第三章　BASIC算法语言

BASIC语言是目前国际通用的计算机算法语言，大多数中小型机上，都配有这种语言。

BASIC语句是"BEGINERS ALL-PURPOSE SYMBOLIC INSTRUCTION CODE"（初学者通用符号指令代码）一词的缩写。目前国内外的BASIC系统大体上分为三类：单用户BASIC，扩展的BASIC和多用户分时BASIC。

SHARP PC—1500机配置有单用户基本BASIC，并配有一部分扩展BASIC。

第一节　BASIC语言在PC—1500机上的使用简介

一、BASIC语言特点：

1.BASIC语言比较简单，它所使用的命令和语句中使用的词（以英语表示）以及运算符号与人们习惯使用的词和符号相似。因此比较直观，易于理解和记忆。例如：INPUT在英文中是输入的意思，在BASIC算法语言中，就是键盘输入的命令。又如END在英文中是结束的意思，在BASIC中它就表示本段程序结束。

2.BASIC语言是一种会话式语言，它通过显示屏或打印机使人与计算机进行对话。当程序送入机器运行时，机器可检查程序中的语法或词法错误（如果有的话）。用户可立即在键盘上修改程序，边改边算，直到得到满意的结果，十分方便。

3.BASIC还可在键盘上直接进行计算和执行某些语句（手算），而不必专门编写一段程序。

如在键盘上打：PRINT（15＋SIN（3.14159））／7，则计算机显示计算结果：2.142857522。

4.BASIC语言是一种小型算法语言，由于它允许的数的范围、简单变量的数量、数组的维数、保留函数、运算速度等，都有一定限制，因此它的应用范围受到一定限制。但是生产、科研中的一些中、小型题目用BASIC还是比较方便和适宜的。

BASIC虽然是一种通用语言，但是每种计算机都有自己特殊的一些规定。本讲义讲的都是SHARP PC—1500机上BASIC语言的具体规定。

二、BASIC语言的结构和使用规则：

先看一个例题，用余弦定理求三角形的边长：

例（1）设三角形二条边长为b＝5，c＝4，夹角E＝52°，求第三边a＝？

根据余弦定理：

$$a=\sqrt{b^2+c^2-2bc\ \cos E}$$
$$=\sqrt{5^2+4^2-2\times4\times5\cos52^\circ}$$
$$=4.046$$

将此题用BASIC语言源程序写出：

```
10  LET  B = 5
20  LET  C = 4
30  LET  E = 52
40  LET  A = √‾ (B * B + C * C - 2 * B * C * COS DEG E)
50  PRINT  A
60  END
```

从上面的程序看，BASIC算法程序的结构有如下特点：

1. 一个BASIC程序是由若干行组成，一行作为一个语句，整个程序就是若干语句组成。

2. 语句结构包括：语句号、语句定义符、语句体三部分。

语句号：或称“行号”。必须是无符号的整数，其范围为1～65279。不同机器上限不一样。计算机依照行号顺序执行。行号不一定连续，以便修改程序时好打入语句。如例（1）中，每两句之间还可插入9个语句之多。

语句定义符：如例中的LET、PRINT、END，它规定该语句执行功能。BASIC最基本语句只有十几种。见表3—1。

基本BASIC语句简表 表3—1

语句定义符	功能	备注
LET	计算并赋值	一般可省略不写
INPUT	由键盘输入数据	
READ	从数据区读数	
DATA	在数据区置数	
PRINT或LPRINT	在显示屏显示数、字符串或在打印机打印数字符串	PC—1500机显示与打印输出是分开的命令，有些机器的命令都是PRINT
GOTO	无条件转移	
IF THEN	条件转移	
FOR TO STEP	设置并进行循环	
NEXT	循环返回及出口	
END	程序终了	
GOSUB	转子程序	
RETURN	由子程序返回主程序	
RESTORE	恢复数据区数据	

续表 3 — 1

语句定义符	功　　能	备　　注
REM	注　　释	
DIM	数组说明	
STOP	使程序暂停	

语句体：跟在语句定义符后的具体内容，如上例中：

A = $\sqrt{}$ (B * B + C * C - 2 * B * C * COS DEG E)

语句体中的内容中，若有算术表达式，一定要写成BASIC表达式

3.程序之后应以END结束该程序语句。

4.程序输入后，打入RUN指令，机器执行程序，运行完毕后，程序并不从机器中消失，再送入RUN命令，机器又执行一遍。

第二节　PC—1500机程序编写与运行的一般规定

1.编写和输入程序时，必须使机器在“程序方式”（PRO）下进行。

2.在编写之前，先按[↑]键，看机器内是否有程序，若有，则需先清除机内程序。（将清除指令NEW输入机器即可清除）。

3.语句内使用的字符应是键盘上有的字符，因此在编源程序时也应该使用这样的符号，以免发生差错，如“×”号必须写成“*”，“÷”号必须写成“/”号，乘方号必须写成“∧”号等等。

4. 作为变量的标识符只能用大写英文字母，小写字母只能用在字符串中，并必须用“ ”号括起来。（在PC—1500机上，引号不分前后，均用″号。）

5. 一个语句必须在一行内写完。PC—1500机一行最多可容纳80个字符。如果一行写不完，可将语句拆为两句。

如：10 LET A = 3.14159 + 2 * 1.23456/3 * SIN (X + Y)

可改写为：

```
10 LET B = 2 * 1.23456/3 * SIN (X + Y)
20 LET A = 3.14159 + B
```

此处第10句与20句等价于上面的第10句。

6.一个语句内，可以不一定只写一种语句定义符，只要它包含的内容在程序运行中没有输入问题就可，但是必须注意，两种定义符之间，一定要用“：”——冒号分开。而且总的字符数应不超过80个。这种在一条语句中有两个以上语句定义符组成的语句，我们叫它复合语句。

例如：
```
70 PRINT″A =″; A
80 END
```

可以合并为复合语句：

70　PRINT"A =";A : END

7.每个语句写完，须按ENTER键（执行键）才能将语句送入机器，否则只在缓冲器中。当按执行键后，在显示屏上该语句号后将出现“：”号。所以，这也是我们了解是否该语句输入了内存的标志。

8.标点符号的用法非常重要，用错了标点符号机器拒绝执行或者造成误操作。

PC—1500机的标点符号有：冒号“：”，分号“；”，逗号“，”，引号“″”（感叹号“！”、问号“？”不在程序之中用）

标点符号的用法，在以后学习中，可逐渐熟悉，在这里，可先记住如下要领：

冒号前后事无关，　　逗号前后一类事，

分号前后紧相连；　　引号专为字符串。

9. 若调回原程序，可在PRO方式下，按变行键即可，按 ↓ 递增，↑ 递减。如果按住变行键不放，语句将依递增或递减的顺序自动跳动，其跳动速度大约每秒10个语句。释放变行键则停止跳动。

10. 编写程序时，语句号的次序可不考虑，因为机器有自动顺序编辑功能。

11. 程序的修改：

（1）修改语句内容时，先用变行键调回要修改的语句，然后按光标键找到需修改的字符，即可进行修改，插入或删除。

（2）删除整个语句时，只需打出该语句号，再按执行键即可。

（3）增添语句时，其语句号一定要插入在原来两个语句号之间，然后写入语句内容，按执行键即可自动编入程序之内。

12. 程序步的计算

所谓程序步，就是在编写程序输入机器时操的作步。每一操作步，（如按字符键或执行键或其他一些键），都是要占用机器内存的。我们把占用内存字节的数量，作为程序步的数量。我们可用程序步的概念来说机器的内存数或程序占用机器的内存数。如，机器还有1200字节的内存，我们就说机器还有1200程序步。又如，某个程序占用内存350字节，那么我们就说其程序步是350步。

所以，PC—1500机在不加模块的情况下，它可用的程序步为1850步。加4K模块时，其可用程序步增加至5946步。加8K模块，其程序步就为10042步。

程序步的计算是为了使我们了解程序用了多少程序步。还有多少可以利用。这样以防止在机器内存用完时产生溢出错误。

PC—1500计算机程序步的计算方法如下：

语句号——不论语句号是几位数，都占3步，（包括语句号后的冒号）。

指令名、函数、语句定义符——不论几个字符组成的，每个均占2步。

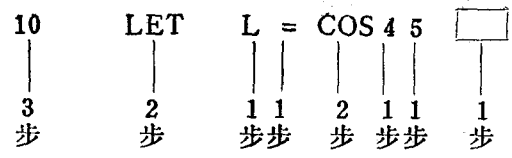
执行键——每个语句输入时按的执行键，均占1步。

SHIFT键不占程序步。

此外，每一个字符，都算1步，不管是变量名还是数字或运算符。如：

10　C = √‾ (A + B) − 45 * B

语句号10算3步，√‾ 1步，其余每个字符1步，执行键也是1步，所以共17步。又如：



共用了12步

我们不必自己去数用了多少程序步，可以用下面两个指令让计算机自动计算并显示给我们已用了多少程序步以及还有多少程序步。

指令STATUS 1 输入后，显示已用了多少程序步，如显示是47，表示已用了46步而下一步就从47步开始。

指令STATUS 0 输入后，显示机器还剩有多少程序步，如显示1783表示还有1783步，可供用户用。

另外，指令MEM也是了解剩余程序步的指令，其功能与STATUS 0 完全一样。

13. 程序的运行

在编完程序输入机器后，须将机器的使用方式改为常规运算方式，即RUN方式。如果程序中使用了自定义标号，还同时要调整到自定义方式（DEF），作好运行准备。

此后，我们可以在键盘上，打出RUN指令并按执行键输入机器，程序即进入运行。显示屏左上角显示BUSY。表示机器正在进行计算。运算结束后BUSY标志消失，其他标志恢复，机器按程序规定显示或打印计算结果。

