB( B): A$(P): TAB(
    20): B(P)
420 P = R(P)
430 IF P < > 0 THEN 410
435 PRINT : PRINT "- - - - -"
    - - - - -
440 RETURN
500 INPUT M
510 P = A
520 IF P = M THEN 560
530 P = R(P)
540 IF P < > 0 THEN 520
550 GOTO 60
560 INPUT A$(M), B(M)
570 A$(P) = A$(M): B(P) = B(M)
580 GOSUB 400
590 GOTO 60
600 DATA 98, MACHINE, 4, 96, MOTOR,
    7, 94, BLOCK, 13, 89, PLATE, 10, 84
    , FRAME, 6
610 DATA 82, FITTEE, 14, 75, TANK, 1,
    70, BRAKES, 5, 63, IGNITION, 9, 58
    , HOUSING, 0

```

ORUN

1--SEARCHING A RECORD

2--ADD A RECORD

3--DELETE A RECORD

4--CHANGE A RECORD

| | | |
|----------|-----------------|----------|
| B(15)=98 | A\$(15)=MACHINE | R(15)=4 |
| B(4)=96 | A\$(4)=MOTOR | R(4)=7 |
| B(7)=94 | A\$(7)=BLOCK | R(7)=13 |
| B(13)=89 | A\$(13)=PLATE | R(13)=10 |
| B(10)=84 | A\$(10)=FRAME | R(10)=6 |
| B(6)=82 | A\$(6)=FITTEE | R(6)=14 |
| B(14)=75 | A\$(14)=TANK | R(14)=1 |
| B(1)=70 | A\$(1)=BRAKES | R(1)=5 |
| B(5)=63 | A\$(5)=IGNITION | R(5)=9 |
| B(9)=58 | A\$(9)=HOUSING | R(9)=0 |

| | | |
|----|----------|----|
| 15 | MACHINE | 98 |
| 4 | MOTOR | 96 |
| 7 | BLOCK | 94 |
| 13 | PLATE | 89 |
| 10 | FRAME | 84 |
| 6 | FITTEE | 82 |
| 14 | TANK | 75 |
| 1 | BRAKES | 70 |
| 5 | IGNITION | 63 |
| 9 | HOUSING | 58 |

71

77

NOW FIND

7 BLOCK 94

72

718

??APPLE-2

??72

| | | |
|----|---------|----|
| 15 | MACHINE | 98 |
| 4 | MOTOR | 96 |
| 7 | BLOCK | 94 |
| 13 | PLATE | 89 |

| | | |
|----|----------|----|
| 10 | FRAME | 84 |
| 6 | FITTEE | 82 |
| 14 | TANK | 75 |
| 18 | APPLE-2 | 72 |
| 1 | BRAKES | 70 |
| 5 | IGNITION | 63 |
| 9 | HOUSING | 58 |

| | | |
|----|----------|----|
| 73 | | |
| 74 | | |
| 15 | MACHINE | 98 |
| 7 | BLOCK | 94 |
| 13 | PLATE | 89 |
| 10 | FRAME | 84 |
| 6 | FITTEE | 82 |
| 14 | TANK | 75 |
| 18 | APPLE-2 | 72 |
| 1 | BRAKES | 70 |
| 5 | IGNITION | 63 |
| 9 | HOUSING | 58 |

| | | |
|-----------|----------|----|
| 74 | | |
| 718 | | |
| 7PC-1500A | | |
| 7774 | | |
| 15 | MACHINE | 98 |
| 7 | BLOCK | 94 |
| 13 | PLATE | 89 |
| 10 | FRAME | 84 |
| 6 | FITTEE | 82 |
| 14 | TANK | 75 |
| 18 | PC-1500A | 74 |
| 1 | BRAKES | 70 |
| 5 | IGNITION | 63 |
| 9 | HOUSING | 58 |

将程序 PRO-88 货物名称改为中文汉字，并在各主要程序段加注简要的汉字说明，则程序阅读和操作将十分方

便。程序PRO-88-1是各种微机及数量的库存情况，利用PRO-88-1可以完成查找、插入、删除、修改四种功能，是一个实用方便的工具软件。程序清单及运行实例，见PRO-88-1。

PRO-88-1

```
2 REM PRO-88-1
5 REM 链接技术
10 DIM B(20),A$(20),R(20)
11 REM 12-18打印菜单
12 PRINT "1--查找记录"
14 PRINT "2--插入记录"
15 PRINT "3--删除记录"
16 PRINT "4--修改记录"
18 PRINT "5--结束运行"
19 PRINT
20 A = 15:P = A
30 READ B(P),A$(P),R(P)
40 P = R(P)
50 IF P < > 0 THEN 30
55 GOSUB 400
58 REM 60-80总控
60 INPUT "请选择1-5:";N
70 ON N GOTO 100,200,300,500
80 PRINT "END": END
90 REM 100-180查找
100 INPUT "请输入查找序号";M
110 P = A
120 IF P = M THEN 160
130 P = R(P)
140 IF P < > 0 THEN 120
```

```

150 PRINT "NOT FIND ": GOTO 60
160 PRINT "NOW FIND"
170 PRINT M; TAB( 8);A$(M); TAB( 25);B(M)
175 PRINT : PRINT "*-*-*-*-*-*-*-*-*-*-*-*-*-*-*"
180 GOTO 60
190 REM 200-280插入
200 INPUT "请输入序号,名称,数量";M,A$(M),B(M)
210 P = A
220 IF B(M) > B(P) THEN 240
230 K = P:P = R(P)
235 IF P < > 0 THEN 220
240 R(M) = P
250 IF A = P THEN 270
260 R(K) = M: GOSUB 400
265 GOTO 60
270 A = M: GOSUB 400
280 GOTO 60
290 REM 300-390删除
300 INPUT "请输入序号";M
310 P = A
320 IF P = M THEN 360
330 K = P:P = R(P)
340 IF P < > 0 THEN 320
350 PRINT "NOT FIND": GOTO 60
360 IF A = P THEN 380
370 R(K) = R(P): GOSUB 400
375 GOTO 60
380 A = R(P): GOSUB 400
390 GOTO 60
395 REM 400-440打印链接表
400 P = A
410 PRINT P; TAB( 8);A$(P); TAB( 25);B(P)

```

```

420 P = R(P)
430 IF P < > 0 THEN 410
435 PRINT : PRINT "- - - - -"
440 RETURN
490 REM 500-590修改
500 INPUT "请输入序号";M
510 P = A
520 IF P = M THEN 560
530 P = R(P)
540 IF P < > 0 THEN 520
550 GOTO 60
560 INPUT "请输入名称,数量";A$(M),B(M)
570 A$(P) = A$(M):B(P) = B(M)
580 GOSUB 400
590 GOTO 60
600 DATA 98,LASER-310,4,96,PC-1500,7,94
605 DATA 中华学习机,13,89,APPLE,10,84,长城,6
610 DATA 82,IBM-XT,14,75,PC-1500A,1,70
615 DATA 紫金-2,5,63,长城286,9,58,中华学习机-2,0

```

IRUN

1--查找记录
 2--插入记录
 3--删除记录
 4--修改记录
 5--结束运行

| | | |
|----|-----------|----|
| 15 | LASER-310 | 98 |
| 4 | PC-1500 | 96 |
| 7 | 中华学习机 | 94 |
| 13 | APPLE | 89 |
| 10 | 长城 | 84 |
| 6 | IBM-XT | 82 |

| | | |
|----|----------|----|
| 14 | PO-1500A | 75 |
| 1 | 紫金-2 | 70 |
| 5 | 长城286 | 68 |
| 9 | 中华学习机-2 | 58 |

程序运行后，首先列印出一个菜单，按1,2,3,4,5列出程序PRO-88-1的几个功能，而数字就是执行该程序的操作说明。然后列出一张明细表，按微机编号（序号）、名称和数量逐行顺序排列。接着是打印一行虚线，并返回总控出现“请选择1-5:”的提示。

```

请选择1-5:1
请输入查找序号9
NOW FIND
9    中华学习机-2    58

```

在出现“请选择1-5:”的提示后，如从键盘键入1并回车，则程序进入第一个功能模块，可以实现查找。

屏幕上又出现“请输入查找序号”的提示，即询问你要查找那一种编号的货物情况，此时若键入9并回车，屏幕出现找到了的英文提示并打印出：

```

9    中华学习机-2    58

```

如果键入的序号，存贮单元中没有，也会通知你“NOT FIND”，并返回总控，可以继续查找。找到后打印有关信息，再返回总控。

```

请选择1-5:2
请输入序号、名称、数量2 IBM/PO-8I,77
15    LASER-310    98
4     PO-1500    96
7     中华学习机    94

```

| | | |
|----|-----------|----|
| 13 | APPLE | 89 |
| 10 | 长城 | 84 |
| 6 | IBM-XT | 82 |
| 2 | IBM/PC-XT | 77 |
| 14 | PC-1500A | 75 |
| 1 | 紫金-2 | 70 |
| 5 | 长城286 | 63 |
| 9 | 中华学习机-2 | 58 |

返回总控后，若键入 2，则进入插入（增加）记录的模块，使用者可根据序号（原来没有的）、微机名称和数量送入机器，回车后立即显示出原来微机存贮情况，以及新增加的记录。最后再返回总控。

上面结果说明，新增加的微机 IBM/PC-XT 是插进去了，由于它的数量是 77，它排在数量 82 和 75 之间，说明本操作完成后，微机仍按数量多少从大到小有序排列。

请选择 1-5:3

请输入序号 1

| | | |
|----|-----------|----|
| 15 | LASER-310 | 98 |
| 4 | PC-1500 | 96 |
| 7 | 中华学习机 | 94 |
| 13 | APPLE | 89 |
| 10 | 长城 | 84 |
| 6 | IBM-XT | 82 |
| 2 | IBM/PC-XT | 77 |
| 14 | PC-1500A | 75 |
| 5 | 长城286 | 63 |
| 9 | 中华学习机-2 | 58 |

第三种操作比较简单，键入 3 ↵，进行删除记录模块，只要再键入要删除的序号，如 1，即可删除编号为 1 的微机存贮

情况，以上结果表明，原来编号为 1 的紫金-2微机被删除了。

请选择1-5:4

请输入序号4

请输入名称,数量R1,85

| | | |
|----|-----------|----|
| 15 | LASER-310 | 98 |
| 4 | R1 | 85 |
| 7 | 中华学习机 | 94 |
| 13 | APPLE | 89 |
| 10 | 长城 | 84 |
| 6 | IBM-XT | 82 |
| 2 | IBM/PC-XT | 77 |
| 14 | PC-1500A | 75 |
| 5 | 长城286 | 63 |
| 9 | 中华学习机-2 | 58 |

键入 4↙ 进行第四种操作，修改记录，如键入序号 4，输入 R₁ 机，数量 55，则原来序号为 4 的 PC-1500 机被删除，代之而起的是 R₁ 机情况。

如果，在进行修改操作时，输入的序号原来没有，则计算机不作处理并返回总控，请求使用者重新操作。

请选择1-5:5

END

结束停机，只要键入 5↙ 即可。

说明：程序 PRO-88-1 操作十分灵活，不一定按 1, 2, 3, 4, 5 顺序操作，程序运行后，即可以随意进行何种操作，当然，增加的记录，序号不应和原记录重复，以示各种不同微机有不同的编号；其它诸如检索、删除和修改记录，也应该是原记录中有的编号。

五、图形数据处理

图形数据处理，是指用计算机控制在显示屏幕上表示曲线、绘制图表、甚至活动目标的方法。

信息表示成图象的形式时，往往使信息更加清晰和易于理解。用图形美化屏幕，再配以文字说明，使图形表达的信息更加生动和丰富多采。

图形数据处理的内容是十分丰富的。本章仅从基本作图语句绘制图形，用键盘控制作图，图形表技术几个方面，介绍绘图的一些实例和工具软件。更深入的内容，在《中华学习机图形管理》一书中给予介绍。

1. 一个表演程序

在低分辨作图和文本方式下，用 HOME 指令，可以将屏幕擦干净。现在，再介绍一个命令：CALL-868，用它可以将文本内容全部擦除干净。为了试验上述指令“擦屏幕”的本领，请看一个表演程序 PRO-89。

为此，可以先在屏幕上写满字符，如 B，只要用一个语句就能达到目的。

如30 FOR I= 1 TO 800: PRINT “B”,:NEXT I
程序是这样运行的：

- RUN↵，屏幕上充满 B 字符。
- 出现? 后，键入 1↵，能把上述字符从屏幕上部到下部逐行擦除。
- 屏幕上又充满 B 字符以及?，键入 2↵，则将屏幕字符

从底部到顶部逐行清除。

- 键入 3 ↵，从上下两个部份向中央逐渐清屏。
- 键入 4 ↵，从右向左将屏幕字符擦除干净。
- 键入 5 ↵，从左向右将屏幕字符擦除干净。
- 键入 6 ↵，是快速清屏。再按 RESET 键，出现提示符。

- 任何时刻，键入 - 1 ↵ 结束。

表演程序见，PRO-89。

PRO-89

```
10 A$ = CHR$ (32)
20 REM FIRST-2
30 FOR I = 1 TO 800: PRINT "B";:
   NEXT I
40 INPUT K
50 IF K = - 1 THEN END
60 ON K GOSUB 80,140,200,350,430
   ,260
70 GOTO 30
80 FOR I = 1 TO 24
90 VTAB I
100 CALL - 868
110 FOR X = 1 TO 300: NEXT X
120 NEXT I
130 RETURN
140 FOR I = 24 TO 1 STEP - 1
150 VTAB I
160 CALL - 868
170 FOR X = 1 TO 300: NEXT X
180 NEXT I
190 RETURN
200 X = 0:Y = 24:P = - 868
210 X = X + 1:Y = Y - 1
220 VTAB X: CALL P: VTAB Y: CALL
   P
230 IF X = 13 THEN RETURN
240 FOR T = 1 TO 500: NEXT T
250 GOTO 210
260 FOR X = 1 TO 40
```

```

270 FOR Z = 1 TO 2000: NEXT Z
280 FOR Y = 1 TO 24
290 HTAB X: VTAB Y
300 CALL - 868
310 NEXT Y
320 NEXT X
330 PRINT
340 RETURN
350 A$ = CHR$ (32)
360 FOR X = 40 TO 1 STEP - 1
370 FOR Y = 24 TO 1 STEP - 1
380 HTAB X: VTAB Y
390 PRINT A$;
400 NEXT Y
410 NEXT X
420 RETURN
430 A$ = CHR$ (32)
440 FOR X = 1 TO 40
450 FOR Y = 1 TO 24
460 HTAB X: VTAB Y
470 PRINT A$;
480 NEXT Y
490 NEXT X
500 RETURN

```

整个程序采用模块化结构，第60句实现总控，由键盘输入的K值（K=1,2,3,4,5,6）通过60句转向各子程序模块，执行完相应的子程序又返回总控。

为了看清各种方式的清屏过程，在子程序中安排了延迟循环。

本程序350—410,430—490句，采用打印空字符 CHR \$ (32)的方法，将屏幕擦除。若删去390,470句 A\$ 后面的分号则又是两种清屏方式。

应该指出的是，本程序不能将高分辨图形清除。

2. 不同画面的显示

当磁盘中已经存有不少绘图程序时，如果我们想逐一显

示其内容，通常的方法是一个一个调进内存再运行，这样做键盘操作次数多，比较烦琐。

简单的方法是，键入程序PRO-90，在所有LOAD后面加上图形程序的名称。

RUN程序PRO-90，这样，就建立了一个名为PPP的T类文件，然后执行EXEC命令：EXEC PPP

这个DOS命令会自动打开名为PPP的T文件，并顺序按文件中的命令一个一个地执行，从屏幕上可以看到各种图形。如果事先接通打印机，则一面在屏幕上看到图形，一面又可以打印出清单。

PRO-90

ULIST

```
5  REM P-27
10  D$ = CHR$ (4)
20  PRINT D$;"OPEN PPP"
25  PRINT D$;"DELETE PPP"
28  PRINT D$;"OPEN PPP"
30  PRINT D$;"WRITE PPP"
35  PRINT "LOAD P-23"
37  PRINT "LIST"
38  GOSUB 200
40  PRINT "RUN"
42  GOSUB 200
45  PRINT "LOAD P-25"
47  PRINT "LIST"
48  GOSUB 200
50  PRINT "RUN"
52  GOSUB 200
65  PRINT D$;"CLOSE"
70  END
200 FOR I = 1 TO 3000: NEXT I: RETURN
URUN
```

UEXEC PPP

U
U

```

200 REM P-23
210 HGR
220 HCOLOR= 3
240 HPLLOT 10,50 TO 260,50 TO 260
    ,150 TO 10,150 TO 10,50
245 FOR I = 1 TO 1000: NEXT
250 TEXT : HOME
260 END

```

Ü

Ü

Ü

```

290 REM P-25
300 HGR2
310 HCOLOR= 3
320 HPLLOT 120,20 TO 150,20 TO 15
    0,170 TO 120,170 TO 120,20
325 FOR I = 1 TO 1000: NEXT
330 TEXT : HOME
340 END

```

若是机器语言绘制的程序，将PRO-90中的LOAD改为BLOAD，RUN改为BRUN，但机器语言程序清单列不出。读者可以仿照本实例，处理多个图形程序。

3. 图形的快速合并

在绘制一幅较为复杂的图画时，可以将这幅画分成几块，由几个人分头完成，然后再合并成一幅完整的画面，这不仅可以发挥各人绘画专长，而且大大加快速度，因此，编制一个能够合并图形的程序，就显得非常必要。

为叙述方便，仅以两幅简单图象为例。

例如，第一幅是画一根横线，它在高分辨率第一页中，程序为PRO-91。

PRO-91

ULIST

```
200 REM PRO-91
210 HGR
220 HCOLOR= 3
230 HPLOT 10,50 TO 260,50
```

第二幅是画一根纵线，它存在高分辨率第二页中，程序为PRO-92：

PRO-92

ULIST

```
290 REM PRO-92
300 HGR2
310 HCOLOR= 3
320 HPLOT 135,10 TO 135,160
```

为合并以上两幅图形，用一个BASIC语言编制的程序，取名为PRO-93。

PRO-93

ULIST

```
10 REM PRO-93
20 INPUT N: IF N < > 1 AND N < >
  2 THEN 20
30 FOR I = 1 TO 10 STEP 3: READ X
  : POKE 767 + I, X: NEXT : DATA
  173,13,141,96
40 FOR I = 8192 TO 16383: A% = I /
  256: B = I - A% * 256: POKE 76
  9, B: POKE 770, A%: POKE 772, B:
  POKE 773, A% + 32: POKE 775, B
  : POKE 776, A% + 32 * (N - 1):
  CALL 768: NEXT
50 POKE - 16304, 0: POKE - 16302
  , 0: POKE - 16297, 0: POKE -
  16301 + N, 0
```

合并图形的步骤：

- RUN PRO-91 ✓
- RUN PRO-92 ✓
- RUN PRO-93 ✓

出现 ? 后, 键入 1 或 2, 经过一段时间, 两幅图形合在一个画面上。

但是, 合并过程时间较长, 共费时 8 分 30 秒。

若改用机器语言编制的图形合并程序, 则速度大大提高, 仍以上述两幅图形为例, 时间仅为 1 秒, 提高速度 510 倍。机器语言子程序取名为 PRO-94, 用 BSAVE PRO-94, A\$ 6000, L\$ 3 B 存盘。

PRO-94

UCALL-151

*6000.603A

```

6000- A9 FF 85 06 85 08 A9 1F
6008- 85 07 A9 3F 85 09 E6 06
6010- D0 02 E6 07 E6 08 D0 02
6018- E6 09 A2 00 A1 06 01 08
6020- 81 08 A9 3F C5 07 D0 E6
6028- A9 FF C5 06 D0 E0 8D 50
6030- C0 8D 52 C0 8D 55 C0 8D
6038- 57 C0 60

```

图形合并步骤:

- RUN PRO-91 ✓
- RUN PRO-92 ✓
- BLOAD PRO-94, A\$ 6000, L\$ 3 B ✓
- CALL 24576 ✓

操作完成后, 图形立即合并。

第四步操作也可以用 CALL-151 ✓, 出现 * 后键入 6000 G ✓ 未实现。

4. 图形的转移

如果希望把高精度作图第一页的图象，转移到第二页显示，应该如何处理呢？

众所周知，高分辨率图形第一页的内存地址是\$ 2000—\$ 3 FFF，其相应的十进制数为：8192—16383，而第二页的内存地址是\$ 4000—\$ 5 FFF，对应的十进制数为16384 —24575。每个区域的长度都是8192个字节。

从上面高分辨率图形的内存分配，可以看到这样一个事实：图形第一页的入口地址加上8192个字节，正好是第二页图形的入口地址16384；同理，第一页图形的尾地址加上8192个字节，又正好是第二页图形的尾地址24575。还可用BASIC程序PRO-95实现第一页图形到第二页图形地址上的转移。这实际上也是一种“搬家”。

PRO-95

```
48 REM PRO-95
50 FOR I = 8192 TO 16383
52 II = I + 8192
54 POKE II, PEEK (I)
56 NEXT I
```

为了实现并检验第一页图象是否转移到第二页，可以用程序PRO-96来验证。

PRO-96

```
5 REM PRO-96
10 HGR : HCOLOR= 3: HPLOT 140,95
20 FOR I = 1 TO 5: X = 140 + 10 *
    I: Y = 95 + 5 * I: HPLOT TO X
    ,Y
30 NEXT I
35 GET A$
```

```

40 HGR2
50 FOR I = 8192 TO 16383:II = I +
    8192
60 POKE II, PEEK (I)
70 NEXT I
75 GET A$
80 TEXT : HOME : PRINT "DONE!": END

```

此程序10—30句，在高分辨第一页画了一根倾斜线，35句等待键盘输入，敲一下空格键即可。40—70句，是将第一页的内容送到第二页，并在屏幕上显示出来。但花时较多，原因是比较清楚的，请看50—70句，循环8000多次，还要一次一次地置数，又一次一次地察看。75句的用法同上，敲一下空格键，执行80句清屏并结束程序。

为了加快图形的转移，可以用程序PRO-97来实现。

PRO-97

```

5 REM PRO-97
10 HGR : HCOLOR= 3
20 FOR I = 0 TO 13: HPLLOT 4, 8 + I
    * 11 TO 184, 8 + I * 11: HPLLOT
    3 + I * 14, 9 TO 3 + I * 14, 15
    0
30 NEXT I
31 PR# 1
32 PRINT CHR$ (17)
35 GET A$
40 HGR2
45 Y$ = "4000<2000.3FFFM N D823G"
50 FOR I = 1 TO LEN (Y$): POKE 5
    11 + I, ASC ( MID$ (Y$, I, 1)) +
    128: NEXT I: POKE 72, 0
60 CALL - 144
64 PR# 1
65 PRINT CHR$ (17)
70 GET A$
80 TEXT : HOME : PRINT "GOOD BYE!
    ": END

```

运行此程序,速度大大加快。其中D 823G是返回BASIC状态。

应该指出的是,第二页图形也可以转移到第一页上显示,只需将程序PRO-97作以下几处改动:

10 HGR2: HCOLOR = 3

40 HGR

45 Y\$ = "2000<4000.5 F F F M N D 823G"

5. 机器语言绘图子程序搬家

已知有一个用机器语言编制的绘图子程序,它的功能是能快速清屏。文件名为PRO-98,入口地址为6ADO,末尾地址是6AEF。现在,希望将上述地址内的机器语言“搬家”到\$ 6000开始的连续单元中。

在监控状态下,用下述命令即可实现:

* (地址1) < (地址2) · (地址3) M ✓

这就是把从地址2(本例是6ADO)到地址3(本例是6AEF)的内容,移到(搬家到)从地址1(据题意有6000)开始的连续地址中。

即] CALL-151 ✓

* 6000 < 6ADO · 6AEF M ✓

*

注意,在进入监控状态前,必须将PRO-98文件调入内存,即BLOADPRO-98 ✓

为了检查文件名为PRO-98的机器语言是否已经搬至\$ 6000开始的连续存贮单元中,可在*后键入6000 · 601F ✓
屏幕上立即显示和PRO-98相同的内容。

PRO-98

*6AD0.6AEF

```
6AD0- A0 00 A9 20 B5 11 A9 00
6AD8- 85 10 91 10 C8 D0 FB A6
6AE0- 11 E0 3F F0 05 E6 11 4C
6AE8- DA 6A 60 00 80 80 00 5E
```

应该说明的是，经上述移动后，从地址 2 到地址 3 的内容并未改变。这就是说，\$6AD0—\$6AEF 中的内容仍然存在，而在 \$6000—\$601F 中的内容又和 \$6AD0—\$6AEF 完全一样。

为了验证上面的结果，并检查 PRO-98 机器语言程序是否具有在高分辨率图形区快速清屏的功能，可运行程序 PRO 99，并键入 CALL 27344。

PRO-99

ULIST

```
10 HGR : HCOLOR=3
20 FOR X = 10 TO 257 STEP 19
30 HPLLOT X,10 TO X,143
40 NEXT X
50 FOR Y = 10 TO 143 STEP 19
60 HPLLOT 10,Y TO 257,Y
70 NEXT Y
80 FOR I = 1 TO 8 STEP 2
90 FOR Y = I * 19 TO (I + 1) * 19
100 FOR J = 0 TO 12 STEP 2
120 FOR X = J * 19 TO (J + 1) * 19
130 HPLLOT X,Y
140 NEXT X,J
150 NEXT Y,I
160 END
```

CALL 后面的 27344，是将快速清幕机器语言子程序入口地址，相当于十六进制数 6AD0。RUN PRO-99 后，我们看到画面一显即灭。说明确实具有快速清屏的功能。

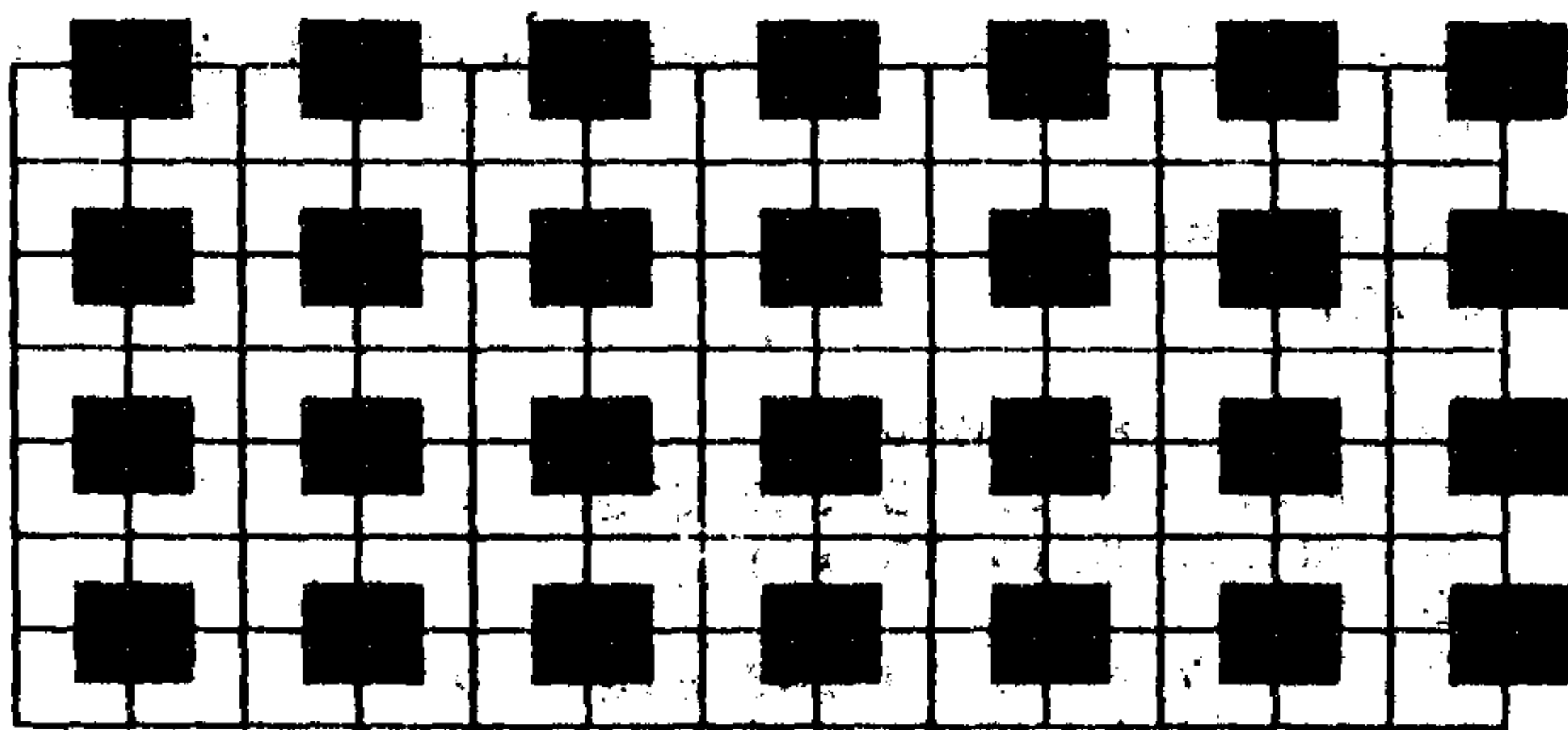
若改动程序PRO-99运行方式，变为CALL 24576，再运行之，则和上面的结果完全一样。24576即十六进制的\$ 6000，这是搬家后的新入口地址。

若要看清上述画面显示和清除过程，可在程序PRO-99中，加上一段延迟程序：

```
155 FOR I=1 TO 1000: NEXT I
```

循环终值愈大，延迟时间愈长，图象也看得愈清楚见程序PRO-99绘制的图形。

注意，可以省去NEXT后面I，这会使循环速度加快。请上机运行，体会二者的差异。你能说出原因吗？



6. 一个简单的绘图程序

程序PRO-100通用性好，既可以写字又可以作图。

PRO-100

LIST

```
50 PRINT
60 READ X
70 IF X < 0 THEN 50
80 IF X > = 35 THEN END
```

```

100 PRINT TAB( X); " ";
110 READ Y
120 FOR I = X TO Y
130 PRINT "B";
140 NEXT I
150 GOTO 60
190 DATA 0,4,35

```

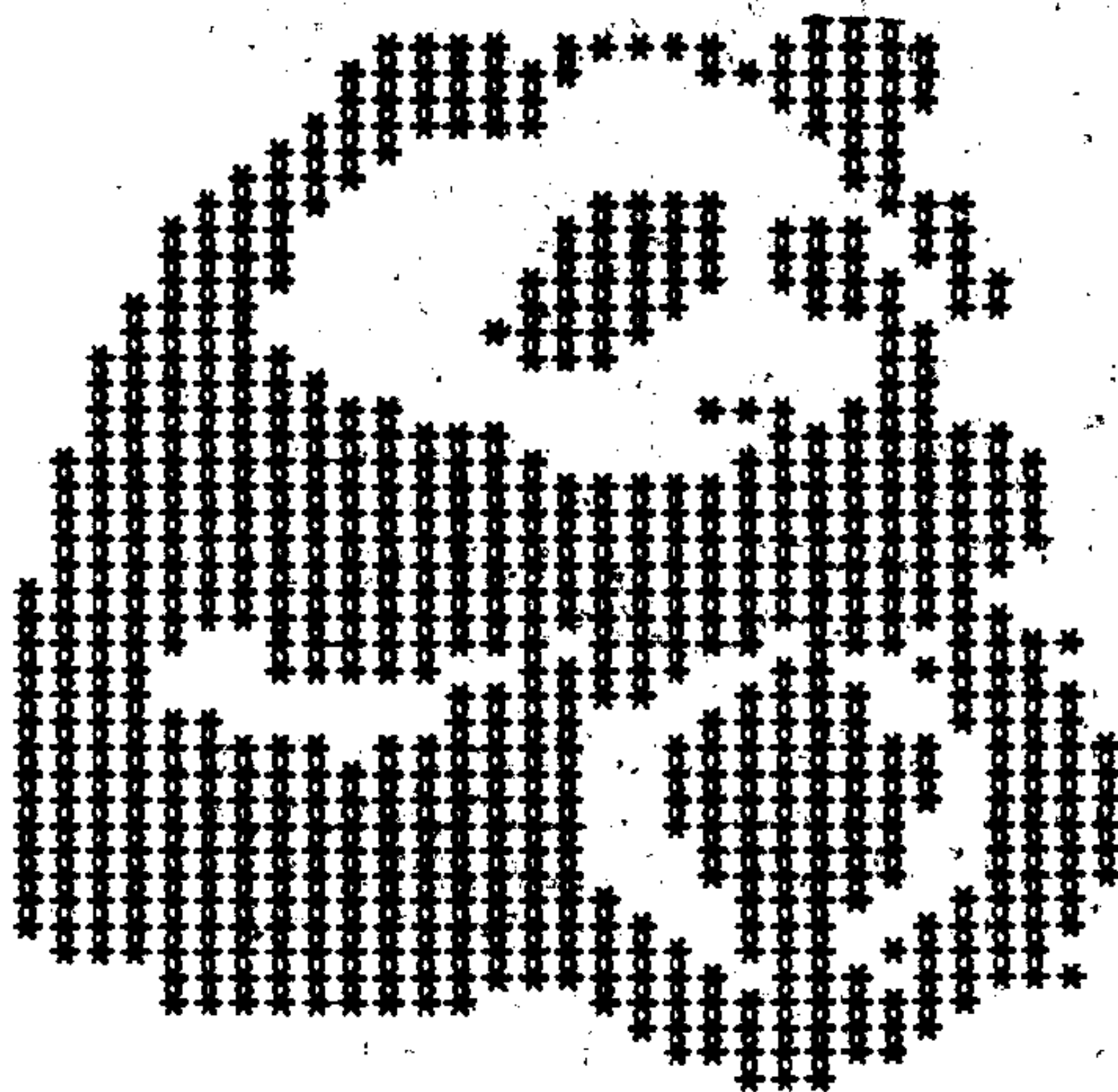
运行PRO-100程序,可以写成由5个B字符组成的“—”字。

程序的执行情况如下:

首先读一个数零,因不满足70句和80句条件,而转向100句,在 $X = 0$ 的位置上打印一个空格,实际上是控制打印针头处在起始位置。执行110句读一个数4,经过120—140句循环,连续打印5个“B”字符,循环结束经150句再转向60句,又读一个值35,满足80句条件结束。

如果DATA区中安排小于零的数,如-1,则满足70句条件而转向50句,50句和70句配合,起控制打印针头换行作用。

URUN



如果我们事先写好字或画好图（最好用方格坐标纸），把数据放在 DATA 区中，则可以利用上述程序，自动完成写字或画图的工作。

上面是一个实例，运行程序 PRO-101 可以画出一个熊猫图案来。

PRO-101

```
18 PRINT
20 READ X: IF X < 0 THEN PRINT
   : GOTO 20
30 IF X > 36 THEN 150
35 PRINT TAB( X); " ";
40 READ Y
50 FOR I = X TO Y
60 PRINT "+";
70 NEXT I
80 GOTO 20
150 RESTORE
180 END
200 DATA 23,25,-1,11,14,16,20,22
      ,26,-1,10,16,20,26,-1,10,15,
      22,26,-1,9,15,23,25,-1
210 DATA 8,11,24,25,-1,7,10,24,2
      5,-1,6,9,17,20,23,27,-1,5,8,
      16,20,22,24,26,27,-1
220 DATA 5,8,16,20,22,24,26,27,
      -1,5,8,15,20,22,25,26,27,-1,
      4,7,15,19,23,25,26,27,-1
230 DATA 4,7,14,18,25,26,-1,3,8
      ,15,17,25,26,-1,3,9,25,26,-1
      ,3,11,20,22,24,26,-1
240 DATA 3,14,22,28,-1,2,15,21,2
      9,-1,2,29,-1,2,29,-1,2,29,-1
      ,2,28,-1
250 DATA 1,27,-1,1,28,-1,1,5,8,
      15,16,20,22,24,26,29,-1,1,4,
      8,12,15,19,22,23,26,29,-1
260 DATA 1,4,13,18,21,24,27,30,-
      1,1,6,13,16,20,24,27,30,-1,1
      ,9,11,16,19,26,28,31,-1
```

```

270 DATA 1,16,19,26,28,31,-1,1,1
    6,19,26,28,31,-1,1,16,19,25,
    28,31,-1,1,16,20,25,28,31,-1
    ,1,16,20,25,28,31,-1
280 DATA 1,17,21,24,27,30,-1,1,1
    8,21,23,26,30,-1,2,19,21,23,
    24,28,-1
290 DATA 5,20,21,23,24,28,-1,5
    ,13,17,27,-1,18,26,-1,19,25,
    -1,21,23,-1,999

```

7. 图形自动显示和打印

在图形程序中，安排一些DOS命令，可使图形边显示边打印。程序实例和运行结果，见PRO-102。

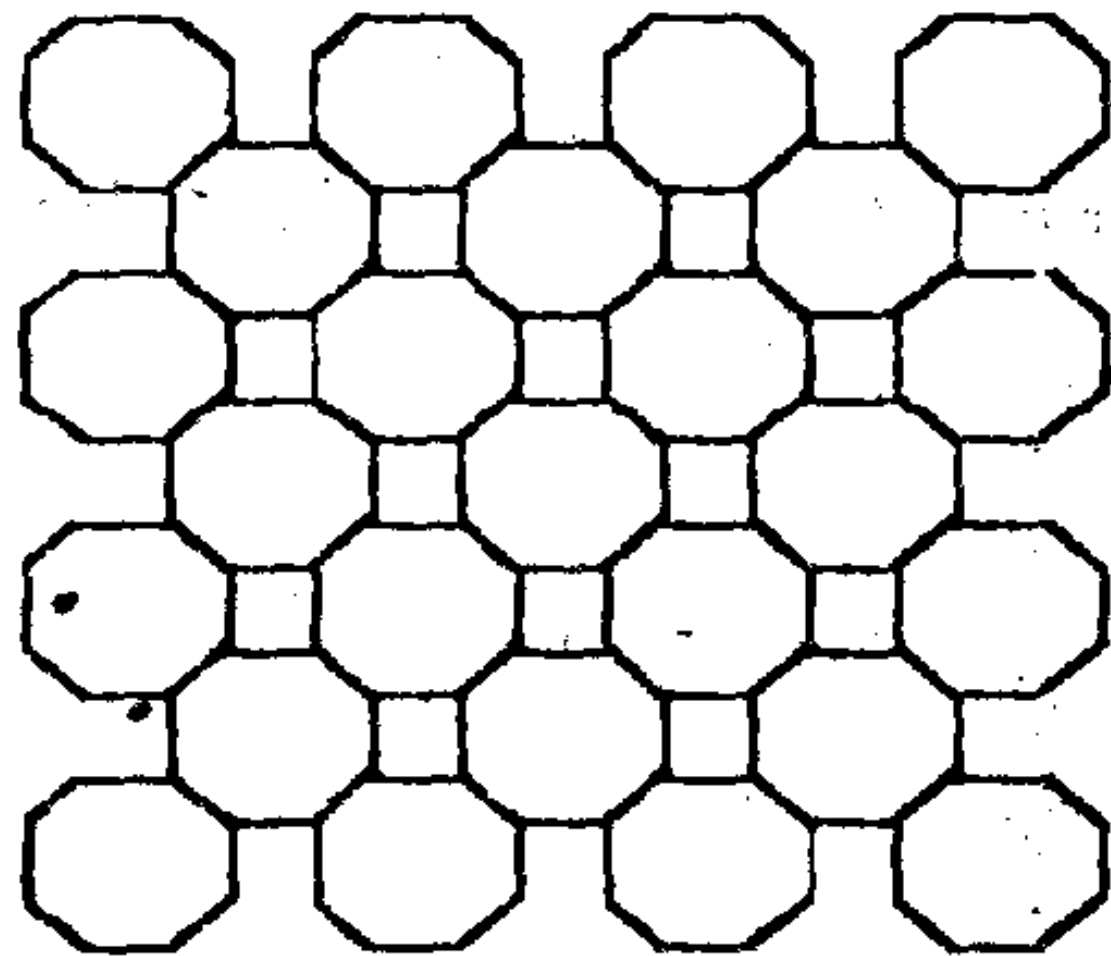
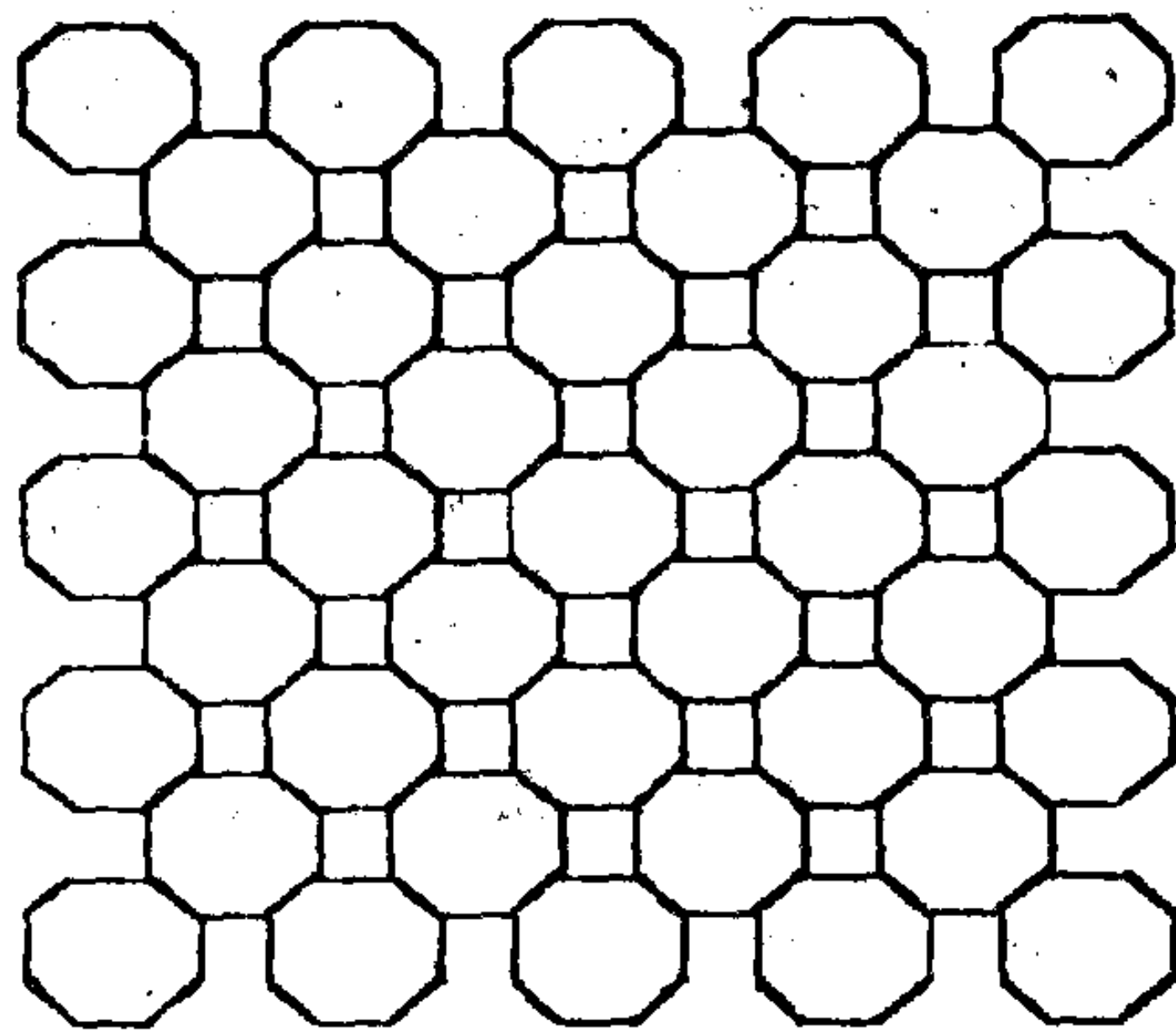
PRO-102

ULIST