（除RUN指令外，GOTO指令也同样可以启动运行程序。）

在有些情况下，我们不需要从头开始运行整个程序，而只要从中间某个地方开始运行。那么我们可以在RUN指令后加上指定开始的语句的语句号，再送入机器，则机器就从我们指定的语句开始运行，如RUN120指令送入机器就从程序的第120句开始执行。

14. 程序运行中错误的显示

PC—1500机使用的是解释BASIC语言，它只能一边运行一边检查程序中是否有这样或那样的错误。如果程序中或计算过程中出现了机器不允许的错误，则计算机自动停机并在显示屏上显示错误类别及地址。显示形式如下：

[ERROR ×× IN ×××]

××——错误类别号。也就是指示的错误性质。

×××——发生错误的语句号。也就是指示的错误地址。

根据这个显示的信息，我们便可很快地改正程序中的问题或错误。

第三节 输出语句

在BASIC中，输出（PRINT）语句是必不可少的，也是用得比较多的，它的用法比较灵活，必须熟练掌握。

在PC—1500机上，输出命令有两个，一是PRINT，表示显示屏显示输出。还有一个是LPRINT，表示在打印机打印输出。它们的用法基本上相同，在程序中一般可互换，所以我们以PRINT为主来介绍输出语句，在打印机一章中，还要对LPRINT命令进一步详细介绍。

（一）PC—1500机上PRINT语句的用途，主要应用在以下三个方面：

1. 显示已经赋值的变量。可以显示一个也可同时显示多个。但必须是显示屏26个字符能容纳下才行。

下面以例题进行操作理解：

我们先给变量赋下列值，然后再让它们编入程序之中运算。

A = 2，B = 3，C = 10.78645312，D = 0.00816，E = − 1.732946815E − 23

例（1） 10 PRINT E

RUN [−1.732946815E − 23]

不用标点，在显示屏右端显示，最多可以显示16个字符，这是PC—1500的标准显示。

例（2） 10 PRINT = A，E

RUN [2 −1.732946E − 23]

可以显示两个数，各占显示器的一半。每个数只能显示13个字符。如果多于13个，则自动减少有效数字。这也是常用显示法，叫“分段格式显示”。

例（3） 10 PRINT A，B，C

RUN [ERROR 1 IN 10]

用逗号分开的三个数要同时显示，机器不接受，显示错误信息。

例（4） 10 PRINT A；B；C；D

RUN [2 3 10.78645312 0.00816]

用分号分开的数，则从左端开始显示，显示数不受限制，每两个数之间留一符号位，但多于显示屏部分不给显示，此叫“紧凑显示格式”。

例（5） 10 PRINT A：B

RUN [2]再按 [ERROR 1 IN 10]

所需输出显示的变量之间用冒号分开了，机器执行时只显示第一个变量A的值，而认为冒号后是另一语句。再继续按执行键后显示错误信息，因为冒号后无语句定义符。

例（6） 10 PPINT A：PRINT B：PRINT C

RUN后只显示[2]，再按执行键显示[3]，再按执行键显示[10.78645312]需显示的数用冒号“：”分开，但必须在每个需显示数前加写PRINT。这种方法显示的数不受限制，每按ENTER键显示一个数，直到最后一个数为止。

以上几个例子所反映的显示情况，都是与标点有关的。

2. 显示表达式和计算出的数值：

下面仍以例题运行来说明。在下面几例中，A，B，C，D的值同前面，另外给F与G赋值为：

F = 38 * 14， G = A * A + 4 * A * B

例（7） 10 PRINT A * A + 4 * A * B

RUN [28]

例（8） 10 PRINT 38 * 14 =

RUN [ERROR 1 IN 10]

例（9） 10 PRINT″38 * 14 =″

RUN [38 * 14 =]

例 (10) 10 PRINT"38 * 14 =", 38 * 14

RUN [38 * 14 = 532]

例 (11) 10 PRINT"38 * 14 ="; 38 * 14

RUN [38 * 14 = 532]

例 (12) 10 PRINT"38 * 14 ="; F

RUN [38 * 14 = 532]

例 (13) 10 PRINT"38 * 14 ="; F, "A * A + 4 * A * B ="; G

RUN [ERROR 1 IN 10]

例 (14) 10 PRINT"38 * 14 ="; F; "A * A + 4 * A * B ="; G

RUN [38 * 14 = 532 A * A + 4 * A * B = 28]

请注意以上几种类型的显示。例 (7) 与例 (8) 都是显示表达式的内容，例 (7) 执行时，机器先计算表达式的值，然后输出。例(8)因为错用了一个等号，所以显示错误信息。

例 (9) 中将表达式加上引号，执行程序时只显示表达式，而不显示计算结果。

例 (10) 与例 (11) 既显示表达式，又显示计算结果，只是因为用的标点不同，一个是分段格式，一个是紧凑格式。

例 (12) 的显示中，引号内的照显，而后面的532是F 存贮器中的数，并不是机器 在执行程序中计算出来的数。

例 (13) 是试图同时显示两组表达式和两组结果，因为中间用逗号联接，机器不接受，所以显示错误信息。应该编成例 (14) 那样。

3. 显示字符串：

PC—1500机的PRINT语句，还可以显示字符串。引号" "为字符串专用符 号。字符串也可以存入存储器，这时叫它字符串变量，字符串变量的标识符形式是在数值变量名后加"$"号。

下面继续通过例题的操作来理解。

A、B、C、D、E、F、G的值仍同前，另给字符串变量赋值：

H$ ="SHARP P", I$ ="C-1500"

例 (15) 10 PRINT"Good morning"

RUN [Good morning]

例 (16) 20 PRINT H$, I$

RUN [SHARP P C-1500]

例 (17) 30 PRINT"WE SAY"THIS IS A STRING""

RUN [ERROR 1 IN 30]

例 (18) 40 PRINT"ABC"; H$; C

RUN [ABCSHARP P 10.78645312]

PRINT后如不带任何语句，如：10 PRINT，则机器作为空输出，显示屏上不显示任何东西。如此时按ON键，则机器显示：

[BREAK IN 〈语句号〉]

表示机器停机在第某句。

（二）PAUSE语句：

PAUSE的功能和用法与PRINT一样，它也是显示输出语句，其用法与PRINT完全一样。只是PAUSE是短暂显示，对要显示的数、变量、表达式等，约每0.85秒自动跳显一个。显示完后自动结束或者执行下面的语句。

（三）PRINT语句的格式输出

在PRINT语句输出时，机器对被输出的数或字符，都不经格式处理。该数是小数几位的数，就输出几位的数。而有时我们并不完全需要全体小数位。例如人民币的单位最小到分，而分以下的数，我们就不需要了，对于任何具体的数，我们往往只需要满足所要求的精度的那部分，而其余的可以舍去不计。

为此，BASIC提供有格式输出语句。

1． 格式输出语句的形式：

〈语句号〉PRINT USING "格式串符号"；〈变量或表达式

" "号中的格式串符号有：

#——数字格式符号　　，——三位分段符

·——小数点　　*——星号（用乘号代），空位补充符

&——字符格式符号　　+——正号，给正数加用的

∧——指数格式符号

2． 格式输出的用法：

"格式串符号"的内容决定输出〈变量或表达式〉的数值格式；输出数时用"#…·#…"的形式；小数点前#的个数表示整数位数及符号位，小数点后的个数表示截留小数位数，如"# # # . # # # #"表示整数二位，正负号一位，小数四位的输出。

例：程序：10 A = -134.5798

20 PRINT USING"#……"; A

30 END

下表3—2中，左边列出此程序中的13种格式串的用法，右边是相应的显示结果。

用作十进制格式输出时，小数点以前的位数必须和现有数字的位数相符，另外加一个符号位（+、-号），所以"#"的数目至少比小数点以前的位数要多一位，如表3—2中(4)，"#"的数目多一位如（5）。但不能少或相等，如（1）、（2）、（3）、（9）。用错了则出现错误信息。。小数点以后的位数可以任意选取，如（6）、（7）、（8）。

用作指数格式输出时，"#"的位数只考虑小数点以后要保留的位数，其他都可以不考虑，在#的最后加"∧"符，一个也行两个也可，都能正确显示，如(10)、(11)、(12)和(13)。

在同一程序中凡在用过USING语句以后的输出中，都将按USING规定的格式显示。因此应注意对各种不同数的输出格式的选取，也可以对不同格式要求重新给出USING格式，如：

20 PRINT USING "# # # . # #"; A

30 PRINT USING "# # · # # #"; B

解除USING格式：解除的方法是，写一个不写具体格式要求的USING语句。如：

40 PRINT USING; C

表 3 — 2

格 式 串 的 用 法	运 行 后 显 示 情 况
(1) "#"	〔ERROR 36 IN 20〕
(2) "# #"	
(3) "# # #"	
(4) "# # # #"	〔 -134〕
(5) "# # # # #"	〔 -134〕
(6) "# # # #. # #"	〔 -134.57〕
(7) "# # # # #. # # # #"	〔 -134.5798〕
(8) "# # # # #. # # # # #"	〔 -134.57980〕
(9) "# #. # # # #"	〔ERROR 36 IN 20〕
(10) "# # # #. # #∧"	〔 -1.34E02〕
(11) "# # # #. # # # #∧"	〔 -1.3457E02〕
(12) "# #. # # #∧∧"	〔 -1.345E02〕
(13) ". # # #∧"	〔 -1.345E02

对于一些数，必须将其正号显示出来时，可以在 USING 的格式串"#….#…"中的#前加一个“+”号，使成为"+#…#.#…"。那么显示出的数若是正数，则在其前面就加上了一个“+”号。

格式串符号的“#”号也可用“*”代替，其功能不变。所不同的是，对于整数位来说，如果格式串符号个数多于应有的位数时，#的位数由空格来补充，而用*号时所多的位数，将有*号来补充。若在格式串中还用了“+”号，那么用#号时，显示的数正负号紧接在数的前面；而用*时，多余位的*号则插在正负号与数之间，如下例：

```
        5  A = -34.5678
       10  USING"+ # # # # # #.# #"
       15  PRINT  A
       20  USING"+ * * * * * *.* *"
       25  PRITN  A
RUN    〔                -34.567〕
       〔              - * *34.567〕
```