```

10 D$ = CHR$(4)
20 INPUT "N<6 ";N
25 FOR I = 1 TO 5
30 A = 2 * N - 1
35 HGR2 : HCOLOR= 7
40 X = 0:Y = 17:P = 0
50 FOR U = 1 TO A
60 FOR V = 1 TO N
70 HPLLOT X,Y
80 FOR W = 1 TO 8
90 READ E,F
100 HPLLOT TO X + E,Y + F: NEXT
    W
110 X = X + 34: RESTORE : NEXT V
120 N = A - N:X = 17 - P:P = X:Y =
    Y + 18: NEXT U
122 PRINT D$;"PR#1"
124 POKE 1913,2: PRINT CHR$(17
    )
125 NEXT I
126 PRINT D$;"PR#0"
130 DATA 0,-12,5,-17,17,-17,24,
    -10,24,0,17,7,7,7,0,0

```

几点说明:

- 运行程序后, 出现 $N < 6$ 时, 可从键盘送入 1—5 的数字并按回车, 计算机循环显示各种变化的图案。若要边显示边打印, 必须先开启打印机电源。

- N 取 8 时, 上述图案正好充满全部屏幕, 出现非法置错误也不要紧, 按 RESET 再按 CTRL-Q, 即可打印一幅全屏幕的图案。

8. 图形的对称处理

(1) 左右对称

在第一页高分辨率图形区,画一 (120,10) — (120, 180) 直线,运行 PRO-103 程序,即可看到另一根对称直线。如果画的图案,也可得到双幅对称图形,但要注意坐标位置的选择。

PRO-103
ULIST

```
100 REM LIFT-RIGHT
110 HGR : HCOLOR= 6
120 ONERR GOTO 170
130 READ X1,Y1,X2,Y2
140 HPLOT X1,Y1 TO X2,Y2
150 HPLOT 139 - (X1 - 139),Y1 TO
    139 - (X2 - 139),Y2
160 GOTO 130
170 END
180 DATA 120,10,120,180
```

(2) 上下对称

画一 (140, 10) — (180, 10) 的直线, 运行程序 PRO-104, 即可得到互相平行的对称输出。

PRO-104

```
5  REM UP-DOWN
10  HGR : HCOLOR= 6
20  ONERR GOTO 70
30  READ X1,Y1,X2,Y2
40  HPLOT X1,Y1 TO X2,Y2
50  HPLOT X1,80 - (Y1 - 80) TO X2
    ,80 - (Y2 - 80)
60  GOTO 30
70  END
80  DATA 140,10,180,10
```

URUN

U?CHR\$(17)

.....

上述两个程序也可改为 FOR /NEXT 循环控制。利用 20 句 ONERR GOTO 70, 可以使输入数据更加灵活而不出错。

9. 按键控制运动方向

程序 PRO-105, PRO-106, 是利用键盘上有关字符键, 实现按键控制图形运动方向, 从而画出简单图形。程序 PRO-105, 用 K、M、J、I 4 个键, 控制一个画点向右、向下、向左、向上 4 个方向运动。程序 PRO-106, 用 Y、G、B、H 4 个键, 实现画点向左上、左下、右下、右上 4 个方

向移动。按其它任意键清除画面，重新作图。

PRO-105

ULIST

```
10  ONERR  GOTO 110
20  HGR2 : HCOLOR= 7
30  X = 140:Y = 95: HPLOT X,Y
40  GET A$
50  IF A$ = "" THEN 40
60  HCOLOR= 0: HPLOT TO X,Y
70  IF A$ = "K" THEN X = X + 1: GOTO
    40
80  IF A$ = "M" THEN Y = Y + 1: GOTO
    40
90  IF A$ = "J" THEN X = X - 1: GOTO
    40
100 IF A$ = "I" THEN Y = Y - 1: GOTO
    40
110 HOME : GOTO 20
```

PRO-106

ULIST

```
10  ONERR  GOTO 110
20  HGR2 : HCOLOR= 7
30  X = 140:Y = 95: HPLOT X,Y
40  GET A$
50  IF A$ = "" THEN 40
60  HCOLOR= 0: HPLOT TO X,Y
70  IF A$ = "Y" THEN X = X - 1:Y =
    Y - 1: GOTO 40
80  IF A$ = "G" THEN X = X - 1:Y =
    Y + 1: GOTO 40
90  IF A$ = "B" THEN X = X + 1:Y =
    Y + 1: GOTO 40
100 IF A$ = "H" THEN X = X + 1:Y =
    Y - 1: GOTO 40
110 HOME : GOTO 20
```

10. 全屏幕作图

下面介绍的是一个实用的全屏幕作图软件。其特点是：

• 用有关按键控制光点移动方向，进行作图或写字。图象水平的高低，由操纵者的绘图技术决定。

• 使用简单。操纵者可以根据菜单提示，只要稍事熟悉，即可运用自如。

• 删改灵活。画的不满意的地方，可以随意擦除。

• 存贮方便。完成或尚未完成的作品，可以随时存贮，使用时可以方便地取出，或继续完成，或继续美化图案。

• 打印多样。需要打印的图案，可以放大也可缩小，可以正相显示，也可反相显示。

使用方法：

• 运行本软件后，即出现一个英文菜单，菜单上详细注明使用方法，如按Y键，光点向左上方移动并作图，擦除先按O键再按其它键，恢复画图先按L键再按其它键。

URUN

```
<==MAKE AN IMAG ==>
KEYS      MANES
-----
Y          :UP-LEFT
G          :DOWN-LEFT
B          :DOWN-RIGHT
H          :UP-RIGHT
I          :UP
J          :LEFT
K          :RIGHT
M          :DOWN
L          :PLOT
O          :UNPLOT
C          :CLEAR SCREEN
T          :LIST ORDERS
A          :DISPLAY AN IMAG
           HAD BEEN MADE
S          :STORE IT INTO DISK
W          :HADR COPY ON PRINTER
RETURN, IF YOU TYPE :SPACE-BAR
```

- 菜单出现后, 按空格 (SPACE) 键, 即出现一个沿屏幕边沿的一个方框, 光点在屏幕的正中央闪耀, 进入绘图状态。

- 画好的图象, 按S 键后, 输入图象名称并按回车, 即自动存盘。

- 调出图象, 按A键后, 送入图象名称, 即把图象显示在屏幕上。

- 任何时刻按C 键, 清除画面, 并重新自动进入绘图状态, 原图象丢失。

- 打印按W键, 分两种方式键入1或2, 自动打印不同形式图案。

程序清单见 PRO-107。

PRO-107

```
10 REM PRO-107
20 HGR2 : HCOLOR= 3: HPLOT 0,0 TO 279,0 TO 279,19
  1 TO 0,191 TO 0,0
30 FOR I = 1 TO 10: HPLOT 140,0 TO 140,191: HPLOT
  0,95 TO 279,95
40 HCOLOR= 0: HPLOT 140,0 TO 140,191: HPLOT 0,95
  TO 279,95: HCOLOR= 3
50 NEXT I
60 X = 140:Y = 95:B$ = "L"
70 HOME : TEXT : INVERSE
80 VTAB 10: HTAB 10: PRINT "<==MAKE AN IMAG ==>"
90 FOR I = 1 TO 1500: NEXT I
100 HOME : NORMAL
110 GOTO 330
120 VTAB 2: HTAB 10: PRINT "KEYS";: HTAB 20: PRIN
  T "MANES"
130 HTAB 10: PRINT "-----"
  ..
140 HTAB 10: PRINT "Y";: HTAB 20: PRINT " ": PRI
  NT "UP-LEFT"
```

```

150 HTAB 10: PRINT "G";: HTAB 20: PRINT ":";: PRI
NT "DOWN-LEFT"
160 HTAB 10: PRINT "B";: HTAB 20: PRINT ":";: PRI
NT "DOWN-RIGHT"
170 HTAB 10: PRINT "H";: HTAB 20: PRINT ":";: PRI
NT "UP-RIGHT"
180 HTAB 10: PRINT "I";: HTAB 20: PRINT ":";: PRI
NT "UP"
190 HTAB 10: PRINT "J";: HTAB 20: PRINT ":";: PRI
NT "LEFT"
200 HTAB 10: PRINT "K";: HTAB 20: PRINT ":";: PRI
NT "RIGHT"
210 HTAB 10: PRINT "M";: HTAB 20: PRINT ":";: PRI
NT "DOWN"
220 HTAB 10: PRINT "L";: HTAB 20: PRINT ":";: PRI
NT "PLOT"
230 HTAB 10: PRINT "O";: HTAB 20: PRINT ":";: PRI
NT "UNPLOT"
240 HTAB 10: PRINT "Q";: HTAB 20: PRINT ":";: PRI
NT "CLEAR SCREEN"
250 HTAB 10: PRINT "I";: HTAB 20: PRINT ":";: PRI
NT "LIST ORDERS"
260 HTAB 10: PRINT "A";: HTAB 20: PRINT ":";: PRI
NT "DISPLAY AN IMAGE": HTAB 22: PRINT "HAD BEEN MAD
E"
270 HTAB 10: PRINT "S";: HTAB 20: PRINT ":";: PRI
NT "STORE IT INTO DISK"
280 HTAB 10: PRINT "W";: HTAB 20: PRINT ":";: PRI
NT "HADR COPY ON PRINTER"
290 VTAB 21
300 HTAB 5: PRINT "RETURN, IF YOU TYPE :SPACE-BAR"
310 GET E$: IF E$ = " " THEN RETURN
320 GOTO 310
330 GOSUB 120
340 GOTO 530
350 W1 = PEEK ( - 16384): IF W1 > 127 THEN A$ =
CHR$ (W1 - 128): POKE - 16368, 0: GOTO 390
360 HPLOT X,Y: HCOLOR= 0: HPLOT X,Y: HCOLOR= 3
370 IF B$ = "L" THEN HPLOT X,Y
380 GOTO 350

```

```

390 IF A$ = "A" THEN 770
400 IF A$ = "Y" OR A$ = "G" OR A$ = "H" OR A$ = "
B" THEN GOSUB 840: GOTO 480
410 IF A$ = "I" OR A$ = "K" OR A$ = "J" OR A$ = "
M" THEN GOSUB 730: GOTO 480
420 IF A$ = "O" THEN 20
430 IF A$ = "T" THEN 520
440 IF A$ = "S" THEN 550
450 IF A$ = "W" THEN 650
460 IF A$ = "L" OR A$ = "D" THEN B$ = A$: GOTO 49
0
470 CALL - 198: GOTO 350
480 O$ = A$
490 IF B$ = "L" THEN HPLLOT X,Y: GOTO 350
500 HPLLOT X,Y: FOR I = 1 TO 5: NEXT I: HCOLOR= 0:
HPLLOT X,Y
510 HCOLOR= 3: GOTO 350
520 HOME : TEXT : GOSUB 120
530 POKE - 16304,0: POKE - 16299,0: POKE - 162
97,0
540 GOTO 350
550 HOME
560 TEXT : VTAB 10: HTAB 10: PRINT "GIVE A NAME F
OR THE IMAG": INVERSE : VTAB 20: PRINT "BE GOING T
O STORE* AFTER KEYING RETURN": INPUT N$: FLASH
570 HOME
580 VTAB 15: HTAB 20: PRINT "STORING"
590 : NORMAL
600 F$ = "BSAVE" + N$ + ",A$4000,L$2000"
610 PRINT CHR$ (4);F$
620 VTAB 22: HTAB 10: PRINT "STORE COMPLETED"
630 FOR I = 1 TO 1000: NEXT I
640 GOTO 530
650 HOME : TEXT : NORMAL
660 VTAB 16: HTAB 10: PRINT "WHICH WAY DO YOU SEL
ECT:NORMAL(1)/INVERSE(2)"
670 GET V$: IF V$ = "1" THEN PR# 1: POKE 1913,2:
PRINT CHR$ (17): GOTO 700
680 IF V$ = "2" THEN PR# 1: POKE 1913,98: PRINT
CHR$ (17): GOTO 700
690 GOTO 660

```



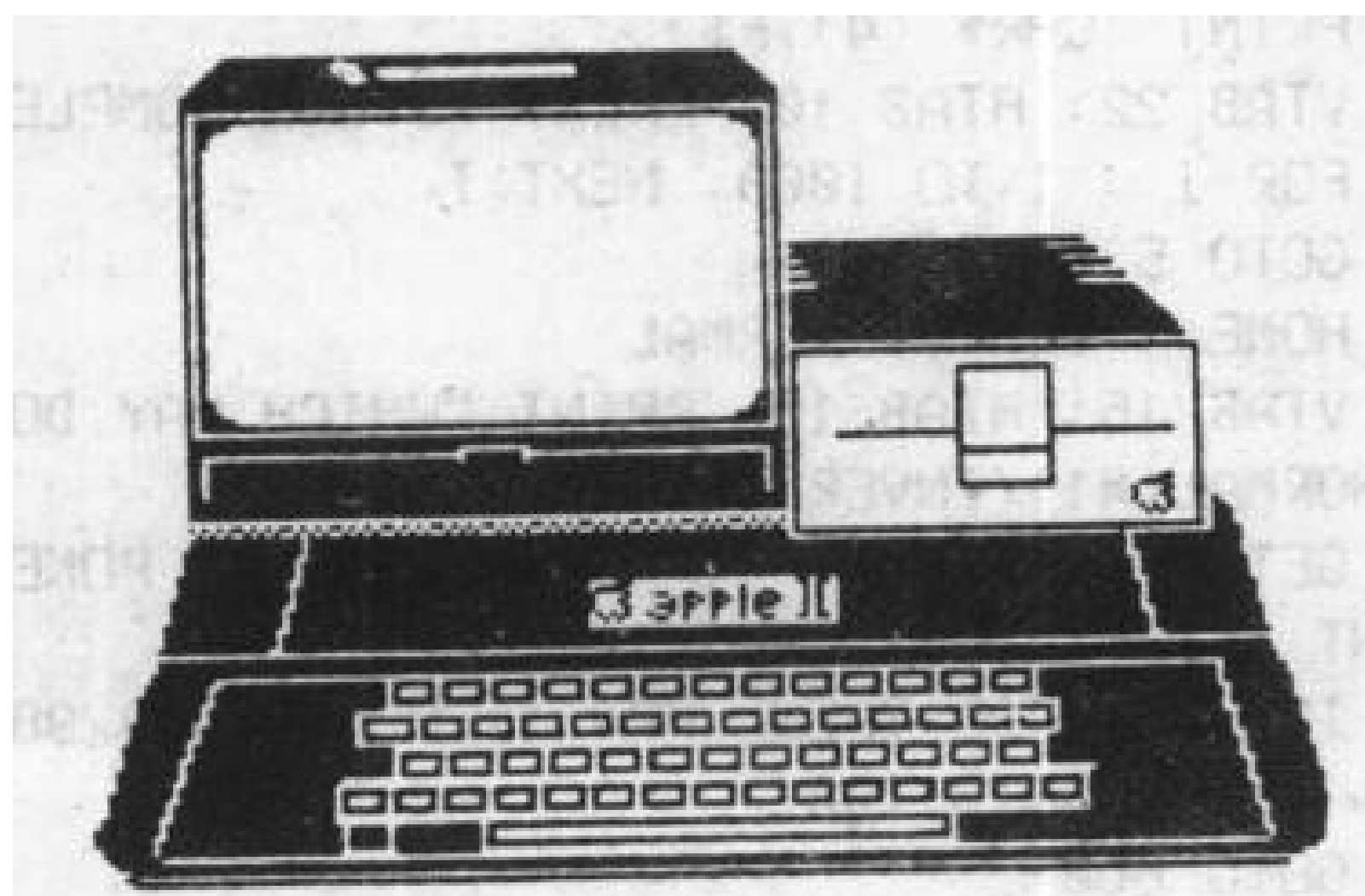
```

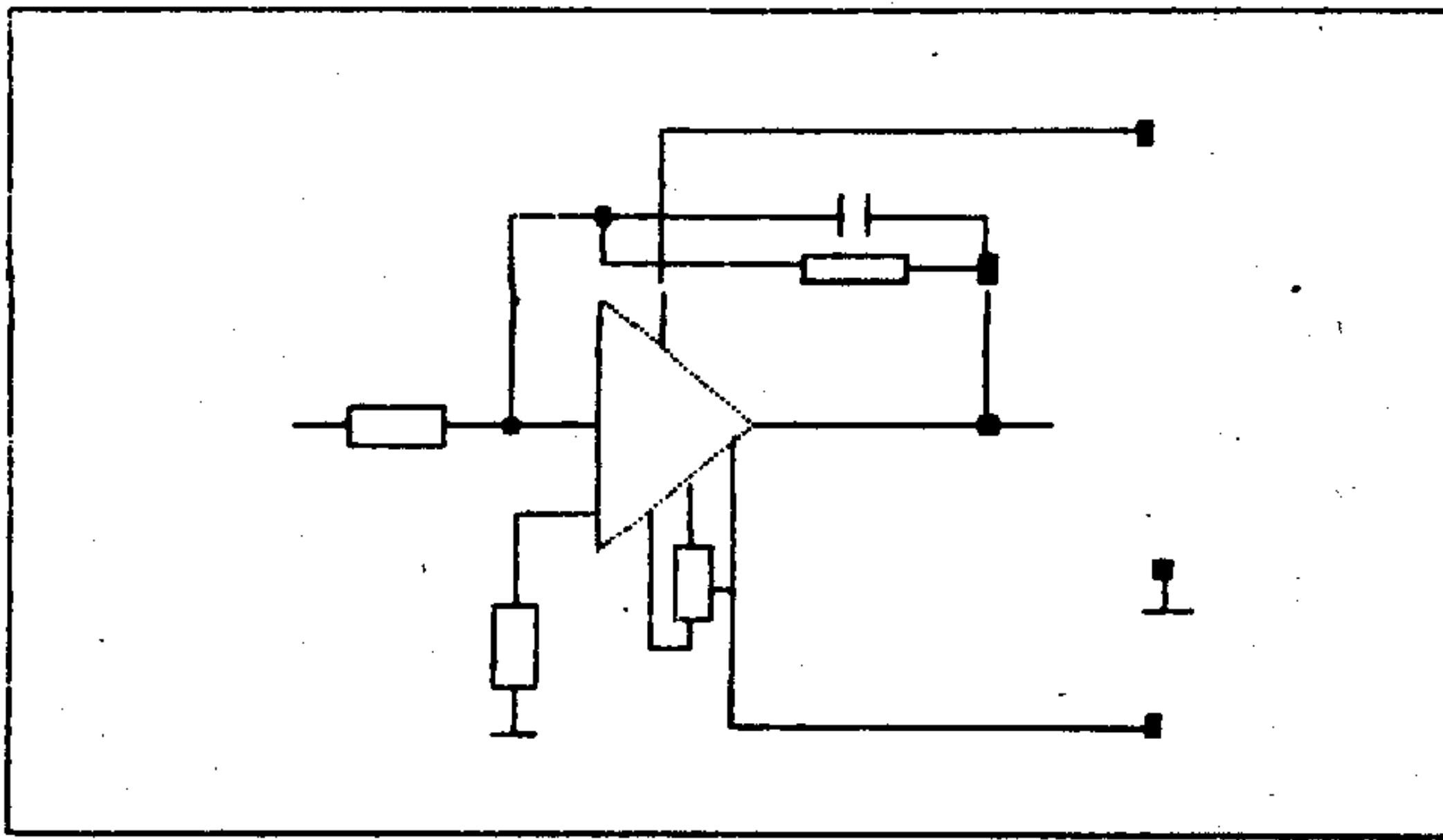
700 VTAB 20: PRINT "HARD COPY COMPLETED"
710 FOR I = 1 TO 1000: NEXT I
720 PR# 0: GOTO 530
730 IF A$ = "I" THEN Y = Y - 1: RETURN
740 IF A$ = "J" THEN X = X - 1: RETURN
750 IF A$ = "K" THEN X = X + 1: RETURN
760 IF A$ = "M" THEN Y = Y + 1: RETURN
770 HOME : TEXT : VTAB 10: HTAB 10
780 PRINT "PLEASE TYPE IN THE IMAG'S NAME"
790 INVERSE : VTAB 20: PRINT "BE GOING TO DISPLAY
    AFTER KEYING RETURN";: NORMAL
800 INPUT M$
810 POKE - 16304,0: POKE - 16299,0: POKE - 162
    97,0
820 PRINT CHR$(4)"BLOAD";M$
830 GOTO 350
840 IF A$ = "Y" THEN X = X - 1:Y = Y - 1: RETURN
850 IF A$ = "G" THEN X = X - 1:Y = Y + 1: RETURN
860 IF A$ = "B" THEN X = X + 1:Y = Y + 1: RETURN
870 IF A$ = "H" THEN X = X + 1:Y = Y - 1: RETURN

```

几个作图实例:

电子技术





11. ASCII 字符造型程序

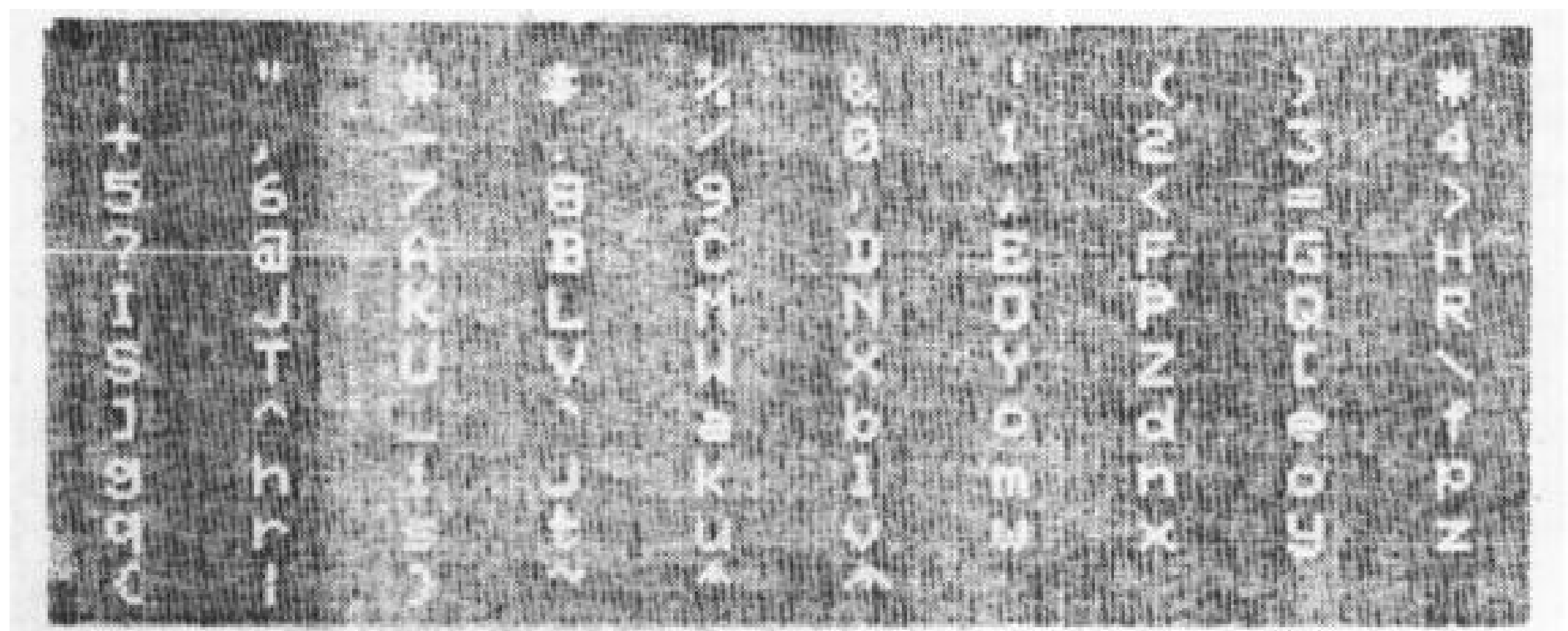
图形表方法，是绘制高清晰图形的又一技巧。这种方法的基本思想是，把图形的轮廓线分解成许多基本矢量的集合，把绘图时笔划移动的过程记录下来，做成图形表（又称造型表）。将造型表放入内存，它就是绘图的依据，用特定的语句和专门的命令，调用造型表，就可以在屏幕上将图形显示出来，而且利用绘图命令，可以使图形旋转、放大。

有关造型表的生成和存贮，造型绘图命令的功能和使用，请参看有关书籍说明，此处从略。

下面提供一个有用的 ASCII 字符造型程序。本程序利用造型表技术，设计了94个 ASCII 字符和两个特殊符号的造型图形，运行本程序后，就在内存中建立了造型表。在有高分辨率作图的程序中，调用本造型表，就可以在屏幕和图形的任意位置，标注说明文字或公式，从而解决高分辨作图和文本方式互不兼容的问题，较好地处理了作图和写字的矛盾“达到”图文并茂”的效果。

程序共分三段：

① 10—270句，是主程序，运行后它调用造型表，并在屏幕上显示信息：



② 280—530句，是造型表子程序。造型表存贮在第二页高分辨率图形区的上部（24576以上的存贮区）。

③ 540—1500句，是存贮的造型。540句DATA中的96，是造型数。550句以后，每一个DATA区只存贮一个造型定义，并以造型结束符8作结尾。

程序清单见PRO-108。

PRO-108

```
10 REM PRO-108
20 REM NUMBERS SHAPE
30 GOSUB 280
40 HGR2 : HCOLOR= 3
50 ROT= 0: SCALE= 1
60 FOR I = 1 TO N
70 IF I > 90 THEN 250
80 IF I > 80 THEN 240
90 IF I > 70 THEN 230
100 IF I > 60 THEN 220
110 IF I > 50 THEN 210
120 IF I > 40 THEN 200
```

```

130 IF I > 30 THEN 190
140 IF I > 20 THEN 180
150 IF I > 10 THEN 170
160 DRAW I AT 22 * I, 10: GOTO 260
170 DRAW I AT 22 * (I - 10), 20: GOTO 260
180 DRAW I AT 22 * (I - 20), 30: GOTO 260
190 DRAW I AT 22 * (I - 30), 40: GOTO 260
200 DRAW I AT 22 * (I - 40), 50: GOTO 260
210 DRAW I AT 22 * (I - 50), 60: GOTO 260
220 DRAW I AT 22 * (I - 60), 70: GOTO 260
230 DRAW I AT 22 * (I - 70), 80: GOTO 260
240 DRAW I AT 22 * (I - 80), 90: GOTO 260
250 DRAW I AT 22 * (I - 90), 100: GOTO 260
260 NEXT I
270 END
280 REM MULTI SHAPE
290 S = 24576: REM FIRST ADDRESS OF SHAPE TABLE
300 S1 = INT (S / 256): S2 = S - S1 * 256
310 POKE 232, S2: POKE 233, S1
320 READ N: K = N: REM SHAPE DEFINIT NUMBERS
330 POKE S, N: POKE S + 1, 0
340 M = S + 2 * (N + 1): S2 = S + 2
350 D = M - S
360 IF D > 255 THEN 390
370 POKE S2, D: POKE S2 + 1, 0
380 GOTO 410
390 D1 = INT (D / 256): D2 = D - D1 * 256
400 POKE S2, D2: POKE S2 + 1, D1
410 S2 = S2 + 2
420 READ A
430 IF A = 8 THEN 500
440 READ
450 IF P = 8 THEN 490
460 X = B * 8 + A
470 POKE M, X: M = M + 1
480 GOTO 420
490 POKE M, A: M = M + 1
500 POKE M, 0: M = M + 1
510 K = K - 1
520 IF K = 0 THEN RETURN
530 GOTO 350

```

540 DATA 96
 550 DATA 1,1,6,6,6,2,2,6,8: REM "1"
 560 DATA 1,6,6,5,1,4,4,4,8: REM ""
 570 DATA 1,5,1,6,7,3,7,2,5,5,5,5,6,3,7,3,7,2,5,5,
 5,5,6,3,7,3,6,5,1,5,8: REM "#"
 580 DATA 1,5,1,2,7,7,7,7,2,5,1,5,2,7,7,6,1,5,1,6,
 3,7,7,7,6,1,1,6,8: REM "\$"
 590 DATA 5,6,7,5,1,1,1,6,3,6,3,6,3,7,2,5,1,1,5,6,
 7,7,8: REM "X"
 600 DATA 1,5,2,7,3,6,5,1,6,3,7,2,5,1,5,1,6,3,7,3,
 3,6,1,5,5,1,5,8: REM "&"
 610 DATA 1,1,1,6,6,6,8: REM ""
 620 DATA 1,1,1,1,6,3,6,3,6,6,5,2,5,2,5,8: REM "("
 630 DATA 1,6,1,6,1,6,6,6,3,6,3,6,8: REM ")"
 640 DATA 1,1,5,1,2,7,3,7,3,6,1,5,5,5,2,7,7,7,7,6,
 1,5,5,5,2,7,3,7,3,6,1,1,6,8: REM "*"
 650 DATA 2,2,2,5,5,4,4,5,1,2,2,7,6,3,6,6,8: REM "+"
 660 DATA 2,2,2,2,2,1,6,6,3,6,8: REM ","
 670 DATA 2,2,2,5,5,5,5,5,8: REM "-"
 680 DATA 2,2,2,2,2,2,1,6,8: REM "."
 690 DATA 1,1,1,1,6,3,6,3,6,3,6,3,6,8: REM "/"
 700 DATA 1,5,5,5,2,6,6,6,6,6,3,7,7,7,0,4,4,4,4,5,
 1,1,2,6,3,6,3,6,8: REM "0"
 710 DATA 1,1,6,7,6,1,6,6,6,6,7,5,5,5,8: REM "1"
 720 DATA 2,5,0,5,5,5,2,6,6,3,7,7,7,2,6,6,5,5,5,5,
 5,8: REM "2"
 730 DATA 5,5,5,5,6,3,7,2,5,2,5,2,6,6,3,7,7,7,0,4,
 8 : REM "3"
 740 DATA 2,2,2,6,5,5,6,2,1,4,4,5,4,3,4,4,4,7,2,7,
 2,7,8: REM "4"
 750 DATA 5,5,5,5,6,3,3,3,3,6,5,5,5,5,2,6,6,6,3,7,
 7,7,0,5,8: REM "5"
 760 DATA 1,1,5,6,3,3,6,3,6,5,5,5,5,2,6,6,3,7,7,7,
 0,4,4,8: REM "6"
 770 DATA 5,5,5,5,6,6,3,6,3,6,3,6,3,6,5,8: REM "7"
 780 DATA 1,5,5,5,2,6,6,2,6,6,3,7,7,7,0,4,4,1,5,5,
 4,3,3,3,4,4,8: REM "8"
 790 DATA 1,5,5,5,2,6,7,3,3,3,4,5,2,2,5,5,5,6,6,3,
 6,3,7,7,8: REM "9"
 800 DATA 2,2,2,1,6,2,6,8: REM ":"
 810 DATA 2,2,1,1,6,2,6,6,3,6,8: REM ";"

820 DATA 1,1,1,6,3,6,3,6,3,6,1,5,2,5,2,5,8:"<"
 830 DATA 2,2,5,5,5,5,6,2,7,7,7,7,7,8:"=" "
 840 DATA 1,6,1,6,1,6,1,6,3,6,3,6,3,6,8:">" "
 850 DATA 2,5,0,5,5,5,2,6,6,3,6,3,6,2,6,8:"?" "
 860 DATA 2,5,0,5,5,5,6,6,6,6,6,6,7,7,7,4,3,4,4,1,
 5,6,6,6,8:"e" "
 870 DATA 1,1,5,2,7,3,7,2,6,6,6,6,5,1,1,1,4,4,4,4,
 7,2,2,7,7,7,8:"A" "
 880 DATA 5,5,5,5,2,7,3,3,6,5,1,1,6,3,7,7,6,6,7,2,
 5,5,5,5,0,4,4,8:"B" "
 890 DATA 1,5,5,5,2,6,2,2,2,6,3,7,7,7,0,4,4,4,4,4,
 8:"C" "
 900 DATA 5,5,5,5,2,6,6,6,6,6,6,3,7,7,7,4,1,4,4,4,4,
 4,8:"D" "
 910 DATA 5,5,5,5,6,3,3,3,3,6,6,5,5,5,6,3,3,3,6,6,
 5,5,5,5,5,8:"E" "
 920 DATA 5,5,5,5,6,3,3,3,3,6,6,5,5,5,6,3,3,3,6,6,
 5,8:"F" "
 930 DATA 1,5,5,5,6,2,2,6,6,6,7,7,7,4,3,4,4,4,4,5,
 1,1,2,2,5,5,8:"G" "
 940 DATA 6,6,6,6,6,6,5,1,1,1,4,4,4,4,4,4,7,2,2,2,
 7,7,7,8:"H" "
 950 DATA 1,5,5,6,3,6,6,6,6,7,2,5,5,5,8:"I" "
 960 DATA 1,1,1,1,6,6,6,6,6,6,3,7,7,7,0,4,8:"J" "
 970 DATA 6,6,6,6,6,6,5,1,1,1,4,3,4,3,4,3,4,1,4,1,
 4,1,4,,8:"K" "
 980 DATA 6,6,6,6,6,6,5,5,5,5,5,8:"L" "
 990 DATA 6,6,6,6,6,6,5,1,1,1,4,4,4,4,4,4,7,2,7,3,
 6,1,5,8:"M" "
 1000 DATA 6,6,6,6,6,6,5,1,1,1,4,4,4,4,4,4,7,3,3,2,
 ,6,1,6,1,6,8:"N" "
 1010 DATA 1,5,5,5,2,6,6,6,6,6,3,7,7,7,0,4,4,4,4,4,
 ,8:"O" "
 1020 DATA 5,5,5,5,2,6,6,3,7,7,7,4,4,5,2,2,2,3,6,6,
 ,6,8:"P" "
 1030 DATA 1,5,5,5,2,6,6,7,3,3,3,4,4,5,2,2,2,3,5,1,
 ,5,1,6,3,7,3,3,6,1,5,5,1,5,8:"Q" "
 1040 DATA 5,5,5,5,2,6,6,3,7,7,7,4,4,5,2,2,2,3,6,6,
 ,5,1,1,1,4,3,4,3,4,8:"R" "
 1050 DATA 1,5,5,5,2,7,3,3,3,6,6,1,5,5,5,2,6,7,3,3,
 ,3,6,1,5,5,5,8:"S" "

1060 DATA 5,5,5,5,6,3,3,6,6,6,6,6,6,8:"I"
 1070 DATA 6,6,6,6,6,5,1,1,1,4,4,4,4,4,7,2,2,2,2,2,2,7,7,7,8:"U"
 1080 DATA 6,6,5,1,1,1,4,4,7,2,2,2,7,3,6,5,1,6,3,6,6,8:"V"
 1090 DATA 6,6,6,6,5,1,5,1,6,7,7,3,7,6,5,1,1,1,4,4,4,4,4,4,8:"W"
 1100 DATA 6,5,1,1,1,4,7,2,2,7,3,6,1,5,2,7,3,7,2,6,5,1,1,1,4,4,8:"X"
 1110 DATA 6,5,1,1,1,4,7,2,2,7,3,6,1,6,6,6,6,8:"Y"
 1120 DATA 5,5,5,5,6,7,2,7,2,7,2,7,2,6,5,5,5,5,5,8:"Z"
 1130 DATA 1,5,5,6,3,3,6,6,6,6,6,5,5,5,8:"["
 1140 DATA 2,5,2,5,2,5,2,5,2,5,8:"\
 1150 DATA 1,5,5,6,6,6,6,6,6,7,7,7,8:"]"
 1160 DATA 1,1,5,2,7,3,7,2,5,1,1,1,5,8:"~"
 1170 DATA 2,2,2,2,2,2,5,5,5,5,5,8:"_
 1180 DATA 1,1,5,2,5,8:"`"
 1190 DATA 2,2,1,5,5,5,2,6,7,7,7,7,2,6,1,5,5,5,4,4,8:"a"
 1200 DATA 6,6,5,1,5,5,2,7,3,3,7,6,5,1,1,1,6,7,3,3,7,6,5,1,5,5,8:"b"
 1210 DATA 2,1,5,5,5,2,7,3,3,3,6,6,5,1,1,1,6,3,7,7,7,8:"c"
 1220 DATA 1,1,1,1,6,6,7,3,7,7,2,5,1,1,5,6,7,3,3,3,6,5,1,1,5,6,7,3,7,7,8:"d"
 1230 DATA 2,2,1,5,5,5,2,7,3,3,3,6,5,5,5,5,6,3,3,3,3,6,1,5,5,5,8:"e"
 1240 DATA 1,1,1,5,2,7,3,7,2,5,5,6,3,6,6,6,6,8:"f"
 1250 DATA 2,1,5,5,1,6,7,7,3,3,6,5,1,1,5,6,7,3,7,7,2,1,1,1,1,6,7,3,3,3,6,1,5,5,5,8:"g"
 1260 DATA 6,6,5,1,5,5,2,6,6,6,7,3,3,3,4,4,4,5,5,8:"h"
 1270 DATA 1,1,6,3,2,5,6,6,6,7,2,5,5,5,8:"i"
 1280 DATA 1,1,1,6,2,6,6,6,7,3,3,3,6,1,5,5,5,8:"j"
 1290 DATA 6,6,5,1,1,6,3,7,3,6,5,5,2,7,3,6,5,1,1,5,8:"k"
 1300 DATA 1,5,6,6,6,6,6,7,2,5,5,5,8:"l"
 1310 DATA 2,2,5,5,1,5,2,7,3,7,3,6,5,1,5,1,6,7,3,7,3,6,5,1,5,1,5,8:"m"