三位数一段的显示法：对于数值的整数部分如果需要按三位一段来显示，如13579.36欲希望显示成13,579.36的形式 则可用“,”号来分段，用法如下：

```
10  PRINT  USING"#, ###, ###, ##"; 13579.36
RUN    [          13,579.36]
```

要注意，分隔的数字只能三位一段，其格式符的段数要比该数可分成的段数多一个段，否则将出现ERROR 36，如：

```
10  PRINT  USING"###, ###.##"; 13579.36
RUN    [ERROR  36  IN  10]
```

格式串的整数位只分成了二个三位段，所以出现36类错误。

USING格式一旦被给出，对其后所有输出语句均有效，直到给出新的格式规定或撤消格式规定为止。因此，对有不同输出要求的数，应分别给出不同的USING格式规定。如：

```
10  A=136.78, B=5.634
20  PRINT  USING"#####.#"; A
30  PRINT  USING"##.#"; B
```

USING 格式规定的撤消方法：在前面如已有了格式规定，而以后的输出想取消这个规定，则可用一个不带格式串的USING语句来实现。如下例中，对X有规定，而对Y则要取消，可用下面的30语句来实现：

```
10  X=34.567, Y125.6873
20  PRINT  USING"###.##"; X
30  USING
40  PRINT  Y
```

如果没有30语句撤消掉格式，当程序运行到第40语句时，将出现36类错误。

对字符串变量的字符输出，也可以用USING格式串，用法较简单，如：

```
50  PRINT  USING"&& &"; A$
```

50语句表示对字符串变量A$，输出显示其前三个字符。其中&符号是字符格式号。

3. 对LPRINT命令，也同样可以用USING格式，USING的用法与PRINT时的用法稍有差异，在打印输出时须注意，在第五章中将结合LPRINT语句介绍。

第四节 输入语句

输入语句是指数据输入机器的语句，它使机器按所编定的程序进行运算，因为计算机只能进行数值的运算，如果不对程序中的变量代入具体数字，机器是无法进行运算的。因此在BASIC语言中输入语句是必不可少的步骤。

在PC—1500机上，输入数据的方法有下列几种：一、程序内输入；二、问答式输入；三、循环输入；四、自动读入；五、READ/DATA语句输入。

（一）程序内输入——LET语句（赋值语句）

程序内输入用赋值语句LET。LET语句是给变量提供数据的最简单的形式。看下面的例子：

例（1）求二次方程的一个根：

$$X=\frac{-B+\sqrt{B^2-4AC}}{2A}$$

```
10  LET  A = 2
20  LET  B = 8
30  LET  C = 5
40  LET  X = (-B+SQR (B↑2-4*A*C)) / (2*A)
50  PRINT"X =";  X
60  END
    RUN [X = -7.752551287E-01          ]
```

（1）LET是赋值符号，它的意思是将等式右边的值赋到左边的变量中去。如A = 2，表示将2赋给A

LET语句中等号的意义与数学上不同。在数学上X = 5 × 3表示等式两边完全相等，可以写成：5 × 3 = X。而在BASIC中，LET X = 5 * 3不是表示X等于5 × 3，而是表示 将5乘3的值送到X之中去，所以等式两边不能互换。

（2）在PC—1500上，LET一般可以省略不写，例（1）可直接写成如下形式：

```
10  A = 2
20  B = 8
30  C = 5
40  X = (-B+SQR (B↑2-4*A*C)) / (2*A)
50  PRINT"X =";  X
60  END
```

（3）可以辗转赋值：即先将一个数值赋给一个变量，再将第一个变量的值赋给第二个变量，……一直继续下去。如果把A⇨X⇨Y⇨Z，等于把A⇨Z。

如：

```
10  LET  X = 10
20  LET  Y = X
30  LET  Z = Y
40  PRINT Z
50  END
    RUN [        10]
```

（4）在RC—1500中不允许下列形式。（不允许几个等号连等）。

```
10  A = B = C = 5
```

也不允许向一个表达式赋值，如：

```
10  X + Y = 5 + 7
20  PRINT  X + Y
    RUN  [ERROR   1 IN  10]
```

（5）在PC—1500中，下列形式是可以的。

```
10  LET  A = 3 , B = 6 , C = B + 5
```

（6）当用LET语句对变量赋值后，不管变量中原来有否数，也不管是什么数，即被

所赋的数所替代，（而不是叠加）。

（二）键盘输入语句（INPUT语句）　　问答式输入

问答式输入用INPUT语句。在程序运行时，计算机执行到INPUT语句时，就停下来在显示屏上显示一个“？”也就是机器发出询问，此时可由键盘上输入一个数值入程序，该数送到存贮器中该变量的单元里，然后继续执行程序。见下例：

```
10  INPUT  A
20  PRINT"A * A ="; A * A
30  END
    RUN     ?, 12, [A * A = 144          ]
```

当程序运行时显示屏显示“？”，在键盘上输入12，则机器显示计算结果A * A = 144。

如果还要输入第二个数直至N个数，则机器在你送入第一个数以后还继续发出询问“？”，直至第N个数全部输入后，再进行程序运算。见例（2）

例（2）求并联电阻的总电阻值。

我们已知并联电阻值公式为：

$$\frac{1}{R}=\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}+\frac{1}{R_4}+\cdots\cdots+\frac{1}{R_n}$$

$$R=\frac{1}{\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}+\frac{1}{R_4}+\cdots\cdots+\frac{1}{R_n}}$$

求4个电阻（n = 4）并联后的总阻值。

设　$R_1=100$，$R_2=150$，$R_3=240$，$R_4=360$

求R的程序如下：

```
10  INPUT  R1, R2, R3, R4
20  R = 1 / (1/R1 + 1/R2 + 1/R3 + 1/R4)
30  PRINT  USING"# # #.# #"; R
40  END
    RUN [   42.35]
```

计算国民经济增长的例题：

例（3）已知某省1980年国民经济总产值为252.7亿元，年增长率为7.2%，问到1985年、1990年、该省的国民经济总值分别为多少？

设：P为1980年的国民经济总产值，在此题中P = 252.7

R为增长率，在此题中R = 7.2%；N为年份数，

E为所求年份的国民经济值

程序编制如下：

```
10  INPUT"YEAR? "; N
20  P = 252.7, R = 0.072, D = N - 1980
30  E = P * (1 + R) ^D
40  PRINT  N; "␣IS"; USING"# # # # # #.#"; E
```

程序执行：

按RUN，显示YEAR？，键盘送入1985，显示计算结果

[1985 IS 357.7]

再按RUN再显示YEAR？，再送1990，显示

[1990 IS 506.4]

如此类推，可计算任一年的产值。

此题可改变输入的数，如求1990年时，在不同增长率时的产值。大家可自己修改程序试一试。

从上面例子可以看出，INPUT语句是一种很方便和直观的问答式输入方式，其优点是：

1、有很大的灵活性，可任意改变变量数值进行同一类计算。

2、可以避免或减少输入时的操作错误。

3、数字不进入程序，占用程序步较少。

但是，这种方法也有它的缺点：

1、由于逐个输入，占用时间长。

2、所有变量占用的存贮器都要在编制程序时加以指定，相当繁琐

3、不太适用于大量数据的题目。

（三）循环输入（GOTO语句在输入时的应用）

GOTO语句是无条件转移语句，在BASIC中用它来改变程序的执行顺序，当程序执行到GOTO语句时，程序就不再按行号顺序执行，而是转到执行GOTO语句指定的那个语句（该语句必须是程序中存在的，否则将显示错误标志）。

在此我们只讲GOTO语句在输入语句中的使用。

前面求并联电阻阻值的例题中，如果我们加入一句

```
35 GOTO 10
```

使程序成为：

```
10 INPUT R1, R2, R3, R4
20 R=1/(1/R1+1/2+1/R3+1/R4)
30 PRINT USING"###.##"; R
35 GOTO 10
40 END
```

当运行时，程序只需运行一次，计算机就不断地向你询问每组的R1～R4的数值，并将该组的等效R值显示出来。它可以无穷多次反复，如果不再向下计算，希望它“脱离”出来，应先按 ON 键，机器显示[BREAK IN 10]，表示在10句中止，然后按 CL 键可以脱离程序。

下面再举一例：

例（4）“加法器程序”，亦即将键盘输入的数进行累加。

```
10 T=0
20 INPUT"X=";X
30 T=T+X
```

```
40  PRINU  T
50  GOTO  20
60  END
```

设X的值从1到36，共36个自然数，求其和T。

大家可以自行计算，并观察计算机的显示情况。

（四）读数和置数语句（READ～DATA语句）

当我们给很多变量(譬如说10个变量)赋值，那么用LET语句或者INPUT语句时，必须写十句LFT语句或者用10次键盘输入，十分不方便，为此BASIC设置了READ～DATA语句。

下面我们仍用求并联电阻等效电阻的阻值的例子来说明RAED～DATA语句的用法。

```
10  READ R1，R2，R3，R4
20  DATA  100，150，240，360，420，180，260，300
30  R=1/(1/R1+1/R2+1/R3+1/R4)
40  PRINT"R="；R
50  GOTO  10
60  EMD
```

```
RUN
 R = 42.25294118
 R = 66.15508885
 ERROR  4 IN  10
```

当运行此程序每当执行到第10语句时，R1、R2、R3、R4分别从20语句DATA中“读”到数值100，150，240，360，亦即给R1～R4赋了值。30语句进行计算，40语句打印出与这组数据相对应的R值。执行50语句时转到10语句，再从DATA中读R1～R4的数值：420、180、260、300，再计算新的R值，然后又转向10语句。当第三次读时，DATA语句中已没有数了，机器显示[ERROR 4 IN 10]表示第4类错误（即DATA数据区已无数可读），于是程序不再执行。

READ语句的各变量从DATA中取数据是一个一个顺序取的。也可认为，在语句的数据表下面有一个无形的“指针”，它的开始位置在第一个数据下面。当READ语句中的第一个变量需要读数据时，就读“指针”所指的第一个数据。当读完该数据后，“指针”就立即自动向右移动到第二个数据的下面。第二个变量需要读数时，就读“指针”所指的数据（第二个数据），以此类推。

对READ～DATA语句有几点说明如下：

1、执行READ～DATA语句，计算机就从DATA语句中顺序取出数据，依次赋给READ中每一个变量，各变量间和各数据间只能以“，”分隔。（但最后一个变量和数据之后不能加“，”号。这一点，任何语句都是如此）

2、DATA中的数据个数不得少于READ中的变量个数。否则会因从DATA中得不到数据而显示[ERROR 4]。

3、READ和DATA必须联合起来使用才能起作用。DATA不一定紧跟READ语句，可以在程序中的任何位置。习惯上放在END之前的程序最后部分或者放在程序最前面。

（五）恢复数据区语句（RESTORE）

DATA中的数据，如果需反复使用，可使用恢复数据区语句。

如例：
```
 10  DATA  8，4
 15  DATA  1，3，6，8E2
100  READ  X，Y
110  READ  M，N，O，P
120  RESTORE  15
130  READ  A，B
```

程序在执行时，100语句读X，Y，分别为8，4，再执行110句读M、N、O、P，分别为1、3、6、8E2，执行到第120句时，表示以后读数又从15句开始，因此A、B读得结果分别为1、3。

如果第120语句改为：

```
120  RESTORE
```

那么在执行第130语句读A与B时，又从最前面的DATA语句取数。所以A与B的值分别读数为8与4。

RESTORE语句的使用，可以使我们重复利用数据语句中的数据（如果需要的话），而不须去写重复的数据语句。

要注意的是，RESTORE语句的作用是使DATA中的数据由RESTORE指定的位置重新读起，而不是使程序回到第一个READ语句从它的变量重新读起。

（六）自动读入（AREAD）语句

有一类题目，数据不多但需经常重复运算，还要改变一个或几个变量，例如下面例(5)计算利息一类问题，可以用自动读入语句简化输入方式，带来很多方便。

用AREAD输入语句编成输入程序如下例（5）：

例（5）　计算复利

$$F=P*(1+R)\wedge N$$

式中：　F　为本利和
　　P　本金
　　R　年利率
　　N　年数

```
10 "A"AREAD  R：R=R/100：END
20 "B"AREAD  N：END
30 "C"AREAD  P：END
40 "F"F=P*(1+R)∧N
50 F=INT (F*100+0.5) /100
60 PRINT  F：END
```

这个程序使用了用户自定义标号键，把年息定名为A，年数定名为B，存入的本金定名为C，运行这个程序必须用〈RUN DEF〉方式，然后写出数据再按相应的标号键(所谓标号键就是先按DEF键，再按相应的字母键)，数据可自动存入（读入）相应的变量单元中去。再启动运算标号，机器自动计算，显示出结果。在年息、年数与本金三个变量中，可随意变更其

中的一个，两个或三个。可以用任意的次序输入，只要在变更后按DEF键与F键，就能立即计算出结果。

运算1：求本金7000元，年息6分，存5年后本利和？

操作：7000 DEF C 5 DEF B 6 DEF A DEF F

显示〔 9367.58〕

运算2：本金与利率不变，存期10年本利和又是多少？

操作：10 DEF B DEF F

显示：〔 12535.93〕

（七）直接输入

除了用程序输入数据之外还有最简单的赋值方法就是“非程序输入”，或叫“直接输入”，直接将数或字符串存入存储器，第二章中曾采用过这种方法。此法虽然简单，但非常重要，在很多情况下甚为有用，例如在调试程序阶段，或单独修改某个数据的操作，用此法非常方便简单和明了。

习 题 二

一、要求计算机显示屏显示下列形式数字，应怎样编写输入与输出语句？

1、〔3.7418E20 4592〕

2、〔0.00000325 -7E90〕

3、〔35.678 〕

4、〔43.5 26.78 51 -27.3〕

5、〔18＊16= 288〕

6、〔18＊16=288 〕

7、〔SIN75=9.659258263E-01 〕

8、〔3175＊15%= 476.25 〕

9、〔 (91-34) /91=62.64% 〕

二、若A=6，B=8.5，希望显示下面结果应如何编写输入输出程序？

〔A＊B= 6 ＊8.5=51 〕

三、下面语句中哪些有错误？有哪些错误？

1、10 LET X=5，Y=4，Z=5，

2、10 LET X∧2=4

3、10 LET X+3=Y-2

4、10 READ A，B，C，D

5、10 READ A、B、C、D

6、10 DATA 4，-8，$1\frac{1}{3}$，A，

7、10 INPUT X；Y，Z

四、下面程序运行结果是什么？并计算程序步是多少？

1、10 READ A，B，C，D，E，N

```
20  LET  T=A+B+C+D÷E
30  LET  U=T/N
40  PRINT"MEAN"; U
50  DATA  4, 8, 6
60  DATA  15, 4, 5
70  END
```

2、
```
10  READ  X, Y
20  READ  A, B, C, D
30  DATA  4.3, 57, 8, 17, 5
40  DATA  19, 30, 40, 25
50  READ  F, E
60  RESTORE  40
70  READ  PQ
80  PRINT"PQ=", PQ
90  END
```

五、写一个把摄氏温度转化为华氏温度的程序。换算公式为C=（5/9）·（F－32）。其中C为摄氏温度，F为华氏温度。要求显示形式为显示屏左端显示摄氏度数，右端显示华氏度数。C的值是给定的并要求用INPUT语句输入。F是计算机计算结果。

第五节　流程和框图

当有了输入语句和输出语句以后，我们就可以编写一些简单的程序了，例如一个求5个数的平均值的程序。

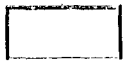
5个数分别为8.7，4.5，－30.1，12，－5.9求其平均的值。

```
10  INPUT  A, B, C, D, E
20  P=（A+B+C+D+E）/5
30  PRINT"Pcp=", P
40  END
```

从这个程序可以看出，首先输入各个数的数据，然后计算它们的均值，最后输出它们的均值。这种计算过程，叫做流程。

在编写程序以前，我们应该把计算的流程描述清楚。在实际生产中，象上面这样简单的流程是很少的，对一些复杂的计算，更应该把流程描述清楚，描述的最好办法是画出“流程框图”。图里用不同的框代表不同的计算过程，中间用带箭头的线联接起来，表示各步骤及流向。因为它们由框组成，所以习惯上叫“框图”。它是计算方案的形象描述，比较直观，因此常用它来帮助进行程序设计及介绍程序的结构。框图的常用符号如下：

椭形框：表示程序的开始和结束



矩形框：表示说明、处理、运算或加工步骤等，它有一个入口，一个出口。


菱形框：表示判断程序的走向，它有一个入口，两个出口。


输出框：表示输出、打印。

对于一个完整的程序，大体可分为三大部分：一、输入部分；二、运算（或处理）部分；三、输出部分。而构成这三大部分的各种计算流程，不外乎三类：一、直线流程；二、循环流程；三、分支流程。

一、流程及其框图

1． 直线流程：直线流程从头到尾都是直线进行的。例如，求平均数的流程框图（图3—1），流程里所有的框只有两个口，一出一入，没有第三条出路。这种流程最简单，不容易出错，也是最基本的流程，不过，实际上这么简单的流程是很少见的。

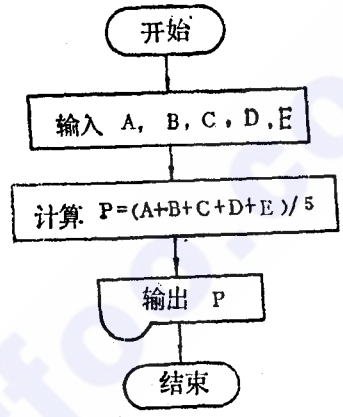


图 3—1

2． 循环流程：程序进行到一定步骤时，返回到指定的步骤又进入程序让计算重复进行。（如图3—2所示）

由于每次计算所使用的变量不同，所以结果也不同，形成“循环流程”。在整个程序里各框都只有两个口，一经循环即成为永久循环，程序里找不到出口，无法脱离循环，所以它不能单独形成一种实用的程序，必须设置循环出口方有意义。

3． 分支流程：程序进行到一定步骤时加入判断条件，如果满足判断条件，程序走指定的一条路，如图3—3中走计算2的路；不满足条件则程序走指定的另一条路，如图3—3

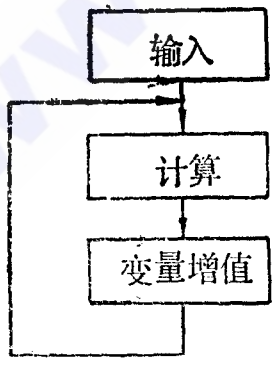


图 3—2

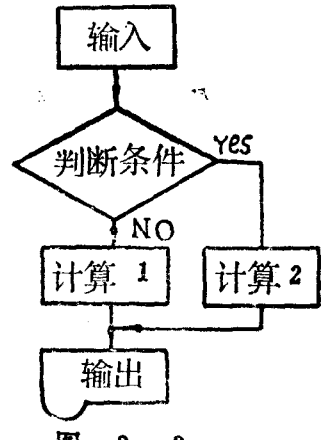
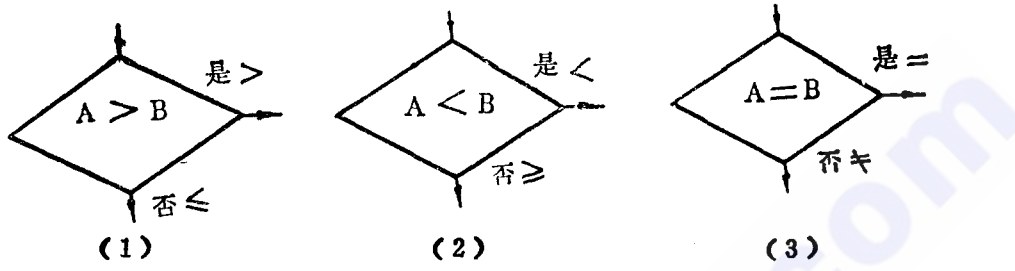