```

1320 DATA 2,2,5,1,5,5,2,7,3,3,7,6,6,6,5,1,1,1,4,4
,4,4,8:"n"
1330 DATA 2,2,1,5,5,5,2,6,6,6,3,7,7,7,0,4,4,4,8:"
o"
1340 DATA 2,5,1,5,5,2,7,3,3,7,6,5,1,1,1,6,7,3,3,7
,6,5,1,5,5,2,,3,3,3,3,6,6,6,8:"p"
1350 DATA 2,1,5,5,1,6,7,7,3,3,6,5,1,1,1,6,7,7,3,3
,6,1,5,5,1,6,6,6,8:"q"
1360 DATA 2,2,5,1,5,5,2,7,3,3,7,6,6,6,6,8:"r"
1370 DATA 2,2,1,5,5,5,6,3,3,3,3,6,1,5,5,5,2,6,3,7
,7,7,7,8:"s"
1380 DATA 1,6,5,1,2,7,7,7,6,1,6,6,6,1,4,1,5,8:"t"
1390 DATA 2,2,6,6,6,6,1,5,5,1,4,7,4,1,4,4,4,8:"u"
1400 DATA 2,2,6,6,6,1,6,1,4,1,4,1,4,4,4,8:"v"
1410 DATA 2,2,6,6,6,6,1,5,1,5,0,7,3,4,5,1,4,4,4,8
:"w"
1420 DATA 2,2,5,1,1,1,6,3,7,3,6,1,5,2,7,3,7,2,5,1
,1,1,5,8:"x"
1430 DATA 2,6,6,6,2,2,6,1,5,5,4,1,4,4,4,4,4,7,2,2
,2,7,7,7,8:"y"
1440 DATA 2,2,5,5,5,5,6,3,6,3,6,3,7,2,5,5,5,5,8
:"z"
1450 DATA 1,1,1,5,6,3,3,6,7,2,6,1,6,6,1,5,5,8:"["
1460 DATA 1,1,6,6,6,2,6,6,6,8:" "
1470 DATA 1,5,5,2,6,6,1,6,3,6,6,3,7,7,8:"]"
1480 DATA 2,4,1,6,1,5,1,6,3,7,8:" "
1490 DATA 2,2,5,1,5,1,4,3,7,7,4,1,6,8:" "
1500 DATA 1,1,1,5,2,7,3,7,2,5,1,5,1,5,2,7,3,3,7,3
,3,7,8

```

12. 电路图绘制

这里介绍一个绘制电路图的实用程序，它可作为电子线路教学的辅助演示，是计算机辅助教学的一种尝试。

利用 APPLE-II 高分辨率作图功能，可以绘制各种各样的模拟电路和数字电路的原理图或逻辑符号。利用造型表技术，在电路的适当位置加注元件参数、文字符号、计算公

式、布尔表达式等。从而在电子线路的电化教学中，做到文图并茂，生动活泼。

本程序共画了与门、或非门、或门、与或非门、异或门、R-S触发器、共发射放大电路等8个电路。实际上以本程序为模式，可以绘制各种各样的电子线路。此外，还可以绘制程序设计框图、印刷线路板等。

为了使用前节介绍的造型表，首先将它存入磁盘备用，取名为PRO-108。

在绘制电路图程序PRO-109开始部份，必须安排以下两个语句：

```
POKE 232, 0
```

```
POKE 233, 96
```

这是因为232和233单元是专门用来存放图形表的首地址的。

从前节程序中知道，图形表的首地址是十进制的24576，对应的十六进制为\$6000，POKE 232后面应放首地址的低字节（十进制数），所以是0；POKE 233后面应放首地址的高字节（十进制数）96，96是十六进制60的十进制表示。

只有这样，将图形表送入内存，并将其首地址记入232和233号单元，上述图形表才能够使用。

另外，为了在电路图的适当位置加注文字说明，在PRO-109程序中安排了诸如X\$ = "P" : XZ = 110 : YZ = 0 : GOSUB 2000 等语句，这个意思是在X = 110, Y = 0的坐标位置上，标注字符P，2000以后的语句，是计算机自动绘出X\$中的字符。

运行本程序的方法：

① 首先运行造型表

RUN PRO-108 ✓

稍等片刻，出现ASCII字符表。

② 调入绘制电路图程序并运行

RUN PRO-109 ✓

③ 出现? 后，可键入1—8中任一数字，计算机自动绘图。

④ 出现“DO-YOU-WANT-PRINT (Y/N)”，若要打印，按Y键，反之按N键或空格键。

⑤ 停机按9 ✓。

程序PRO-109的清单和运行结果。

PRO-109

```
10 REM PRO-109
20 POKE 232,0: POKE 233,96
30 INPUT X
40 IF X = 9 THEN END
50 ON X GOSUB 70,190,340,480,620,830,940,1170,900
60 GOTO 30
70 HGR : HCOLOR= 3
80 HPLOT 110,8 TO 110,50
90 HPLOT 50,50 TO 170,50 TO 170,100 TO 50,100 TO
50,50
100 HPLOT 80,100 TO 80,150
110 HPLOT 140,100 TO 140,150
120 X$ = "P":XZ = 110:YZ = 0: GOSUB 1690
130 X$ = "A":XZ = 80:YZ = 155: GOSUB 1690
140 X$ = "B":XZ = 140:YZ = 155: GOSUB 1690
150 X$ = "AND-CIRCUIT":XZ = 75:YZ = 75: GOSUB 1690
160 X$ = "P=A.B":XZ = 20:YZ = 20: GOSUB 1690
170 GOSUB 1610
180 RETURN
190 HGR : HCOLOR= 3
200 HPLOT 110,8 TO 110,44
210 X$ = "O":XZ = 111:YZ = 43: GOSUB 1690
220 HPLOT 50,50 TO 170,50 TO 170,100 TO 50,100 TO
```

```

50,50
230 HPLOT 80,100 TO 80,150
240 HPLOT 140,100 TO 140,150
250 X$ = "+":XZ = 110:YZ = 75: GOSUB 1690
260 X$ = "P":XZ = 110:YZ = 0: GOSUB 1690
270 X$ = "A":XZ = 80:YZ = 155: GOSUB 1690
280 X$ = "B":XZ = 140:YZ = 155: GOSUB 1690
290 X$ = "P=A+B":XZ = 20:YZ = 20: GOSUB 1690
300 HPLOT 30,18 TO 50,18
310 X$ = "NOR-CIRCUIT":XZ = 73:YZ = 85: GOSUB 1690
320 GOSUB 1610
330 RETURN
340 HGR : HCOLOR= 3
350 HPLOT 110,8 TO 110,50
360 HPLOT 50,50 TO 170,50 TO 170,100 TO 50,100 TO
50,50
370 HPLOT 80,100 TO 80,150
380 HPLOT 140,100 TO 140,150
390 HPLOT 106,75 TO 114,75
400 HPLOT 110,70 TO 110,80
410 X$ = "P":XZ = 110:YZ = 0: GOSUB 1690
420 X$ = "A":XZ = 80:YZ = 155: GOSUB 1690
430 X$ = "B":XZ = 140:YZ = 155: GOSUB 1690
440 X$ = "OR-CIRCUIT":XZ = 75:YZ = 85: GOSUB 1690
450 X$ = "P=A+B":XZ = 20:YZ = 20: GOSUB 1690
460 GOSUB 1610
470 RETURN
480 HGR : HCOLOR= 3
490 HPLOT 110,8 TO 110,40
500 HPLOT 108,40 TO 112,40 TO 112,50 TO 108,50 TO
108,40
510 HPLOT 50,50 TO 170,50 TO 170,100 TO 50,100 TO
50,50
520 HPLOT 80,100 TO 80,150
530 HPLOT 140,100 TO 140,150
540 X$ = "P":XZ = 110:YZ = 0: GOSUB 1690
550 X$ = "A":XZ = 80:YZ = 155: GOSUB 1690
560 X$ = "B":XZ = 140:YZ = 155: GOSUB 1690
570 X$ = "NAND-CIRCUIT":XZ = 74:YZ = 75: GOSUB 169
0
580 X$ = "P=A.B":XZ = 20:YZ = 20: GOSUB 1690

```

```

590 HPLLOT 30,18 TO 50,18
600 GOSUB 1610
610 RETURN
620 HGR : HCOLOR= 3
630 HPLLOT 110,8 TO 110,40
640 HPLLOT 108,40 TO 112,40 TO 112,50 TO 108,50 TO
    108,40
650 HPLLOT 50,50 TO 170,50 TO 170,100 TO 50,100 TO
    50,50
660 HPLLOT 50,100 TO 170,100 TO 170,150 TO 50,150
    TO 50,100
670 HPLLOT 110,100 TO 110,150
680 HPLLOT 70,150 TO 70,170
690 HPLLOT 90,150 TO 90,170
700 HPLLOT 130,150 TO 130,170
710 HPLLOT 150,150 TO 150,170
720 HPLLOT 106,75 TO 114,75
730 HPLLOT 110,70 TO 110,80
740 X$ = "P":XZ = 110:YZ = 0: GOSUB 1690
750 X$ = "A":XZ = 70:YZ = 175: GOSUB 1690
760 X$ = "B":XZ = 90:YZ = 175: GOSUB 1690
770 X$ = "C":XZ = 130:YZ = 175: GOSUB 1690
780 X$ = "D":XZ = 150:YZ = 175: GOSUB 1690
790 X$ = "ANDR-CIRCUIT":XZ = 70:YZ = 85: GOSUB 169
    0
800 X$ = "P=AB+CD":XZ = 20:YZ = 20: GOSUB 1690
810 HPLLOT 30,18 TO 63,18: GOSUB 1610
820 RETURN
830 HGR : HCOLOR= 3
840 HPLLOT 110,8 TO 110,50
850 HPLLOT 50,50 TO 170,50 TO 170,100 TO 50,100 TO
    50,50
860 HPLLOT 80,100 TO 80,150
870 HPLLOT 140,100 TO 140,150
880 HPLLOT 106,75 TO 114,75
890 HPLLOT 110,70 TO 110,80
900 HPLLOT 104,65 TO 116,65 TO 116,85 TO 104,85 TO
    104,65
910 X$ = "EXOR-CIRCUIT":XZ = 73:YZ = 88: GOSUB 169
    0
920 GOSUB 1610

```

```

930 RETURN
940 HGR : HCOLOR= 3
950 HPLOT 70,20 TO 70,50
960 HPLOT 150,20 TO 150,50
970 HPLOT 50,50 TO 170,50 TO 170,100 TO 50,100 TO
    50,50
980 HPLOT 70,100 TO 70,140
990 HPLOT 150,100 TO 150,140
1000 HPLOT 110,100 TO 110,167
1010 HPLOT 104,100 TO 110,95 TO 116,100
1020 HPLOT 68,145 TO 72,145
1030 HPLOT 68,145 TO 68,149 TO 72,149 TO 72,153 T
    O 68,153
1040 HPLOT 148,145 TO 152,145 TO 152,149 TO 148,1
    49 TO 152,153
1050 HPLOT 148,145 TO 148,153
1060 HPLOT 104,170 TO 108,170
1070 HPLOT 104,170 TO 104,178 TO 108,178
1080 HPLOT 110,170 TO 110,178
1090 HPLOT 110,170 TO 114,170 TO 114,174 TO 110,1
    74
1100 HPLOT 68,60 TO 72,60 TO 72,68 TO 68,68 TO 68
    ,60
1110 HPLOT 70,64 TO 74,68
1120 HPLOT 148,60 TO 152,60 TO 152,68 TO 148,68 T
    O 148,60
1130 HPLOT 148,57 TO 152,57: HPLOT 150,64 TO 154,
    68
1140 X$ = "R-S-FLIP-FLOPS":XZ = 68:YZ = 85: GOSUB
    1690
1150 GOSUB 1620
1160 RETURN
1170 HGR : HCOLOR= 3
1180 HPLOT 70,10 TO 70,30
1190 HPLOT 68,30 TO 72,30 TO 72,50 TO 68,50 TO 68
    ,30
1200 HPLOT 70,50 TO 70,70
1210 HPLOT 68,70 TO 72,70 TO 72,90 TO 68,90 TO 68
    ,70
1220 HPLOT 70,90 TO 70,110
1230 HPLOT 20,60 TO 40,60: HPLOT 45,60 TO 110,60

```

```

1240 HPLOT 40,55 TO 40,65: HPLOT 45,55 TO 45,65
1250 HPLOT 110,52 TO 110,68: HPLOT 110,60 TO 120,
52: HPLOT 110,60 TO 120,68
1260 HPLOT 120,68 TO 118,62: HPLOT 120,68 TO 118,
70
1270 HPLOT 120,68 TO 120,78: HPLOT 118,78 TO 122,
78 TO 122,88 TO 118,88 TO 118,78
1280 HPLOT 120,89 TO 120,110
1290 HPLOT 120,72 TO 135,72 TO 135,85
1300 HPLOT 132,85 TO 138,85: HPLOT 132,90 TO 138,
90
1310 HPLOT 135,91 TO 135,110
1320 HPLOT 20,110 TO 210,110
1330 HPLOT 70,10 TO 210,10
1340 HPLOT 120,50 TO 120,40
1350 HPLOT 118,40 TO 122,40 TO 122,25 TO 118,25 T
O 118,40
1360 HPLOT 120,25 TO 120,10
1370 HPLOT 120,45 TO 140,45: HPLOT 140,40 TO 140,
50
1380 HPLOT 145,40 TO 145,50
1390 HPLOT 145,45 TO 170,45
1400 HPLOT 170,45 TO 170,65
1410 HPLOT 168,65 TO 172,65 TO 172,85 TO 168,85 T
O 168,65
1420 HPLOT 170,85 TO 170,110
1430 X$ = "o":XZ = 20:YZ = 57: GOSUB 1690
1440 X$ = "o":XZ = 20:YZ = 107: GOSUB 1690
1450 X$ = "o":XZ = 210:YZ = 7: GOSUB 1690
1460 X$ = "Vi":XZ = 20:YZ = 80: GOSUB 1690
1470 X$ = "C1":XZ = 40:YZ = 45: GOSUB 1690
1480 X$ = "R1":XZ = 80:YZ = 33: GOSUB 1690
1490 X$ = "R2":XZ = 80:YZ = 75: GOSUB 1690
1500 X$ = "Rc":XZ = 130:YZ = 28: GOSUB 1690
1510 X$ = "C2":XZ = 150:YZ = 31: GOSUB 1690
1520 X$ = "Re":XZ = 107:YZ = 80: GOSUB 1690
1530 X$ = "Ce":XZ = 150:YZ = 85: GOSUB 1690
1540 X$ = "RL":XZ = 185:YZ = 75: GOSUB 1690
1550 X$ = "+Ec":XZ = 213:YZ = 17: GOSUB 1690:X$ =
"Vo":XZ = 213:YZ = 75: GOSUB 1690
1560 HPLOT 170,45 TO 210,45

```

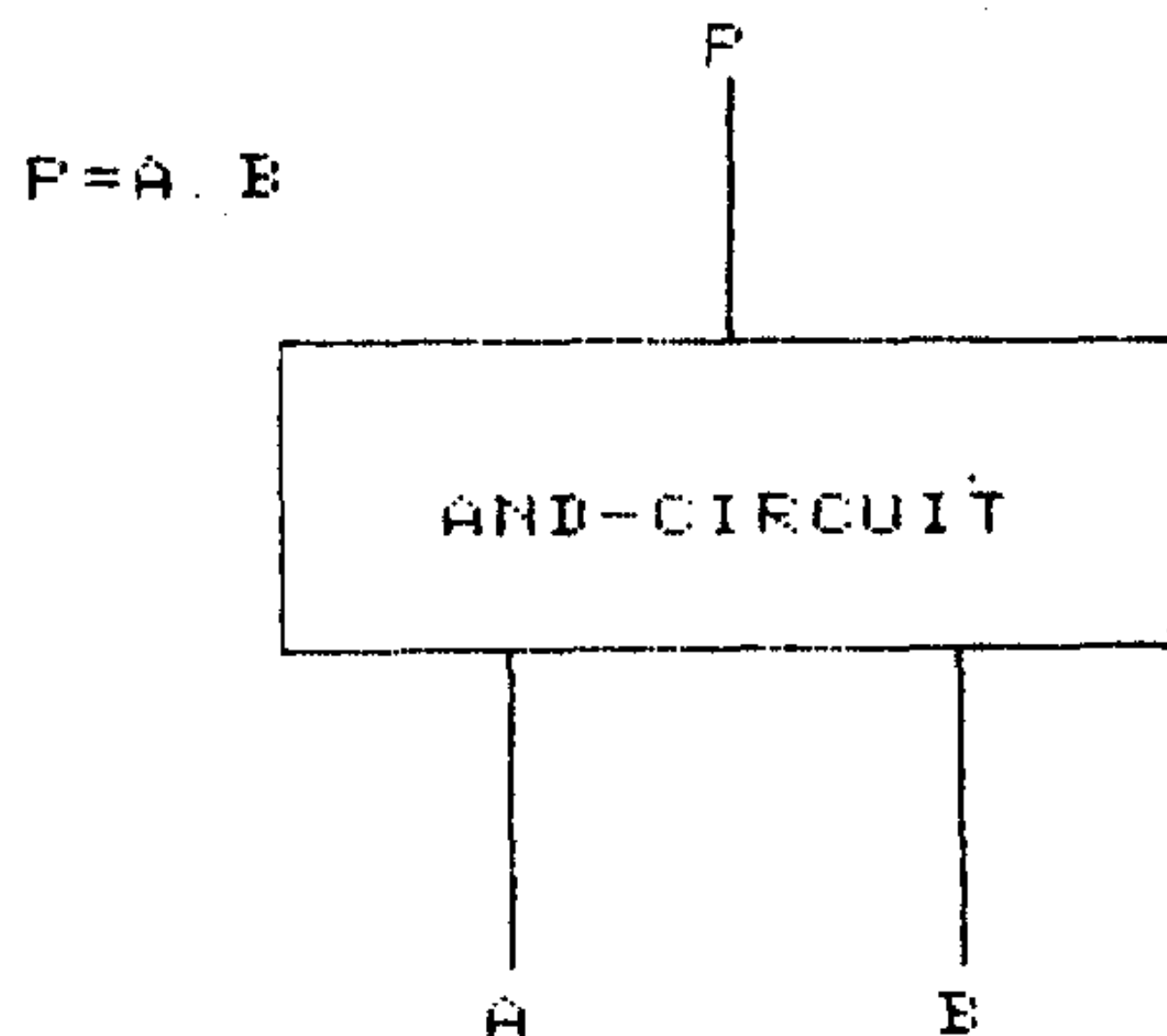
```

1570 X$ = "o":XZ = 214:YZ = 42: GOSUB 1690
1580 X$ = "o":XZ = 214:YZ = 107: GOSUB 1690
1590 GOSUB 1620
1600 RETURN
1610 X$ = "DO-YOU-WANT-PRINT(Y/N)":XZ = 120:YZ = 3
5: GOSUB 1690
1620 GOTO 1640
1630 X$ = "DO-YOU-WANT-TO-PRINT(Y/N)":XZ = 70:YZ =
20: GOSUB 1690
1640 GET A$: IF A$ = "N" OR A$ = " " THEN TEXT :
HOME : PRINT "GOOD BYE!": RETURN
1650 IF A$ < > "Y" THEN PRINT CHR$ (7): RETURN

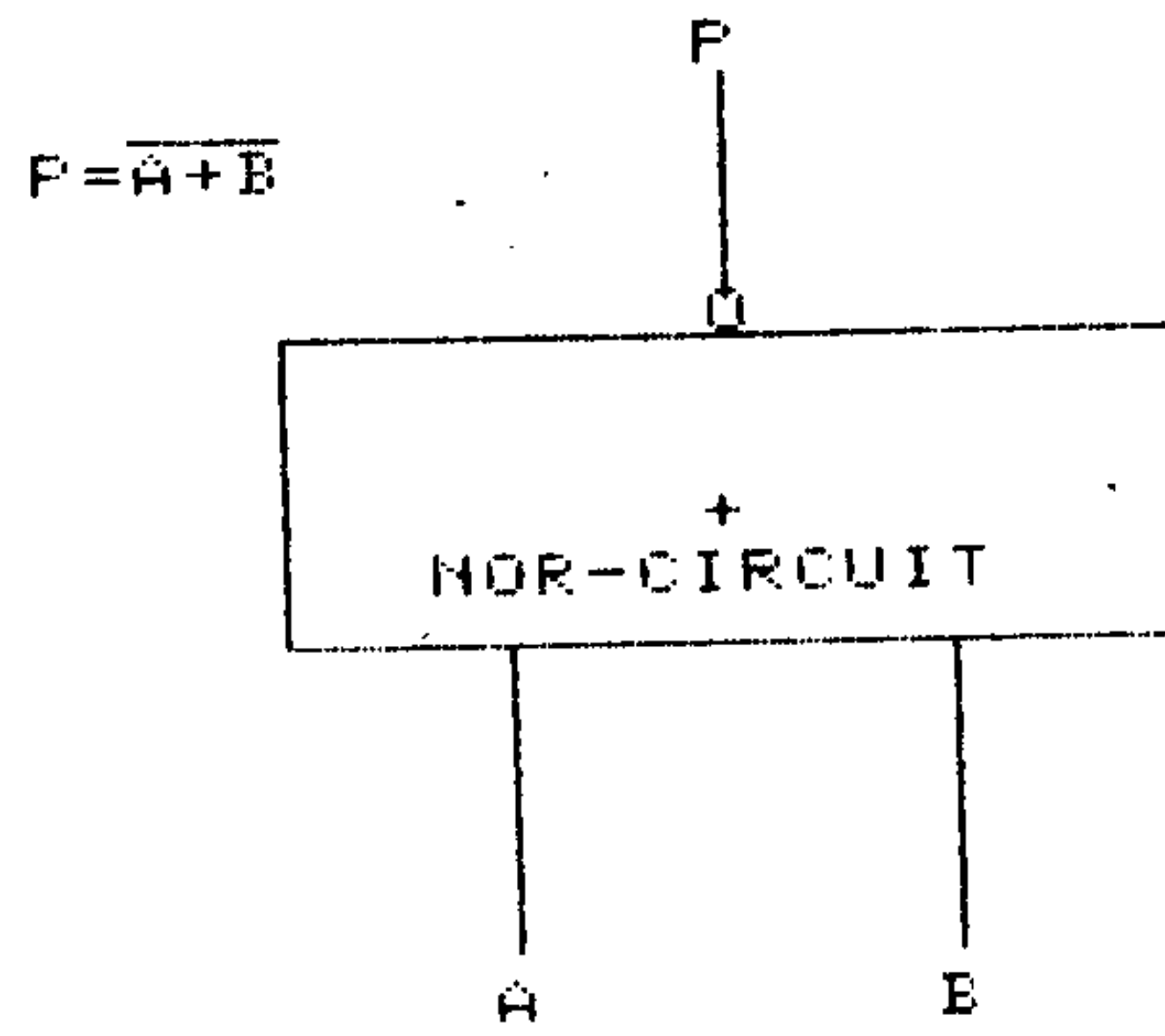
1660 HCOLOR= 0:X$ = "DO-YOU-WANT-PRINT(Y/N)":XZ =
120:YZ = 35: GOSUB 1690
1670 PR# 1: PRINT CHR$ (4): POKE 1913,1
1680 POKE 1913,1: PRINT CHR$ (17): PR# 0: FOR K
= 0 TO 10: PRINT CHR$ (7): NEXT K: HCOLOR= 3: RET
URN
1690 ROT= 0:XX = LEN (X$): IF XX = 3 THEN XZ = X
Z - 11: GOTO 1720
1700 IF XX = 2 THEN XZ = XZ - 4: GOTO 1720
1710 XZ = XZ - 3
1720 FOR I1 = 0 TO XX - 1:AS = ASC ( MID$ (X$,I1
+ 1,1)) - 32: DRAW AS AT XZ + I1 * 7,YZ: NEXT I1:
RETURN

```

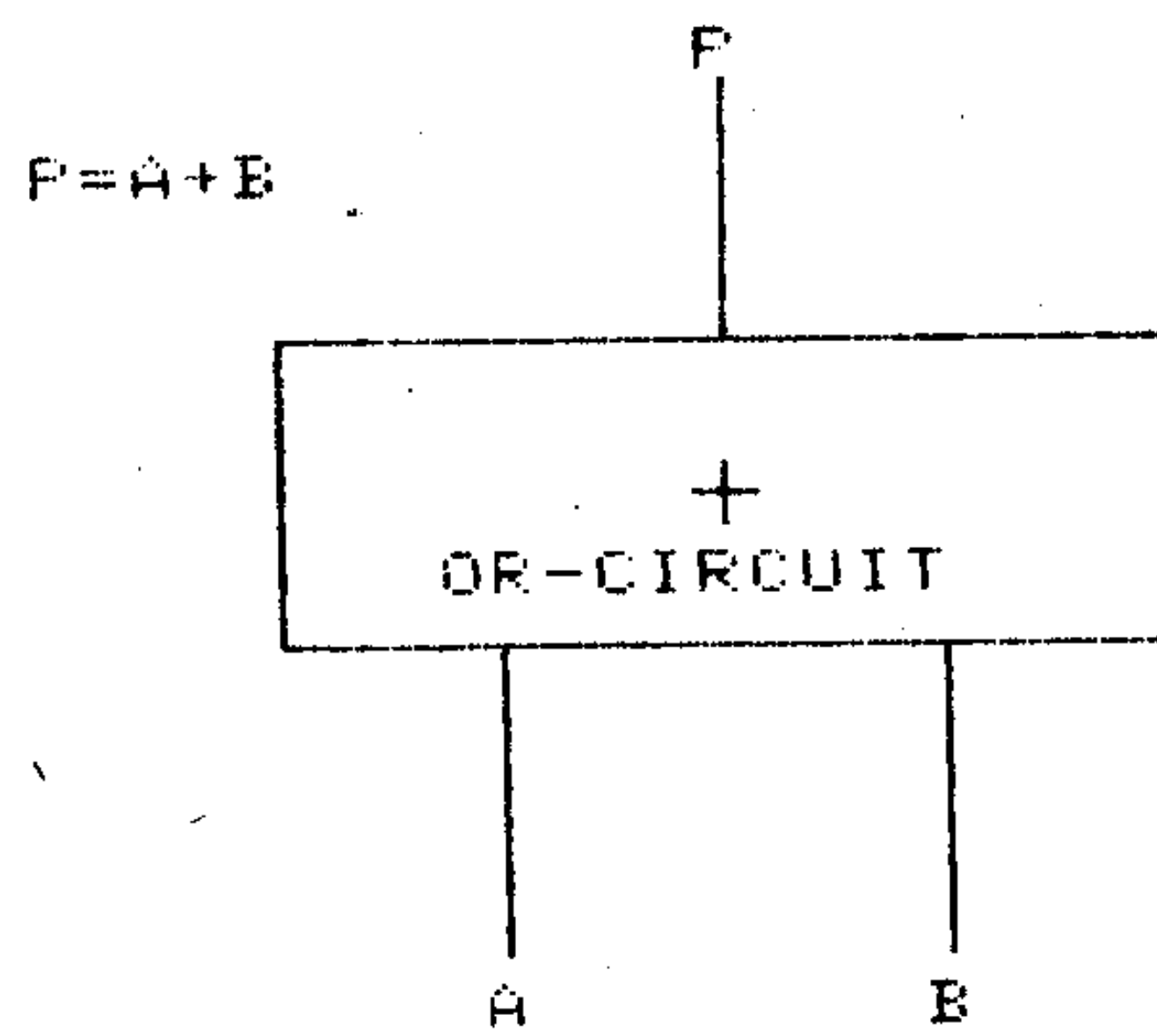
URUN
 ?1



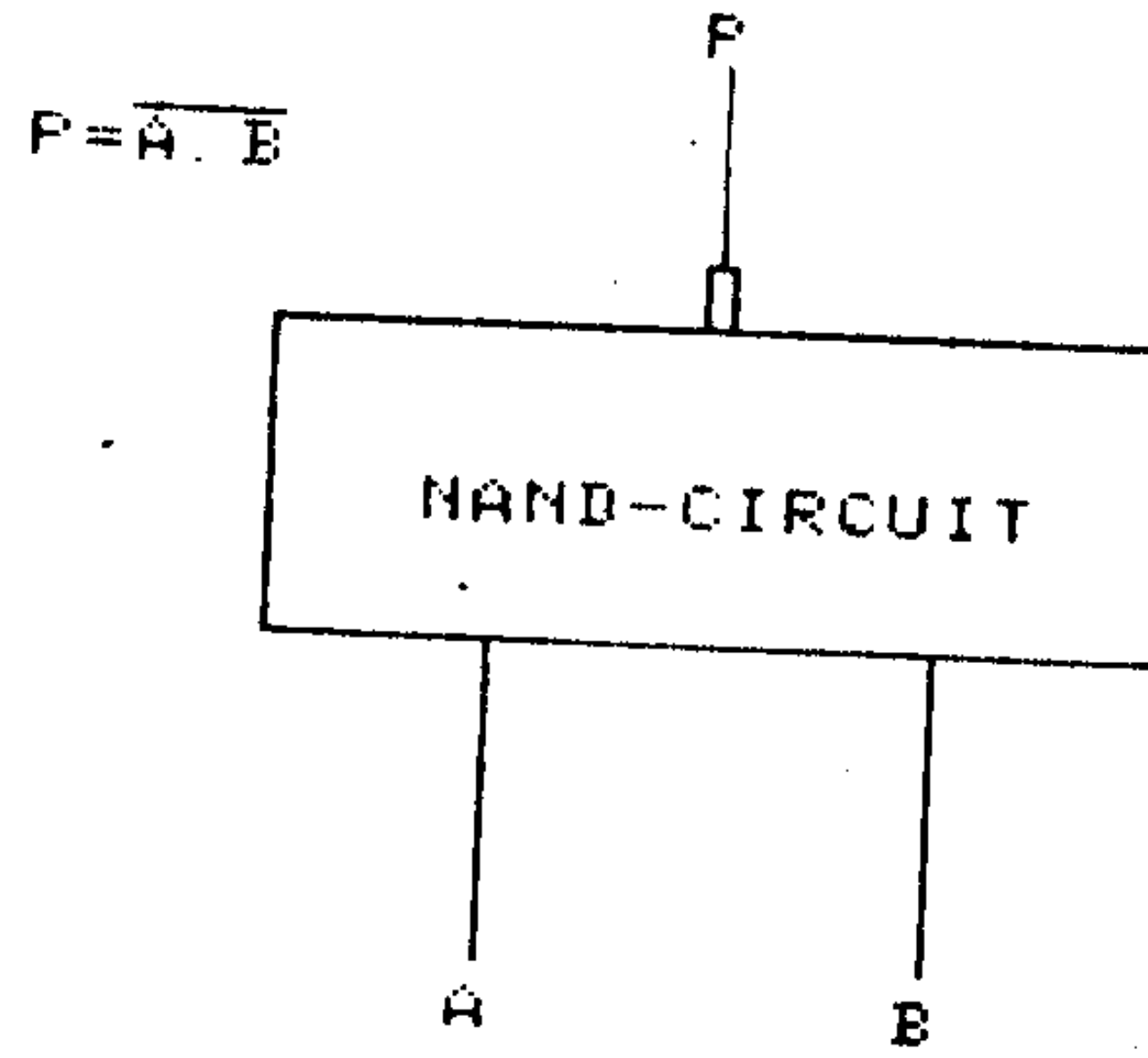
2



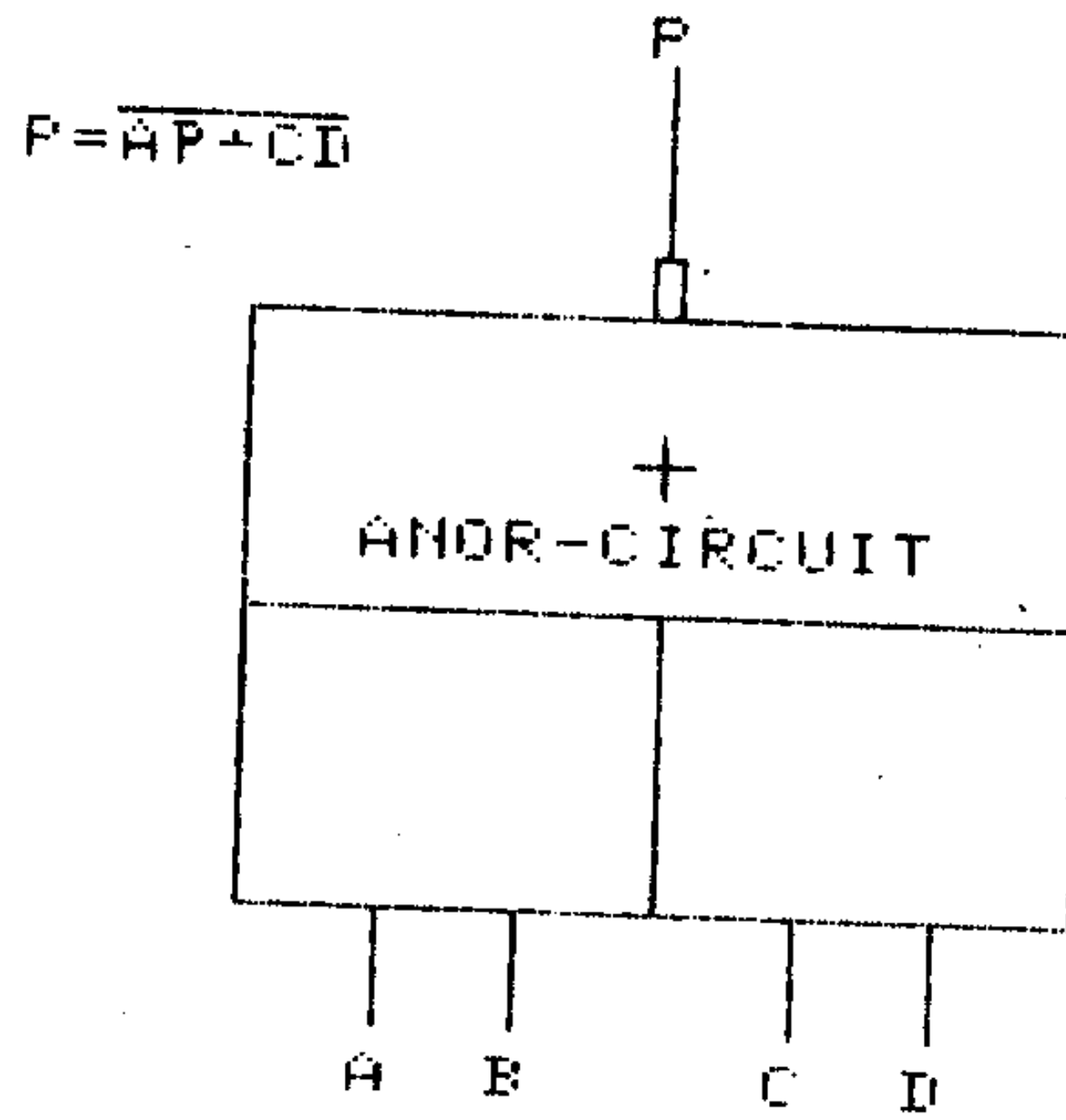
3



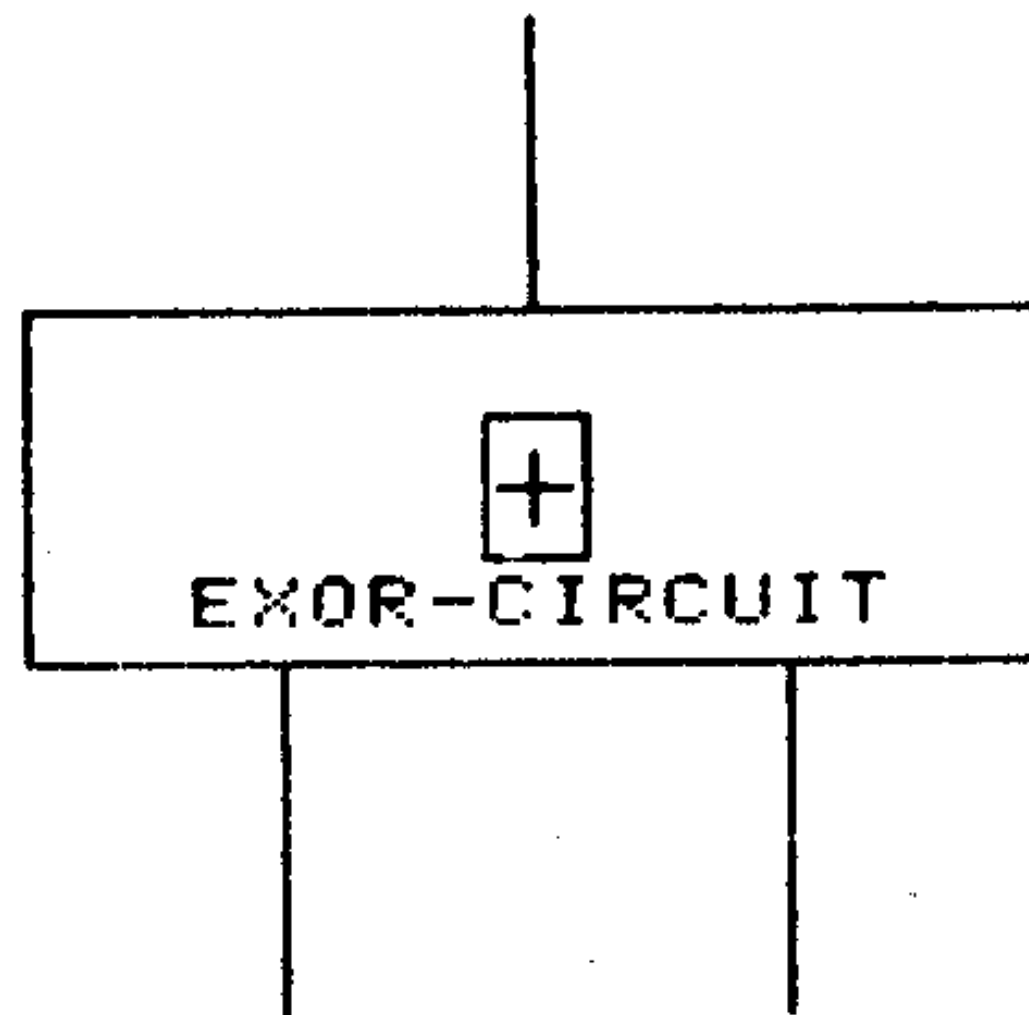
4



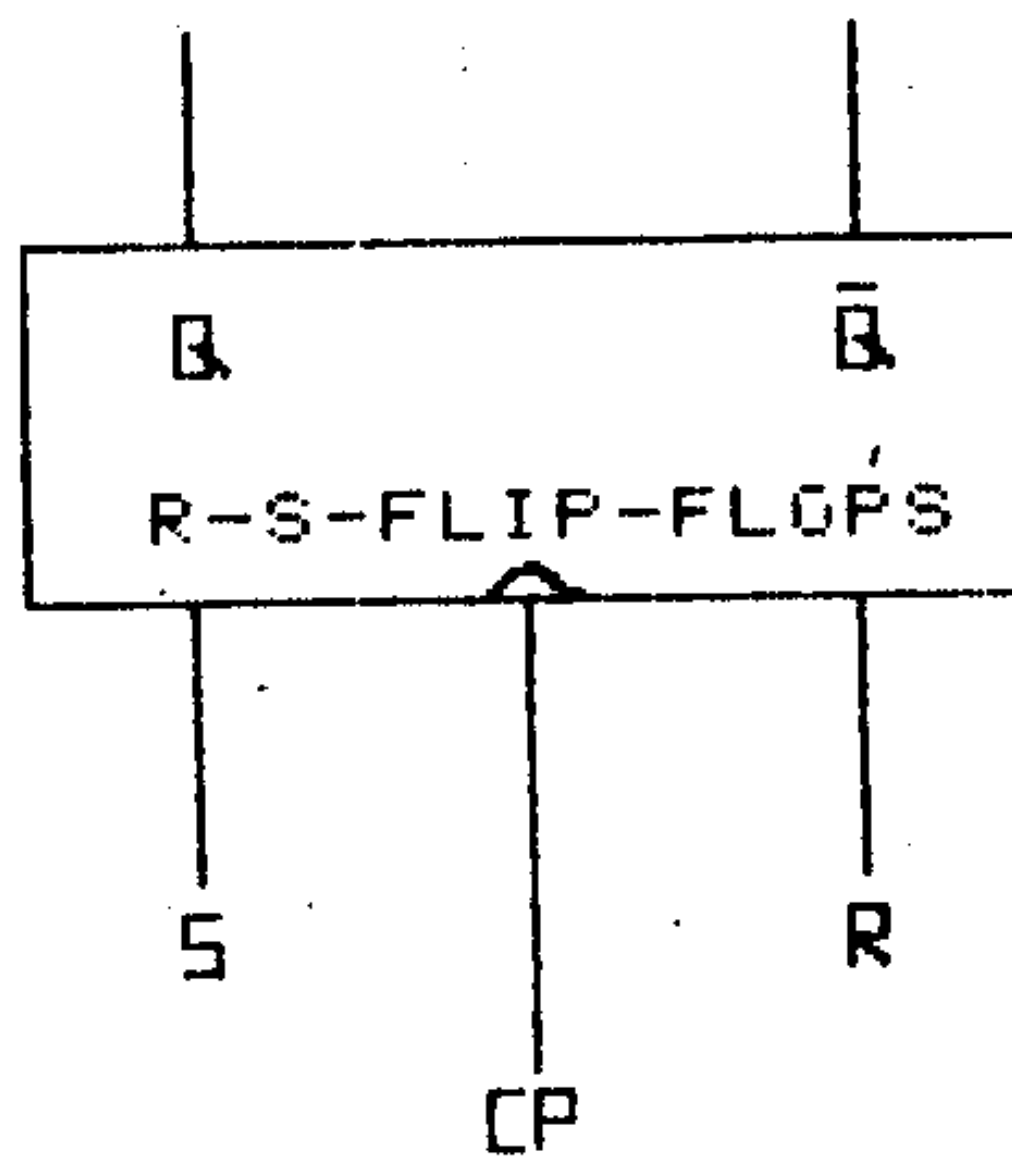
5

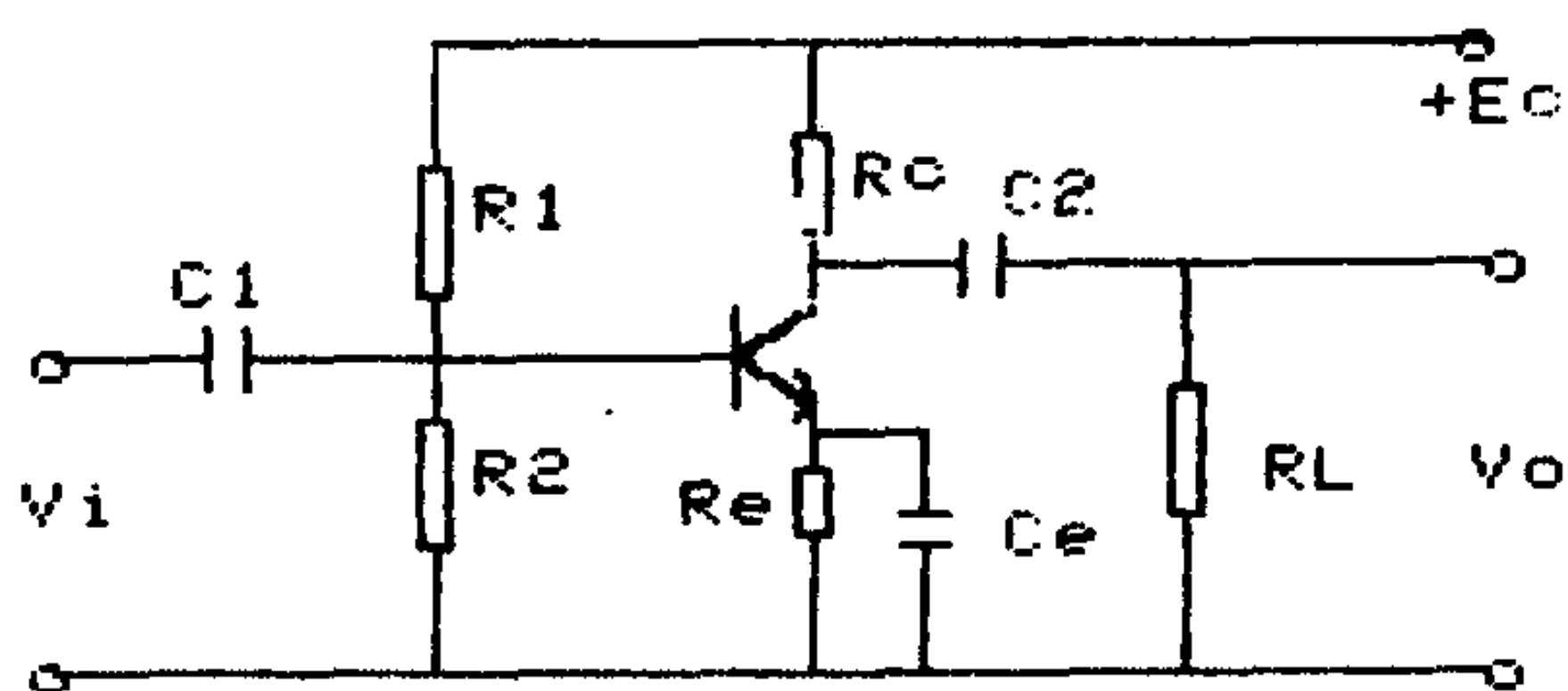


6



7





13. 回归曲线拟合程序

这里介绍一个关于回归曲线拟合的程序，本程序以一元线性回归曲线拟合为实例，其功能如下：

- ① 按需要画 X 轴，Y 轴；
- ② 按需要在轴上画刻度；
- ③ 在刻度旁边标注坐标值；
- ④ 点绘 X、Y 的实验数据；
- ⑤ 画 $Y = AX + B$ 的回归直线；
- ⑥ 打印回归系数 A、B 及相关系数 R 值；
- ⑦ 给出 $Y = AX + B$ 的方程；
- ⑧ 进行统计预报分析。

本程序取名 PRO-110，操作方法：

- 运行造型表：RUN PRO-108 ✓.
- 调线性回归程序进内存并运行之：RUN PRO-110 ✓.
- 屏幕出现 UNITS AT THE AXES (U,V) 时，键入 0.2, 1 ✓.

- 计算机自动绘图，并给出 $Y = AX + B$ 的表达式，A、B、R 值。

- 出现？此时即可预报，敲入 X 值，打印 Y 值

- 如需打印输出预报结果，可在运行本程序前，接通打印机。

注：实际应用程序 PRO-110，只要将实验数据放在 198 句开始的 DATA 区。若因实验数值过大或过小，可适当调整窗口 W3，W4 的值，同时坐标值及位置也应适当改变，这可以通过修改 363 句至 378 句的数据来完成。如果实验数据过多，请改变 140 句 DIM 语句。

程序 PRO-110 的清单和执行结果。

PRO-110