图 3—3

中走计算1的路。这样判断条件框以后产生了分支，关键在于“判断条件”框有三个口：一

个入口，二个出口。程序到此要进行逻辑或条件判断，自动区分要走那条路，于是形成了“分支流程”。这种判断分支的功能非常重要，有了它才能使整个程序灵活起来，可以说它是程序的灵魂。

判断条件有各种各样，但是最基本的有三种，即＞、＜和＝。作成框图如下：



（1）　（2）　（3）

图　3—4

图3—4表示三种基本判断条件，满足条件从横的出口走，不满足条件从竖的出口走，当然，这种规定并非是绝对的，我们也可以把竖的作为满足条件走的出口，横的作为不满足条件的出口，为了避免发生误会，画制框图时，可以在两个出口旁标出“是”和“否”或“Y”和“N”。

利用三种基本判断条件还以可组成下列三种判断条件：

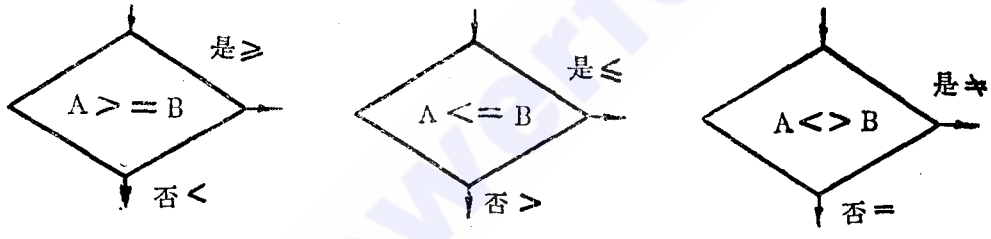


图　3—5

运用基本条件还能组成复合判断条件，一种是“逻辑加”，另一种是“逻辑乘”。逻辑加如：（A＜0）＋（A＞8）其含义是“A小于0或大于8”，逻辑乘如：（A＞0）＊（A＜8）其含义是“A大于0并小于8”。复合判断条件的内容为“或”和“并”，前者的两个条件具备其一即可，后者的两个条件必须全具备方可。满足复合条件时，计算机就显示1，不满足条件时，显示0。

（A＜0）＋（A＞8）＝0 表示两个条件全不符合。

（A＜0）＋（A＞8）＝1 表示两个条件有一符合即可。

还可写成：（A＜0）OR（A＞8）也表示两个条件有一符合即可。

（A＞0）＊（A＜8）＝1 表示两个条件要同时满足。

还可写成：（A＞0）AND（A＜8）也表示两个条件同时满足。

实用程序总是三种流程有效地组合，图3—6即为一例，程序中有两次判断，两次循环，内循环和外循环程序运算在内循环完毕，从循环出口转到外循环，然后再到内循环循环一遍，直到外循环完毕，才从外循环出口离开循环，这样的双重循环叫做“嵌套循环”。还可以设计成三重循环以至多重循环。程序从循环完毕到停机还要经过一段直线程序流程，最后形成一个完整的程序。

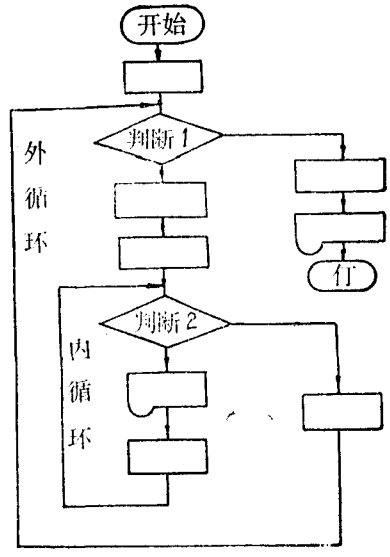


图 3—6

二、编绘框图的实例

例（1）编绘解二次方程$AX^2+BX+C=0$的方程的框图。

二次方程的根是$\frac{-B\pm\sqrt{B^2-4AC}}{2A}$，其中（$B^2-4AC$）是判别式。（$B^2-4AC$）$\geqslant 0$，有两个实根；（$B^2-4AC$）$<0$，有两个虚根。编成的程序要能判别是那一种根，并打印出结果。

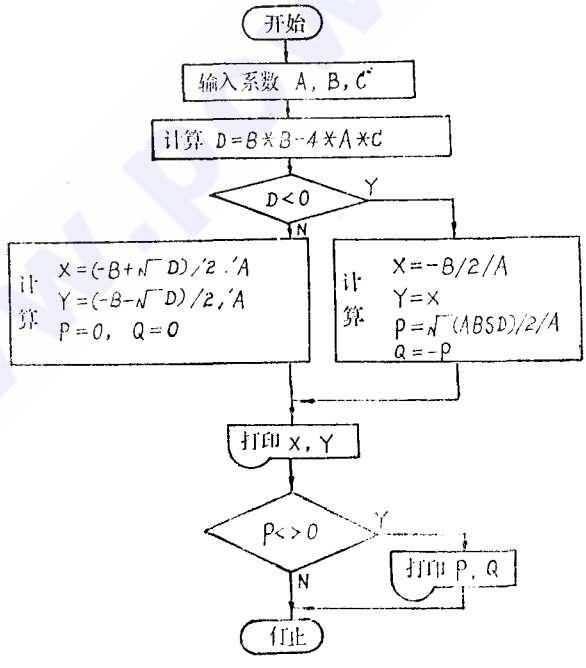


图 3—7

例（2）编绘求N! 的程序框图（见图3—8）

输入N之外，另输入两个数，F是阶乘数，I是循环次数，应赋初值1，每循环一次计算一次F的值，计算完一次在循环次数上加1，什么时候I>N，也就是最多循环N次，停止计算，打印出F值，这时F就是N的阶乘。

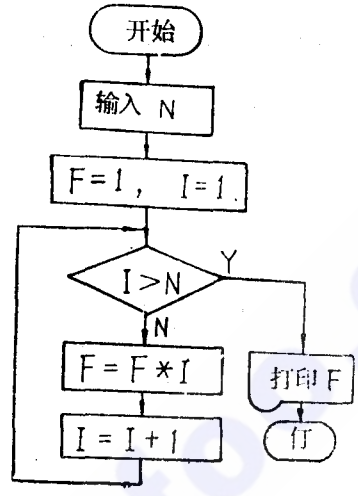


图 3—8

例（3）编制求均值程序的框图（见图3—9）

S存贮器放入数的总和，N存贮器放入数的个数，所以在输入数之前应对这两个存贮单元清零，或叫赋初值。然后输入第1个数，以后继续输入第2个、第3个数直到输完所有的数。最后输入一个0，这个0不是数，而是一个结束信号。当最后输入了0后，程序从循环出口出来，打印平均值及数据的个数。

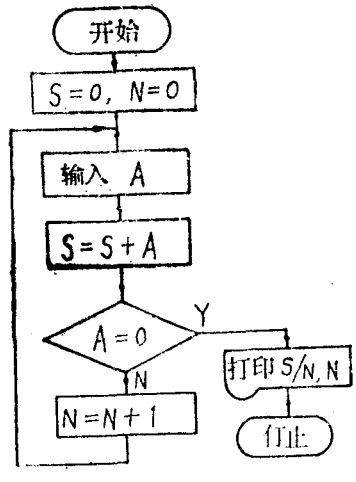


图 3—9

例(4) 编绘求3154452的因子的程序框图(见图3—10)

开始用2试除3154452，商1577226,商是整数，所以2是一个因子；再用2试除1577226，商788613，仍是整数，2是重因子；再用2试除788613，商394306.5不是整数，2不是它的因子。然后用3试除788613，商262871，是整数,所以3又是一个因子。如此继续试除,商是整数都是因子,不是整数的，改数试除，最后可以找出所有的因子，并且，打印出来。

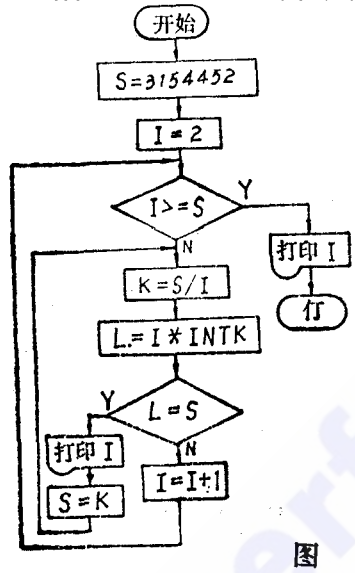


图 3—10

第六节 转 移 语 句

我们知道，程序的运行，一般总是按语句号的大小顺序执行的。但是，有时需要改变这种顺序，那么我们可以在程序中加入转移指令，让程序从另一个新规定的语句号开始顺序执行，这就是无条件转移（GOTO）语句，在输入语句一节中介绍循环输入时曾作过一些介绍。在另一种情况下，有时程序的运算过程须依据某些给定的条件是否满足而判断是否需要转移及要转移到程序的何处。这就是条件转移语句（IF～THEN语句）。

一、无条件转移语句（GOTO语句）

GOTO语句在程序中的作用是改变程序的执行顺序，当程序执行到GOTO语句时，程序就不在顺序往下执行，而转到GOTO语句指定的那个语句标号或自定义标号处开始执行。

无条件转移语句的形式：

（语句）GOTO（语句号或自定义标号）

例：编制求平均数的程序

```
10    A = 0 , B = 0
20    INPUT"DATA =" ; D
30    A = A + 1 , B = B + D
40    C = B/A
50    PRINT  B, C : GOTO  20
```