```
95 REM PRO-110
100 REM LINEAR REGRESSION
110 REM =====
120 POKE 232,0: POKE 233,96
130 TEXT : HOME
140 DIM X(100),Y(100)
150 W1 = 0:W2 = 0:W3 = 10:W4 = 15
160 SX = 0:SY = 0:SP = 0:S2 = 0:N = 0:S3 = 0
170 PRINT "INPUT THE COUPLES (X,Y)-MAX=100": PRINT
180 PRINT "INPUT(-1,-1)FOR FINISH": PRINT : PRINT

195 READ X,Y
198 DATA -0.8395,4.3723,1.7506,8.0120,1.9546,8.1
458,1.4542,7.6103,0.8159,7.1753,0.4648,6.5469
208 DATA -1,-1
210 IF X = -1 AND Y = -1 THEN 290
220 N = N + 1:X(N) = X:Y(N) = Y
230 SX = SX + X:SY = SY + Y:SP = SP + X * Y:S2 = S
2 + X * X
232 S3 = S3 + Y * Y
```

```

240 IF X < W1 THEN W1 = X
250 IF X > W2 THEN W2 = X
260 IF Y < W3 THEN W3 = Y
270 IF Y > W4 THEN W4 = Y
280 GOTO 195
285 REM ....THE AXES MUST BE TRACED CORRECTLY....
288 REM U=.2:V=2
290 PRINT : INPUT "UNITS AT THE AXES(U,V)";U,V
295 IF W1 > 0 THEN W1 = 0
296 IF W3 > 0 THEN W3 = 0
300 D = N * S2 - SX * SX
310 A = (N * SP - SX * SY) / D:B = (SY * S2 - SX *
    SP) / D
318 R = (N * SP - SX * SY) / SQR ((N * S2 - SX *
    SX) * (N * S3 - SY * SY))
320 READ V1,V2,V3,V4: DATA 1,279,32,191
330 HGR : HCOLOR= 3: GOSUB 60000: GOSUB 62500: GO
    SUB 63000
335 REM ....TRACE OF POINT
340 FOR I = 1 TO N:X = X(I):Y = Y(I):W$ = "U": GO
    SUB 62300
345 W$ = "D": GOSUB 62300: NEXT I
348 REM ....TRACE OF STRAIGHT LINE....
350 X = W1:Y = A * X + B:W$ = "U": GOSUB 61000
360 X = W2:Y = A * X + B:W$ = "D": GOSUB 61000
363 X$ = ".2":XZ = 103:YZ = 145: GOSUB 2000
364 X$ = ".4":XZ = 123:YZ = 145: GOSUB 2000
365 X$ = ".6":XZ = 143:YZ = 145: GOSUB 2000
366 X$ = ".8":XZ = 163:YZ = 145: GOSUB 2000
367 X$ = "1.0":XZ = 187:YZ = 145: GOSUB 2000
368 X$ = "1.2":XZ = 208:YZ = 145: GOSUB 2000
369 X$ = "1.4":XZ = 229:YZ = 145: GOSUB 2000
370 X$ = "1.6":XZ = 250:YZ = 145: GOSUB 2000
371 X$ = "1.8":XZ = 271:YZ = 145: GOSUB 2000
372 X$ = "-.2":XZ = 63:YZ = 145: GOSUB 2000
373 X$ = "-.6":XZ = 23:YZ = 145: GOSUB 2000
374 X$ = "4":XZ = 80:YZ = 114: GOSUB 2000
375 X$ = "8":XZ = 80:YZ = 72: GOSUB 2000
376 X$ = "12":XZ = 75:YZ = 28: GOSUB 2000
378 X$ = "Y=A*X+B":XZ = 200:YZ = 100: GOSUB 2000
379 POKE 1913,1: PR# 1: PRINT CHR$ (17)

```

```

380 VTAB 24: PRINT "Y=";A;"*X+";B
382 PRINT "A=";A
384 PRINT "B=";B
386 PRINT "R=";R
400 INPUT X: PRINT CHR$ (14);X
405 IF X = - 5 THEN PRINT "END": END
407 Y = A * X + B
409 PRINT CHR$ (14);
410 PRINT "Y=";Y
415 GOTO 400
2000 ROT= 0:XX = LEN (X$): IF XX = 3 THEN XZ = X
Z - 11: GOTO 2010
2002 IF XX = 2 THEN XZ = XZ - 4: GOTO 2010
2004 XZ = XZ - 3
2010 FOR I1 = 0 TO XX - 1:AS = ASC ( MID$ (X$,I1
+ 1,1)) - 32: DRAW AS AT XZ + I1 * 6,YZ: NEXT I1:
RETURN
59999 REM S.R.0 INITIALISATION AT LINE 60000
60000 A8 = (V2 - V1) / (W2 - W1):B8 = (V1 * W2 - V
2 * W1) / (W2 - W1)
60010 A9 = (V4 - V3) / (W4 - W3):B9 = (V3 * W4 - V
4 * W3) / (W4 - W3): RETURN
60998 REM S.R.1 PRINCIPAL AT LINE 61000
60999 REM
61000 X5 = X:Y5 = Y: GOSUB 61500: IF W$ = "D" THEN
61020
61001 C8$ = C5$:X8$ = C5$:X8 = X:Y8 = Y: IF C8$ <
> "0000" THEN RETURN
61005 C8$ = C5$:X8$ = C5$:X8 = X:Y8 = Y: IF C8$ <
> "0000" THEN RETURN
61010 X6 = X:Y6 = Y:W$ = "U": GOSUB 62400: RETURN
61020 C9$ = C5$:X9$ = C5$:X9 = X:C7 = X:Y9 = Y:D7
= Y
61030 GOSUB 62000:C8$ = X9$:X8$ = X9$:X8 = C7:Y8
= D7: RETURN
61498 REM S.R.2 BINARY CODE AT LINE 61500
61500 G9$ = "0":D9$ = "0":B9$ = "0":H9$ = "0"
61505 IF X5 < W1 THEN G9$ = "1": GOTO 61520
61510 IF X5 > W2 THEN D9$ = "1"
61520 IF Y5 < W3 THEN B9$ = "1": GOTO 61540
61530 IF Y5 > W4 THEN H9$ = "1"

```

```

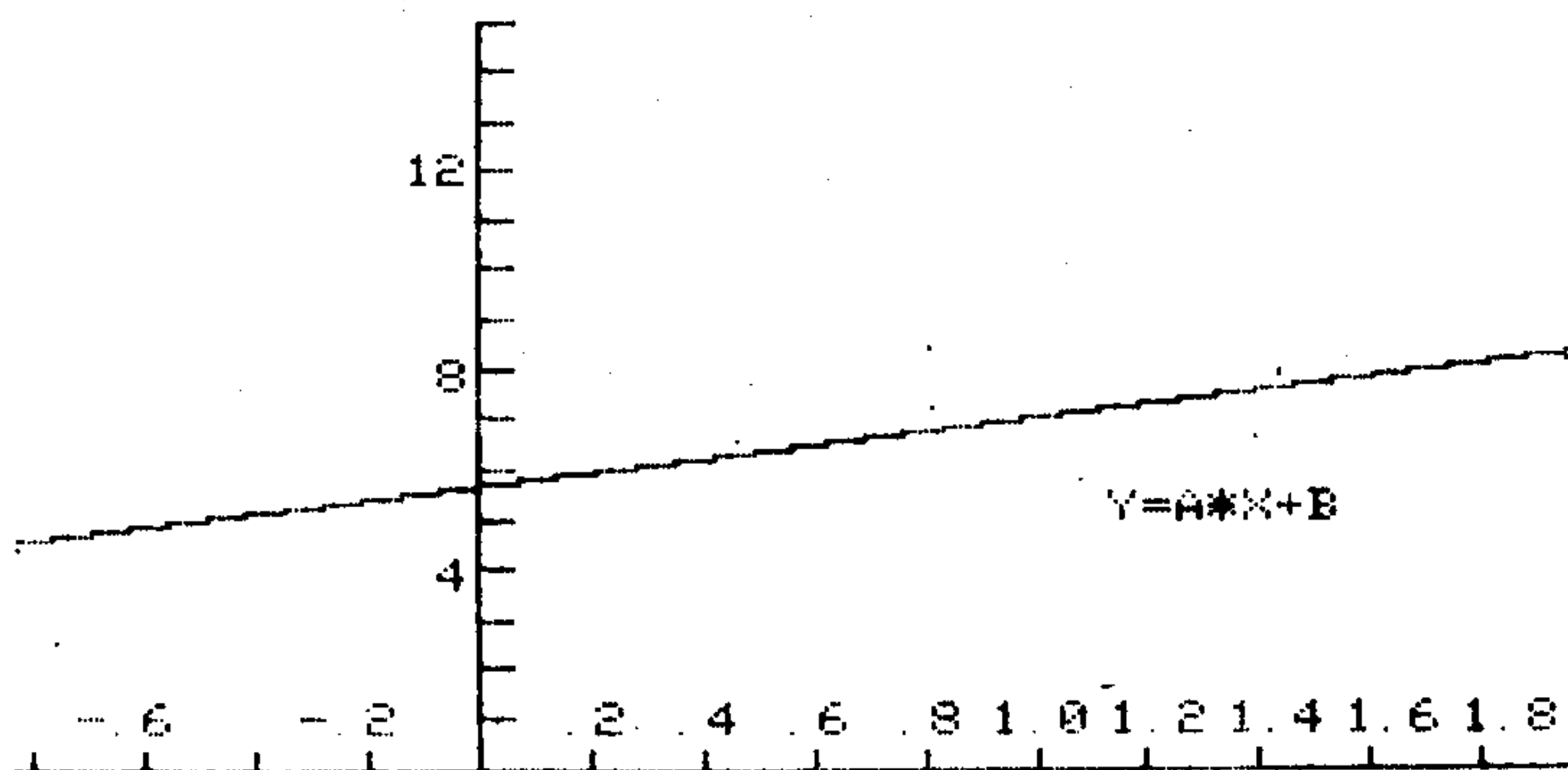
61540 C5$ = H9$ + B9$ + D9$ + G9$: RETURN
61998 REM S.R.3 CUTTING AT LINE 62000
62000 IF C8$ = "0000" AND C9$ = "0000" THEN 62100
62010 ET = 0: FOR K9 = 1 TO 4: X$ = MID$ (C8$, K9, 1)
      Y$ = MID$ (C9$, K9, 1)
62012 ET = ET + VAL (X$) * VAL (Y$): NEXT K9
62015 IF ET < > 0 THEN RETURN
62020 C7$ = C8$: IF C7$ = "0000" THEN G7$ = C9$
62030 G9$ = MID$ (C7$, 4, 1): IF G9$ < > "1" THEN
62040
62035 Y6 = Y8 + (Y9 - Y8) * (W1 - X8) / (X9 - X8):
X6 = W1: GOTO 62070
62040 D9$ = MID$ (C7$, 3, 1): IF D9$ < > "1" THEN
62050
62045 Y6 = Y8 + (Y9 - Y8) * (W2 - X8) / (X9 - X8):
X6 = W2: GOTO 62070
62050 B9$ = MID$ (C7$, 2, 1): IF B9$ < > "1" THEN
62060
62055 X6 = X8 + (X9 - X8) * (W3 - Y8) / (Y9 - Y8):
Y6 = W3: GOTO 62070
62060 H9$ = MID$ (C7$, 1, 1): IF H9$ < > "1" THEN
62070
62065 X6 = X8 + (X9 - X8) * (W4 - Y8): Y6 = W4
62070 IF C7$ = C8$ THEN X8 = X6: X5 = X6: Y8 = Y6: Y
5 = Y6: GOSUB 61500: C8$ = C5$: GOTO 62000
62080 X9 = X6: X5 = X6: Y9 = Y6: Y5 = Y6: GOSUB 61500
: C9$ = C5$: GOTO 62000
62100 IF X8$ < > "0000" THEN X6 = X8: Y6 = Y8: W$
= "U": GOSUB 62400
62110 X6 = X9: Y6 = Y9: W$ = "D": GOSUB 62400: RETUR
N
62298 REM S.R.4 PROJECTION+TRACE AT LINE 62400
62300 X6 = X: Y6 = Y: REM ....ENTRY FOR QUICK TRA
CE NO CUTTING
62400 X5 = A8 + X6 + B8: Y5 = 191 - (A9 + Y6 + B9)
62410 IF W$ = "U" THEN HPLOT X5, Y5: RETURN
62420 HPLOT TO X5, Y5: RETURN
62498 REM S.R.5 TRACE OF AXES AT LINE 62500
62500 P9 = W1: Q9 = W3
62510 IF W1 < 0 AND W2 > 0 THEN P9 = 0
62520 IF W3 < 0 AND W4 > 0 THEN Q9 = 0

```

```

62540 X6 = W1:Y6 = Q9:W$ = "U": GOSUB 62400:X6 = W
2:W$ = "D": GOSUB 62400
62550 X6 = P9:Y6 = W3:W$ = "U": GOSUB 62400:Y6 = W
4:W$ = "D": GOSUB 62400
62560 RETURN
62998 REM S.R.6 GRADUATION AT LINE 63000
63000 T8 = .02 * (W2 - W1):T9 = 0.02 * (W4 - W3)
63010 U8 = ((P9 - W1) / U - INT ((P9 - W1) / U))
* U
63015 V8 = ((Q9 - W3) / V - INT ((Q9 - W3) / V))
* V
63020 FOR X6 = W1 + U8 TO W2 STEP U
63030 Y6 = Q9:W$ = "U": GOSUB 62400:Y6 = Q9 + T9:W
$ = "D": GOSUB 62400
63040 NEXT X6
63050 FOR Y6 = W3 + V8 TO W4 STEP V
63060 X6 = P9:W$ = "U": GOSUB 62400:X6 = P9 + T8:W
$ = "D": GOSUB 62400
63070 NEXT Y6: RETURN
63498 REM S.R.7 TRACE OF FRAME AT LINE 63500
63500 X6 = W1:Y6 = W3:W$ = "U": GOSUB 62400:X6 = W
2:W$ = "D": GOSUB 62400
63510 Y6 = W4: GOSUB 62400:X6 = W1: GOSUB 62400:Y6
= W3: GOSUB 62400: RETURN
ORUN
INPUT THE COUPLES (X,Y)-MAX=100
INPUT (-1,-1)FOR FINISH
UNITS AT THE AXES(U,V)0.2,1

```




```

Y=1.33719658*X+5.72891614
A=1.33719658
B=5.72891614
R=.986415697
20.6
-6
Y=6.53123409
21.4
1.4
Y=7.60099135
20.3
-3
Y=6.13007511
?-1
-1
Y=4.39171956

```

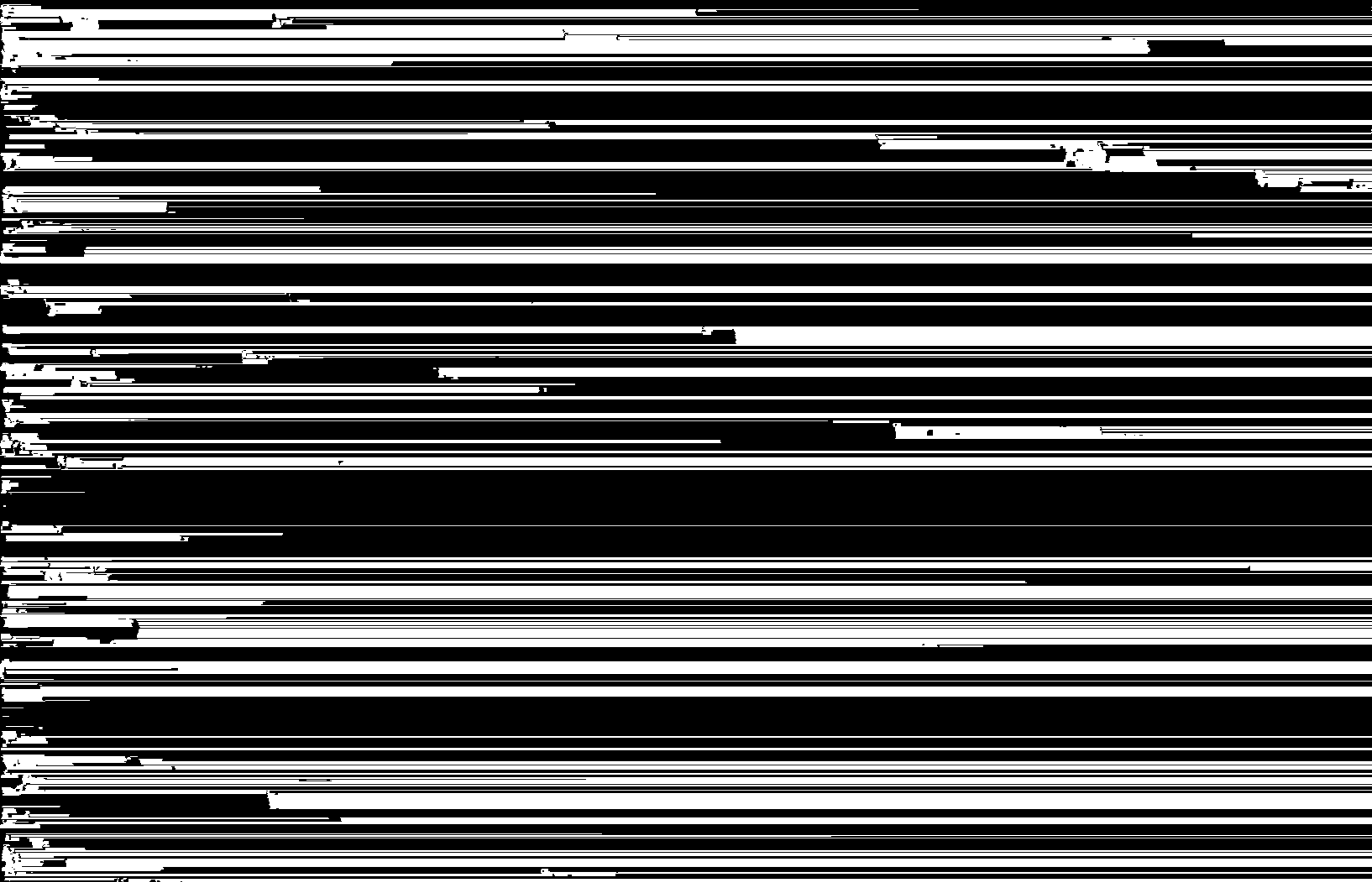
最后，应该指出的是，上述程序中59999—63500这一程序段是一个非常有用的绘图子程序，利用它常常可以得到令人惊奇的美好图形，作出所想要的所有的绘图，特别是这个子程序几乎适用于任何微型计算机，尤其是中华学习机、APPLE-II和紫金-II。当然，你要绘制的图形或报表，必须设计出相应的主程序。

14. 机器语言编制的造型表

程序 PRO-111 是一个用机器语言编制的 95 个 ASCII 码造型程序，它和前面介绍的程序 PRO-108 相比，有一个十分显著的优点，运行速度特别快。

程序 PRO-111 的起始地址：35946 (\$8C6A)，它的存贮长度为：2454 个字节 (\$995)，即造型表的末尾地址是：35946 + 2454 = 38400 (\$9600)。

程序 PRO-112 是一个调用机器语言子程序 PRO-111 的 BASIC 程序，为了正确实现调用，在 PRO-112 程序中首先应安排：



| | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|
| 8D28- | BE | 07 | D1 | 07 | E4 | 07 | 09 | 09 |
| 8D30- | 1A | 1B | 13 | 09 | 09 | 1A | 1B | 13 |
| 8D38- | 09 | 09 | 1A | 1B | 13 | 09 | 09 | 1A |
| 8D40- | 00 | 09 | 0D | 1A | 3B | 13 | 09 | 0D |
| 8D48- | 1A | 3B | 13 | 09 | 09 | 1A | 1B | 13 |
| 8D50- | 09 | 0D | 1A | 00 | 29 | 29 | 1A | 1F |
| 8D58- | 17 | 29 | 29 | 1A | 1B | 13 | 09 | 09 |
| 8D60- | 1A | 1B | 13 | 09 | 09 | 1A | 00 | 29 |
| 8D68- | 29 | 1A | 1F | 17 | 2D | 2D | 1E | 1F |
| 8D70- | 17 | 2D | 2D | 1E | 1F | 17 | 29 | 29 |
| 8D78- | 1A | 00 | 09 | 0D | 3A | 3F | 17 | 0D |
| 8D80- | 0D | 3A | 3F | 17 | 09 | 0D | 1E | 3F |
| 8D88- | 37 | 09 | 0D | 1A | 00 | 2D | 09 | 1E |
| 8D90- | 1F | 37 | 09 | 0D | 1A | 1B | 17 | 0D |
| 8D98- | 09 | 3A | 1F | 13 | 09 | 29 | 1E | 00 |
| 8DA0- | 29 | 09 | 1A | 3B | 33 | 0D | 0D | 3A |
| 8DA8- | 1B | 17 | 0D | 0D | 1E | 1F | 33 | 29 |
| 8DB0- | 0D | 1E | 00 | 09 | 29 | 1A | 3B | 13 |
| 8DB8- | 29 | 09 | 1A | 1B | 13 | 09 | 09 | 1A |
| 8DC0- | 1B | 13 | 09 | 09 | 1A | 00 | 09 | 29 |
| 8DC8- | 1A | 3B | 13 | 29 | 09 | 1A | 1B | 17 |
| 8DD0- | 29 | 09 | 1A | 3B | 13 | 09 | 29 | 1A |
| 8DD8- | 00 | 29 | 09 | 1A | 3B | 13 | 09 | 29 |
| 8DE0- | 1A | 1F | 13 | 09 | 29 | 1A | 3B | 13 |
| 8DE8- | 29 | 09 | 1A | 00 | 09 | 0D | 3A | 3B |
| 8DF0- | 33 | 29 | 2D | 1A | 3B | 13 | 29 | 2D |
| 8DF8- | 3A | 3B | 33 | 09 | 0D | 1A | 00 | 09 |
| 8E00- | 09 | 1A | 3B | 13 | 09 | 0D | 3A | 3F |
| 8E08- | 37 | 09 | 0D | 1A | 3B | 13 | 09 | 09 |
| 8E10- | 1A | 00 | 09 | 09 | 1A | 1B | 13 | 09 |
| 8E18- | 09 | 1A | 1B | 13 | 29 | 0D | 1A | 3B |
| 8E20- | 13 | 29 | 09 | 1A | 00 | 09 | 09 | 1A |
| 8E28- | 1B | 13 | 09 | 09 | 3A | 3F | 37 | 09 |
| 8E30- | 09 | 1A | 1B | 13 | 09 | 09 | 1A | 00 |
| 8E38- | 09 | 09 | 1A | 1B | 13 | 09 | 09 | 1A |
| 8E40- | 1B | 13 | 09 | 09 | 1A | 1B | 13 | 09 |
| 8E48- | 0D | 1A | 00 | 09 | 09 | 3A | 1B | 13 |
| 8E50- | 09 | 29 | 1A | 3B | 13 | 29 | 09 | 1A |
| 8E58- | 1B | 33 | 09 | 09 | 1A | 00 | 29 | 2D |
| 8E60- | 3A | 1B | 33 | 0D | 29 | 3E | 3B | 33 |
| 8E68- | 2D | 09 | 3E | 1B | 33 | 29 | 2D | 1A |
| 8E70- | 00 | 09 | 0D | 1A | 3B | 17 | 09 | 0D |
| 8E78- | 1A | 3B | 13 | 09 | 0D | 1A | 3B | 13 |
| 8E80- | 29 | 2D | 1A | 00 | 29 | 2D | 3A | 1B |

| | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|
| 8E88- | 33 | 09 | 09 | 1E | 1F | 13 | 09 | 0D |
| 8E90- | 1A | 1B | 17 | 2D | 2D | 1E | 00 | 29 |
| 8E98- | 2D | 3A | 1B | 33 | 09 | 09 | 1E | 3F |
| 8EA0- | 13 | 09 | 09 | 3E | 1B | 33 | 29 | 2D |
| 8EAB- | 1A | 00 | 09 | 29 | 1A | 3F | 13 | 29 |
| 8EB0- | 29 | 1A | 1F | 33 | 2D | 2D | 1E | 1F |
| 8EBB- | 13 | 09 | 29 | 1A | 00 | 2D | 2D | 1E |
| 8ECO- | 1B | 33 | 2D | 2D | 3A | 1B | 13 | 09 |
| 8ECB- | 09 | 3E | 1B | 33 | 29 | 2D | 1A | 00 |
| 8EDO- | 09 | 2D | 1E | 1B | 17 | 0D | 09 | 1A |
| 8EDB- | 3F | 37 | 0D | 09 | 3E | 1B | 33 | 29 |
| 8EE0- | 2D | 1A | 00 | 2D | 2D | 3E | 1B | 13 |
| 8EEB- | 09 | 29 | 1A | 3B | 13 | 29 | 09 | 1A |
| 8EFO- | 1B | 17 | 29 | 09 | 1A | 00 | 29 | 2D |
| 8EF8- | 3A | 1B | 33 | 0D | 09 | 1E | 3F | 17 |
| 8F00- | 0D | 09 | 3E | 1B | 33 | 29 | 2D | 1A |
| 8F08- | 00 | 29 | 2D | 3A | 1B | 33 | 0D | 09 |
| 8F10- | 3E | 3F | 17 | 09 | 09 | 1E | 1F | 13 |
| 8F1B- | 2D | 0D | 1A | 00 | 09 | 09 | 1A | 1B |
| 8F20- | 13 | 09 | 0D | 1A | 1B | 13 | 09 | 0D |
| 8F2B- | 1A | 1B | 13 | 09 | 09 | 1A | 00 | 09 |
| 8F30- | 09 | 1A | 1B | 13 | 09 | 0D | 1A | 1B |
| 8F3B- | 13 | 09 | 0D | 1A | 3B | 13 | 29 | 09 |
| 8F40- | 1A | 00 | 09 | 29 | 1A | 3B | 13 | 29 |
| 8F4B- | 09 | 1A | 1B | 33 | 29 | 09 | 1A | 3B |
| 8F50- | 13 | 09 | 29 | 1A | 00 | 09 | 09 | 1A |
| 8F5B- | 1B | 13 | 2D | 2D | 1E | 1B | 13 | 2D |
| 8F60- | 2D | 1E | 1B | 13 | 09 | 09 | 1A | 00 |
| 8F6B- | 29 | 09 | 1A | 3B | 13 | 09 | 29 | 3A |
| 8F70- | 1B | 13 | 09 | 29 | 1A | 3B | 13 | 29 |
| 8F7B- | 09 | 1A | 00 | 29 | 2D | 3A | 1B | 33 |
| 8F80- | 09 | 09 | 1E | 1F | 13 | 09 | 0D | 1A |
| 8F8B- | 1B | 13 | 09 | 0D | 1A | 00 | 29 | 2D |
| 8F90- | 3A | 1B | 33 | 0D | 0D | 3E | 3F | 33 |
| 8F9B- | 0D | 2D | 1A | 1B | 33 | 29 | 2D | 1E |
| 8FA0- | 00 | 09 | 0D | 1A | 1F | 17 | 0D | 09 |
| 8FAB- | 3E | 1B | 33 | 2D | 2D | 3E | 1B | 33 |
| 8FB0- | 0D | 09 | 1E | 00 | 2D | 2D | 3A | 1B |
| 8FBB- | 33 | 0D | 09 | 1E | 3F | 37 | 0D | 09 |
| 8FC0- | 3E | 1B | 33 | 2D | 2D | 1A | 00 | 29 |
| 8FCB- | 2D | 3A | 1B | 33 | 0D | 09 | 1A | 1B |
| 8FDO- | 33 | 0D | 09 | 3A | 1B | 33 | 29 | 2D |
| 8FDB- | 1A | 00 | 2D | 2D | 3A | 1B | 33 | 0D |
| 8FE0- | 09 | 3E | 1B | 33 | 0D | 09 | 3E | 1B |
| 8FEB- | 33 | 2D | 2D | 1A | 00 | 2D | 2D | 1E |

| | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|
| 8FF0- | 1B | 33 | 0D | 09 | 1A | 3F | 37 | 0D |
| 8FF8- | 09 | 1A | 1B | 33 | 2D | 2D | 1E | 00 |
| 9000- | 2D | 2D | 1E | 1B | 33 | 0D | 09 | 1A |
| 9008- | 3F | 37 | 0D | 09 | 1A | 1B | 33 | 0D |
| 9010- | 09 | 1A | 00 | 29 | 2D | 1E | 1B | 33 |
| 9018- | 0D | 09 | 1A | 1B | 33 | 0D | 29 | 3E |
| 9020- | 1B | 33 | 29 | 2D | 1E | 00 | 0D | 09 |
| 9028- | 3E | 1B | 33 | 0D | 09 | 3E | 3F | 37 |
| 9030- | 0D | 09 | 3E | 1B | 33 | 0D | 09 | 1E |
| 9038- | 00 | 29 | 2D | 1A | 3B | 13 | 09 | 0D |
| 9040- | 1A | 3B | 13 | 09 | 0D | 1A | 3B | 13 |
| 9048- | 29 | 2D | 1A | 00 | 09 | 09 | 3E | 1B |
| 9050- | 13 | 09 | 09 | 3E | 1B | 13 | 09 | 09 |
| 9058- | 3E | 1B | 33 | 29 | 2D | 1A | 00 | 0D |
| 9060- | 09 | 1E | 1F | 33 | 0D | 0D | 1A | 1B |
| 9068- | 37 | 0D | 0D | 1A | 1F | 33 | 0D | 09 |
| 9070- | 1E | 00 | 0D | 09 | 1A | 1B | 33 | 0D |
| 9078- | 09 | 1A | 1B | 33 | 0D | 09 | 1A | 1B |
| 9080- | 33 | 2D | 2D | 1E | 00 | 0D | 09 | 3E |
| 9088- | 1F | 37 | 0D | 0D | 3E | 3B | 33 | 0D |
| 9090- | 09 | 3E | 1B | 33 | 0D | 09 | 1E | 00 |
| 9098- | 0D | 09 | 3E | 1B | 33 | 2D | 09 | 3E |
| 90A0- | 3B | 33 | 0D | 29 | 3E | 1B | 33 | 0D |
| 90A8- | 09 | 1E | 00 | 29 | 2D | 3A | 1B | 33 |
| 90B0- | 0D | 09 | 3E | 1B | 33 | 0D | 09 | 3E |
| 90B8- | 1B | 33 | 29 | 2D | 1A | 00 | 2D | 2D |
| 90C0- | 3A | 1B | 33 | 0D | 09 | 1E | 3F | 37 |
| 90C8- | 0D | 09 | 1A | 1B | 33 | 0D | 09 | 1A |
| 90D0- | 00 | 29 | 2D | 3A | 1B | 33 | 0D | 09 |
| 90D8- | 3E | 1B | 33 | 0D | 0D | 1E | 1F | 33 |
| 90E0- | 29 | 0D | 1E | 00 | 2D | 2D | 3A | 1B |
| 90E8- | 33 | 0D | 09 | 1E | 3F | 37 | 0D | 0D |
| 90F0- | 1A | 1F | 33 | 0D | 09 | 1E | 00 | 29 |
| 90F8- | 2D | 3A | 1B | 33 | 0D | 09 | 1A | 3F |
| 9100- | 17 | 09 | 09 | 3E | 1B | 33 | 29 | 2D |
| 9108- | 1A | 00 | 2D | 2D | 1E | 3B | 13 | 09 |
| 9110- | 0D | 1A | 3B | 13 | 09 | 0D | 1A | 3B |
| 9118- | 13 | 09 | 0D | 1A | 00 | 0D | 09 | 3E |
| 9120- | 1B | 33 | 0D | 09 | 3E | 1B | 33 | 0D |
| 9128- | 09 | 3E | 1B | 33 | 29 | 2D | 1A | 00 |
| 9130- | 0D | 09 | 3E | 1B | 33 | 0D | 09 | 3E |
| 9138- | 1B | 33 | 0D | 09 | 1E | 1F | 17 | 09 |
| 9140- | 0D | 1A | 00 | 0D | 09 | 3E | 1B | 33 |
| 9148- | 0D | 09 | 3E | 3B | 33 | 0D | 0D | 3E |
| 9150- | 1F | 37 | 0D | 09 | 1E | 00 | 0D | 09 |

9158- 3E 1B 33 29 29 1A 3B 13
 9160- 29 29 3A 1B 33 0D 09 1E
 9168- 00 0D 09 3E 1B 33 29 29
 9170- 1A 3B 13 09 0D 1A 3B 13
 9178- 09 0D 1A 00 2D 2D 3E 1B
 9180- 13 09 29 1A 3B 13 29 09
 9188- 1A 1B 33 2D 2D 1E 00 29
 9190- 2D 1A 1B 17 29 09 1A 1B
 9198- 17 29 09 1A 1B 17 29 2D
 91A0- 1A 00 09 09 1A 1B 33 29
 91A8- 09 1A 3B 13 09 29 3A 1B
 91B0- 13 09 09 1A 00 29 2D 1A
 91B8- 1F 13 09 29 1A 1F 13 09
 91C0- 29 1A 1F 13 29 2D 1A 00
 91C8- 09 0D 1A 1F 17 0D 09 1E
 91D0- 1B 13 09 09 1A 1B 13 09
 91D8- 09 1A 00 09 09 1A 1B 13
 91E0- 09 09 1A 1B 13 09 09 1A
 91E8- 1B 13 2D 2D 1E 00 29 09
 91F0- 1A 3B 13 09 29 1A 1B 13
 91F8- 09 09 1A 1B 13 09 09 1A
 9200- 00 09 09 1A 1B 13 29 2D
 9208- 3A 1B 13 29 2D 3E 1B 33
 9210- 29 2D 1E 00 0D 09 1A 1B
 9218- 33 2D 2D 3A 1B 33 0D 09
 9220- 3E 1B 33 2D 2D 1A 00 09
 9228- 09 1A 1B 13 29 2D 1E 1B
 9230- 33 0D 09 1A 1B 33 29 2D
 9238- 1E 00 09 09 3E 1B 13 29
 9240- 2D 3E 1B 33 0D 09 3E 1B
 9248- 33 29 2D 1E 00 09 09 1A
 9250- 1B 13 29 2D 3A 1B 33 2D
 9258- 2D 1E 1B 33 29 2D 1E 00
 9260- 09 2D 3A 1B 17 29 09 1A
 9268- 3F 37 29 09 1A 1B 17 29
 9270- 09 1A 00 09 09 1A 3F 17
 9278- 0D 09 3E 1B 33 29 2D 3E
 9280- 1B 13 29 2D 1A 00 0D 09
 9288- 1A 1B 33 2D 2D 3A 1B 33
 9290- 0D 09 3E 1B 33 0D 09 1E
 9298- 00 09 0D 1A 1B 13 29 0D
 92A0- 1A 3B 13 09 0D 1A 3B 13
 92A8- 29 2D 1A 00 09 29 1A 1B
 92B0- 13 09 29 1A 1F 13 09 29
 92B8- 1A 1F 33 29 0D 1A 00 0D

| | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|
| 92C0- | 09 | 1A | 1B | 33 | 0D | 09 | 1E | 1F |
| 92C8- | 33 | 2D | 0D | 1A | 1F | 33 | 0D | 09 |
| 92D0- | 1E | 00 | 29 | 0D | 1A | 3B | 13 | 09 |
| 92D8- | 0D | 1A | 3B | 13 | 09 | 0D | 1A | 3B |
| 92E0- | 13 | 29 | 2D | 1A | 00 | 09 | 09 | 1A |
| 92E8- | 1B | 13 | 2D | 2D | 3A | 3B | 33 | 0D |
| 92F0- | 0D | 3E | 3B | 33 | 0D | 0D | 1E | 00 |
| 92F8- | 09 | 09 | 1A | 1B | 13 | 29 | 2D | 3A |
| 9300- | 1B | 17 | 29 | 09 | 3E | 1B | 17 | 29 |
| 9308- | 09 | 1E | 00 | 09 | 09 | 1A | 1B | 13 |
| 9310- | 29 | 2D | 3A | 1B | 33 | 0D | 09 | 3E |
| 9318- | 1B | 33 | 29 | 2D | 1A | 00 | 09 | 09 |
| 9320- | 1A | 3F | 37 | 0D | 09 | 3E | 1B | 33 |
| 9328- | 2D | 2D | 1A | 1B | 33 | 0D | 09 | 1A |
| 9330- | 00 | 09 | 09 | 3A | 3F | 17 | 0D | 09 |
| 9338- | 3E | 1B | 33 | 29 | 2D | 3E | 1B | 13 |
| 9340- | 09 | 09 | 1E | 00 | 09 | 09 | 1A | 1B |
| 9348- | 13 | 0D | 2D | 1E | 1B | 37 | 0D | 09 |
| 9350- | 1A | 1B | 33 | 0D | 09 | 1A | 00 | 09 |
| 9358- | 09 | 1A | 1B | 13 | 29 | 2D | 1E | 1B |
| 9360- | 33 | 29 | 2D | 3A | 1B | 13 | 2D | 2D |
| 9368- | 1A | 00 | 29 | 09 | 1A | 1B | 17 | 2D |
| 9370- | 2D | 1A | 1B | 17 | 29 | 09 | 3A | 1B |
| 9378- | 17 | 09 | 2D | 1A | 00 | 09 | 09 | 1A |
| 9380- | 1B | 13 | 0D | 09 | 3E | 1B | 33 | 0D |
| 9388- | 09 | 3E | 1F | 33 | 29 | 0D | 1E | 00 |
| 9390- | 09 | 09 | 1A | 1B | 13 | 0D | 09 | 1E |
| 9398- | 1F | 33 | 29 | 29 | 1A | 3B | 17 | 09 |
| 93A0- | 0D | 1A | 00 | 09 | 09 | 1A | 1B | 13 |
| 93A8- | 0D | 09 | 3E | 1B | 33 | 0D | 0D | 3E |
| 93B0- | 3B | 33 | 2D | 29 | 1E | 00 | 09 | 09 |
| 93B8- | 1A | 1B | 13 | 0D | 09 | 1E | 1F | 17 |
| 93C0- | 09 | 0D | 1A | 1F | 17 | 0D | 09 | 1E |
| 93C8- | 00 | 09 | 09 | 3A | 1B | 33 | 0D | 29 |
| 93D0- | 1A | 1F | 17 | 29 | 0D | 1A | 3B | 13 |
| 93D8- | 2D | 09 | 1A | 00 | 09 | 09 | 1A | 1B |
| 93E0- | 13 | 2D | 2D | 1E | 1F | 13 | 09 | 0D |
| 93E8- | 1A | 1B | 17 | 2D | 2D | 1E | 00 | 09 |
| 93F0- | 2D | 1E | 3B | 17 | 29 | 0D | 1A | 1B |
| 93F8- | 37 | 29 | 0D | 1A | 3B | 17 | 09 | 2D |
| 9400- | 1E | 00 | 09 | 0D | 1A | 3B | 13 | 09 |
| 9408- | 0D | 1A | 3B | 13 | 09 | 0D | 1A | 3B |
| 9410- | 13 | 09 | 0D | 1A | 00 | 2D | 0D | 1A |
| 9418- | 3F | 13 | 09 | 2D | 3A | 1F | 13 | 09 |
| 9420- | 2D | 1A | 3F | 13 | 2D | 0D | 1A | 00 |

| | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|
| 9428- | 29 | 0D | 1E | 3F | 33 | 09 | 09 | 1A |
| 9430- | 1B | 13 | 09 | 09 | 1A | 1B | 13 | 09 |
| 9438- | 09 | 1A | 00 | 2D | 2D | 3E | 3F | 37 |
| 9440- | 2D | 2D | 3E | 3F | 37 | 2D | 2D | 3E |
| 9448- | 3F | 37 | 2D | 2D | 1E | 00 | 2D | 2D |
| 9450- | 3E | 3F | 37 | 2D | 2D | 3E | 3F | 37 |
| 9458- | 2D | 2D | 3E | 3F | 37 | 2D | 2D | 1E |
| 9460- | 00 | 13 | 09 | 09 | 1A | 1B | 1B | 1B |
| 9468- | 1F | 1B | 1B | 17 | 29 | 2D | 2D | 29 |
| 9470- | 2D | 2D | 2D | 1A | 1B | 1F | 1B | 1F |
| 9478- | 1F | 1B | 17 | 29 | 09 | 29 | 29 | 09 |
| 9480- | 29 | 09 | 1A | 1B | 1F | 1B | 1F | 1F |
| 9488- | 1B | 17 | 29 | 09 | 29 | 29 | 09 | 29 |
| 9490- | 09 | 1A | 1B | 1F | 1B | 1F | 3F | 3F |
| 9498- | 17 | 29 | 09 | 09 | 29 | 09 | 29 | 09 |
| 94A0- | 1A | 1B | 1F | 1B | 1F | 1B | 1B | 17 |
| 94AB- | 29 | 09 | 09 | 0D | 09 | 29 | 09 | 1A |
| 94B0- | 1B | 1F | 1B | 1B | 1F | 1B | 33 | 00 |
| 94B8- | 20 | C4 | 94 | 4C | D0 | 94 | 20 | C4 |
| 94C0- | 94 | 4C | 64 | 95 | A0 | 03 | B1 | 69 |
| 94C8- | 85 | 06 | C8 | B1 | 69 | 85 | 07 | 60 |
| 94D0- | A2 | 00 | A0 | 00 | A9 | 04 | 8D | FF |
| 94D8- | 95 | B1 | 06 | 85 | 09 | 06 | 09 | 06 |
| 94E0- | 09 | 06 | 09 | 20 | 10 | 95 | 20 | 10 |
| 94E8- | 95 | 20 | 44 | 95 | C8 | B1 | 06 | 85 |
| 94F0- | 09 | 20 | 53 | 95 | A9 | 1B | 20 | 2B |
| 94F8- | 95 | A9 | 13 | 20 | 2B | 95 | C8 | CE |
| 9500- | FF | 95 | F0 | 03 | 4C | D9 | 94 | CA |
| 9508- | CA | A9 | 00 | 9D | 4E | 94 | 60 | 00 |
| 9510- | A9 | 09 | 85 | 08 | 06 | 09 | 90 | 06 |
| 9518- | A9 | 04 | 05 | 08 | 85 | 08 | 06 | 09 |
| 9520- | 90 | 04 | A9 | 20 | 05 | 08 | 9D | 4E |
| 9528- | 94 | E8 | 60 | 85 | 08 | 46 | 09 | 90 |
| 9530- | 06 | A9 | 04 | 05 | 08 | 85 | 08 | 46 |
| 9538- | 09 | 90 | 04 | A9 | 20 | 05 | 08 | 9D |
| 9540- | 4E | 94 | E8 | 60 | A9 | 1A | 85 | 08 |
| 9548- | 06 | 09 | 90 | 06 | A9 | 04 | 05 | 08 |
| 9550- | 85 | 08 | 60 | 46 | 09 | 90 | 06 | A9 |
| 9558- | 20 | 05 | 08 | 85 | 08 | A5 | 08 | 9D |
| 9560- | 4E | 94 | E8 | 60 | A2 | 00 | A0 | 00 |
| 9568- | A9 | 07 | 8D | FF | 95 | B1 | 06 | 85 |
| 9570- | 09 | 06 | 09 | 06 | 09 | 20 | 10 | 95 |
| 9578- | 20 | 10 | 95 | 20 | 10 | 95 | C8 | B1 |
| 9580- | 06 | 85 | 09 | 06 | 09 | 06 | 09 | 20 |
| 9588- | 10 | 95 | 20 | 10 | 95 | 20 | 10 | 95 |


```

9590- C8 B1 06 85 09 06 09 06
9598- 09 06 09 06 09 06 09 20
95A0- 10 95 20 44 95 C8 C8 C8
95A8- B1 06 85 09 20 53 95 A9
95B0- 1B 20 2B 95 88 B1 06 85
95B8- 09 A9 1B 20 2B 95 A9 1B
95C0- 20 2B 95 A9 1B 20 2B 95
95C8- 88 B1 06 85 09 A9 1B 20
95D0- 2B 95 A9 1B 20 2B 95 A9
95D8- 13 20 2B 95 C8 C8 C8 CE
95E0- FF 95 F0 03 4C 6D 95 A9
95E8- 00 9D 4E 94 60 A2 6A BD
95F0- 4D 94 9D 3A 94 CA D0 F7
95F8- 60 A2 13 4C EF 95 00 00
9600- A0

```

PRO-1 12

```

10 REM TABLE-1
20 POKE 232,106: POKE 233,140
30 SCALE= 1: ROT= 0: HCOLOR= 3
40 HGR2
50 A$ = "CODE":X = 5:Y = 6:W = 7: GOSUB 390
60 A$ = "ASCII CHARACTER TABLE":X = 65: GOSUB 390
70 HPLOT 5,20 TO 270,20: HPLOT 35,5 TO 35,170
80 A$ = " 32":Y = 8:Y = 24: GOSUB 390
90 J = 1:L = 10:X = 60: GOSUB 360
100 A$ = " 42":X = 8:Y = 38: GOSUB 390
110 J = 11:X = 60: GOSUB 360
120 A$ = " 52":X = 8:Y = 52: GOSUB 390
130 J = 21:X = 60: GOSUB 360
140 A$ = " 52":X = 8:Y = 64: GOSUB 390
150 J = 31:X = 60: GOSUB 360
160 A$ = " 62":X = 8:Y = 80: GOSUB 390
170 J = 41:X = 60: GOSUB 360
180 A$ = " 72":X = 8:Y = 94: GOSUB 390
190 J = 51:X = 60: GOSUB 360
200 A$ = " 82":X = 8:Y = 108: GOSUB 390
210 J = 61:X = 60: GOSUB 360
220 A$ = " 92":X = 8:Y = 122: GOSUB 390

```

```

230 J = 71: X = 60: GOSUB 360
240 A$ = "102": X = 8: Y = 136: GOSUB 390
250 J = 81: X = 60: GOSUB 360
260 A$ = "112": X = 8: Y = 150: GOSUB 390
270 J = 91: L = 6: X = 60: GOSUB 360
280 HPLLOT 5,170 TO 270,170
290 A$ = "DO YOU WANT TO PRINT (Y/N) ": X = 60: Y = 175
    : GOSUB 390
300 GET X$: PRINT D$: IF X$ = "N" OR X$ = "" THEN TEXT
    : HOME : PRINT "GOODBYE!": PR# 6
310 IF X$ < > "Y" THEN PRINT CHR$ (7): GOTO 40
320 HCOLOR= 0: A$ = "DO YOU WANT TO PRINT (Y/N) ": X =
    60: Y = 175: GOSUB 390
330 PR# 1: PRINT CHR$ (4): POKE 1913,66
340 POKE 1913,66: PRINT CHR$ (17): PR# 0: FOR I = 0
    TO 10: PRINT CHR$ (7): NEXT : HCOLOR= 3: GOTO 4
    0
350 STOP
360 K = 0: FOR I = J TO J + L - 1
370 DRAW I AT X + 18 * K, Y: K = K + 1: NEXT
380 RETURN
390 FOR I = 1 TO LEN (A$)
400 DRAW ASC ( MID$ (A$, I, 1)) - 31 AT X + (I - 1) *
    W, Y: NEXT : RETURN

```

| CODE | ASCII CHARACTER TABLE | | | | | | | | | |
|------|-----------------------|-----|---|----|---|---|---|---|---|---|
| 32 | | " | # | \$ | % | & | ' | < | > | |
| 42 | 1 | + | , | - | . | / | 0 | 1 | 2 | 3 |
| 52 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = |
| 52 | > | ? | @ | A | B | C | D | E | F | G |
| 62 | H | I | J | K | L | M | N | O | P | Q |
| 72 | R | S | T | U | V | W | X | Y | Z | [|
| 82 | \ |] ^ | _ | ` | a | b | c | d | e | |
| 92 | f | g | h | i | j | k | l | m | n | o |
| 102 | p | q | r | s | t | u | v | w | x | y |
| 112 | z | { | | } | ~ | | | | | |

六、软件功能扩展技术

某种特定计算机系统的功能和效用主要取决于当时设计的情况，因而当投入使用后，由于硬软件技术的迅速发展和用户使用水平的提高，总是不可避免地存在着局限性。这种局限性促使专门的计算机设计者和用户一起进行不断的研究和探索。一方面通过改进外设、增加外设、联机等手段加强硬件功能，另一方面又不断地改进老的操作系统、开发新的操作系统和各种应用程序，在软件方面扩充其性能。这样不断地改进和扩充，最后引起硬软件的换代。

那么，怎样发挥处于生命周期中的某种计算机系统的作用呢？

从目前的情况来看，开发功能较强又比较实用的各种应用程序，尽可能使用软件技术来加强系统功能是一个简捷而行之有效的方法。因为硬件的价格虽然在大幅度下降，但与软件相比，仍然非常昂贵。国内的情况尤其如此。

本章所要介绍的内容正是以6502为CPU的计算机系统的软件功能扩展技术，我们还将提供一些有较高实用价值的功能扩充软件。有兴趣的读者不妨上机试一试，相信它能给你带来益处。

1. APPLESOFT 中的ERASE 语句模拟

在许多 APPLESOFT BASIC 的参考书中都提到这样一个变量使用的技巧，后面的程序尽可能使用前面已经使用过的变量。一方面节省主机内存（如使用新的变量，则必

须开辟新的存贮空间),另一方面加快变量的搜索速度,因为在变量表中新开辟的变量总是在老变量之后,通过指针进行的变量值搜索也就更慢。

但实际情况往往并不这么简单。比如两个较大的连接程序或两个程序段中变量的类型不一样,或数组的格式不一样,就不能使用这种覆盖的方法。但内存空间又不允许再定义更多的变量。于是,有不少应用程序便通过磁盘来进行变量的筛选和传递,即将当前程序或程序段中有用的变量以磁盘文件的形式保存起来,再用 RUN 或 CLEAR 指令清除所有内存变量,在被连接的程序或需要传递变量的程序段中重新定义并从磁盘取回变量值。显然,这种方法不仅麻烦而增加了程序长度,且大大限制了程序的执行速度,缩短了磁盘的使用寿命。

因此,有必要增加一个变量删除语句。目前的许多 BASIC 版本中都已有了这个 ERASE 语句,它的作用是从内存空间中清除其后所列出的所有变量,并释放相应的存贮单元,以作他用。APPLESOFT 中由于没有这个语句,使不同程序或程序段的变量使用很不方便,尤其给变量空间的灵活开辟带来不少困难。

为此,我们使用 6502 机器语言编写了一段程序。这样,一方面可以节省内存空间,另一方面,被删除的变量可以在以后的程序段中按要求重新定义和赋值。

我们知道,程序在运行过程中始终有一个动态的变量表,其在内存中存放的起始位置由 LOMEM: 的值来决定(一般指向当前程序尾的三个 \$00 字节之后的第一个单元)。这个表有两个部分:一是简单变量表,二是数组变量表。根据系统所规定的表结构,可以查出某一变量所占的内存单元数。将

指定的简单变量或数组变量区之后直至变量表尾的内容往前搬移一个位置，以覆盖这一变量所占用的空间，并修改有关的指针，这样，就达到了删除这个变量的目的。

设将该机器语言程序存放在 \$ 6000 (24576) 开始的地址单元中，则其清单见程序 PRO-113。

PRO-113

*6000.616A

| | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|
| 6000- | A0 | 00 | A9 | 00 | 85 | E1 | B1 | B8 |
| 6008- | C9 | 2C | F0 | 18 | C9 | 00 | D0 | 0B |
| 6010- | 98 | 18 | 65 | B8 | 85 | B8 | 90 | 02 |
| 6018- | E6 | B9 | 60 | C9 | 3A | F0 | F1 | A2 |
| 6020- | 10 | 4C | 12 | D4 | C8 | B1 | B8 | 85 |
| 6028- | E0 | 4C | 5C | 61 | F0 | 23 | C9 | 3A |
| 6030- | F0 | 1F | C9 | 30 | B0 | 0F | C9 | 24 |
| 6038- | F0 | 1E | C9 | 25 | F0 | 23 | C9 | 28 |
| 6040- | F0 | 28 | 4C | 1F | 60 | 85 | E1 | C8 |
| 6048- | B1 | B8 | C9 | 30 | B0 | F9 | 4C | 2C |
| 6050- | 60 | A2 | 01 | 86 | E2 | 4C | 94 | 60 |
| 6058- | A5 | E1 | 09 | 80 | 85 | E1 | 4C | 87 |
| 6060- | 60 | A5 | E0 | 09 | 80 | 85 | E0 | 4C |
| 6068- | 58 | 60 | C8 | B1 | B8 | C9 | 29 | F0 |
| 6070- | 03 | 4C | 1F | 60 | C8 | A2 | 00 | 86 |
| 6078- | E2 | 8D | 6B | 00 | 9D | E3 | 00 | E8 |
| 6080- | E0 | 04 | D0 | F5 | 4C | A1 | 60 | C8 |
| 6088- | B1 | B8 | C9 | 28 | F0 | 03 | 4C | 51 |
| 6090- | 60 | 4C | 6A | 60 | A2 | 00 | BD | 69 |
| 6098- | 00 | 9D | E3 | 00 | E8 | E0 | 04 | D0 |
| 60A0- | F5 | 98 | 48 | A5 | E4 | C5 | E6 | D0 |
| 60A8- | 04 | A5 | E3 | C5 | E5 | 90 | 03 | 4C |
| 60B0- | 1F | 60 | A0 | 00 | B1 | E3 | C5 | E0 |
| 60B8- | D0 | 0A | C8 | B1 | E3 | C5 | E1 | D0 |
| 60C0- | 03 | 4C | F0 | 60 | 20 | CA | 60 | 4C |
| 60C8- | A3 | 60 | A5 | E2 | D0 | 16 | A0 | 02 |
| 60D0- | B1 | E3 | 18 | 65 | E3 | 85 | E3 | 90 |
| 60D8- | 02 | E6 | E4 | C8 | B1 | E3 | 18 | 65 |
| 60E0- | E4 | 85 | E4 | 60 | A5 | E3 | 18 | 69 |
| 60E8- | 07 | 85 | E3 | 90 | 02 | E6 | E4 | 60 |
| 60F0- | A5 | E3 | 85 | E5 | A5 | E4 | 85 | E6 |
| 60F8- | A0 | 02 | B1 | E3 | 85 | E9 | C8 | B1 |
| 6100- | E3 | 85 | EA | 20 | CA | 60 | A5 | E4 |

```

6108- C5 6E D0 06 A5 E3 C5 6D
6110- F0 02 B0 15 A0 00 B1 E3
6118- 91 E5 E6 E3 D0 02 E6 E4
6120- E6 E5 D0 02 E6 E6 4C 06
6128- 61 A5 E2 D0 10 A5 6D 38
6130- E5 E9 85 6D A5 6E E5 EA
6138- 85 6E 4C 57 61 A5 6D 38
6140- E9 07 85 6D A5 6E E9 00
6148- 85 6E A5 6B 38 E9 07 85
6150- 6B A5 6C E9 00 85 6C 68
6158- A8 4C 02 60 C8 B1 B8 C9
6160- 2C D0 03 4C 51 60 C9 00
6168- 4C 2C 60

```

*

假设这一程序已调入内存（用BLOAD指令），则模拟语句的格式为：

CALL24576, Variable, Variable, ..., Variable

其中，Variable为要删除的变量名及其类型，类型说明符号系统约定一致（\$表示字符串变量，%表示整型变量）。如果某一变量是数组变量，则其后必须跟“（）”字符。

程序①

```

1000 REM SAMPLE-1
1010 DIM A(10),B$(5),CD%(12)
1020 F = 1000:CD%(1) = 10
1030 CALL 24576,A(),F,CD%()
1040 PRINT "F=";F
1050 DIM A(4,8),CD%(7)
1060 PRINT "CD%(1)='";CD%(1)
1070 END

```

RUN

F=0

CD%(1)=0

]

在1010句定义了三个不同类型的数组变量，1020句给简单变量F赋值，经执行1030句的数组删除后，F的值被清为

零，1050句又对已删除的数组变量重新定义。由运行结果可知，系统默认这种处理。

程序②

```
1000 REM SAMPLE-2
1010 DIM AB(4000)
1020 PRINT FRE (0)
1030 CALL 24576,AB()
1040 PRINT FRE (0)
1050 END
```

```
]RUN
16266
-29258
```

在这一例子中，结果显示 AB 数组删除前后的自由空间字节数分别是 16266 和 $65536 - 29258 = 36278$ ，也即释放的内存字节数为 $36278 - 16266 = 20012$ 。而 AB 数组所占的内存字节数为： $4001 \times 5 + 7$ （其中 7 个字节为数组索引，以后每个数占 5 个字节），也正好是 20012，可见该数组确已从内存中删除。

顺便指出，当需要在两个程序间进行部分变量传递的链接时，可先用该模拟语句对不用的变量清除，再用 APPL-ESoft BASIC 中的 CHAIN 指令来完成。

2. 程序运行中的变量表列印

程序调试是软件设计者的主要工作之一，其基本方法有两种：一是静态分析法，即通过对列印程序的阅读、分析，发现其中存在的错误和必须改进的地方；二是动态跟踪法，即在程序运行过程中，通过运行情况追踪、变量查找以及结果输出等发现问题。前一种方法难度较大，许多问题不容易找出，因而一般很少单独使用。后一种方法直观、简便，而

且容易使调试者很快掌握整个程序的基本状况，为大多数调试过程所采用。但是，即便采用这种方法，大程序的调试仍然是相当复杂而费时的。在程序运行出现错误时，有时还很难找出错误原因以及解决办法。

比如，出现“内存空间溢出”的错误，是否由变量定义太多而引起？程序中使用了多少变量？是否存在一些已经无效而可以删除或覆盖的变量？这些问题都无法从运行状态中看出。为了加强程序调试的功能，增加一些 TRACE、STOP、CTRL-C 之外的调试手段，在此我们提供一个内存变量列印的工具软件。

其清单见 PRO-114。

PRO-114

*300.3EB

```
0300- 20 42 FC 20 A7 03 A5 69
0308- 85 E0 A5 6A 85 E1 C5 6C
0310- F0 05 B0 21 4C 1D 03 A5
0318- E0 C5 6B B0 18 20 B8 03
0320- A5 E0 18 69 07 85 E0 A5
0328- E1 69 00 85 E1 20 8E FD
0330- A5 E1 4C 0E 03 20 A7 03
0338- A5 6B 85 E0 A5 6C 85 E1
0340- C5 6E F0 05 B0 71 4C 4F
0348- 03 A5 E0 C5 6D B0 68 20
0350- B8 03 A9 A8 20 ED FD C8
0358- C8 C8 B1 E0 0A 84 E2 18
0360- 65 E2 A8 B1 E0 AA 88 B1
0368- E0 84 E3 CA E0 FF D0 03
0370- 18 E9 01 20 24 ED A4 E3
0378- 88 C4 E2 F0 08 A9 AC 20
0380- ED FD 4C 63 03 A9 A9 20
0388- ED FD A4 E2 EA 88 88 B1
0390- E0 C8 18 65 E0 48 B1 E0
0398- 65 E1 85 E1 68 85 E0 20
03A0- 8E FD A5 E1 4C 40 03 20
03A8- 8E FD A2 0A A9 AA 20 ED
```



```

03B0- FD CA D0 FA 20 8E FD 60
03B8- A0 00 B1 E0 10 0B 20 ED
03C0- FD A2 A5 C8 B1 E0 4C DC
03C8- 03 09 80 20 ED FD C8 B1
03D0- E0 30 07 A2 00 09 80 4C
03D8- DC 03 A2 A4 C9 80 F0 03
03E0- 20 ED FD 8A C9 00 F0 03
03E8- 20 ED FD 60

```

★

这一机器语言程序占用 \$ 300—\$ 3EB 共 236 个字节的内存空间，分四个小段。

第一段从 \$ 300 到 \$ 334 为列印简单变量表的程序段；

第二段从 \$ 335 到 \$ 3A6 为列印数组变量表的程序段；

第三段从 \$ 3A7 到 \$ 3B7 为显示列印格式的子程序；

第四段从 \$ 3B8 到 \$ 3EB 为显示变量名及类型说明符的子程序。

将此程序输入主机，并存贮在磁盘上。当需要列印当前内存中 APPLESOFT 程序的运行变量时，只需将其调入内存(用 BLOAD 命令)，然后用 CALL768 指令执行即可。CALL768 可以作为程序的一个语句，也可作为一个键盘指令直接启动。其执行过程对用户程序的状态不加任何干预，也就是说，如果用户程序由 STOP 或 CTRL-C 等中断，利用 CALL768 执行完这一机器语言程序后，用户程序仍可由 CONT 继续。

实际使用证明，这一辅助调试手段对了解程序的基本情况、分析内存空间分配以及合理使用程序变量等方面很有好处。

3. 磁盘文本文件的列印

在上一部分中我们介绍了 APPLESOFT BASIC 程

序的调试问题，并提供了一个列印内存变量表的辅助调试工具。这一部分我们将简单说明一下磁盘文本文件的存取过程及其结构和存放格式，然后提供一个分析磁盘文本文件存贮情况、列印其内容的程序。

(1) 文本文件的存贮格式

文本文件通常分为顺序和随机文本文件两种类型。这两种文件都是存贮数据信息的，其在磁盘中的格式并不相同，但却都是ASCII码形式，是将用户资料转换成相应的ASCII码再送至磁盘的。为说明它们不同的存贮格式，现假设有三个字段的数据：ONE、TWO、THREE，在顺序和随机文件中，格式分别如下，见图23。

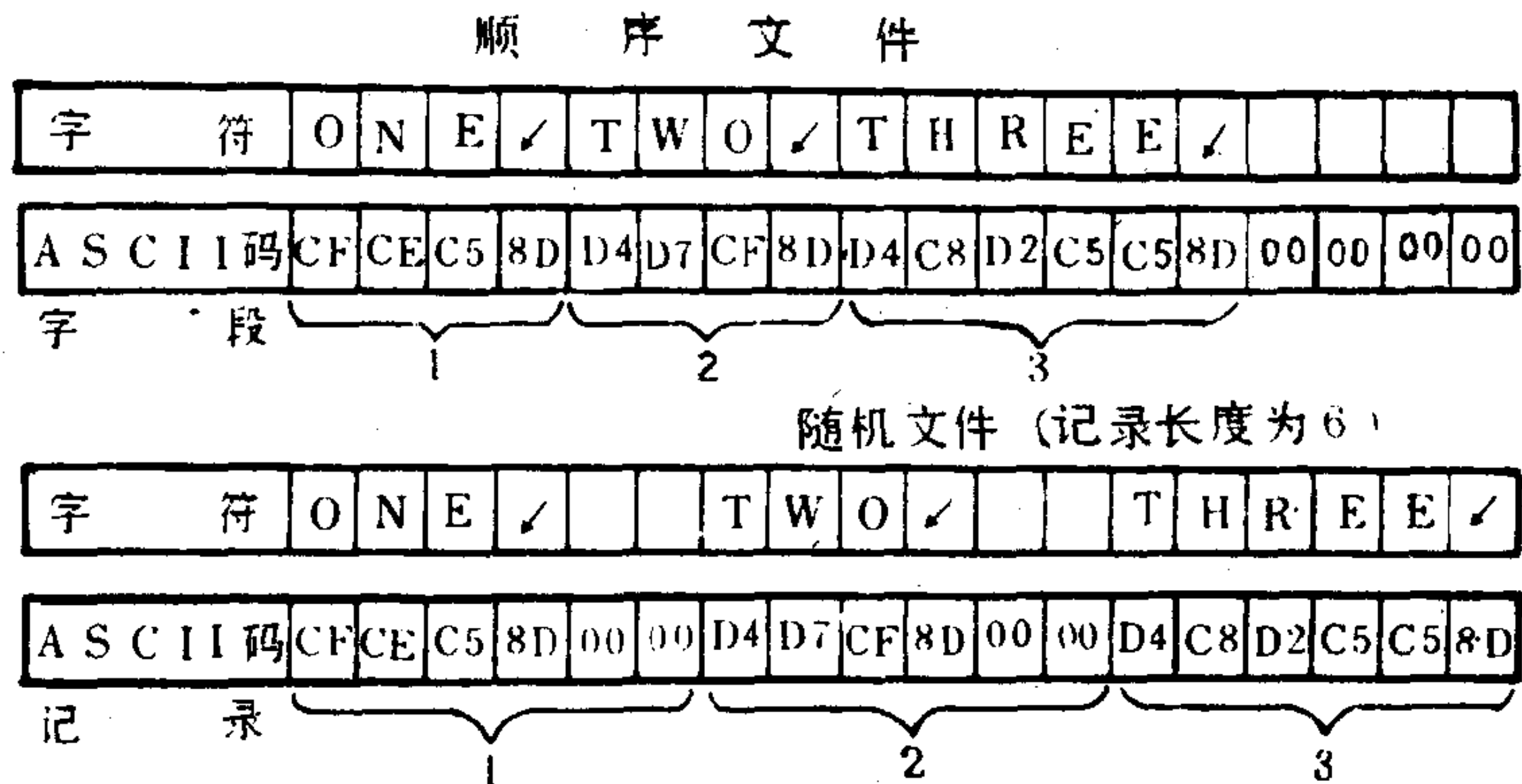


图 23

其实，图23中的ASCII码一栏就是它们在磁盘中存贮的实际形式。由此可以看出：顺序文件的每一个字段是紧接着前一个字段存放的，每个字段之间以回车符作为分隔；随机文件则是以定长的记录存放的，一个字段之后为回车符，如其长度不足规定长度，其后为“空位”，下一个记录仍以标准长度计算从下一个记录位置开始存贮。顺序文件占磁盘空

间少，批量存取速度快，使用比较简单，但不适宜对任意指定字段的修改和存取；随机文件占磁盘空间多，结构更复杂一些，但因其每一记录的存贮位置可以通过定长来计算，所以特别适合于对其中部分数据的修改和存取操作。在实际使用时，一般应根据具体情况选择一种形式。

(2) 文本文件的存贮过程

顺序文件和随机文件的存贮过程基本上是一样的。首先，DOS在遇到第一个指向文本文件的PRINT语句时，清除键盘缓冲区的原有内容，并将PRINT后的数据移入该区，如缓冲区尚未存满(数据不足256个字节)，则继续等待下一个PRINT。直至存满256个字节时，DOS开始寻找磁盘上的一个空扇区，将缓冲区内容存贮到空扇区中，再返回，继续等待PRINT输出。

对一个空磁盘来说，用户空间共有31个磁道（另有3个磁道为DOS占用，1个磁道为目录索引区占用），每一磁道为16个扇区。存贮时，DOS按一定的先后次序来进行：对磁道来说，首先是\$12道，然后递增一直到\$22道，存满后，又转为第\$10道，再递减一直到\$03道；对扇区来说，无论哪一个磁道，均从第\$F区递减至第\$0区。由于文件删除、修改等原因，实际存贮过程往往并不全按这种约定顺序，因而一个文件在盘上的分配可能是杂乱无章的，为了记录这一情况，DOS在文件存贮结束后(CLOSE)自动建立磁道/扇区一览表，并将此一览表的位置存在目录区中（这也就是文件操作后必须CLOSE，否则可能全部或部分丢失数据的原因）。

(3) 磁盘文本文件的列印程序

建立磁盘文本文件的目的是为各种应用程序提供所

需的数据。在编写应用程序时，首先必须清楚数据文件中保存了哪些资料，其具体的格式如何，然后才能相应地设置文件取数操作。如果文件不是由编写应用程序的人员建立，则可能就不知道这些情况，因而使取数失败，既影响了应用程序的设计，也降低了数据文件的使用率。

为解决这一问题，我们通过分析数据文件的磁盘存贮过程和格式，设计了一个列印文件内容的机器语言程序。不管是何种文件，格式如何，只要运行这一程序，即可在屏幕或打印机上输出其内容。

这一程序主要是利用系统内部的 R W T S 子程序来完成磁盘读写工作的，从其功能来分，有五段：

第一段从 \$6000—\$6022为从键盘接受文件名的程序。文件名存于键盘缓冲区中，如其字符数不足30，以空格补充；

第二段从 \$602D—\$6054指定 R W T S 子程序有关的 I O B 参数(磁盘卷号、读写状态、读写的磁道号和扇区号、资料存取的缓冲区起始、地址等)，取得 I O B 表地址，并调用 R W T S 进行磁盘的读写操作；

第三段从 \$6055—\$60B8：读出磁盘目录区内容，从中搜索指定名称的文本文件；

第四段从 \$60B9—\$60E0：读出指定文件的磁道/扇区一览表，并从中取得文件存贮的实际地址（磁道号、扇区号）；

第五段从 \$60E1—结束：按磁道/扇区一览表的顺序读出文件内容，并显示。

将此程序(取名为 P R O - 115)输入 \$6000 (24576)开始的内存单元中，使用 C A L L 24576 (B A S I C 态下)或 6000 G (监控态下)指令即可使之运行。运行第一步显示：

“INPUT FILE NAME:”, 要求输入文件名, 之后程序自动查找该文件并列印其内容。

如指定文件不存在或文件类型不符 (不是文本文件) 则显示:

“FILE NOT FOUND”

返回原状态。

该程序的运行不影响当前内存中的 BASIC 程序及其所有变量 (除非程序和变量的存放地址超过 \$6000), 因此, 可以在程序设计、调试或运行等过程中使用程序 PRO-115。

PRO-115

*6000.614F

| | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|
| 6000- | 20 | 42 | FC | 20 | 33 | 61 | A2 | 00 |
| 6008- | BD | 0B | 61 | 20 | ED | FD | E8 | E0 |
| 6010- | 10 | D0 | F5 | 20 | 6F | FD | A9 | A0 |
| 6018- | 9D | 00 | 02 | E8 | E0 | 1E | 90 | F8 |
| 6020- | 20 | 8E | FD | 20 | E3 | 03 | 85 | E1 |
| 6028- | 84 | E0 | 4C | 55 | 60 | A0 | 03 | A9 |
| 6030- | 00 | 91 | E0 | C8 | A9 | 00 | 91 | E0 |
| 6038- | C8 | A9 | 00 | 91 | E0 | A0 | 0C | A9 |
| 6040- | 01 | 91 | E0 | A9 | 00 | A0 | 08 | 91 |
| 6048- | E0 | A9 | 96 | C8 | 91 | E0 | 20 | E3 |
| 6050- | 03 | 20 | D9 | 03 | 60 | A9 | 00 | 85 |
| 6058- | E2 | A9 | 96 | 85 | E3 | A9 | 11 | 8D |
| 6060- | 35 | 60 | A9 | 0F | 8D | 3A | 60 | A9 |
| 6068- | 01 | 8D | 40 | 60 | 20 | 2D | 60 | A0 |
| 6070- | 01 | B1 | E2 | 85 | E4 | C8 | B1 | E2 |
| 6078- | 85 | E5 | A0 | 0B | 84 | E7 | B1 | E2 |
| 6080- | 8D | 35 | 60 | C8 | B1 | E2 | 4C | 3C |
| 6088- | 61 | C8 | B1 | E2 | F0 | 07 | C9 | 80 |
| 6090- | F0 | 03 | 4C | A8 | 60 | C8 | A2 | 00 |
| 6098- | BD | 00 | 02 | D1 | E2 | D0 | 09 | C8 |
| 60A0- | E8 | E0 | 1E | D0 | F3 | 4C | B9 | 60 |
| 60A8- | C0 | DD | B0 | 0A | A4 | E7 | 98 | 18 |
| 60B0- | 69 | 23 | A8 | 4C | 7C | 60 | 4C | 1B |
| 60B8- | 61 | 20 | 2D | 60 | A0 | 01 | B1 | E2 |
| 60C0- | 85 | E4 | C8 | B1 | E2 | 85 | E5 | A0 |


```

60C8- 0C B1 E2 8D 35 60 C8 B1
60D0- E2 8D 3A 60 18 6D 35 60
60D8- 4C 48 61 C8 84 E6 EE 4A
60E0- 60 20 2D 60 E6 E3 A0 00
60E8- B1 E2 F0 03 20 ED FD C8
60F0- D0 F6 C6 E3 CE 4A 60 A4
60F8- E6 F0 03 4C C9 60 A5 E4
6100- 8D 35 60 A5 E5 8D 3A 60
6108- 4C B9 60 C9 CE D0 D5 D4
6110- A0 C6 C9 CC C5 A0 CE C1
6118- CD C5 BA A5 E4 A6 E5 8D
6120- 35 60 8E 3A 60 18 6D 3A
6128- 60 F0 03 4C 6C 60 A9 3E
6130- 4C 04 A7 A9 96 8D 4A 60
6138- 20 8E FD 60 8D 3A 60 18
6140- 6D 35 60 F0 E9 4C 89 60
6148- D0 03 4C FF FF 4C DB 60

```

*

为方便不熟悉机器语言的读者使用，下面列出相应的 BASIC 语言程序，其功能和运行过程和上述的机器语言程序相同。

见程序 PRO-116。

PRO-116

```

10 REM LIST MEMORY
20 CALL - 958: PRINT : FOR I =
   1 TO 3: POKE 1000 + I, PEEK
   (984 + I): NEXT
30 INPUT "INPUT FILE NAME:";FIL$

40 IF LEN (FIL$) < 30 THEN FIL$
   = FIL$ + " ": GOTO 40
50 PRINT : IOB = PEEK (43713) +
   PEEK (43714) * 256: KEY = 38
   400: Q = 150
60 TRA = 17: SEC = 15
70 GOSUB 290
80 ITRA = PEEK (KEY + 1): ISEC =
   PEEK (KEY + 2)
90 P = 11
100 PP = P: TRA = PEEK (KEY + P):
   P = P + 1: SEC = PEEK (KEY +

```

```

P): IF TRA + SEC = 0 THEN PRII
"FILE NOT FOUND": END
110 P = P + 1: IF PEEK (KEY + P)
    = 0 OR PEEK (KEY + P) = 12
    8 THEN 130
120 GOTO 250
130 FOR I = 1 TO 30: P = P + 1
140 IF MIDS$ (FIL$, I, 1) < > CHR$
    ( PEEK (KEY + P) - 128) THEN
    250
150 NEXT
160 GOSUB 290: P = 1: ITRA = PEEK
    (KEY + P): P = P + 1: ISEC = PEEK
    (KEY + P): P = 12
170 TRA = PEEK (KEY + P): P = P +
    1: SEC = PEEK (KEY + P)
180 IF TRA + SEC = 0 THEN END
190 P = P + 1: Y = P: Q = Q + 1: GOSUB
    290
200 KEY = 38656: P = 0
210 X = PEEK (KEY + P): IF X < >
    0 THEN PRINT CHR$ (X);
220 P = P + 1: IF P < 256 THEN 21
    0
230 Q = Q - 1: KEY = 38400: IF Y <
    = 256 THEN 170
240 TRA = ITRA: SEC = ISEC: GOTO 1
    60
250 IF P > 221 THEN 270
260 P = PP + 35: GOTO 100
270 TRA = ITRA: SEC = ISEC: IF TRA
    + SEC = 0 THEN PRINT "FILE
    NOT FOUND": END
280 GOTO 70
290 REM SUB
300 T = 3: POKE (IOB + T), 0
310 T = T + 1: POKE (IOB + T), TRA
    : T = T + 1: POKE (IOB + T), S
    EC
320 T = 12: POKE (IOB + T), 1: T =
    8: POKE (IOB + T), 0
330 T = T + 1: POKE (IOB + T), Q
340 CALL 995
350 RETURN

```


程序③