求平均值的程序和循环输入程序基本相同，只是在数据输入之后，不存入存储器而作累

加，在作累加的同时，还把输入次数（即数据数）累加起来，计算平均值C，然后输出数据的总和、平均数。10语句给A赋值0，B赋值0，其目的是将A、B存储器冲0，免得该存储器原来有数而肇致错误。行号10语句处在循环之外，仅在程序开始时做一次赋值，叫做“初值”，20，30，40和50语句构成循环。

用上面的程序求217，631，820，791，69，72，48，55，19，27十个数的和及平均数。

```
第 1 个数          [217                217]
第 2 个数          [848                424]
第 3 个数          [1668               556]
第 4 个数          [2459             614.75]
第 5 个数          [2528              505.6]
第 6 个数          [2600         433.3333333]
第 7 个数          [2648         378.2857143]
第 8 个数          [2703              337.875]
第 9 个数          [2722         302.4444444]
第10个数           [2749              274.9]
```

二、条件转移语句（IF～THEN语句）

条件转移语句（简称条件语句）的作用，是根据给出的条件，判断程序的去向。

例如，从一组数，8，7.6，－3.5，46，－87，－5，0中把负数挑选输出来的程序：

```
10    READ  X
20    IF  X>=0  THEN  10
30    PRINT  X;
40    GOTO  10
50    DATA  8，7.6，－3.5，46，－87，－5，0
```

此程序中的第20句即是一个条件语句：

```
20  IF  X>=0  TNEN  10
```

它的含义是：如果X大于等于0，即转向第10语句重新读数，如果X小于0则执行30句即输出X的值。

20句中的X>=0（即关系式）就是给出的条件。对于任何条件，都只有两种可能，即肯定或否定，如果肯定的（条件符合），程序转去执行THEN后面指定标号的语句，如果是否定的（条件不符合），程序就执行下面一条语句（下一条行号的语句）。

1. 条件及其组成

在第五节中曾作介绍，在此再归纳一下。所谓条件，就是由比较符组成的关系式，可分为简单条件与复合条件两种。

（1）简单关系式（基本条件）：最基本的，也是用得最多的。

比较符	构成关系式	关系式的含意
=	A=B	A等于B
<	A<B	A小于B
>	A>B	A大于B

<=	A<=B	A小于等于B
>=	A>=B	A大于等于B
<>	A<>B	A不等于B

（2）复合关系式：（即复合条件）

构成的关系式	关系式的含意
A>B OR A<C	A大于B或A小于C
(A>B) + (A<C) = 1	A大于B或A小于C
A>B AND A<C	A大于B且A小于C
(A>B) * (A<C) = 1	A大于B且A小于C

2. 条件语句的一般形式：

形式1：IF〈关系式〉THEN〈语句号或“标号”〉

形式2：IF〈关系式〉GOTO〈语句号或“标号”〉

形式3：IF〈关系式〉〈其他语句〉

几点说明：①形式3中的其他语句若是赋值语句，其LET指令不能省略。

②BASIC语言规定，只有条件语句可以用比较符，其他任何语句不允许使用比较符，仅有=号作为赋值符号时例外。

③前面关系式中A、B、C可以是变量，数值与算术表达式。

④如果条件是（表达式）<>0，则可省去<>0。如：

IFA<>0 THEN70可以简化为IF A THEN 70

⑤在某些情况下，THEN后也可不写语句号或标号，直接写一些BASIC的语句，如GOTO，END，BEEP，PRINT，PASUE等，则可使程序更简单明了。

三、条件语句的实例

例（1） 在三个数中选出最大的数(不保留其他数)先画出框图，然后根据框图编写程序。

```
10  INPUT  A, B, C
20  IF  A>B  THEN  40
30  A=B
40  IF  A>C  THEN  60
50  A=C
60  PRINT  A
70  END
```

框图见图3—11

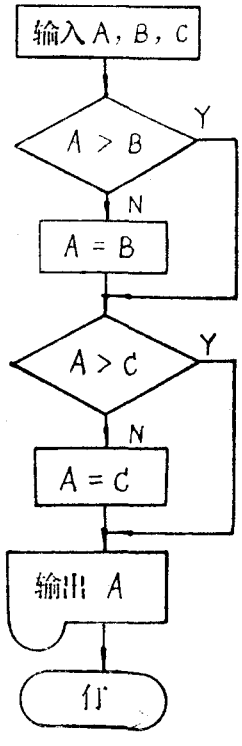


图3—11

例（2） 在若干个数中选最出大的数（不保留其他数）

同例（1） 用INPUT语句输入数据。但在此题中，用一特别的数 1E99来结束输入循环，也就是用1E99作为结束信号。这也是一种常用的方法。特别是数据很多。而且事先不便确定数目时，便可用这种方式结束程序运行。但是须注意，这个结束的数应很特别，以区别于正常数目。

下面仍先画框图，注意下面程序只能在正数范围内使用。

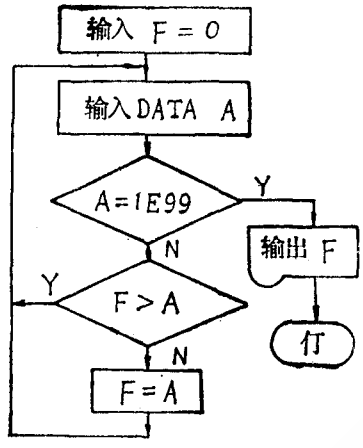


图 3—12

```
10  F = 0
20  INPUT"DATA = "; A
30  IF  A = 1E99  THEN  70
40  IF  F>A  THEN  60
50  F = A
60  GOTO  20
70  PRINT  F
80  END
```

例(3) 在3个数中选出最大的数，并保留其他的数。

```
10  INPUT  A, B, C
20  IF  A>B  THEN  40
30  T = B : GOTO  50
40  T = A
50  IF  C>T  THEN  70
60  PRINT  T : GOTO  80
70  PRINT  C
80  END
```

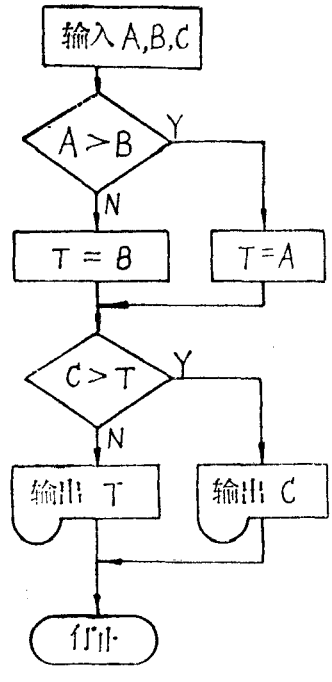


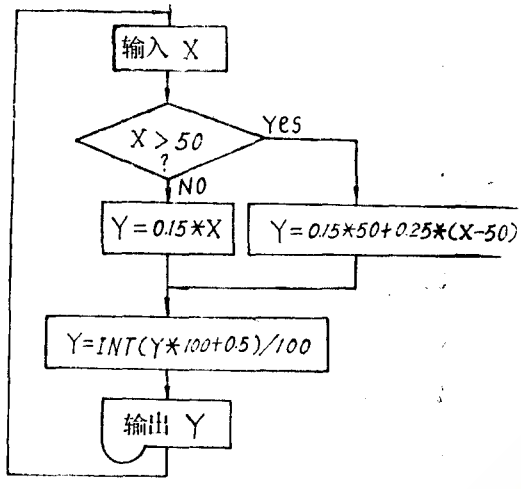
图 3—13

例（4） 某车站托运旅客行李至甲地，每张车票在50公斤以内时，每公斤运费0.15元，超过50公斤时，超过的部分每公斤0.25元，编出计算该车站至甲地的行李托运费的程序。

设Y为运费，X为行李重量

$$\text{则：}Y=\begin{cases}0.15X & X\leqslant 50\\ 0.15\times 50+0.25\ (X-50) & X>50\end{cases}$$

编制框图：




```
10  INPUT"WEIGHT=";  X
20  IF  X>50  THEN  40
30  Y=0.15*X : GOTO  50
40  Y=0.15*50+0.25*  (X-50)
50  Y=INT  (Y*100+0.5)  /100
60  PRINT"Y=";  Y
70  GOTO  10
```

图 3—14

某旅客托运行李32.5公斤，运费应为多少？（计算应为4.88元）

另一旅客托运行李175公斤，运费又应为多少？（计算应为38.75元）

例（5） 设计一个求任一自然数因子的程序

先设计求因子程序的框图

```
10  INPUT"S=";  A
20   I=2
30  IF  I>=A  THEN  110
40  K=A/I
50  IF   K=1   THEN  110
60  L=I*INT  K
70  IF  L=A  THEN  90
80  I=I+1 : GOTO  30
90  PRINT  I;
100 A=K : GOTO  40
110 BEEP 3 : PRINTI : END
```