```
1000 REM SAMPLE-3
1010 D$ = CHR$(4):F$ = "NAME"
1020 PRINT D$;"OPEN";F$;"",L10"
1030 FOR I = 1 TO 5: READ X$
1040 PRINT D$;"WRITE";F$;"",R";I
1050 PRINT X$: NEXT
1060 PRINT D$;"CLOSE": END
1070 DATA OPEN,WRITE,READ,CATALOG,
      G,PRINT
```

CALL 24576

INPUT FILE NAME:NAME

OPEN
WRITE
READ
CATALOG
PRINT

这样一个程序所建立的随机文件由于 0 记录为空，其它记录也因字符数少于记录长度而有“空位”，因而不了解其格式就无法使用。利用 CALL24576 指令即能列出其内容。

4. 磁盘信息的显示和修改

上一部分中简单讲述了磁盘文本文件的存贮过程和格式，并提供了列印文件内容的工具软件，这一软件对于文件资料的使用以及掌握文件的存放格式起到了很好的作用，提高了数据文件的利用率。那么，读者可能碰到过这样一个问题，即由于各种各样的原因，磁盘文件（包括程序文件）的部分内容（有时可能只有一个或几个单元）受到破坏，致使整个文件无法使用。由于 DOS 命令无法处理被破坏点，因而文件出现截断或混乱的现象。为了找出并修改这些被破坏点，

使文件恢复使用,就必须对磁盘内容进行直接的读写和修改。另外,分析磁盘的信息记录方法、记录格式、各种索引表以及许多类型的保密、解密问题都必须通过对磁盘内容的直接读写和修改这一方法。

(1) RWTS 子程序及其参数表

在文章的内容中,曾几次提到过系统的 RWTS 子程序,这一子程序是DOS的一个特殊组成部分,它使得软件人员可以不通过正常DOS指令而对磁盘信息进行处理。RWTS子程序的处理体是磁盘中某一指定的扇区(DOS命令的处理体是磁盘中某一指定的文件)。

RWTS意即“Read or Write a Track and Sector”,其执行的起始地址在\$3D9(985)中。为使用这一子程序,必须先建立输入输出表(IOB表)、设备特征表及调用控制程序,为RWTS的执行提供所需的参数。

作为一个例子,我们现在从\$6000(24576)开始建立这些参数,并说明各参数的意义。

① 调用控制程序的建立

调用控制程序首先将IOB表存放地址的高位存到计算器A中,低位存到暂存器Y中,然后转RWTS入口,完成调用工作。

\$6000—A 9 60 LDA # \$60 将IOB表存放地址高位存入A中

\$6002—A 0 80 LDY # \$08 低位存入Y中

\$6004—20 D9 03 JSR \$03D9 转RWTS入口

\$6007—60 RTS 返回

② IOB表的建立

IOB表共占17个单元,指明插槽号、驱动器号、缓冲区

地址（用于存放读出的资料或要写入的资料）及读写状态等参数。

其中资料缓冲区地址一般不用键盘缓冲区（\$200—\$2FF）。因为在处理过程中可能要求一些输入，这些输入是由键盘缓冲区接受的，这样就会破坏读写内容。

| 位置 | 内容 | 意义 |
|--------|----|--------------------|
| \$6008 | 01 | IOB表状态指示器，为\$01 |
| \$6009 | 60 | 驱动器插槽号乘以16 |
| \$600A | 01 | 驱动器号 |
| \$600B | 00 | 磁盘卷号 |
| \$600C | 11 | 要读写的磁道号（\$00—\$22） |
| \$600D | 0F | 要读写的扇区号（\$00—\$0F） |
| \$600E | 19 | 设备特征表存放地址低位 |
| \$600F | 60 | 设备特征表存放地址高位 |
| \$6010 | 00 | 资料缓冲区地址低位 |
| \$6011 | 96 | 资料缓冲区地址高位 |
| \$6012 | 00 | 不用 |
| \$6013 | 00 | 不用 |
| \$6014 | 01 | 读命令，写命令为\$02 |
| \$6015 | 00 | 错误码 |
| \$6016 | 00 | 实际磁盘卷号 |
| \$6017 | 60 | 实际驱动器插槽号 |
| \$6018 | 01 | 实际驱动器号 |

③ 设备特征表的建立

| 位置 | 内容 | 意义 |
|--------|----|--------------|
| \$6019 | 00 | 设备状态标志，为\$00 |
| \$601A | 01 | 磁道现象数，为\$01 |

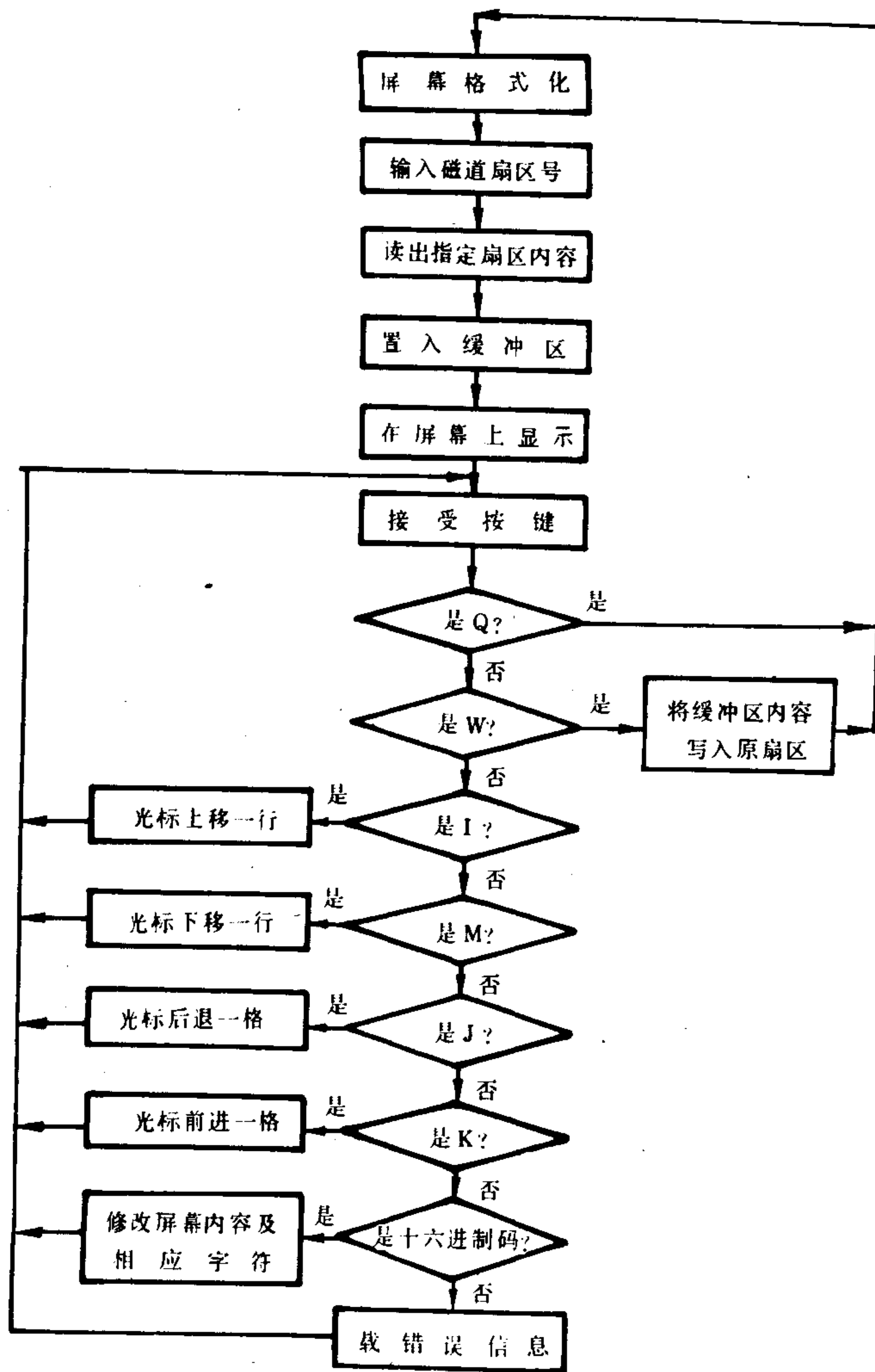


图 24

\$ 601 B EF 马达计时, 为 \$ EF

\$ 601 C D 8 马达计时, 为 \$ D 8

将以上内容输入指定地址, 用 600 G 或 CALL 24576 即可进行 IOB 表中指定磁盘位置的读或写动作。

不过, 实际上 IOB 表及设备特征表并不需要设计人员去建立, 在系统的 DOS 区中已经给出。表的起始地址在单元 \$ AAC1、\$ AAC2 中, 可使用指令从此取得, 或直接调用 \$ 03 E 3 的子程序, 以取得表的存贮地址 (执行 \$ 03 E 3 开始的子程序使表地址高位存于 A, 低位存于 Y 中)。

这样一来, R W T S 子程序的使用就显得非常简单。

第一步: 指定部分调用参数 (插槽号、驱动器号、磁盘卷号、磁道号、扇区号、读写标志等);

第二步: 使用 20 E 3 03 指令, 取得表地址;

第三步: 使用 20 D 9 03 指令, 调用 R W T S;

第四步: 60 (返回)。

(2) 磁盘信息的读写、显示和修改软件

通过以上分析, 我们对 R W T S 子程序及其参数已有了较多的认识, 现在详细介绍下面列出的磁盘信息处理软件。

这一软件具有灵活的光标移动方式、直观的屏幕显示和简便的操作使用功能。

粗框图见图 24。

从图 24 中可以看出软件执行的几个主要过程, 其中屏幕显示的格式如下:

```
1
00 D4C8C9D3A0C9D3A0C1A0D3CF THIS.IS.A.SO
0C C6D4D7C1D2C5A0D4CFCFCC8D FTWARE.TOOL.
18 D7C9D4C8A0C9D4A0D9CFD5A0 WITH.IT.YOU.
24 C3C1CEA0C4C9D2C5C3D4CCD9 CAN.DIRECTLY
30 A0CDCFC4C9C6D98DD2C5C1C4 .MODIFY.READ
```


除上述指令外,可按入十六进制码0—9,A—F的字符,以修改光标位置上的字符,屏幕右栏的显示将根据这一修改进行自动调整。

(3) 查找特定文件

上面所列出的软件在磁盘扇区内容的读写、显示和修改方面比较方便,它的处理体是磁盘中指定的一个扇区。但我们有时却需要处理一些特定文件的内容,而这一文件的存贮地址并不知道,如何查找这一文件呢?可以设计一段程序来完成这一工作,但实际上使用上面这个软件已能够胜任这一工作。

假设为前盘中有一名为“NAME”的文件,我们来列出它的内容。

第一步:读取磁盘目录区。通常目录区存放在\$11(17)磁道中,扇区的存放顺序是\$0F—\$00(15—0)。即先输入磁道17,扇区15,待屏幕显示后,从右栏中查找所需的文件名。如没有找到,用Q退回,再输入17磁道,14扇区,继续查找。读取磁盘目录区情况如下所示:

```
]CALL 24576
```

```
INPUT TRACK:17
```

```
INPUT SECTOR:15
```

```
00 00110E00000000000000000012 .....
0C 0F02C8C5CCCCCFA0A0A0A0A0 ..HELLO.....
18 A0A0A0A0A0A0A0A0A0A0A0A0 .....
24 A0A0A0A0A0A0A0A0A00600130F .....
30 02D3C1CDD0CCC5A0A0A0A0A0 .SAMPLE.....
3C A0A0A0A0A0A0A0A0A0A0A0A0 .....
48 A0A0A0A0A0A0A0A00200140F00 .....
54 CEC1CDC5A0A0A0A0A0A0A0A0 NAME.....
60 A0A0A0A0A0A0A0A0A0A0A0A0 .....
```


从中可见，要查找的“NAME”已经在右栏中出现。

第二步：如已查找到文件名，则中间一栏文件名第一个字符右边的一个单元内容是文件类型标志。其值为\$00、\$01、\$02、\$04时分别表示是未锁住的文本文件、整型BASIC程序、APPLESOFT程序和二进制文件；为\$80、\$81、\$82、\$84时表示锁住的文本文件、整型BASIC程序，APPLESOFT程序和二进制文件（本例中该单元的序号显然是\$53，其值为\$00，所以，“NAME”是未锁住的文本文件）。

再右边的2个单元是该文件磁道/扇区一览表的存放位置：第一个为磁道号，第2个为扇区号，记下这2个单元的值。本例中一览表在\$14(20)磁道、\$0F(15)扇区。

第三步：用Q退回，输入上面记下的磁道号和扇区号，读出一览表内容：

```
]CALL 24576
```

```
INPUT TRACK:20
```

```
INPUT SECTOR:15
```

```
00 00000000000000000000000000 .....  
0C 140E0000000000000000000000 .....  
18 00000000000000000000000000 .....
```

第四步：一览表中\$01、\$02单元为一览表续表的磁道、扇区号，若都为\$00，则表示无续表。从\$0C单元开始，每2个单元表示该文件256个字节内容的磁道、扇区号。本例中，文件只占一个扇区，即\$14(20)磁道的\$0E(14)扇区。

第五步：根据一览表中提供的磁道、扇区号，读出文件

内容，查找需修改的地方，进行修改：

```
]CALL 24576
```

```
INPUT TRACK:20
```

```
INPUT SECTOR:14
```

```
00 D4C8C9D3A0C9D3A0C1A0D3CF THIS.IS.A.SO  
0C C6D4D7C1D2C5A0D4CFCFCC8D FTWARE.TOOL.  
18 D7C9D4C8A0C9D4A0D9CFD5A0 WITH.IT.YOU.  
24 C3C1CEA0C4C9D2C5C3D4CCD9 CAN.DIRECTLY  
30 A0CDCFC4C9C6D98DD2C5C1C4 .MODIFY.READ  
3C A0C1CEC4A0D7D2C9D4C5A0C9 .AND.WRITE.I  
48 CEC6CFD2CDC1D4C9CFCE8DCF NFORMATION.O  
54 CEA0C1A0C4C9D3CB8D000000 N.A.DISK....
```

右栏中所列即是“NAME”文件的内容。

第六步：修改完后，用W命令写回磁盘。

至此，读者已经可以通过这一软件对磁盘信息进行随心所欲的处理了。作为一种简单的保密措施，它可以修改文件类型、长度以及文件名（使它违反DOS的规定），也可以通过它分析磁盘的启动原理以及进行更高层的保密、解密。

软件清单见PRO-117：

PRO-117

***6000.629D**

```
6000- 4C 59 60 A2 00 BD 35 62  
6008- 20 ED FD E8 E0 0E D0 F5  
6010- 20 6F FD E0 03 B0 EC E0  
6018- 00 EA F0 E7 E0 02 F0 06  
6020- AD 00 02 29 0F 60 AD 00  
6028- 02 29 0F 0A 0A 0A 0A 85  
6030- E0 AD 01 02 29 0F 18 65  
6038- E0 C9 35 90 03 4C 03 60  
6040- C9 10 B0 01 60 C9 20 B0
```

| | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|
| 6048- | 04 | 38 | E9 | 06 | 60 | C9 | 30 | B0 |
| 6050- | 04 | 38 | E9 | 0C | 60 | 38 | E9 | 12 |
| 6058- | 60 | 20 | 42 | FC | A9 | 81 | 8D | 06 |
| 6060- | 60 | 20 | 03 | 60 | 48 | A9 | 8F | 8D |
| 6068- | 06 | 60 | 20 | 03 | 60 | 48 | 20 | E3 |
| 6070- | 03 | 85 | E1 | 84 | E0 | A0 | 05 | 68 |
| 6078- | 91 | E0 | 88 | 68 | 91 | E0 | 88 | A9 |
| 6080- | 00 | 91 | E0 | A0 | 08 | A9 | 00 | 91 |
| 6088- | E0 | C8 | A9 | 96 | 91 | E0 | A0 | 0C |
| 6090- | A9 | 01 | 91 | E0 | 20 | E3 | 03 | 20 |
| 6098- | D9 | 03 | A0 | 0D | B1 | E0 | C9 | 10 |
| 60A0- | D0 | 05 | A9 | 24 | 4C | 04 | A7 | C9 |
| 60A8- | 20 | D0 | 05 | A9 | 4C | 4C | 04 | A7 |
| 60B0- | C9 | 40 | D0 | 05 | A9 | 5B | 4C | 04 |
| 60B8- | A7 | C9 | 80 | F0 | F7 | 20 | 58 | FC |
| 60C0- | A9 | 00 | AA | A8 | 48 | 20 | DA | FD |
| 60C8- | 20 | F4 | FB | B9 | 00 | 96 | 48 | 20 |
| 60D0- | DA | FD | 68 | EA | EA | EA | EA | C9 |
| 60D8- | A1 | B0 | 02 | A9 | AE | 48 | 84 | E0 |
| 60E0- | 8A | 18 | 69 | 1C | A8 | 68 | 91 | 28 |
| 60E8- | A4 | E0 | E8 | C8 | F0 | 10 | E0 | 0C |
| 60F0- | D0 | D9 | 20 | 8E | FD | A2 | 00 | 68 |
| 60F8- | 18 | 69 | 0C | 4C | C4 | 60 | 20 | 35 |
| 6100- | FD | C9 | D1 | D0 | 06 | 20 | 8E | FD |
| 6108- | 4C | 00 | 60 | C9 | 88 | D0 | 28 | A5 |
| 6110- | 25 | C9 | 01 | B0 | 0C | A5 | 24 | C9 |
| 6118- | 04 | B0 | 06 | 20 | 3A | FF | 4C | FE |
| 6120- | 60 | A5 | 24 | C9 | 04 | B0 | 0A | 20 |
| 6128- | 1A | FC | A9 | 1A | 85 | 24 | 4C | FE |
| 6130- | 60 | 20 | 10 | FC | 4C | FE | 60 | C9 |
| 6138- | CA | F0 | D4 | C9 | C9 | D0 | 12 | A5 |
| 6140- | 25 | C9 | 01 | B0 | 06 | 20 | 3A | FF |
| 6148- | 4C | FE | 60 | 20 | 1A | FC | 4C | FE |
| 6150- | 60 | C9 | CD | D0 | 12 | A5 | 25 | C9 |
| 6158- | 15 | 90 | 06 | 20 | 3A | FF | 4C | FE |
| 6160- | 60 | 20 | 66 | FC | 4C | FE | 60 | C9 |
| 6168- | 95 | D0 | 2B | A5 | 25 | C9 | 15 | 90 |
| 6170- | 12 | A5 | 24 | C9 | 1A | 90 | 06 | 20 |
| 6178- | 3A | FF | 4C | FE | 60 | 20 | F4 | FB |
| 6180- | 4C | FE | 60 | A5 | 24 | C9 | 1A | 90 |
| 6188- | 0A | 20 | 66 | FC | A9 | 03 | 85 | 24 |
| 6190- | 4C | FE | 60 | 4C | 7D | 61 | C9 | CB |
| 6198- | F0 | D1 | C9 | B0 | 90 | 0C | C9 | C7 |
| 61A0- | B0 | 5C | C9 | BA | 90 | 0A | C9 | C1 |

```

61A8- B0 06 20 3A FF 4C FB 60
61B0- A4 24 91 28 A5 24 38 E9
61B8- 02 F0 09 C9 01 D0 F7 A4
61C0- 24 4C C7 61 A4 24 88 B1
61C8- 28 C9 C0 B0 06 38 E9 B0
61D0- 4C D6 61 38 E9 B7 0A 0A
61D8- 0A 0A 85 E0 C8 B1 28 C9
61E0- C0 B0 06 38 E9 B0 4C EC
61E8- 61 38 E9 B7 18 65 E0 48
61F0- 98 4A 18 69 1A A8 68 91
61F8- 28 A9 CB 4C 01 61 C9 D7
6200- D0 A8 20 E3 03 85 E1 84
6208- E0 A9 00 A0 03 91 E0 A0
6210- 08 91 E0 C8 A9 96 91 E0
6218- A0 0C A9 02 91 E0 20 1A
6220- FC A5 25 D0 F9 20 10 FC
6228- A5 24 C9 03 D0 F7 A2 00
6230- A4 24 B1 28 C9 C0 B0 06
6238- 38 E9 B0 4C 41 62 38 E9
6240- B7 0A 0A 0A 0A 85 E0 20
6248- F4 FB A4 24 B1 28 C9 C0
6250- B0 06 38 E9 B0 4C 5B 62
6258- 38 E9 B7 18 65 E0 9D 00
6260- 96 E8 F0 14 20 F4 FB A5
6268- 24 C9 1B D0 C3 A0 10 20
6270- F4 FB 88 D0 FA 4C 30 62
6278- 20 E3 03 20 D9 03 4C 00
6280- 60 8D 8D C9 CE D0 D5 D4
6288- A0 D4 D2 C1 C3 CB BA 8D
6290- C9 CE D0 D5 D4 A0 D3 C5
6298- C3 D4 CF D2 BA FF

```

★

5. 数据管理功能之扩充

以6502为CPU的低档八位微机数据管理功能（如数据的输入、输出及屏幕编辑等）较差，而且没有随机的数据管理软件。为方便广大用户进行大量数据的处理，充分挖掘其潜力，我们设计了这一功能扩充软件。

(1) 设计思路

磁盘数据文件的操作必须借助于程序，只有通过程序的

运行才能打开、关闭或读写文件，这就给大批量的数据管理带来一定困难，而且这样建立的数据文件没有较为统一的结构，难以为各种应用程序所共享。

另外，一般建立文件的程序是通过 DATA 语句或 INPUT 语句从键盘上接受数据的，这就给数据的编辑带来很大不便。一是系统本身的编辑操作过于繁杂，二是容易出错而达不到目的，且数据的输入速度也大大受到限制。

本软件正是在解决以上问题的基础上研制而成的，它采用行编辑的方式进行数据文件的管理，除了操作简便、执行速度快之外，还具有以下主要特点。

① 数据的输入速度快。本软件在处理同一文句的不同数据时使用空格作为分隔符，而且允许空格多于一个。这样，既提高了输入速度，也减少了 DATA 语句中多打逗号的出错机率。

② 能进行任意位置的文句删除和内插，且删除后文句自动前推以填补被删除的空间，内插后文句自动后移以挤入被插数据。

③ 在对指定文句进行编辑时，能快速将光标移至末尾进行数据续加和在文句中任意位置进行字元的删除、插入和修改，被编辑的文句及其在屏幕上的显示方式根据键盘接受信息随时被调整。

④ 在对较长的文句进行编辑时，为避免对逐个字元不必要的扫描，可直接将光标进行相邻行的上下移动。

⑤ 在对转换后的文件数据进行检索时，同一记录中的数据既能顺序进行，也可使用定长字节计数，有利于在各种情况下提高检索速度。

⑥ 在需要给出文件名时，即可不加指定使用默认驱动器，

也可直接指定驱动器号。

(2) 使用指令

为用户操作方便, 本软件多采用单字符指令, 其中一部分指令还具有多种参数选择方式, 下面逐一加以说明。

① 文句续加: A↵, 用于对指定文件进行资料续加。

② 文件建立: C↵, 用于建立一个新的磁盘文件。

③ 文句删除: D, 用于对指定文件进行文句删除, 共有四种形式:

- Di-↵: 删除第i句以后的资料;

- Di↵: 删除第i个文句;

- Di-j↵: 删除第i到j的文句;

- D-i↵: 删除1到i的文句。

④ 文句修改: E, 用于修改指定文件中的文句资料, 共有四种形式, 同上。

⑤ 文句内插: I, 用于在指定文件中进行文句插入, 共有四种形式, 同上。

⑥ 屏幕显示: L, 用于显示指定文件中的文句资料。共有五种形式:

- L↵: 显示全部文句。

⑦ 打印输出: P, 用于打印指定文件中的文句资料, 形式同⑥。

⑧ 清除内存: Q↵, 用于清除内存中的文件资料。

⑨ 字串替换: R, 使用指定字符串替换文件中的某一字符串, 共有五种形式, 同上。

⑩ 类型转换: T, 用于将建立好的文件转换成可供随机检索的目标文件, 共有五种形式, 同上。

⑪ 文件删除: W↵, 用于删除磁盘文件。

⑫ 文件链接: X↵, 用于链接两个同类型的文本文件。

⑬ 单句显示数字键↵, 用于显示相应的单个文句。

⑭ 循环显示: ↵, 用于顺序显示指定文件的文句资料, 按一次回车键显示一句。

⑮ 编辑指令:

- CTRL-A: 设置块起始位置;

- CTRL-B: 设置块结束位置;

- CTRL-C: 在当前光标位置上复制标记块;

- CTRL-D: 字符删除, 每按一次删除光标位置上的一个字符;

- CTRL-E: 将光标移至文句末尾;

- CTRL-I: 字符内插, 按一次进入内插状态, 可插入若干个字符, 插入位置在光标之前;

- CTRL-L: 光标下移一行;

- CTRL-O: 光标上移一行;

- CTRL-P: 光标前进十个位置;

- CTRL-Q: 放弃已作修改, 退出本句编辑;

- CTRL-S: 将光标移至文句首;

- CTRL-Y: 光标后退十个位置;

- ← 光标后退一个位置;

- → 光标前进一个位置;

- RETURN: 本句编辑结束。

(3) 操作说明

本软件从文件的建立、显示、修改、增删到随机检索的整个过程都具有比较完善的功能, 而且, 只要掌握了前述的操作指令, 使用非常简便。现将其中几个主要步骤说明如下。

① 文句续加: 键入 A ↵, 则显示下一文句号, 等待数据输入。输入结束只要按回车键退出。

② 文句删除:

- 删除第 i 句以后, 键入: Di - ↵;
- 删除第 i 句, 键入: Di ↵;
- 删除第 i 到 j 句, 键入: Di - j ↵;
- 删除第 1 到 i 句, 键入: D - i ↵。

之后, 自动调整文句结构。其它文句修改、内插、打印、显示等操作与此基本相同。

③ 文句编辑: 自动显示编号及相应的文句, 并将光标定于句首等待按键。

- 将光标移至需要修改、插入或删除字符的位置上 (使用 ←、→、CTRL-L、CTRL-O、CTRL-E、CTRL-S、CTRL-P、CTRL-Y 等指令键);

- 使用编辑指令进行修改、插入或删除。

在操作过程中, 系统随时检测是否有非法输入, 提供以下信息:

- Syntax error: 非法指令;
- Bad file name: 非法文件名;
- Bad file information: 文件参数错误;
- Illegal quantity: 参数溢出;
- Enter record: 输入记录参数;
- Bad string length: 字符串超长;
- no string mark: 无字符串标志。

(4) 软件评价

① 本软件在执行过程中只有两种状态, 即执行或输入状态、指令状态, 而且随时可由一种状态退出进入另一种状态。

、这样,一旦发现错误即可进行编辑,编辑后又可随时进行资料的再输入。

② 使用指令操作方式,既便于掌握也避免了转换功能时回到菜单程序进行选择的繁琐操作,而且作为软件本身略去了菜单程序的篇幅和执行时间。

③ 本软件特别考虑了系统编辑功能的缺陷,设计出一组功能较强的光标移动和编辑指令。

④ 由于系统本身的磁盘操作速度较慢,在大文件中进行记录内插或删除的文句调整速度也较慢,而且拉长了磁盘的动作时间。

6. 全屏幕程序编辑系统

调试是软件开发过程中的一个重要环节,而编辑则是实现调试的基本手段,系统编辑功能的强弱直接影响到设计的效率和软件的可靠性,因此是使用人员较为关心的问题。

众所周知,APPLE-Ⅱ及其兼容机的屏幕编辑功能较差,主要表现在两个方面:一是编辑操作比较复杂,如字符内插、删除及程序行接收等都必须通过从键盘上输入一系列指令字才能实现;二是屏幕显示方式与内存不一致,容易造成编辑错误,而降低了信息的可靠性。

为加强其屏幕编辑功能,提高使用效率,设计出这一全屏幕程序编辑软件。这一软件使用6502机器语言编写,占用主机内存6K左右。使用证明,这一软件在BASIC程序的输入和修改方面大大加强了原有系统程序的功能。

(1) 设计思路

APPLE-Ⅱ系列机从开机到输入处理主要完成以下工作:

① 产生复位信号,执行RESET开工程序,包括设置正常显示方式、正常文本窗口、输入输出设备、屏幕显示格式及访问磁盘驱动器。

② 初始化主机内存,包括设立零页扫描子程序,测定用户最高地址,建立初始化系统指针等。

③ 接收键盘输入,包括接收输入字符,显示正常字符,处理特殊字符等。特别地,当遇到输入为ESC编辑键时,转ESC子程序入口。

ESC子程序采用光标位置字符扫描方式,将输入字符收入键盘缓冲区。因此,在任何位置按回车键后,其缓冲区指针和光标在缓冲区中的序号是相同的,光标位置后的所有字符不复有效。这就是其屏幕编辑后必须将光标移至末尾再按回车键的原因,也是其插入字符采用跳出、输入、跳回方式的原因。

另外,由于LIST入口程序(\$D6A5)的不足,使程序行显示在屏幕两端留有若干空格,给编辑带来很多麻烦。

这种程序编辑过程的复杂性和不可靠性使许多初学者难以掌握,也使熟练的操作员无法得心应手。因为它既减慢了程序的输入速度,还容易造成出错现象。

本系统软件正是考虑到原有系统的以上缺陷而设计的。它具有快速方便的屏幕编辑功能和键盘保留字功能,灵活多样的光标移动方式和行号自动编排功能,且改变了原有程序行的显示方式。操作简便易学,指令设计合理。

(2) 程序块结构

本软件共有10个主要的功能程序块和3个资料表,各程序块分别完成独立的工作,以便于运行时随时调用。

功能程序块:

① 键盘保留字处理

为使程序和指令的输入更加方便迅速,该软件加强了原有的键盘保留字,另外设计了常用的18个保留字。它们是: AUTO、CHAIN、CLOSE、DATA、FOR、GOSUB、GOTO、INPUT、LEFT\$、MID\$、NEW、NEXT、OPEN、RETURN、RIGHT\$、STOP、THEN、VAL等,是由CTRL+字符或CTRL-SHIFT+字符构成的。当操作者从键盘输入控制字符时,屏幕上即显示相应的保留字字符串,同时内存中进行代码转换处理。

② 接受屏幕程序行

为了达到全屏幕编辑的效果,程序必须能够对屏幕显示的逻辑程序行进行处理,使其自动移入键盘缓冲区,供操作者编辑,这就是该程序块的功能。在操作者通过上、下光标移动键将光标移至屏幕程序行的一定位置进行编辑时,如键盘缓冲区的指针为零,则无法直接接受编辑,而是先快速将此程序行通过扫描接收进键盘缓冲区,然后再转向相应的编辑指令入口,实现编辑功能。

③ 程序编辑

这一功能程序块包含一整套较为完善的编辑指令执行程序,如光标移动、字符增删和修改、输入、显示、程序行屏幕显示格式的调整等,是主要程序块之一。

④ 不同数制间的转换

用户输入的程序行行号总是十进制形式的,而它在程序区存放时则是十六进制形式的,因此必须将此十进制的行号转换为十六进制。相反,将程序区中的内容显示到屏幕上时,其行号必须从十六进制转为十进制。

⑤ 内存搬移

程序在程序区是以一种链式结构，按行号从小到大的顺序存放的。当操作者需要删除或插入某些内容时，程序区必须进行相应的前推或后移。同时，链指针和程序参数指针（程序尾、变量表首、数组表首、数组表尾）也发生相应的变化，这种调整使程序区始终保持其固定的结构方式。另外，在一个指令或程序行输入结束前，其内容是存放在键盘缓冲区中的，这时如需要删除或插入某些字符，键盘缓冲区的内容也需进行相应的调整。这些调整实际上就是内存的搬移。内存搬移程序必须在搬移内容和地址上没有任何限制，即将内存中任一段的内容搬移至另一个地方，包括前移、后移和有部分地址重叠的搬移。

⑥ 程序行显示

根据键入的指令实现相应的程序行显示方式，包括单个程序行显示、循环显示和连续显示等，以便于校对和编辑。显示过程是通过搜索程序区，将程序区相应内容映射到屏幕上来实现的。

⑦ 程序行删除

在操作者对内存程序发出删除某些行的指令时，系统首先从程序区中找出需要删除的起始地址和终止地址。然后，将有关的链指针和程序参数指针进行相应调整，并将终止地址至程序尾的所有内容顺序搬移至起始地址开始的内存单元中，以覆盖这一段空间，从而完成程序行的删除。

⑧ 自动行号编排

这一功能包括行号自动编写和行号自动调整两个方面，是提高输入速度和合理调整用户程序的有效措施。

⑨ 磁盘操作功能

为方便用户操作,减少不必要的运行中断,设计了部分常用的磁盘操作功能,包括列目录、文件锁存、文件解锁、文件删除、文件输入输出等。

⑩ 常规侦错

这一部分模仿人工阅读、分析的步骤,使用静态分析法对内存程序进行错误检测。检测的主要内容包括:

- 语法分析:错误指令、非法变量名、非法文件名、非法表达式及标点符号等;
- 程序结构分析:子程序调用和返回、转向语句、匹配结构。

主要资料表:

① 出错信息表

在出现用户输入错误指令及常规侦错结果时,根据出错信息表提供相应的错误信息,以便于改正或重新输入。

② 保留字及其代码表

用作接受程序输入时,将保留字转换为内部代码和程序行显示时将内部代码转换为保留字。

③ 数制转换表

在不同数制的转换中使用进位查表方式组合而成。

(3) 运行流程

软件执行的第一步为内存检测,即检测主机内存、程序存放等。在结构不符或程序过长时将给出无法管理的信息而返回BASIC。

图25是其执行过程中主要步骤的粗框图。

框图中的任一过程都有一程序段与之对应,从而使整个系统在主流程的基础上形成较为完善的结构和功能。下面介绍程序编辑和程序区处理这两个核心模块的运行流程,以便

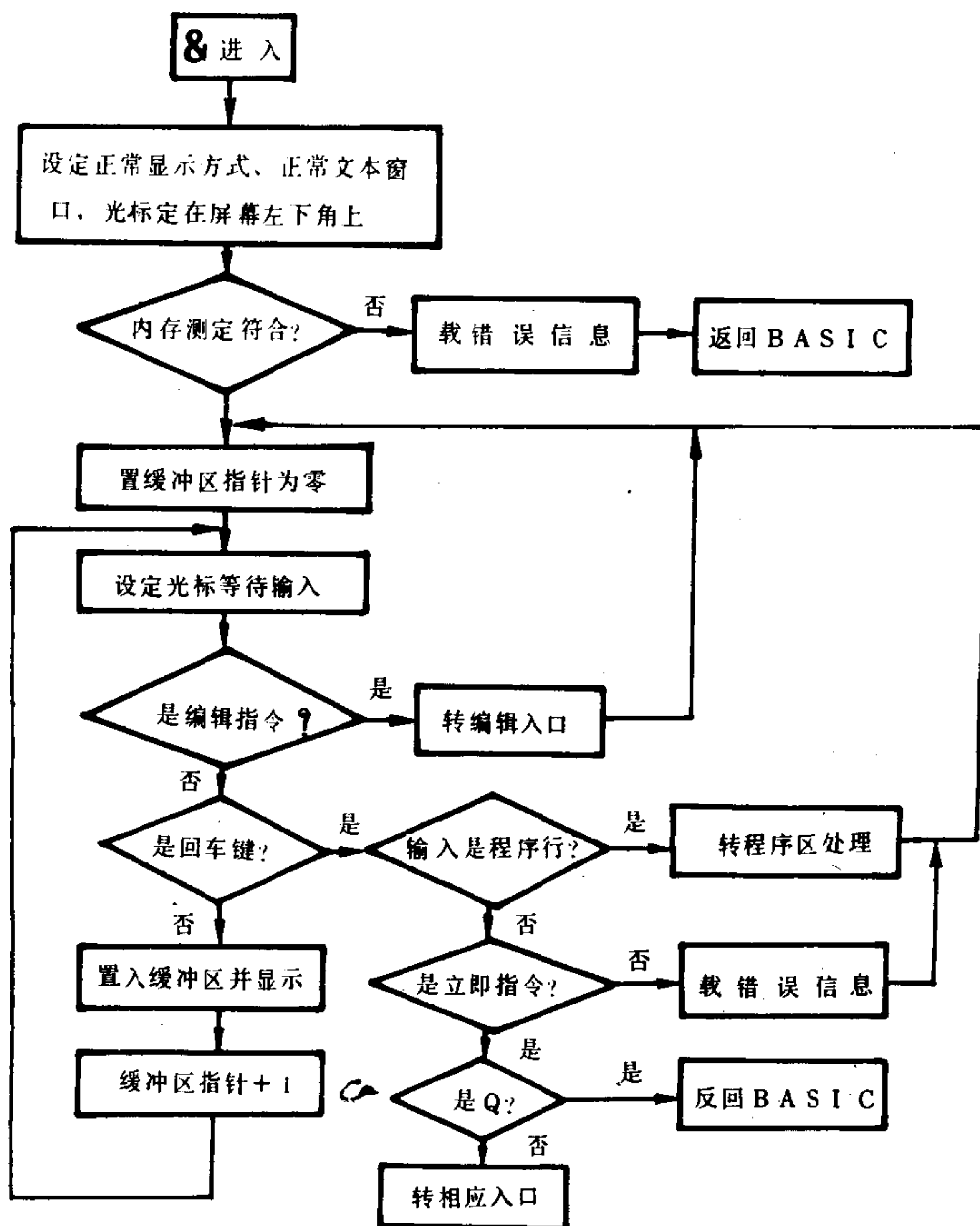
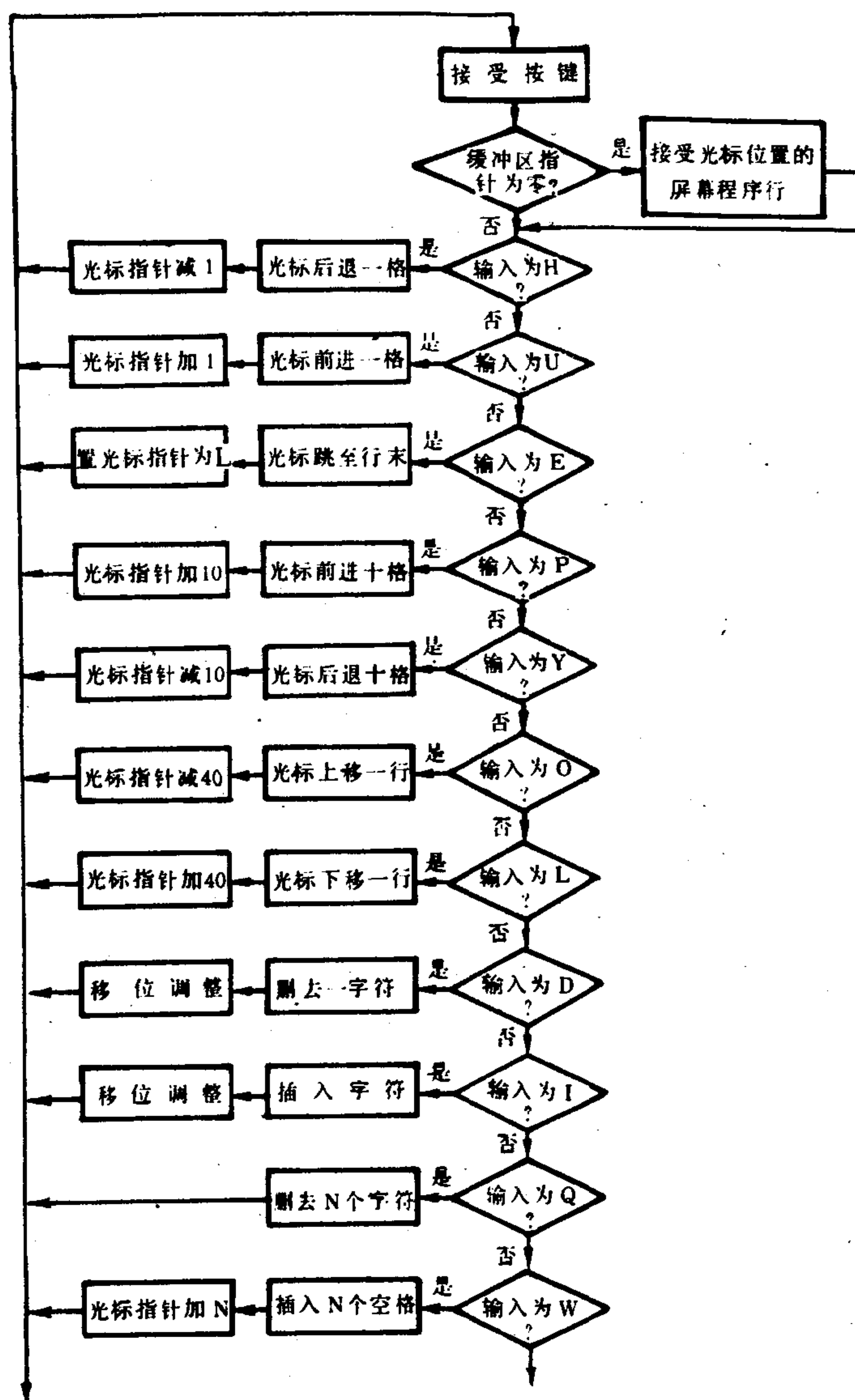


图 25

让读者了解操作指令及其执行情况。

① 程序行编辑

接受输入的程序在检测到有键盘输入时首先判断其是否属于编辑指令，如是则转为执行编辑处理程序，实现编辑操作，见图 26。



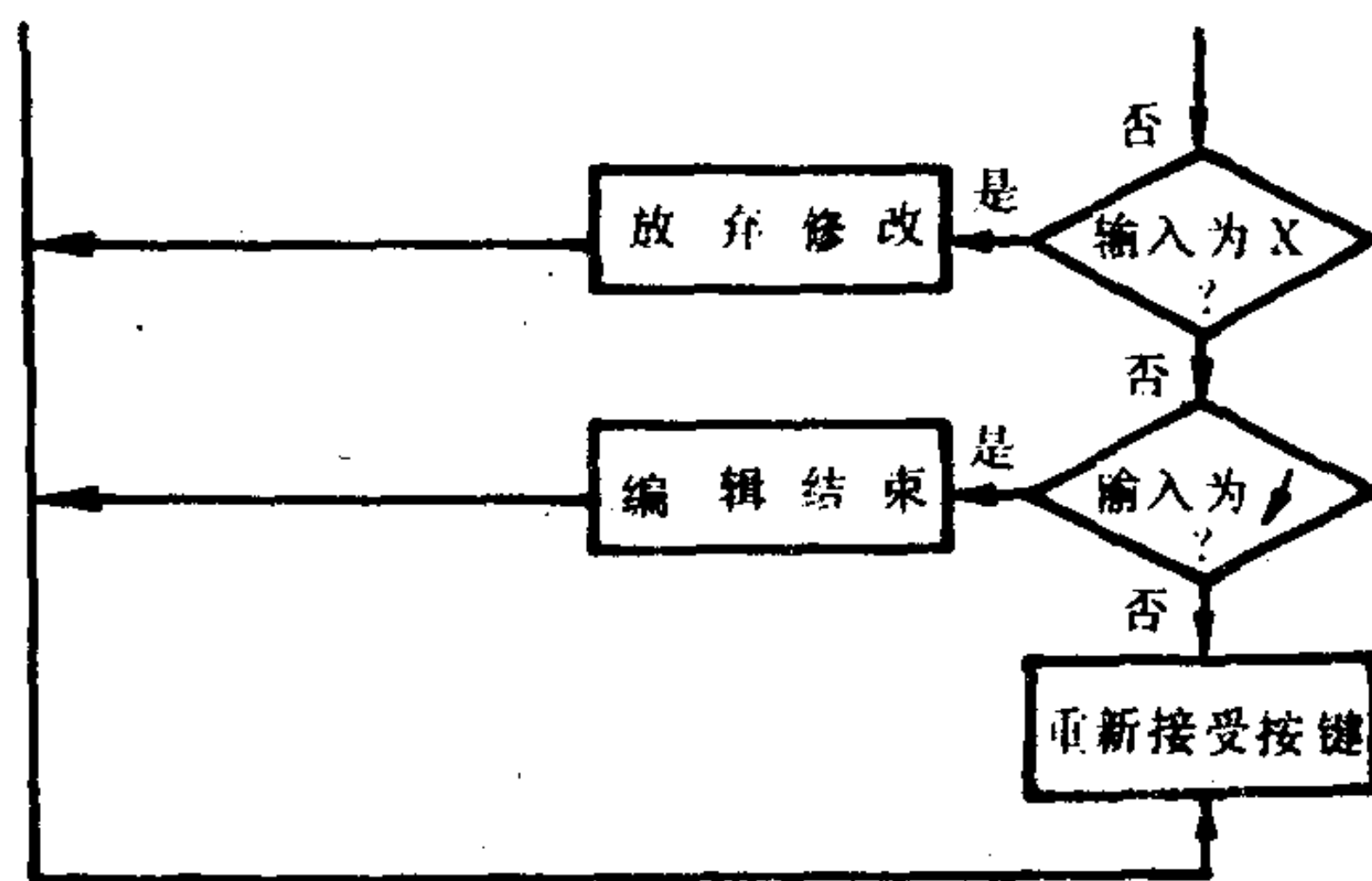


图 26

② 程序区处理

程序区处理的内容比较复杂,它包括行长检测、ASCII 码转换及程序区搬移、存放等。

框图如图 27 所示。

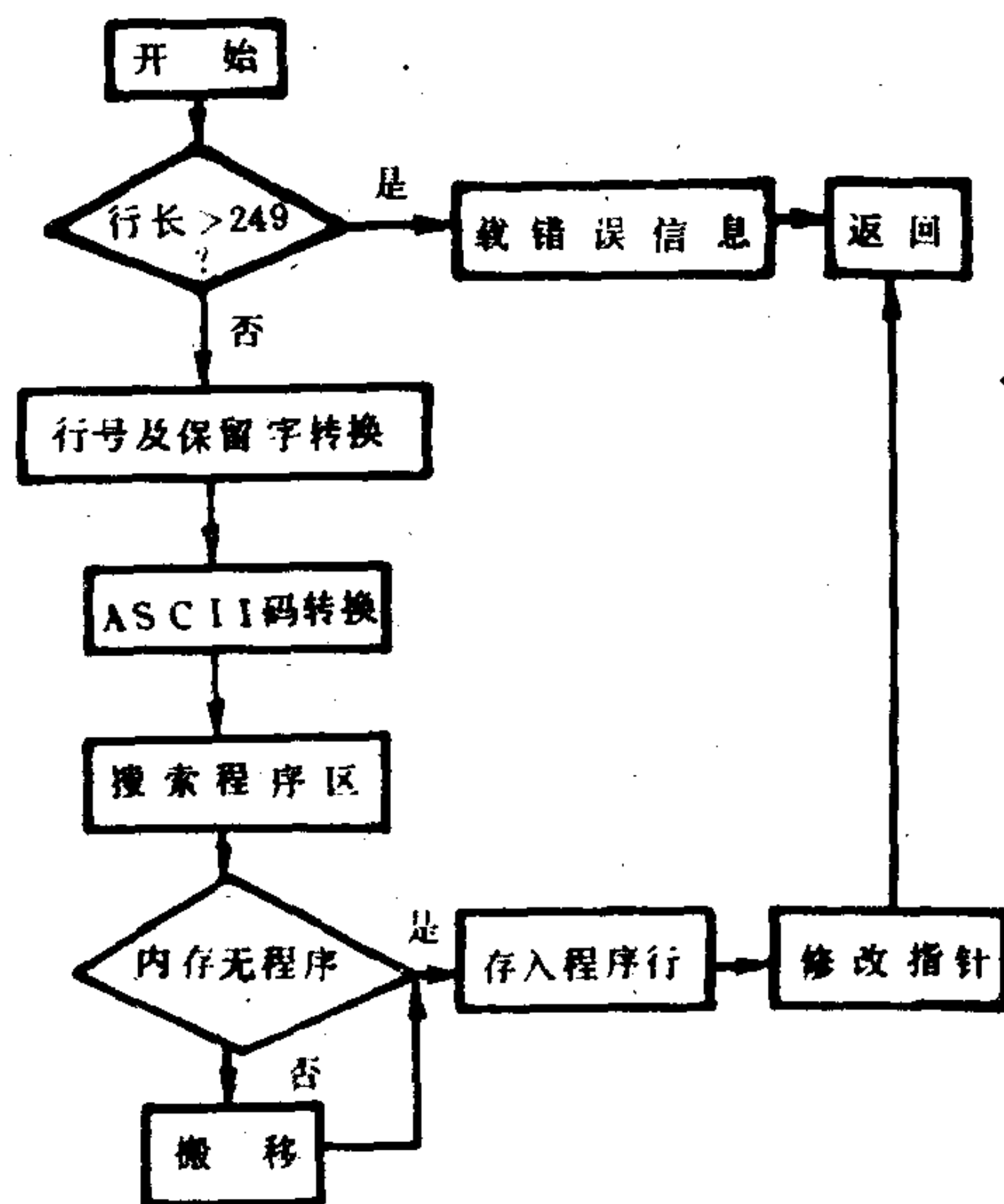


图 27

(4) 目的与评价

设计本软件的目的,是为提高程序管理效率提供一个有效的手段。实际使用说明,其全屏幕编辑功能与系统原编辑功能相比,确有很大的改进。

① 操作更为简便灵活。由于软件设计了较强的编辑功能以及灵活的光标移动方式,编辑操作极为简便。如插入字符的原过程是:先将光标移至行首;用→键移至一定位置;按ESC I上跳一行;输入字符;按ESC M下移;按↑键复位;用→键移光标至末尾;按回车键。这样一个繁琐的操作过程(实际编辑时还要对屏幕两端的空位进行ESCK删除处理)在软件下面变为:将光标直接移至一定位置;按CTRL-I进入内插状态;输入内插字符;直接按回车键。

无疑,是简单多了。还有其它一些功能,都在不同程度上简化了操作手续。

② 提高了程序编辑的速度。系统原编辑功能的实现没有借助于屏幕显示的逻辑程序行内容,而是通过键盘输入和光标进入键重新建立缓冲区内容和指针的。这样,编辑速度就会受到限制。本软件则首先借助于屏幕显示,自动建立缓冲区的基本内容和指针,然后进行任意修改。也即编辑过程可以排开不需修改的内容。

③ 提高了程序编辑的可靠性。原编辑过程容易造成出错,而其缓冲区内容与屏幕显示的不一致(如在字符内插、删除时)使错误不能立即被发现。这样整个软件的可靠性势必因此而降低。针对这些问题,本软件的解决办法是:

- 尽可能简化程序编辑过程,减少编辑错误;
- 使缓冲区内容与屏幕显示格式一致,便于发现并修改错误;

• 设计灵活多样的指令形式，使编辑功能更加完善实用。

由于编写软件的尽可能简炼和较多地使用了零页指令，执行速度较快。另外，自编行号功能和键盘保留字功能大大地提高了用户程序的输入速度。

(5) 操作指令

为实现屏幕操作、程序编辑等功能，本软件设计了多种操作指令，其中一部分指令还具有灵活的参数选择方式。

① 行号自动编写：指令 (CTRL-X退出)

AUTO <起始行号> <, 增量>

其中起始行号和增量两个参数可任意省略。省略时默认值为10。如

AUTO ✓ 表示行号从10开始，每次递增10；

AUTO, 20 ✓ 表示行号从20开始，每次递增20。

② 程序行删除：指令

D <起始行号> <, 结束行号> 或 D <起始行号> <- 结束行号>

其中起始行号和结束行号两个参数可省略任意一个。省略起始行号时删除从程序头开始到指定行；省略结束行号时删除从指定行开始到程序尾。

③ 程序行编辑：指令

EDIT <行号>

按回车后，自动显示出指定的程序行，并将光标定在行首，等待按键编辑。

④ 清洗屏幕：指令

HOME ✓

按回车后清除当前的屏幕显示，并将提示符及光标置于

屏幕的左上角。

⑤ 程序行显示：指令

L 〈起始行号〉〈, 结束行号〉或L 〈起始行号〉〈- 结束行号〉

其中起始行号和结束行号两个参数可任意省略。省略起始行号时表示显示从程序头开始到指定行；省略结束行号时显示从指定行开始到程序尾；全部省略时显示从程序头开始到程序尾。

在显示期间,可使用CTRL-S、暂停,也可使用CTRL-C中断。

⑥ 内存清除：指令

NEW ✓

其执行效果同APPLES OF T中的NEW指令。

⑦ 返回BASIC：指令

Q ✓

⑧ 屏幕格式化：指令

TEXT ✓

按回车后,屏幕返回文本状态,并置显示窗口、显示状态为正常,光标停在左下角。

⑨ 循环显示：指令

回车

循环显示指使用回车符对程序区进行逐句显示,每显示完一句,指针拨向下一程序行。到程序尾时又从头开始。

循环显示的指针受行输入、显示等的影响。

⑩ 单行显示：指令

行号

按回车后在屏幕当前位置上显示指定的程序行。

⑪ 编辑指令

编辑指令主要是操作者在进行程序行编辑时所发出的要求移动光标、增删或修改字符等的指令。

- CTRL-B: 将光标从当前位置直接移至行首;
- CTRL-E: 将光标从当前位置直接移至行末;
- CTRL-D: 删除当前光标位置上的一个字符, 后面的字符自动前推;

• CTRL-I: 进入内插状态 (屏幕显示没有变化), 以后的正常字符都将插在当前光标前, 光标及其后面的字符自动后移。

内插状态一直保持到下一个非正常显示字符的输入;

- CTRL-H或←: 光标后退一格;
- CTRL-U或→: 光标前进一格;
- CTRL-O: 光标上移一行;
- CTRL-L: 光标下移一行;
- CTRL-P: 光标前进十格;
- CTRL-Y: 光标后退十格;
- CTRL-X: 放弃当前行的编辑;
- 回车: 当前行编辑结束;
- ESC: 清除光标位置后的字符 (当前行), 光标位置不变。

⑫ 常规侦错: 指令

DETE ✓

⑬ DOS 操作指令

指令格式与DOS 3.3约定一致。

错误信息

当系统检测到用户有错误操作或出现常规侦错结果时,

将随时向用户报告错误信息，以便于立即更正。下面是系统所使用的错误信息集。

- NO PROGRAM IN MEMORY：内存无 BASIC 程序；

- SYNTAX ERROR：错误指令或程序行语法错误；

- PROGRAM TOO LONG：程序太长；

- UNDEFINED LINE NUMBER：指定行不存在；

- ILLEGAL LINE NUMBER：行号溢出；

- IF WITHOUT THEN：IF 无相应的 THEN 匹配；

- THEN WITHOUT IF：THEN 无相应的 IF 匹配

另外，还有一些 BASIC 和 DOS 设置的错误信息，在此就不赘述。

结 束 语

在本章的内容中，我们介绍了语句模拟、程序调试、文件操作、磁盘信息处理、数据管理以及全屏幕程序编辑等功能扩充问题，并提供了相应的软件，以实现这些扩充功能。

这些软件能够完成一定的功能扩充，解决一些比较实际的问题，但软件功能扩充是一门较深的技术，列出它们目的是为抛砖引玉，更多的工作还靠广大的计算机用户去完成。

七、文件管理

1. 随机处理文件

下面是一个实用的随机处理文件程序。本程序可以自由地去读、修改、加入以及建立随机文件和对文件记录进行列表。每一个记录都存贮一行由键盘键入的信息。

随机文件最重要的特点是它的记录最大长度是固定的，而且对记录的访问是随机的。因此，对那些处理工作无特殊规律，需要随机进行的实际问题，使用随机文件的结构形式，将是适宜的。

随机文件使用中应注意：

- 随机文件由记录组成，记录的最大允许长度由用户自行设定，其值在 1 至 32767 之间；
- 记录由用户编号，对于文件中的记录，只要用户指定了记录编号，则可以对该记录进行随机访问；
- 随机文件的总长度不能超过磁盘的最大容量；
- 记录可分若干项，项与项之间可用“，”号或回车符分隔。每个项只能是字符。数值必须转换成字符串才能存入随机文件。
- 随机文件在给出 OPEN 命令时，要宣告记录的长度，在 READ 和 WRITE 命令中要使用记录号码；
- 随机文件只有打开后才能进行读写，而且只有对某一记录写入信息后，才会读出信息；
- 通常用 0 号记录作为存放最后一个记录的序号，0 号

记录还可以用来存放其他统计等方面内容，以利快速和方便地读写资料。

下面简要说明随机文件的各种操作语句。

(1) 随机文件打开语句

OPEN 命令——打开文件。

对于随机文件的处理，首先要打开文件，随机文件打开语句的格式为：

〈行号〉 PRINT D\$; “OPEN 〈文件号〉, Sn,
Dm, Lp”

OPEN: 是语句定义符，打开文件的命令。

〈文件名〉: 是用户希望打开的文件或者是准备建立的文件的名字。

S: 中华机接口标志字。是系统规定的，不能更改。

n: 一个整数，代表磁盘驱动器所连接的接口序号，中华机 $n=6$ 。

D: 磁盘控制插板上机号标志字。是系统规定的，不能更改。

m: 代表接在磁盘控制板上磁盘驱动器的机号，中华机代一个驱动器 $m=1$ 。

L: 是长度标志字。

p: 是文件记录的最大长度值，表示字节数，是 1 到 32767 之间的整数。

Lp 项是随机文件记录的长度说明，此项绝不能遗漏，它是说明此文件是否为随机文件的唯一说明，也是区别顺序文件的主要标志。

Sn, Dm 是磁盘参数表中的参数。

上述随机文件打开语句的功能是：

① 如果以〈文件名〉为名的文件已存在〈磁盘参数表〉所指定的磁盘中，则系统打开这个文件；

② 如果此随机文件不存在于盘中，则建立一个以〈文件名〉为名的随机文件并打开它，该文件记录的最大字节数是p值。该文件名已被登记在磁盘目录中。

例如，我们要建立一个关于学生考试成绩资料的随机文件，取名为STUDENT，记录的最大长度是200个字节，若拟将该文件存在6号接口1号驱动器所装的磁盘上，而有关工作已准备就绪，则可以用下述语句：