在这个程序的第 110 句中，使用了音响命令 BEEP，其目的是让程序计算完毕后，机器发出三声音响，告

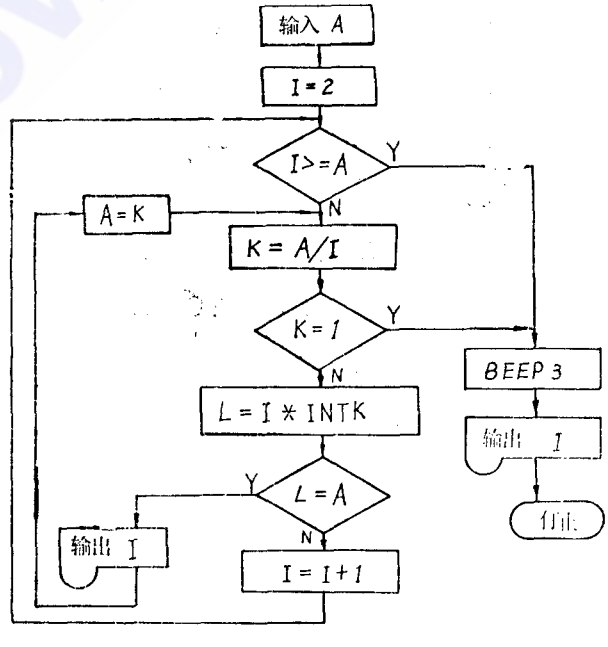


图 3—15

诉我们计算结束了。在这里BEEP后的数字是音响次数。（有关 BEEP的详细介绍，请看其他功能语句一节）

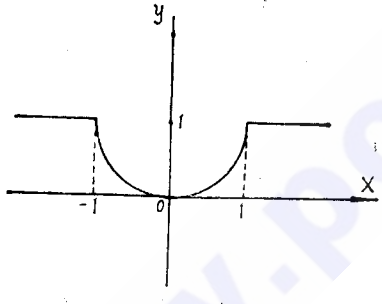
例（6） 编解二次方程，$AX^2+BX+C=0$ 的程序

框图已在第五节流程与框图中绘出，这里仅编写程序

```
10  INPUT"A=";  A, "B=";  B, "C=";  C
20  D=B*B-4*A*C
30  IF  D<0  GOTO  70
40  X=  (-B+√ D) /2/A
50  Y=  (-B-√ D) /2/A
60  BEEP  3 :PRINT  X, Y :GOTO  120
70  U=-B/2/A
80  P=√ (ABS  D) /2/A
90  Q=-P
100 PRINT  U, U
110 BEEP  3 :PRINT  P, Q
120 END
```

例（7）用程序来描述下列方程

$$Y=\begin{cases}1 & \text{当 } -1>X>1\\ X^2 & \text{当 } -1\leqslant X\leqslant 1\end{cases}$$



此题可用复合条件关系式转移语句来编写，从而使得程序更为简洁明了。

```
10  INPUT"X=";  X
20  IF  X<-1  OR  X>1  THEN 50
30  Y=X*X
40  PRINT"Y=";  Y :GOTO  60
50  PRINT"Y=";  1
60  END
```

程序还可写成如下形式：

```
10  INPUT"X=";  X
20  IF  X<=1  AND  X>=-1  THEN40
30  PRINT"Y=";  1 :GOTO  60
40  Y=X*X
50  PRINT"Y=";  Y
60  END
```

这个程序中的第20句，还可改写为下列形式，其作用完全一样：

```
20  IF (X<=1) * (X>=-1) =1  THEN  40
```

综观以上各例，充分看出条件语句是一种非常重要的语句，任何计算机的程序里，如若没有它，只能相当于一架快速算盘，充其量也就解决一些直线程序，根本无法解决计算数学方面

的问题。有了条件转移语句，实际上是给计算机增添了逻辑能力，它可以按照我们事先规定的条件，自行判断和实施操作步骤，完成非常复杂的运算，计算机好象有了活力。从实践来看几乎所有实用程序都离不了条件转移语句。因此，对条件语句一定要彻底弄清，并且能熟练地运用到程序中去。

第七节　循 环 语 句

前一节中，曾讲到无条件转移语句GOTO以及借助于GOTO语句进行程序执行顺序的改变，以达到循环地进行某一重复计算的目的。例如要对从1到100的自然数进行叠加，用GOTO语句进行循环的方式如下：

例（1）
```
10  S = 0，A = 1
20  IF  A>100  THEN  50
30  S = S + A
40  A = A + 1 ：GOTO  20
50  PRINT"S = "；S
60  END
```

程序在执行时，先对变量A赋值为1（即初值为1），判断A是否超过规定的最后一个数100(即终值为100)，再将A的值累加到S之中去，然后将A的值增加一个1（增量为1），再转移到20句进行下一轮计算。当A增加到101时（即已超过规定的终值），跳出循环，输出叠加值S = 5050。

下面介绍另一种语句，即FOR～NEXT语句。它也可达到例（1）的目的，并且更为简单明了，见例（2）。

例（2）
```
10  FOR  A = 1  TO  100  STEP  1
20  S = S + A
30  NEXT  A
40  PRINT"S = "；S
```

程序执行时，自动对A从1变化到100，每次增加一个1，计算完毕，输出S的值。

一、循环语句的结构

FOR～NEXT语句可分为两部分，一是FOR语句部分，一是NEXT语句部分。

.FOR语句的一般形式和意义。

FOR语句一形式为：

FOR〈I〉=〈A〉TO〈B〉STEP〈C〉

其中：〈I〉为循环变量，一般用简单变量。

〈A〉为循环变量初值。

〈B〉为循环变量终值。

〈C〉为循环变量增量（或称步长）。

FOR语句的意义是：对循环变量I的值从A到B，每次间隔（步长）为C。

A、B、C可以是具体的数，可以是表达式，也可以是被赋了值的变量，（表达式中的变

量也必须在此语句以前被赋值）。PC—1500规定C的值不得为小数。

如：FOR I=3*2 TO TAN 12 STEP-2这种形式都是可以的。

2.NEXT语句的一般形式及意义

NEXT语句的一般形式是：

NEXT 〈I〉

其中：〈I〉为与FOR语句中相对应的循环变量

NEXT I的意义是：执行到此语句，先判断循环变量是否超过终值，如不超过则循环变量I就加上一个增量（步长）值，返回去执行下一循环。注意，NEXT后的变量I不能省略。

在FOR与NEXT语句之间的语句，叫作“循环体”也就是被循环(运算或操作)的部分。

二、循环语句的执行过程

我们以下例来说明：

```
10 FOR I=1 TO 5 STEP 1
20 PRINT I
30 NEXT I
40 END
```

运算结果，分五次分别显示1，2，3，4，5。

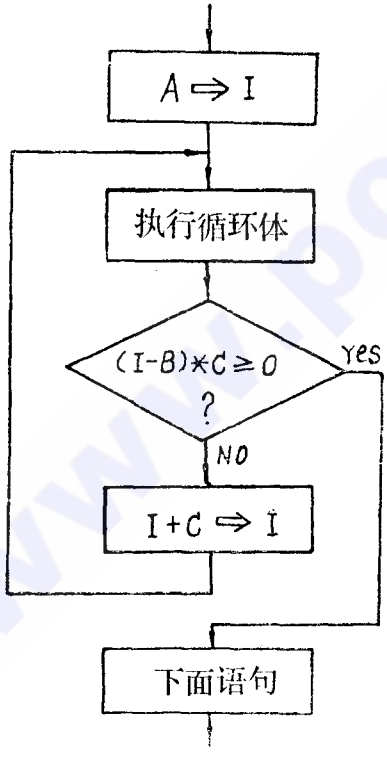


图3—16

1. 在执行FOR语句时，把初值1赋给变量I，并将终值与步长记录下来。

2. 接着执行循环体内的语句，（在此例中，循环体就是一条语句，即第20句PRINT I）循环变量在这之中保持不变，直到NEXT语句。

3. 执行NEXT语句时，先判断循环变量是否超过终值，（也就是看I≥B否），如果没有超过，则环循变量加上一个步长，然后返到FOR语句下面执行下一个循环，如果超过了终值，则跳出循环执行后面的语句。

图3—16是循环语句的框图（图中A、B、C、I的意义同前）

可以看出，FOR～NEXT语句的框图与用IF语句及GOTO语句构成的循环，框图基本上相同。

下面例(2)给出两种不同方法编写的求阶乘的程序。

例(2) 阶乘程序

方法1：用FOR～NEXT语句

```
10 INPUT "F=";F
20 X=1
30 FOR I=1 TO F STEP 1
40 X=X*I
50 NEXT I
60 PRINT F;"!=";X
```

方法2：用IF语句与GOTO语句构成循环

```
10  INPUT"F=";F
20  X=1,I=1
30  IF  I>F  THEN  60
40  X=X*I
50  I=I+1 :GOTO30
60  PRINT  F;"!  =";X
```

例（3）计算$A=1^2+2^2+3^2+\cdots\cdots N^2$

```
10  INPUT "N=" ; N
20  A=0
30  FOR  I=1  TO  N
40  A=A+I*I
50  NEXT  I
60  PRINT  A :END
```

三、PC—1500计算机关于FOR～NEXT语句的一些规定

1．循环变量的初值、终值、步长可以是数或表达式，也可以是已赋了值的变量，如例（3）中的N。循环变量必须是简单变量。在PC—1500机上，规定FOR语句的终值与步长的取值范围是－32768～32767，超过此范围显示〔ERROR 19〕。

2．循环语句的FOR与NEXT必须配对使用，缺一不可。循环开始的变量与NEXT返回的变量名必须一致，否则显示〔ERROR 2〕

3．步长为1时，可以省去STEP 1，而循环体仍按步长为1执行下去。例（3）中就省去了 STEP 1

4．步长可以是正数，也可以是负值。步长是负值时，表示每一循环变量是递减的。步长不得为小数。如为小数，机器对它自动取整；如为小于1的小数，则要出现错误显示〔ERROR 19〕。对于一些计算题，步长为小数时，可以在程序设计中进行处理，如程序：

```
10  FOR  I=1  TO  10  STEP  0.5
20  PRINT  I;
30  NEXT  I
40  END
```

可改写为：

```
10  FOR  I=0  TO  18
20  P=1+0.5*I
30  PRINT  P;
40  NFXT  I
50  END
```

或改为：

```
10  FOR  I=10  TO  100  STEP  5
20  PRINT  (I/10);
```

```
30  NEXT  I
40  END
```

5．循环参数要合理，如果不合理，则循环体只执行一次，下次不再执行循环，而去执行NEXT以下的语句，如例（4）程序就是如此。它的初值为10，终值是3，而步长是2，是不合理的。