```
100 PRINT D$; "OPEN STUDENT, S6, D1,  
L 200"
```

实现。此后，即可以向此文件中写入内容了。

(2) 随机文件关闭语句

CLOSE 命令——关闭文件。

文件一旦被开启，在处理工作完毕后，要及时关闭，否则文件会受到不必要的损伤甚至丢失。

随机文件关闭语句的格式为：

〈行号〉PRINT D\$; "CLOSE 〈文件名〉"

或 〈行号〉PRINT D\$; "CLOSE"

上述两种形式在使用中是有所区别的。

若在CLOSE命令后给定文件名称，则所关闭的只限于给定的文件；

若在CLOSE命令后不给定文件名称，则所关闭的是目前已打开的全部文件。

被关闭了的文件，除非重新打开，否则不能再用。

此外，关闭的动作是相对前面打开的文件而言，所以该文件的所在盘号、机号、接号点都是已经明确了，因此，

关闭语句中不允许〈磁盘参数表〉。

例如，要将前面打开的文件 STUDENT 关闭，可用下述语句实现：

```
500 PRINT D$;"CLOSE STUDENT"
```

如果要将其他打开的文件同时关闭，或者此时在系统中只有一个文件打开，或者在程序最后完成所有文件操作，要关闭它，则用下述语句同样可以达到目的：

```
1000 PRINT D$;"CLOSE"
```

(3) 随机文件写入语句

WRITE 命令——宣告写文件。

向随机文件中写入内容，要使用写入语句作为先导，然后通过前导语句后面的 PRINT 语句将内容写入。但是，这种写入必须指明操作是对哪一个记录进行的。这是和顺序文件的主要区别，同时应注意随机文件只接受字符信息，而对数字应先转换成字符串处理。

随机文件写入的一般形式为：

```
〈行号〉PRINT D$;"WRITE 〈文件名〉, R r"
```

在这个语句的后面，应当把需要写入的内容以字符型量的形式排列在一个 PRINT 语句之中。

实际写入的语句形式应为：

```
〈行号〉PRINT 〈字符串表〉
```

所以输出资料前，必须使用 WRITE 命令，它宣告要写文件。但执行这项命令并不能真正输出资料；执行 PRINT 才是实际输出资料。为什么要宣告写文件呢？这是因为系统可以同时打开和使用多个文件，而每个文件均用 PRINT 命令输出资料，但 PRINT 语句又无法指定要输出到哪一个文件中，因此在 PRINT 之前必须使用 WRITE 命令来宣告

所要输出的文件，且指定文件名称，这样，在WRITE后面的PRINT执行写时，对输出资料到什么地方才有依据。

在WRITE命令中，R r是一个宣告记录号码的参数，R是记录标志字，r代表记录号，是一个正整数。

因此，整个语句的含义是给指定的文件中的第r个记录作写入准备。在执行此语句后再执行第二个语句时，则把字符串表中的各变量的值作为一个记录写入原文件的第r个记录中。

(4) 随机文件读出语句

READ 命令——宣告读文件。

随机文件内容的读出也要以随机文件读出语句作为前导，后面用输入语句接收读出的内容。其一般形式为：

〈行号〉PRINT D\$;“READ〈文件名〉, R r”

〈行号〉INPUT〈串变量表〉

如果r是变量或数值型表达式，也可用下述形式：

〈行号〉PRINT D\$;“READ〈文件名〉, R”; r

第一个语句是随机文件读出语句，它的含义是读出由〈文件名〉指定的文件上第r个记录做准备。在执行到第二个语句时，该记录内容全被读出至缓冲区，再赋给INPUT语句中所列的各个字符串变量。

最后，关于整个随机文件的管理，还应注意：

① 如果需要列印结果，可在程序运行前键入命令PR# 1（西文状态），或POKE 1659, 1，而在程序运行后再键入命令PR# 0（西文状态），或POKE 1659, 0即可达到列印结果的目的。上述命令也可以放在程序中，POKE 1659, n对中文状态有效。

② 为使使用随机文件更为灵活和方便，文件名可以不放

在“”中，而改用 INPUT K\$ 输入，其余语句中的文件名可用下述方式，如：

```
PRINT D$;"OPEN";K$;","S6,D1,L 256"
```

```
PRINT D$;"READ";K$;","R";R
```

```
PRINT D$;"WRITE";K$;","R";M+1 等形式。
```

③ 如果不要列印，而在屏幕显示，程序中所有 GOSUB 170 句之前，最好加一个清屏语句 HOME 或 CALL -936。如 230 HOME: GOSUB，如果要打印输出结果，则所有 HOME 指令应删去（指中文输出），否则打印机会乱走纸（系指用中华学习机控制）。

下面给出一个实用的随机文件管理程序 PRO-118，各程序段及主要语句都作了详细而简明的中文注释，阅读和理解不会有太大的困难。最好反复操作多次，以利对该软件有更多的了解。

PRO-118

```
2 REM PRO-118
5 REM 随机文件一
9 POKE 1659,5
10 D$ = CHR$(4)
13 INPUT "输入文件名:";K$
14 PRINT "文件名:";K$
15 PRINT : PRINT
20 PRINT "1-读记录"
30 PRINT "2-增加记录"
40 PRINT "3-修改记录"
50 PRINT "4-列出全部记录"
60 PRINT "5-建立随机文件"
70 PRINT "6-结束"
80 INPUT "请选择1-6:";D
90 ON D GOTO 230,330,440,600,110,710
```

```

100 GOTO 15
110 PRINT D$;"OPEN";K$;"",S6,D1,L3000": REM 建立随机文件
120 PRINT D$;"WRITE";K$;"",R0": REM 写零号记录
130 PRINT 0: REM 记录内容为零
140 PRINT D$;"CLOSE": REM 关闭文件
150 GOTO 15
160 REM 170-210读零号记录的内容
170 PRINT D$;"OPEN";K$;"",S6,D1,L3000"
180 PRINT D$;"READ";K$;"",R0": REM 读零号记录
190 INPUT M: REM 已写入的记录数
200 PRINT D$: REM 取消读命令
210 RETURN
220 REM 230-310 读记录
230 GOSUB 170
240 INPUT "输入记录数(输0返回):":R
250 IF R < 1 THEN 15: REM 返回
260 IF R > M THEN 240: REM 无此记录数
270 PRINT D$;" READ";K$;"",R":R
280 INPUT B$: REM 读数据
290 PRINT D$
300 PRINT : PRINT B$: PRINT : REM 显示数据
310 GOTO 240: REM 继续
320 REM 330-420增加记录
330 GOSUB 170
340 INPUT "输入增加的记录内容(按RETURN键返回):":B$
350 IF B$ = "" THEN 15
360 PRINT D$;"WRITE";K$;"",R":M + 1
370 PRINT B$
380 M = M + 1
390 PRINT D$;"WRITE";K$;"",R0"
400 PRINT M: REM 写记录数
410 PRINT D$: REM 取消写命令
420 GOTO 340: REM 继续
430 REM 440-580修改记录
440 GOSUB 170

```



```

450 INPUT "输入修改的记录数(输0返回):";R
460 IF R < 1 THEN 15
470 IF R > M THEN 450
480 PRINT D$;"READ";K$";",R";R
490 INPUT B$
500 PRINT D$
510 PRINT : PRINT B$: PRINT
520 INPUT "输入要修改的数据(不改按RETURN键):";C$
530 IF C$ < > "" THEN 550: REM 将修改后数据写入
540 GOTO 450
550 PRINT D$;"WRITE";K$";",R";R
560 PRINT C$
570 PRINT D$
580 GOTO 450: REM 继续
590 REM 600-700列出全部记录
600 GOSUB 170
610 R = 0
620 R = R + 1
630 IF R > M THEN 680
640 PRINT D$;"READ";K$";",R";R
650 INPUT B$
660 PRINT B$
670 GOTO 620
680 PRINT D$
690 INPUT "按RETURN键返回:";B$
700 GOTO 15
710 PRINT D$;"CLOSE": REM 关闭全部文件
720 POKE 1659,0
730 END

```

下面是几个操作方法及实例的说明。

① 文件建立

如想建立一个文件名为“中华”的随机文件，则在运行上述程序后，屏幕立即显示“输入文件名”的提示，这时从键盘送入“中华”两字（在中文状态下），机器立即打出文件

名：中华，并给出菜单和请选择 1—6 的提示，键入 5 ↵，就执行“建立随机文件”的操作，驱动器红灯亮，待灯熄灭后宣告文件建立完成。建立文件的程序段是 110—140 句。

```
]RUN  
输入文件名:中华  
文件名:中华
```

```
1-读记录  
2-增加记录  
3-修改记录  
4-列出全部记录  
5-建立随机文件  
6-结束  
请选择1-6:5
```

已经建立的文件（第一次）是不能执行读操作的，因为第一次写零号记录，而记录内容为“0”，所以在“请选择 1—6”的提示后，键入 1 ↵ 再输入记录数“1”时，机器没有反应，又出现“输入记录数”的提示，表明无记录可读，键入“0” ↵ 返回总控。

```
1-读记录  
2-增加记录  
3-修改记录  
4-列出全部记录  
5-建立随机文件  
6-结束  
请选择1-6:1  
输入记录数(输0返回):1  
输入记录数(输0返回):0
```

② 写入文件

返回总控后，键入 2↵，这时进入写文件模块，就是菜单上写明的第 2 个功能项——增加记录。

当出现“输入增加的记录内容”时，由键盘送入信息，如中华学习机的汉语拼音：

ZHONG HUA XUE XI JI

按 RETURN 键返回，再写，如 BASIC↵，不写按 RETURN 键返回总控，列出菜单。

增加记录的功能程序段在 330—420 句。

1-读记录
2-增加记录
3-修改记录
4-列出全部记录
5-建立随机文件
6-结束
请选择 1-6:2
输入增加的记录内容(按 RETURN 键返回):ZHONG HUA XUE XI JI
输入增加的记录内容(按 RETURN 键返回):BASIC
输入增加的记录内容(按 RETURN 键返回):

③ 读出记录

当上述文件名为“中华”的文件，已经写入信息后，就可以按 1↵ 读出，例如我们想读出刚才写进去的第一个记录：ZHONG HUA XUE XI JI，则在“输入记录数”的指示后键入 1↵ 即可，若不需要再读按 0↵。程序为 230—310 句。

1-读记录
2-增加记录
3-修改记录
4-列出全部记录
5-建立随机文件
6-结束
请选择 1-6:1
输入记录数(输 0 返回):1

ZHONG HUA XUE XI JI

输入记录数(输0返回):0

④ 记录列表

这是对已经写入文件的全部记录列表，例如我们已经写入的信息共 2 项，即：

ZHONG HUA XUE XI JI
BASIC

则在“请选择 1 — 6”提示后，键入 4 ↵ 即可。而按 RETURN 键再回总控。全部结束按 6 ↵。列表程序段：
600—700 句。

1-读记录
2-增加记录
3-修改记录
4-列出全部记录
5-建立随机文件
6-结束
请选择1-6:4
ZHONG HUA XUE XI JI
BASIO
按RETURN键返回:

1-读记录
2-增加记录
3-修改记录
4-列出全部记录
5-建立随机文件
6-结束
请选择1-6:6

⑤ 修改记录

例如我们已建立了另一个随机文件 L L L，其中已写入

信息:

中华学习机

建立随机文件

ZHONG HUA XUE XI JI

则可以键入 3 进行修改。

如将 ZHONG HUA XUE XI JI 改为APPLE可按下面实例执行。修改记录程序段在440句到580句。

]RUN

输入文件名:LLL

文件名:LLL

1-读记录

2-增加记录

3-修改记录

4-列出全部记录

5-建立随机文件

6-结束

请选择1-6:4

中华学习机

建立随机文件

ZHONG HUA XUE XI JI

按RETURN键返回:

1-读记录

2-增加记录

3-修改记录

4-列出全部记录

5-建立随机文件

6-结束

请选择1-6:3

输入修改的记录数(输入6返回):3

ZHONG HUA XUE XI JI

输入要修改的数据(不改按RETURN键):APPLE
输入修改的记录数(输0返回):0

1-读记录
2-增加记录
3-修改记录
4-列出全部记录
5-建立随机文件
6-结束
请选择1-6:4
中华学习机
建立随机文件
APPLE
按RETURN键返回:

1-读记录
2-增加记录
3-修改记录
4-列出全部记录
5-建立随机文件
6-结束
请选择1-6:6

2. 顺序处理文件

下面是一个实际使用的顺序处理文件程序。共有文件建立、读出、增写、删除以及文件跳区处理五项功能。程序采用模块结构，有中文菜单，操作方便。

所谓顺序文件，顾名思义是存取文件资料时，需按照先后顺序处理，无论是读取文件资料还是写入文件资料，都必须按次序一个一个地进行。因此，这种文件结构主要适用于资料量变化不大的工作。而对于每次均需处理文件中的大部份资料，仍不失为一种较好的文件结构。

顺序文件的缺点是不能快速取存所需资料，也不容易增加、删除和更新资料。

顺序文件内容的处理，要按下述步骤进行：打开文件、处理记录（读或写的操作）、关闭文件。而这些步骤的实现，则通过在程序中使用有关文件操作语句进行。各种操作语句的功能，除个别语句外，均与随机文件相类似，这里不再一一重复，今将顺序处理文件的程序格式小结如下：

(1) 将资料存入顺序文件

```
10 D$ =CHR$(4)
20 PRINT D$;"OPEN 文件名"
30 PRINT D$;"DELETE 文件名"
40 PRINT D$;"OPEN 文件名"
50 PRINT D$;"WRITE 文件名"
60 PRINT 资料
70 PRINT D$;"CLOSE"
```

(2) 从顺序文件中取资料

```
10 D$ =CHR$(4)
20 PRINT D$;"OPEN 文件名"
30 PRINT D$;"READ 文件名"
40 INPUT 资料
50 PRINT D$;"CLOSE"
```

(3) 在顺序文件中增加资料

```
10 D$ =CHR$(4)
20 PRINT D$;"APPEND 文件名"
30 PRINT D$;"WRITE 文件名"
40 PRINT 资料
50 PRINT D$;"CLOSE"
```


(4) 从顺序文件中任意记录读资料

```
10 D$ =CHR$(4)
20 PRINT D$;"OPEN 文件名"
30 PRINT D$;"POSITION 文件名, R "
40 PRINT D$;"READ 文件名"
50 INPUT 资料
60 PRINT D$;"CLOSE"
```

当然, 还有其它格式, 这里不再叙述。上面几种格式, 在我们介绍的程序中均有应用, 请对照上述格式和程序, 可以更好地理解整个程序的编制结构。

顺序处理文件程序见PRO-119:

PRO-119

```
5 REM PRO-119
10 REM 顺序文件
20 D$ = CHR$(4)
30 INPUT "输入文件名:";K$
40 PRINT "文件名:";K$
50 PRINT : PRINT
60 PRINT "A-建立顺序文件"
70 PRINT "B-文件读出"
80 PRINT "C-文件填写"
90 PRINT "D-文件展区"
100 PRINT "E-文件删除"
110 PRINT "F-结束"
120 INPUT "请选择A-F:";C$
130 ON ASC (C$) - 64 GOTO 150,310,400,500,580,610
140 GOTO 50
150 PRINT D$;"OPEN";K$;"",S6,D1": REM 建立顺序文件
160 PRINT D$;"DELETE";K$
170 PRINT D$;"OPEN";K$;"",S6,D1"
180 PRINT D$;"WRITE";K$
```

```

190 FOR I = 1 TO 5
200 READ M$,N$
210 PRINT M$;"",N$
220 NEXT I
230 PRINT D$;"CLOSE": REM 关闭文件
240 DATA 中华学习机,800
250 DATA 紫金-II,6000
260 DATA IBM-PC/XT,25000
270 DATA 长城-286,33000
280 DATA PC1500-A,2200
290 PRINT "顺序文件已建立": FOR I = 1 TO 1000: NEXT I
300 GOTO 50
310 PRINT D$;"OPEN";K$;"",S6,D1": REM 读顺序文件
320 PRINT D$;"READ";K$
330 ONERR GOTO 50
340 FOR I = 1 TO 50
350 INPUT M$,N$
360 PRINT M$; TAB( 15);N$
370 NEXT I
380 PRINT D$;"CLOSE";K$
390 GOTO 50
400 PRINT D$;"APPEND";K$;"",S6,D1"
410 INPUT "输入增加的记录个数:K=";K
420 FOR I = 1 TO K
430 INPUT "输入增加的记录内容:";M$,N$
440 PRINT D$;"WRITE";K$
450 PRINT M$;"",N$
460 PRINT D$
470 NEXT I
480 PRINT D$;"CLOSE";K$
490 GOTO 50
500 INPUT "输入记录指针向前移动的位置:";P
510 PRINT D$;"OPEN";K$;"",S6,D1"
520 PRINT D$;"POSITION";K$;"",R";P - 1
530 PRINT D$;"READ";K$

```

```

540 INPUT M$,N$
550 PRINT M$; TAB( 15);N$
560 PRINT D$;"CLOSE";K$
570 GOTO 50
580 PRINT D$;"OPEN";K$;"",S6,D1"
590 PRINT D$;"DELETE";K$
600 GOTO 50
610 END

```

JRUN

输入文件名:QA
文件名:QA

A-建立顺序文件
B-文件读出
C-文件增写
D-文件跳区
E-文件删除
F-结束

请选择A-F:A
顺序文件已建立

A-建立顺序文件
B-文件读出
C-文件增写
D-文件跳区
E-文件删除
F-结束

请选择A-F:B

| | |
|-----------|-------|
| 中华学习机 | 800 |
| 紫金-II | 6000 |
| IBM-PC/XT | 25000 |
| 长城-286 | 33000 |
| PC1500-A | 2200 |

输入文件名。

按 A↵, 进入建文件操作, 并提示“文件已建立”信息。

按 B↵, 读出文件全部内容。

A-建立顺序文件
 B-文件读出
 C-文件增写
 D-文件跳区
 E-文件删除
 F-结束
 请选择A-F:C
 输入增加的记录个数:K=2
 输入增加的记录内容:APPLE,4500
 输入增加的记录内容:R1,450

A-建立顺序文件
 B-文件读出
 C-文件增写
 D-文件跳区
 E-文件删除
 F-结束

请选择A-F:D
 中华学习机 800
 紫金-II 5000
 IBM-PC/XT 25000
 长城-286 33000
 PC1500-A 2200
 APPLE 4500
 R1 450

A-建立顺序文件
 B-文件读出
 C-文件增写
 D-文件跳区
 E-文件删除
 F-结束
 请选择A-F:D
 输入记录指针向前移动的位置:6
 APPLE 4500

按C↵，输入增写记录个数2，
 增写内容为APPLE，4500和
 R1，450

按B↵，除读出原来的存贮信
 息外，又增加了两个新记录，
 共7个。

按D↵，并输入6↵，这时读
 出第6项记录内容。

A-建立顺序文件
 B-文件读出
 C-文件增写
 D-文件跳区
 E-文件删除
 F-结束
 请选择A-F:E

按 E ↵, 全部文件内容被删除。

A-建立顺序文件
 B-文件读出
 C-文件增写
 D-文件跳区
 E-文件删除
 F-结束
 请选择A-F:F

按 F ↵, 程序结束停机。

关于顺序处理文件程序的几点说明:

① 各程序段语句安排

- 20—140句: 菜单和总控
- 150—300句: 建立顺序文件
- 310—390句: 读取顺序文件
- 400—490句: 文件增写
- 500—570句: 文件跳区处理
- 580—600句: 文件删除

② 总控说明

总控安排在 120 — 140 句, 采用开关转向语句 ON < 表达式 > GOTO < 行号 1 >, < 行号 2 >, < 行号 n > 的形式。

其中 C\$ 为输入的字符, 即选用菜单中的 A、B、.....、E、F, 用 ASC(C\$) - 64 处理后, 字符 A、B、.....、E、F 分别为 1、2、.....、5、6。这样做的好处是, 菜



INPUT K就增加了使用的灵活性,同时 420 句循环终值也用K表示,这样相互配合,使程序简洁又不易出错,并省去了附加条件语句的麻烦。

⑧ APPEND 命令

顺序文件建立后,如想增写一些记录,应如何做?显然,新的记录只能补充到原有记录的最后一个的后面。方法之一是使用APPEND命令。如本程序第三个模块中,设置了400句:

```
400 PRINT D$; "APPEND"; K$; ", S6, D1"
```

APPEND语句的作用与OPEN语句的作用有相似之处,也有差别。它们都有打开由〈文件名〉所指定的文件的功能,但是, APPEND语句只能用于已经存在的文件,如果由〈文件名〉所指定的文件并不存在于该磁盘,则 APPEND语句不能象OPEN语句一样建立起新文件。而系统对此将作出“FILE NOT FOUND”的出错处理。这是两者使用中的第一个差别。其二,执行OPEN语句后,文件记录的指针是指向文件的开始处(即第0号记录),而执行APPEND语句后,文件记录的指针指向文件的最后一个记录之后。

正因为 APPEND 语句执行后,文件记录指针指向文件的最末尾,所以可以实现新记录的增写工作。但必须在 APPEND 语句后面,加一写入语句,如本程序 440 句 PRINT D\$; "WRITE"; K\$。如果在 APPEND 语句后面,接了 READ 语句,只能得到“END OF DATA”的信息,而永远不能增写文件资料。

⑨ POSITION 语句

POSITION 语句(或称命令),是指文件指针位移的。

它的形式如本程序第四个模块中的 520 句: PRINT D\$;
"POSITION"; K\$;",R"; P-1。

其中R 是定义符应照写, P 是一个数值, 是指记录内容向前移动的位置。

上述语句的功能是: 记录指针向前移动P-1个位置(即移过P-1个记录)。

由于顺序文件记录的编号是自 0 号开始, 因此, 编号为 P-1 的记录中内容, 就是第P 项内容。

另外, 在使用中还应注意两点:

- 520句的POSITION语句不能和530句的READ语句对调位置, 否则READ语句的作用将被后面的POSITION语句取消;

- POSITION语句只能将文件记录指针向前移动, 但不能使其向后, 即语句中的P 值, 必须是非负整数。如果P 为负值, 必将导致出错停机。

⑩ 记录修改

顺序处理文件不容易修改记录, 这是顺序处理文件的一大缺陷, 补救的方法之一, 是在建立的最初文件的DATA语句中进行。

八、实用汉字程序

在我国，大量的信息是中文汉字，更好地解决计算机中文信息处理，不仅是计算机应用和研究的一个重要领域，而且也是计算机在我国普及和推广的前提。事实上，许多有价值的实用软件，由于缺少中文汉字的必要说明，使得它们的使用、推广变得困难和受到限制。

为此，这里通过几个实例，说明汉字程序或软件的编制方法，更详细的内容，我们将在《中华学习机汉字软件》一书中详细说明。

在程序的适当地方，加注中文汉字的说明性语句，使程序或软件的易读性好，而用中文菜单的方式给出提示，既可以了解程序或软件的功能，又可以掌握程序或软件的流程及操作方法。这样，对未学过程序设计的读者，也可以非常直观地按照中文提示，进行游戏，学会管理；而对懂得程序设计语言的读者，则可以通过对程序设计思想和基本算法的介绍，更好地提高程序设计能力和编制软件的技巧。

在程序中如何加注汉字说明？一般有下面几种方法：

(1) 用REM语句

如：REM图书资料管理程序

REM人事档案管理程序

这样，人们一打开程序，立即就了解程序的性质、功能。REM语句常常放在程序的开头部份。

当程序采用模块化结构编制时，也可以在程序的相应地方，说明每段程序的功能。例如：

300 REM 功能选择 (见310—400句)

500 REM 数据分类 (见510—600句)

700 REM 数据检索 (见710—800句)

这样, 就能了解每段程序的功能, 程序从那里开始, 又到那里结束。

(2) 用PRINT语句

如: 300 REM 功能选择

310 PRINT“图书资料管理程序”

320 PRINT“1: 书目检索”

330 PRINT“2: 逐册检索”

340 PRINT“3: 单册检索”

350 PRINT“4: 退出检索”

这样, 操作者可以根据以上中文提示, 方便地选择各种操作, 而只要从键盘上输入1或2或3或4任何一个数字, 即可运行程序。

为使屏幕显示更为美观, 也可选用定位语句, 如:

VTAB3:HTAB5:PRINT“书名检索”

为更方便读者运行程序, 也可加注必要的操作说明, 如:

VTAB9:HTAB5:PRINT“按空格键继续查找, 按Z键返回总控”

(3) 用INPUT语句

如: 410 INPUT“请问找那一本书?”; A

650 INPUT“是否需要打印?”; K\$

730 INPUT“请输入书号:”; X

(4) 用DATA语句

如: 800 DATA 1, 中华学习机数据处理, 朱国江、占丰兴, 2.45元, 气象出版社, 1989.2。

810 DATA 2, 中华学习机汉字软件, 朱国江, 2.18元,
象出版社, 1989.4

这样, 就可以方便地安排 READ 语句和 PRINT 语句
读出并打印, 编号、书名、作者、定价、出版社、出版时间
等内容。

(5) 用字符串变量

如: 160 IF N = 0 THEN B\$ = "RUN": REM
运行程序

170 IF N = 1 THEN B\$ = "LOAD":
REM 调进内存

180 IF N = 2 THEN B\$ = "LOCK":
REM 文件加锁

190 IF N = 3 THEN B\$ = "DELETE":
REM 删除文件

总之, 根据程序的实例需要, 可以灵活运用上述语句,
加注简单明了的汉字说明, 将使程序结构看得更加清楚, 操
作方法更加明白。即使程序放置很长一段时间, 也因文字说
明清楚, 不需要再花时间阅读理解程序, 给学习和使用带来
方便。

下面我们给出一些汉字程序实例, 由于详尽地加注了汉
字说明, 阅读、理解和运行都不会发生困难。

1. 中华学习机 LOGO 语言引导程序

运行程序 PRO-120 后, 首先显示一个带字屏的方框, 然
后自动进入 LOGO 语言状态, 出现 LOGO 状态的提示符?,
就可以建立过程, 编制程序了。

PRO-120

```
5 REM PRO-120
10 PR# 3: CALL 43089
20 HOME : HGR2 : HCOLOR= 7
30 HPLOT 5,5 TO 270,5 TO 270,160 TO 5,160 TO 5,5
40 HTAB 5: VTAB 2: PRINT "中华学习机LOGO语言引导程序"
50 HTAB 10: VTAB 5: PRINT "欢迎使用LOGO语言"
60 HTAB 11: VTAB 8: PRINT "一九八八年五月"
70 FOR I = 1 TO 3000: NEXT
80 PRINT CHR$(4); "MAXFILES1"
90 LG : PRINT CHR$(13)
```

中华学习机LOGO语言引导程序

欢迎使用LOGO语言

一九八八年五月

2. 评委亮分

PRO-121, 是一个评委亮分程序。这里以歌手比赛评分为例实际上可以以本程序为模式, 适当修改即可作为各类比赛时评委亮分程序。

PRO-121

```
5 REM PRO-121
10 REM 评委亮分
20 L = 0: H = 10: N = 0
30 INPUT "输入评委人数: "; M
35 IF M = 0 THEN END
```

```

38 INPUT "输入歌手号码:";X
40 FOR I = 1 TO M
50 PRINT I;"号评委亮分:";: INPUT S
60 IF S > L THEN L = S: REM 放大数
70 IF S < H THEN H = S: REM 放小数
80 N = N + S: REM 放总数
90 NEXT I
92 PRINT : PRINT
95 PRINT "去掉一个最高分,再去掉一个最低分,";X;"号歌手得分:";
100 PRINT INT ((N - L - H) / (M - 2) * 100) / 100;: PRINT "分"
103 PRINT : PRINT
105 FOR I = 1 TO 3000: NEXT
110 GOTO 10

```

```

IRUN
输入评委人数: 8
输入歌手号码: 7
1号评委亮分: 99.25
2号评委亮分: 99.30
3号评委亮分: 99.50
4号评委亮分: 99.65
5号评委亮分: 99.20
6号评委亮分: 99.35
7号评委亮分: 99.40
8号评委亮分: 99.55

```

去掉一个最高分,再去掉一个最低分,7号歌手得分:9.39分

3. 预测身高

PRO-122是一个预测青少年、儿童、成年人身高的程序。男孩在7到21周岁之间,女孩在7到19岁之间,输入实足年龄和实测足长,即可预测将来的身高,据国家体委选拔

运动员试用证明，误差 $\pm 2\text{cm}$ 不到。

PRO-122

```
2 REM PRO-122
5 REM 预测身高
10 DIM A(15),B(15)
20 FOR I = 1 TO 15: READ A(I): NEXT
30 FOR I = 1 TO 13: READ B(I): NEXT
40 PRINT CHR$(26)
50 PRINT "预测男孩身高按 1"
60 PRINT "预测女孩身高按 2"
70 GET P$
80 IF P$ < ">" "1" AND P$ < ">" "2" THEN 50
90 IF P$ = "2" THEN 160
100 INPUT "输入男孩年龄<周岁> : ";N
110 IF N < 7 OR N > 21 THEN 100
120 INPUT "输入足长<cm> = ";X
130 C = INT (X * 10 / (0.146 * A(N - 6)) + 0.5) / 10
140 PRINT "男孩成年时身高 : ";C;" CM"
150 GOTO 210
160 INPUT "输入女孩年龄<周岁> : ";N
170 IF N < 7 OR N > 19 THEN 160
180 INPUT "输入足长<cm> = ";X
190 C = INT (X * 10 / (0.144 * B(N - 6)) + 0.5) / 10
200 PRINT "女孩成年时身高 : ";C;"
210 FOR I = 1 TO 3000: NEXT
220 GOTO 40
230 DATA 0.734,0.767,0.799,0.831,0.859,0.896,0.931,0.964,0.984
235 DATA 0.992,0.992,0.992,0.996,0.996,1.000
240 DATA 0.795,0.825,0.860,0.895,0.926,0.961,0.978,0.987,0.996
250 DATA 0.996,0.996,0.996,1.000

JRUN

预测男孩身高按 1
预测女孩身高按 2
```


输入男孩年龄(周岁): 7
 输入足长(cm) = 18.4
 男孩成年时身高: 171.7 CM

预测男孩身高按 1
 预测女孩身高按 2
 输入女孩年龄(周岁): 8
 输入足长(cm) = 18.7
 女孩成年时身高: 157.4

预测男孩身高按 1
 预测女孩身高按 2
 输入男孩年龄(周岁): 7
 输入足长(cm) = 19.0
 男孩成年时身高: 177.3 CM

预测男孩身高按 1
 预测女孩身高按 2

4. 中华学习机多功能引导程序

程序PRO-123有5个功能, 键入0—5中的任一数字而不需要回车, 即可执行运行程序、调入内存、文件加锁、文件解锁、删除文件的各种操作。同时, 本程序运行后, 可以定义10个功能键, 例如: 键入& 1↵, 程序自动列出磁盘文件目录, 键入& 2↵, 程序会自动运行内存中的程序等等。

PRO-123

```
5 REM PRO - 123
10 FOR I = 768 TO 851: READ X: POKE I,X: NEXT I
20 POKE 1014,0: POKE 1015,3
30 D$ = CHR$(4): PRINT D$;"PR#3": PRINT
35 HGR2 : HOME
40 PRINT "***中华学习机多功能引导程序***"
50 PRINT
70 PRINT "0 : 运行程序-----RUN或BRUN"
```

```

80 PRINT "1 :调入内存-----LOAD或BLOAD"
90 PRINT "2 :文件加锁-----LOCK"
100 PRINT "3 :文件解锁-----UNLOCK"
110 PRINT "4 :删除文件-----DELETE"
120 PRINT "5 :返回BASIC"
125 PRINT : PRINT "      请选择:";
130 GET C$
150 H = VAL (C$): IF H < 0 OR H > 5 THEN 130
160 IF H = 0 THEN B$ = "RUN"
170 IF H = 1 THEN B$ = "LOAD"
180 IF H = 2 THEN B$ = "LOCK"
190 IF H = 3 THEN B$ = "UNLOCK"
200 IF H = 4 THEN B$ = "DELETE"
210 IF H = 5 THEN TEXT : END
220 TEXT : HOME : PRINT : PRINT D$;"CATALOG"
225 A = PEEK (37) - 2: IF A > 22 THEN A = 22
230 FLASH : HTAB 30: VTAB 4: PRINT B$: NORMAL
235 B = 0:C = 4
240 FOR D = 0 TO 23: GOSUB 400: IF E < > 160 THEN POKE F - 1,219
245 POKE F,B + 193: POKE F + 1,221:B = B + 1:G = D
250 NEXT : VTAB 24
255 A$ = "&1=CATALOG &2=RUN &3=LIST &4=CALL-151&5=HOME &6=TRACE &7=NOT
TRACE&8=INVERSE &9=FLASH &0=NORMAL***"
260 HTAB 1: PRINT LEFT$ (A$,39);:A$ = MID$ (A$,2) + LEFT$ (A$,1)
265 H = PEEK (49152): IF H < 128 THEN FOR I = 1 TO 80: NEXT I: GOTO
260
270 HTAB 1: PRINT B$;: GET C$:H = ASC (C$) - 48
275 IF H < 17 OR H > B + 16 THEN PRINT D$;"PR#3": PRINT : GOTO 30
280 C = 1:D = G - B + H - 16: GOSUB 400
285 IF E = 194 AND (B$ = "RUN" OR B$ = "LOAD") THEN B$ = "B" + B$
290 FOR C = 6 TO 39: GOSUB 400:B$ = B$ + CHR$ (E): NEXT C
300 HTAB 1: CALL - 868: PRINT B$: PRINT D$;B$: GOTO 30
320 DATA 160,0,177,184,72,200, 32,152,217,104,201,48,208,3,76,115
326 DATA 242,201,49,208,3,76,110,165
327 DATA 201,50,208,3,76,18,217,201,51,208,3,76,165,214,201,52,208
328 DATA 3,76,105,255,201,53,208,3,76
329 DATA 88,252,201,54,208,3,76,109 ,242,201
330 DATA 55,208,3,76,111,242,201,56,208,3,76
340 DATA 128,254,201,57,208,3,76,128,242,76,201,220,0
400 K = INT (D / 8):L = D - K * 8
410 F = 1024 + 128 * L + 40 * K + C:E = PEEK (F): RETURN

```

5. PC-1500袖珍机语句及键名称一览表

运行程序PRO-124后,即可打印一张PC-1500袖珍机基

本 BASIC 语句和键的名称, 其中页码和《袖珍计算机 PC-1500 编写程序语言及外围设备》(气象出版社 1984 年 12 月出版)一书的页码完全一致。

应该注意的是, 在运行程序打印输出前, 必须先键入:
POKE 2043, 60

PRD-124

```
JPoke 2043,60
```

```
JLIST
```

```
10 REM
20 DIM A$(70),B$(70),C$(70)
30 FOR M = 1 TO 70
40 READ A$(M),B$(M),C$(M)
45 IF A$(M) = "END" THEN 60
47 T = T + 1
50 NEXT
60 REM
70 POKE 1659,5: POKE 1787,2: POKE 1915,3: POKE 2043,100
75 PRINT TAB( 19);"PC-1500袖珍机语句及键名称一览表"
76 PRINT
80 PRINT TAB( 1);"键/语句名称"; TAB( 20);"功    能";
  TAB( 60);"页 码"
90 PRINT
100 FOR X = 1 TO T
110 PRINT TAB( 1);A$(X); TAB( 20);B$(X); TAB( 60);C$(X)
115 IF X = 27 THEN POKE 1659,0: GET NN$: IF NN$ = "Y"
  THEN POKE 1659,5: 能"; TAB( 60);"页 码": PRINT
  PRINT TAB( 1);"键/语
120 NEXT
130 POKE 1659,0
200 DATA ON/OFF 键, 开关计算机, 8
205 DATA CL 键, 清屏语句, 9
210 DATA MODE 键, 控制状态语句, 9
215 DATA SHIFT 键, 第二功能键, 9
220 DATA SML 键, 大小写字母转换, 9
225 DATA DEL/INS 键, 插入/删除语句, 9
230 DATA 上/下移行键, 显视机内程序, 9
```

| | | |
|-----|------|------------------------------|
| 235 | DATA | ENTER 执行键, 输入键, 10 |
| 240 | DATA | DEF 键, 启动永存键, 10 |
| 245 | DATA | RCL 键, 控制备用键, 10 |
| 250 | DATA | LET 语句, 赋值, 36 |
| 255 | DATA | INPUT 语句, 输入语句, 39 |
| 260 | DATA | PRINT 语句, 输出语句, 40 |
| 265 | DATA | END 语句, 结束语句, 42 |
| 270 | DATA | STOP 语句, 暂停语句, 42 |
| 275 | DATA | GOTO 语句, 无条件转向语句, 44 |
| 280 | DATA | IF... THEN 语句, 条件判断语句, 45 |
| 285 | DATA | DIM 语句, 数组说明语句, 53 |
| 290 | DATA | FOR... NEXT 语句, 循环语句, 55 |
| 300 | DATA | GOSUB 语句, 转子语句, 64 |
| 305 | DATA | RETURN 语句, 子程序返回语句, 64 |
| 310 | DATA | ON... GOSUB 语句, 计算转移语句, 68 |
| 315 | DATA | ON... GOTO 语句, 计算转移语句, 68 |
| 320 | DATA | ONERR... GOTO 语句, 出错转移语句, 68 |
| 325 | DATA | CLEAR 语句, 清除存储语句, 69 |
| 330 | DATA | DEGREE 语句, 角度状态语句, 69 |
| 335 | DATA | RADIAN 语句, 角度状态语句, 69 |
| 340 | DATA | GRAD 语句, 角度状态语句, 69 |
| 345 | DATA | ASC 函数, 字符函数语句, 72 |
| 350 | DATA | CHR\$ 函数, 字符函数语句, 72 |
| 355 | DATA | INKEY\$ 函数, 输入字符, 73 |
| 360 | DATA | LEN 函数, 测字符串长度, 73 |
| 365 | DATA | LEFT\$ 函数, 左字符串函数, 73 |
| 370 | DATA | MID\$ 函数, 中字符串函数, 73 |
| 380 | DATA | RIGHT\$ 函数, 右字符串函数, 73 |
| 385 | DATA | RND 函数, 随机函数语句, 74 |
| 390 | DATA | RANDOM 函数, 随机函数语句, 74 |
| 395 | DATA | STR\$ 函数, 数 - 字互换函数, 75 |
| 400 | DATA | VAL 函数, 数 - 字互换函数, 75 |
| 405 | DATA | STATUS 函数, 查内存容量, 75 |
| 410 | DATA | DATA 语句, 数据语句, 77 |
| 415 | DATA | READ 语句, 数据语句, 77 |
| 420 | DATA | RESTORE 语句, 数据恢复语句, 78 |
| 425 | DATA | AREAD 语句, 自读语句, 79 |
| 430 | DATA | PAUSE 语句, 暂停显视语句, 80 |
| 135 | DATA | WAIT 语句, 显视时间语句, 80 |
| 140 | DATA | CURSOR 语句, 显视定位语句, 81 |

450 DATA GCURSOR 语句, 显视定位语句, 81
 460 DATA GPRINT 语句, 点阵显视语句, 82
 470 DATA POINT 函数, 光点显视函数, 85
 475 DATA USING 语句, 打印规格化语句, 86
 480 DATA BEEP 语句, 音响语句, 87
 485 DATA TEST 语句, 打印笔检验语句, 96
 490 DATA LLIST 语句, 程序列表语句, 96
 500 DATA TEXT 语句, 行文状态语句, 98
 510 DATA GRAPH 语句, 绘图状态语句, 98
 520 DATA COLOR 语句, 设颜色语句, 99
 10000 DATA END, E, E

PC-1500 袖珍机 语句及键名称一览表

| 键/语句名称 | 功 能 | 页 码 |
|---------------|---------|-----|
| ON/OFF 键 | 开关计算机 | 3 |
| CL 键 | 清屏语句 | 9 |
| MODE 键 | 控制状态语句 | 9 |
| SHIFT 键 | 第二功能键 | 9 |
| SML 键 | 大小写字母转换 | 9 |
| DEL/INS 键 | 插入/删除语句 | 9 |
| 上/下移行键 | 显视机内程序 | 9 |
| ENTER 执行键 | 输入键 | 10 |
| DEF 键 | 启动永存键 | 10 |
| RCL 键 | 控制备用键 | 10 |
| LET 语句 | 赋值 | 36 |
| INPUT 语句 | 输入语句 | 39 |
| PRINT 语句 | 输出语句 | 40 |
| END 语句 | 结束语句 | 42 |
| STOP 语句 | 暂停语句 | 42 |
| GOTO 语句 | 无条件转向语句 | 44 |
| IF...THEN 语句 | 条件判段语句 | 45 |
| DIM 语句 | 数组说明语句 | 53 |
| FOR...NEXT 语句 | 循环语句 | 55 |

| | | |
|---------------|---------|----|
| GOSUB 语句 | 转子语句 | 64 |
| RETURN 语句 | 子程序返回语句 | 64 |
| ON...GOSUB 语句 | 计算转移语句 | 68 |
| ON...GOTO 语句 | 计算转移语句 | 68 |
| ONERR.GOTO 语句 | 出错转移语句 | 68 |
| OLEAR 语句 | 清除存储语句 | 69 |
| DEGREE 语句 | 角度状态语句 | 69 |
| RADIAN 语句 | 角度状态语句 | 69 |

| 键/语句名称 | 功 能 | 页 码 |
|--------|-----|-----|
|--------|-----|-----|

| | | |
|------------|---------|----|
| GRAD 语句 | 角度状态语句 | 69 |
| ASC 函数 | 字符函数语句 | 72 |
| CHR\$ 函数 | 字符函数语句 | 72 |
| INKEY\$ 函数 | 输入字符 | 73 |
| LEN 函数 | 测字符串长度 | 73 |
| LEFT\$ 函数 | 左字符串函数 | 73 |
| MID\$ 函数 | 中字符串函数 | 73 |
| RIGHT\$ 函数 | 右字符串函数 | 73 |
| RND 函数 | 随机函数语句 | 74 |
| RANDOM 函数 | 随机函数语句 | 74 |
| STR\$ 函数 | 数~字互换函数 | 75 |
| VAL 函数 | 数~字互换函数 | 75 |
| STATUS 函数 | 查内存容量 | 75 |
| DATA 语句 | 数据语句 | 77 |
| READ 语句 | 数据语句 | 77 |
| RESTORE 语句 | 数据恢复语句 | 78 |
| AREAD 语句 | 自读语句 | 79 |
| PAUSE 语句 | 暂短显视语句 | 80 |
| WAIT 语句 | 显视时间语句 | 80 |
| CURSOR 语句 | 显视定位语句 | 81 |
| GCURSOR 语句 | 显视定位语句 | 81 |
| GPRINT 语句 | 点阵显视语句 | 82 |

| | | |
|----------|---------|----|
| POINT 函数 | 光点显示函数 | 85 |
| USING 语句 | 打印规格化语句 | 86 |
| BEEP 语句 | 音响语句 | 87 |
| TEST 语句 | 打印笔检验语句 | 86 |
| LLIST 语句 | 程序列表语句 | 96 |
| TEXT 语句 | 行文状态语句 | 98 |
| GRAPH 语句 | 绘图状态语句 | 98 |
| COLOR 语句 | 设颜色语句 | 99 |

6. 中华学习机区位码

对于不熟悉汉语拼音的读者，或者有一些特殊汉字，用拼音的方法不能查找到，程序 PRO-125 将为你解除困境。根据程序的中文提示，你可以打印出某一区号内的所有字符或者汉字。

程序清单和部分运行结果见 PRO-125。应注意体会本程序在打印格式的技巧，无论在屏幕上显示结果，或者从打印机上输出结果，格式都是非常整齐的。

PRO-125

```

1 REM PRO-125
2 POKE 2043,50
4 POKE 1787,7
6 POKE 1915,5
10 REM CEC-I 中华学习机
20 REM 区位字符及汉字查找
30 INPUT "输入区号:1-67 ";A
40 IF A < 1 OR A > 87 THEN 30
50 PRINT "第 ";A;" 区"
55 PRINT
60 J = 0
70 F = A: GOSUB 500

```



```

80 Q = FF
90 FOR B = 1 TO 94
100 F = B: GOSUB 500
110 W = FF
120 A$ = CHR$(127) + CHR$(Q) + CHR$(W)
130 IF B < 10 THEN B$ = "0" + STR$(B): GOTO 150
140 B$ = STR$(B)
150 IF A < 10 THEN A$ = "0" + STR$(A): GOTO 170
160 A$ = STR$(A)
170 X = 7 * (K + 1)
174 PRINT TAB(X);
176 PRINT A$;B$;A$;
180 J = J + 1: IF J = JJ THEN PRINT :J = 0
185 K = K + 1
187 IF X > 34 THEN PRINT :K = 0:X = X - 34
190 NEXT B
195 PRINT
200 PRINT : PRINT "是否还要查找 (Y/N)": INPUT G$
210 IF G$ = "Y" THEN PRINT :K = 0: GOTO 30
220 END
500 IF F > 0 AND F < 6 THEN FF = 28 + F
510 IF F > 5 AND F < 15 THEN FF = 29 + F
520 IF F > 14 AND F < 28 THEN FF = 30 + F
530 IF F > 27 AND F < 95 THEN FF = 31 + F
540 RETURN

```

IRUN

输入区号: 1-67 3

第 3 区

| | | | | | | | | | |
|------|---|------|---|------|---|------|---|------|---|
| 0301 | ! | 0302 | " | 0303 | # | 0304 | ¥ | 0305 | % |
| 0306 | & | 0307 | ' | 0308 | (| 0309 |) | 0310 | * |
| 0311 | + | 0312 | , | 0313 | - | 0314 | . | 0315 | / |
| 0316 | 0 | 0317 | 1 | 0318 | 2 | 0319 | 3 | 0320 | 4 |
| 0321 | 5 | 0322 | 6 | 0323 | 7 | 0324 | 8 | 0325 | 9 |

| | | | | | | | | | |
|------|---|------|---|------|---|------|---|------|---|
| 0326 | : | 0327 | ; | 0328 | < | 0329 | = | 0330 | > |
| 0331 | ? | 0332 | @ | 0333 | A | 0334 | B | 0335 | C |
| 0336 | D | 0337 | E | 0338 | F | 0339 | G | 0340 | H |
| 0341 | I | 0342 | J | 0343 | K | 0344 | L | 0345 | M |
| 0346 | N | 0347 | O | 0348 | P | 0349 | Q | 0350 | R |
| 0351 | S | 0352 | T | 0353 | U | 0354 | V | 0355 | W |
| 0356 | X | 0357 | Y | 0358 | Z | 0359 | [| 0360 | \ |
| 0361 |] | 0362 | ^ | 0363 | _ | 0364 | ` | 0365 | a |
| 0366 | b | 0367 | c | 0368 | d | 0369 | e | 0370 | f |
| 0371 | g | 0372 | h | 0373 | i | 0374 | j | 0375 | k |
| 0376 | l | 0377 | m | 0378 | n | 0379 | o | 0380 | p |
| 0381 | q | 0382 | r | 0383 | s | 0384 | t | 0385 | u |
| 0386 | v | 0387 | w | 0388 | x | 0389 | y | 0390 | z |
| 0391 | { | 0392 | | 0393 | } | 0394 | ~ | | |

是否还要查找 (Y/N)

?Y

输入区号: 1-67 43

第 43 区

| | | | | | | | | | |
|------|---|------|---|------|---|------|---|------|---|
| 4301 | 恕 | 4302 | 刷 | 4303 | 耍 | 4304 | 拌 | 4305 | 袁 |
| 4306 | 甩 | 4307 | 帅 | 4308 | 栓 | 4309 | 拴 | 4310 | 霜 |
| 4311 | 双 | 4312 | 爽 | 4313 | 谁 | 4314 | 水 | 4315 | 睡 |
| 4316 | 税 | 4317 | 吮 | 4318 | 瞬 | 4319 | 顺 | 4320 | 舜 |
| 4321 | 说 | 4322 | 硕 | 4323 | 朔 | 4324 | 烁 | 4325 | 斯 |
| 4326 | 撕 | 4327 | 嘶 | 4328 | 思 | 4329 | 私 | 4330 | 司 |
| 4331 | 丝 | 4332 | 死 | 4333 | 肆 | 4334 | 寺 | 4335 | 嗣 |
| 4336 | 四 | 4337 | 伺 | 4338 | 似 | 4339 | 伺 | 4340 | 巳 |
| 4341 | 松 | 4342 | 耸 | 4343 | 怱 | 4344 | 颂 | 4345 | 送 |

| | | | | |
|--------|--------|--------|--------|--------|
| 4346 宋 | 4347 讼 | 4348 通 | 4349 搜 | 4350 艘 |
| 4351 撒 | 4352 嗽 | 4353 苏 | 4354 酥 | 4355 俗 |
| 4356 素 | 4357 速 | 4358 栗 | 4359 僦 | 4360 塑 |
| 4361 溯 | 4362 宿 | 4363 诉 | 4364 肃 | 4365 酸 |
| 4366 蒜 | 4367 算 | 4368 虽 | 4369 隋 | 4370 随 |
| 4371 绥 | 4372 髓 | 4373 碎 | 4374 岁 | 4375 穗 |
| 4376 遂 | 4377 隧 | 4378 崇 | 4379 孙 | 4380 损 |
| 4381 笋 | 4382 簣 | 4383 梭 | 4384 唆 | 4385 缩 |
| 4386 琐 | 4387 索 | 4388 锁 | 4389 所 | 4390 塌 |
| 4391 他 | 4392 它 | 4393 她 | 4394 塔 | |

是否还要查找 (Y/N)

?N

7. 图书资料管理程序

PRO-126和PRO-127是小型图书、资料管理程序。虽然仅以三本书检索为例，但原则上在DATA区中可以放几百本、几千本书，这是因为在125句安排了防止出错语句，实用时可根据需要任意加添新书，十分方便。

本程序操作方便，只要输入1，2，3等数字，即可转入相应的功能模块，实现书目、逐册、单册的检索，而退出检索，只要输入数字“4”即可。而这一切均因为安排了总控（见370句）的原因。

本程序使用灵活，不需要顺序检索，这可以通过按空格键转换，而要退出某一功能的检索，只要按“Z”键即可。

每本书共分7个记录，它们是书号、书名、作者、定价、出版社、出版时间、购书年月、分别用A\$，B\$，C\$，D\$，E\$，F\$，G\$存贮。下面给出两个完整的程序清

单, 程序 PRO-126 在屏幕上显示, 程序 PRO-127 在打印机上输出。

PRO-126

```
2 REM PRO-126
5 PR# 3: PRINT : PRINT CHR$(18): HOME
10 REM 图书资料管理
15 DIM A$(100), B$(100), C$(100), D$(100), E$(100), F$(100), G$(100)
20 HOME
30 VTAB 3: HTAB 9: PRINT "中华学习机实用软件": HTAB 7: PRINT "-"
40 VTAB 6: HTAB 10: PRINT "图书资料管理程序"
80 FOR I = 1 TO 2000: NEXT
90 GOSUB 300
100 REM 书名检索
110 HOME : PRINT "编 号"; TAB( 10); "书 名"
120 READ A$, B$, C$, D$, E$, F$, G$
125 ONERR GOTO 160
130 PRINT TAB( 2); A$; TAB( 6); B$
150 GOTO 120
160 FOR I = 1 TO 4000: NEXT : PRINT CHR$(7): RESTORE : GOTO 300
200 REM 全面检索
205 READ A$, B$, C$, D$, E$, F$, G$
208 GOSUB 1000
250 VTAB 10: HTAB 5: PRINT "*按空格键转换 按Z键退出*"
260 IF G$ = "" THEN CHR$(19): RESTORE : GOTO 300
270 GET Z$: IF Z$ = " " THEN 205
280 IF Z$ = "Z" THEN RESTORE : PRINT CHR$(7): GOTO 300
290 IF Z$ < > " " THEN 270
300 REM 功能选择
310 HOME : VTAB 2: HTAB 10: PRINT "图书资料管理程序"
320 VTAB 4: HTAB 10: PRINT "1 书目检索"
330 VTAB 5: HTAB 10: PRINT "2 逐册检索"
340 VTAB 6: HTAB 10: PRINT "3 单册检索"
```

```

350 VTAB 7: HTAB 10: PRINT "4 退出检索"
360 GET X$:N = ASC (X$) - 48: IF N < 1 OR N > 4
THEN 360
365 REM 总控
370 ON N GOTO 100,200,400,500
400 REM 单册检索
405 HOME : VTAB 3: HTAB 5: PRINT "单册检索"
410 VTAB 5: HTAB 5: INPUT "检索第几本书?";A
420 FOR I = 1 TO A: READ A$,B$,C$,D$,E$,F$,G$: NE
XT
425 BDSUB 1000
480 VTAB 9: HTAB 5: PRINT "按空格再查,按Z键退出"
490 GET S$: IF S$ = " " THEN RESTORE : GOTO 400
495 IF S$ = "Z" THEN RESTORE : PRINT CHR$ (7):
GOTO 300
500 HOME : END
610 DATA 1,中华学习机编程技巧,朱国江,1.25,气象出
版社,1988.2,1988.3
620 DATA 2,CEC-I技术参考手册(硬件),联合设计组,2.5
0,清华大学,1987.10,1988.5
630 DATA 3,CEC-I技术参考手册(软件),联合设计组,3.5
0,清华大学,1987.10,1988.5
1000 REM 子程序
1010 HOME : VTAB 1: HTAB 11: PRINT "编 号:";A$
1020 VTAB 2: PRINT "书 名:";B$
1030 VTAB 3: PRINT "著译者:";C$
1040 VTAB 4: PRINT "出版社:";E$
1050 VTAB 5: PRINT "出版时间:";F$
1060 VTAB 6: PRINT "购书时间:";G$; TAB( 22);"书
价:¥";D$
1070 FOR I = 1 TO 2000: NEXT
1080 RETURN

```

PRO-127

```

5 REM PRO-127
10 REM 图书资料管理
15 DIM A$(100),B$(100),C$(100),D$(100),E$(100),F$
(100),G$(100)
20 PRINT
30 PRINT "中华学习机实用软件": PRINT "- - - - -"
- - - - -

```

```

40 PRINT "图书资料管理程序"
90 GOSUB 300
100 REM 书名检索
110 PRINT "编 号"; TAB( 10); "书      名"
120 READ A$, B$, C$, D$, E$, F$, G$
125 ONERR GOTO 160
130 PRINT TAB( 2); A$; TAB( 6); B$
150 GOTO 120
160 PRINT CHR$ (7); RESTORE : GOTO 300
200 REM 全面检索
205 READ A$, B$, C$, D$, E$, F$, G$
208 GOSUB 1000
250 PRINT "*按空格键转换 按Z键退出*"
260 IF G$ = "" THEN CHR$ (19); RESTORE : GOTO 300
270 GET Z$: IF Z$ = " " THEN 205
280 IF Z$ = "Z" THEN RESTORE : PRINT CHR$ (7);
GOTO 300
290 IF Z$ < > " " THEN 270
300 REM 功能选择
310 PRINT "图书资料管理程序"
320 PRINT "1 书目检索"
330 PRINT "2 逐册检索"
340 PRINT "3 单册检索"
350 PRINT "4 退出检索"
360 GET X$: N = ASC (X$) - 48: IF N < 1 OR N > 4
THEN 360
365 REM 总控
370 ON N GOTO 100, 200, 400, 500
400 REM 单册检索
405 PRINT "单册检索"
410 INPUT "检索第几本书?"; A
420 FOR I = 1 TO A: READ A$, B$, C$, D$, E$, F$, G$: NE
XT
425 GOSUB 1000
480 PRINT "按空格再查, 按Z键退出"
490 GET S$: IF S$ = " " THEN RESTORE : GOTO 400
495 IF S$ = "Z" THEN RESTORE : PRINT CHR$ (7);
GOTO 300
500 HOME : END
610 DATA 1, 中华学习机编程技巧, 朱国江, 1.25, 气象出
版社, 1988.2, 1988.3

```

```

620 DATA 2,CEC-I技术参考手册(硬件),联合设计组,2.5
0,清华大学,1987.10,1988.5
630 DATA 3,CEC-I技术参考手册(软件),联合设计组,3.5
0,清华大学,1987.10,1938.5
1000 REM 子程序
1010 PRINT "编 号:";A$
1020 PRINT "书 名:";B$
1030 PRINT "著译者:";C$
1040 PRINT "出版社:";E$
1050 PRINT "出版时间:";F$
1060 PRINT "购书时间:";G$; TAB( 22);"书 价:¥";D$
1080 RETURN

```

下面是执行程序 PRO-127的部份结果。

程序运行后, 出现一个菜单, 键入 1, 打印出全部库存书目。

JRUN

中华学习机实用软件

图书资料管理程序
图书资料管理程序

1 书目检索
2 逐册检索
3 单册检索
4 退出检索

| 编 号 | 书 名 |
|-----|-----------------|
| 1 | 中华学习机编程技巧 |
| 2 | CEC-I技术参考手册(硬件) |
| 3 | CEC-I技术参考手册(软件) |

上述工作完成后, 又出现菜单, 键入 2, 并不断按空格键, 计算机顺序打印每本书的全部信息, 而按“Z”键又返回总控。

图书资料管理程序

1 书目检索
2 逐册检索
3 单册检索
4 退出检索

编 号:1

书 名:中华学习机编程技巧

著译者:朱国江

出版社:气象出版社

出版时间:1988.2
 购书时间:1988.3 书价:¥1.25
 按空格键转换 按Z键退出
 编号:2
 书名:CEC-I技术参考手册(硬件)
 著译者:联合设计组
 出版社:清华大学
 出版时间:1987.10
 购书时间:1988.5 书价:¥2.50
 按空格键转换 按Z键退出
 编号:3
 书名:CEC-I技术参考手册(软件)
 著译者:联合设计组
 出版社:清华大学
 出版时间:1987.10
 购书时间:1988.5 书价:¥3.50
 按空格键转换 按Z键退出

若要单册检索每本书的情况,可键入数字“3”,此时出现“检索第几本书?”的提示,可键入数字(实际上是书号,本程序为说明原理,书号仅以1,2,3表示),即可查到您需要的书,见图书资料管理程序PRO-127。

图书资料管理程序
 1 书目检索
 2 逐册检索
 3 单册检索
 4 退出检索
 单册检索
 检索第几本书?2
 编号:2
 书名:CEC-I技术参考手册(硬件)
 著译者:联合设计组
 出版社:清华大学
 出版时间:1987.10
 购书时间:1988.5 书价:¥2.50
 按空格再查,按Z键退出
 单册检索
 检索第几本书?1
 编号:1
 书名:中华学习机编程技巧
 著译者:朱国江
 出版社:气象出版社
 出版时间:1988.2
 购书时间:1988.3 书价:¥1.25
 按空格再查,按Z键退出
 单册检索

检索第几本书?3
编 号:3
书 名:CEC-I技术参考手册(软件)
著译者:联合设计组
出版社:清华大学
出版时间:1987.10
购书时间:1988.5 书 价:¥3.50
按空格再查,按Z键退出

8. 计算机辅助教学

下面是一个计算机辅助教学的实例,进行四则运算,计算机自动出题并评分,每次出题是随机的,任何一种计算可以按需要灵活进行一次或多次。

清单见PRO-128:

PRO-128

```
5 REM PRO-128
20 INPUT "请输入您的姓名: ";N$
27 PRINT TAB(10);"目      录"
29 PRINT
30 PRINT "  1 ----- 加法"
32 PRINT "  2 ----- 减法"
34 PRINT "  3 ----- 乘法"
36 PRINT "  4 ----- 除法"
38 PRINT "  5 ----- 结束"
39 PRINT : INPUT "      请您选择 [1-5] ";K
60 ON K GOSUB 200,400,600,800,65
65 GOSUB 1010
70 PRINT "姓名: ";N$
74 ZF = S(1) + S(2) + S(3) + S(4)
75 PRINT "总分: ";ZF
80 PRINT : INPUT "再来一次吗? ";K$
100 IF K$ = "Y" THEN 27
110 END
200 REM JIA FA
210 FOR I = 1 TO 5
220 GOSUB 1010
225 PRINT "第 [";I;"]题"
```

```

230 PRINT : PRINT : PRINT TAB( 4);A;"+";B;"=";
235 INPUT C
250 IF C < > A + B THEN PRINT "对不起, 您做错了": FOR K = 1 TO 100: NEXT K: GOTO 280
260 PRINT "很好! 对了, 加5分": FOR K = 1 TO 100: NEXT K
270 S(1) = S(1) + 5
280 NEXT I
290 RETURN
400 REM JIAN FA
410 FOR I = 1 TO 5
420 GOSUB 1010
425 PRINT "第[";I;"]题"
430 PRINT : PRINT A;"-";B;"=";
440 INPUT C
450 IF C < > A - B THEN PRINT "对不起, 您做错了": FOR K = 1 TO 200: NEXT K: GOTO 480
460 PRINT "对了! 很好, 加5分"
465 FOR K = 1 TO 200: NEXT K
470 S(2) = S(2) + 5
480 NEXT I
490 RETURN
600 REM CHENG FA
610 FOR I = 1 TO 5
620 GOSUB 1010
625 PRINT : PRINT "第 [";I;"] 题"
630 PRINT : PRINT A;"×";B;"=";
640 INPUT C
650 IF C = A * B THEN PRINT "对了! 很好, 加5分": FOR K = 1 TO 200: NEXT K: S(3) = S(3) + 5: GOTO 670
660 PRINT "对不起, 您做错了"
665 FOR K = 1 TO 200: NEXT K
670 NEXT I
680 RETURN
800 REM CHU FA
810 FOR I = 1 TO 5
820 GOSUB 1010
825 PRINT "第 [";I;"] 题"
830 PRINT : PRINT A;" / ";B;"=";
840 INPUT C
850 IF C = A / B THEN PRINT "对了! 很好, 加5分":

```

```

FOR K = 1 TO 100: NEXT K: S(4) = S(4) + 5: GOTO 870
860 PRINT " 遗憾! 您错了"
865 FOR K = 1 TO 100: NEXT K
870 NEXT I
880 RETURN
1010 A = INT (100 * RND (1)): B = INT (100 * RND (1)) + 1
1015 IF INT (A / B) < > A / B THEN 1010
1040 RETURN
1050 GOTO 70

```

程序PRO-128运行实例:

(1) 做加法

IRUN

请输入您的姓名: 苏 苏
目 录

| | | |
|---|-------|----|
| 1 | ----- | 加法 |
| 2 | ----- | 减法 |
| 3 | ----- | 乘法 |
| 4 | ----- | 除法 |
| 5 | ----- | 结束 |

请您选择 [1-5] 1

第 [1] 题

21+3=?24

很好! 对了, 加5分

第 [2] 题

0+6=?6

很好! 对了, 加5分

第 [3] 题

0+12=?12

很好! 对了, 加5分

第 [4] 题

$0+98=?98$
很好! 对了, 加5分
第 [5] 题

$77+11=?87$
对不起, 您做错了
姓名: 苏 苏
总分: 20

再来一次吗? Y

(2) 做减法

目 录

| | | |
|---|-------|----|
| 1 | ----- | 加法 |
| 2 | ----- | 减法 |
| 3 | ----- | 乘法 |
| 4 | ----- | 除法 |
| 5 | ----- | 结束 |

请您选择 [1-5] 2

第[1]题

$90-45=?45$
对了! 很好, 加5分
第[2]题

$80-4=?76$
对了! 很好, 加5分
第[3]题

$72-18=?68$
对不起, 您做错了
第[4]题

$0-37=?-37$
对了! 很好, 加5分
第[5]题

$17-17=?0$

对了！很好，加5分
姓名： 苏 苏
总分： 40

再来一次吗？ Y

(3) 做乘法

目 录

| | | |
|---|-------|----|
| 1 | ----- | 加法 |
| 2 | ----- | 减法 |
| 3 | ----- | 乘法 |
| 4 | ----- | 除法 |
| 5 | ----- | 结束 |

请您选择 [1-5] 3

第 [1] 题

$$28 \times 2 = ? 56$$

对了！很好，加5分

第 [2] 题

$$50 \times 5 = ? 259$$

对不起，您做错了

第 [3] 题

$$60 \times 5 = ? 300$$

对了！很好，加5分

第 [4] 题

$$36 \times 12 = ? 372$$

对不起，您做错了

第 [5] 题

$$35 \times 35 = ? 1345$$

对不起，您做错了

姓名：苏 苏
总分：50

再来一次吗？ Y

(4) 做除法

目 录

| | | |
|---|-------|----|
| 1 | ----- | 加法 |
| 2 | ----- | 减法 |
| 3 | ----- | 乘法 |
| 4 | ----- | 除法 |
| 5 | ----- | 结束 |

请您选择 [1-5] 4

第 [1] 题

90/2=?45
对了！很好，加5分

第 [2] 题

28/2=?14
对了！很好，加5分

第 [3] 题

51/51=?1
对了！很好，加5分

第 [4] 题

48/12=?4
对了！很好，加5分

第 [5] 题

68/17=?4
对了！很好，加5分

姓名：苏 苏

总分：75

再来一次吗？ N

[G e n e r a l I n f o r m a t i o n]

书名 = 中华学习机数据处理

作者 = B E X P

页数 = 3 4 5

下载位置 = h t t p : / / w w w . m y e l i b . c o m / d i s k e q 0 1 / e q
7 8 / 1 3 / ! 0 0 0 0 1 . p d g