例（4）
```
10  FOR  I=10  TO  3  STEP  2
20  PRINT  I
30  NEXT  I
40  BEEP 3 : END
```

6．循环变量的终值与初值之差，可以不是步长的整数倍，如例（5）所示。循环从1开始，然后3、5、7、9，共作五次循环，这时循环变量仍然小于终值10，还要再做一次循环，变量X成为11，然后才停止。所以这类问题在设计循环要特别注意。如把终值改为8或者改为9就可以了。

例（5）输出1～10之间各奇数的平方根
```
10  FOR  X=1  TO  10  STEP  2
20  PRINT√X
30  NEXT  X
40  END
```

在最近出产的PC—1500机及一些其它机器上（如DJS—130机），对这个问题作了改进。把循环体的执行与否，改在判断循环变量是否超过终值之后，而不是象图3—16那样，先执行循环体，后进行判断。图3—17是这些机器循环语句的框图。

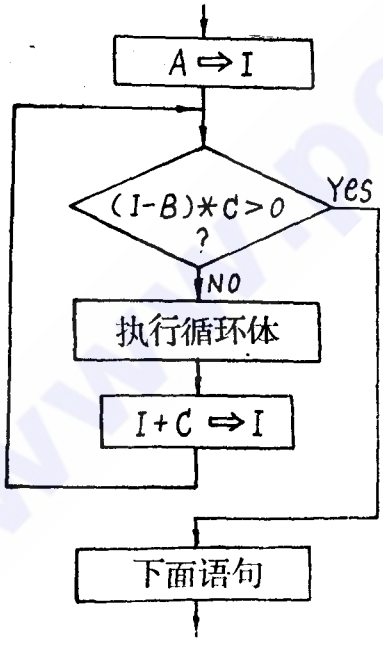


图3—17

7．在循环体内，循环变量可以参与计算，如例（2）、例（3）等。循环变量也可以不参与计算，仅仅起控制循环次数的作用，或者说仅作为一个计数器在使用。如例（6）：

例（6）
```
10  EOR  I=6  TO  10
20  READ  X
30  PRINT  X
40  NEXT  I
50  DATA  1，2，3，4，5
60  END
```

四、多重循环（循环嵌套）

前面介绍的是单重循环，它固然可以解决一些较为简单的问题，但在实际工作中，有很多问题需要两重以至多重循环才能解决

1．关于多重循环的概念

对于1×1＝1到1×9＝9的乘法表，我们可以用单重循环列出下面程序：

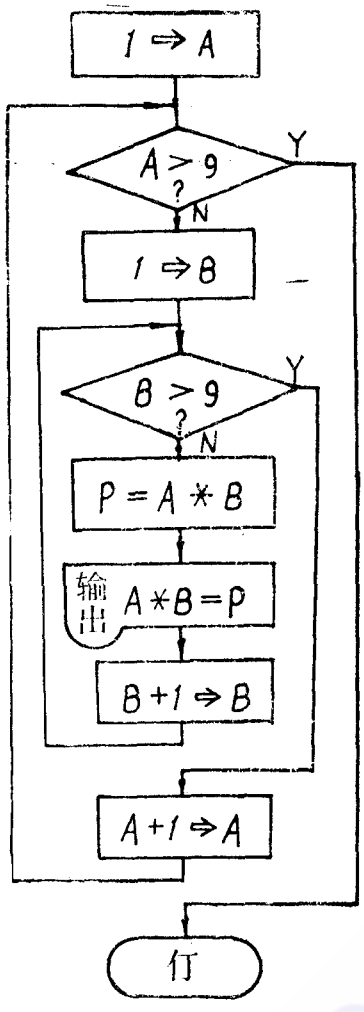


图 3—18

```
10  A = 1
20  FOR  B = 1  TO  9
30  P = A * B
40  LPRINT  A; "*"; B; "="; P
50  NEXT  B
60  END
```

如果要打印 2 × 1 = 2 到 2 × 9 = 18，则要改第10句，使A = 2。要打印整个“九九表”，则要使 A先后等于 1、2、3、，…9，每个A都要执行20句到第50句。

如用双重循环，程序和运算就可以大大地简化，如将上例改为：

例（7）

```
10  FOR  A = 1  TO  9
20  FOR  B = 1  TO  9
30  P = A * B
40  LPRINT  A; "*"; B; "="; P
50  NEXT  B
60  NEXT  A
70  END
```

这个程序和前面程序比较，只修改了第10语句，增加了一个60句，形成了一个双重循环，它的运行结果是自动打印出1×1 = 1 到 9 × 9 = 81的整个“九九表”。

这里第40句LPRINT是打印机打印输出命令。

它的框图见图 3—18

例（8） 用台劳级数展开公式$e \approx 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots\cdots + \frac{1}{N!}$近似求自然对数的底数e的数值（取N = 10）

```
10  E = 1
20  FOR  N = 1  TO  10
30  T = 1
40  FOR  K = 1  TO  N
50  T = T * K
60  P = 1 / T
70  NEXT  K
80  E = E + P
90  NEXT  N
100 PRINT  "E ="; E : END
```

以上两例，都是双重循环，这种外循环套内循环我们叫“嵌套”。由于计算的需要，我们还可以有三重以至多重循环，不同计算机有不同多重循环极限规定。

在PC—1500计算机上，循环嵌套重次由它的堆栈的暂挂功能所决定，同时还决定于程序占用了多少堆栈的暂挂功能。1500机的暂挂堆栈一共有196字节。而循环语句每对要占用12字节，如果在没有其他暂挂功能时（这是不可能的），应可嵌套（196÷12≈16）16重循环。不过实际上不可能，因为其他运算功能还要占用一些堆栈。

2. 关于多重循环的一些规定。

多重循环除去前面所述的单重循环的那些规定外，它还有一些特殊的规定。

（1）循环嵌套时，内外循环不许交叉。

例如：

```
FOR  A
 ⋮
FOR  B
 ⋮
NEXT B
 ⋮
NEXT A
```

允许

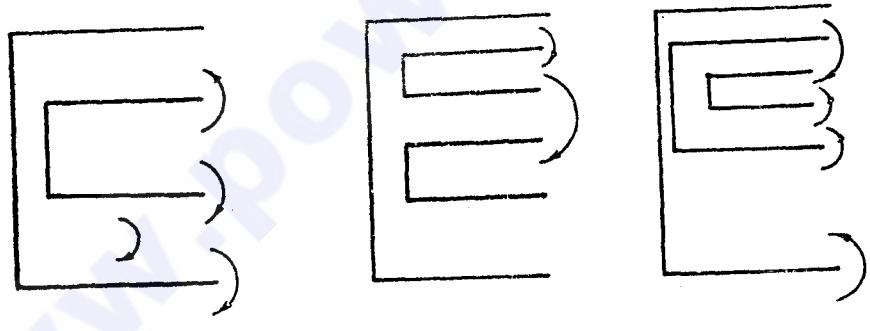
```
FOR  A
 ⋮
EOR  B
 ⋮
NEXT A
 ⋮
NEXT B
```

错误

```
FOR  A
 ⋮
FOR  B
 ⋮
NEXT B : NEXT A
```

允许

（2）循环语句和条件语句或转移语句联合使用时，可以从循环体内转出，但不允许从循环体外转入循环体内。可以从内循环转到外循环，不允许从外循环转入内循环。



允许　　不允许　　不允许

五、循环语句的应用举例

例（9）　用循环语句计算多项式的程序。

求多项式　$Y=4.7X^7+8.35X^6-5.44X^5+34.7X^4+5.5X^3-10.4X^2+5X-4$ 的值，对于这一个多项式，我们作一点变化：

$$Y=((((((4.7X+8.35)X-5.44)X+34.7)X+5.5)X-10.4)X+5)X-4$$

从中可以看出，这实际是通过一个一次式的乘和加的反复计算，逐步计算到高次多项式，也就是将一个n次多项式的计算，改为重复计算n个一次式的计算，这个一次式可将其写成通式：

(a_iX+a_{i-1})

也可将多项式写成通式：

$$Y=(((a_nX+a_{n-1})X+a_{n-2})X+\cdots\cdots+a_1)X+a_0$$

（(a_nX+a_{n-1})：第一次循环计算；$(\cdots\cdots)X+a_0$：第二次循环计算）

可以看出，对于每一循环只需计算

$$Y=a_iX+a_{i-1} \qquad i=n,\ n-1,\ n-2\cdots\cdots 1$$

即可。因此我们可以根据以上的分析先编制框图，见图3—19。

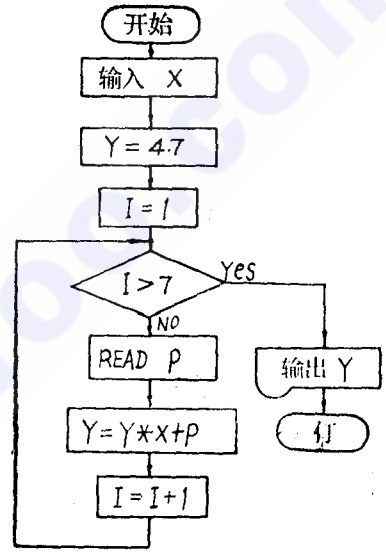


图 3—19

编写程序时，我们将各项系数放在DATA语句之中。

程序如下：

```
10  INPUT  "X=";  X
20  READ  Y
30  FOR  I=1  TO  7
40  READ  P
50  Y=Y*X+P
60  NEXT  I
70  PRINT  "Y=";  Y
80  DATA  4.7,  8.35,  -5.44,
          34.7,  5.5,  -10.4,  5,  -4
90  END
```

设X=4运行程序结果为：

Y=114720.64

例（10）　用循环语句编写程序，计算下式的值（取n=10）

$$Y=X-\frac{X^3}{3!}+\frac{X^5}{5!}-\frac{X^7}{7!}+\frac{X^9}{9!}-\frac{X^{11}}{11!}+\frac{X^{13}}{13!}\cdots\cdots-\frac{X^{4n-1}}{(4n-1)!}+\frac{X^{4n+1}}{(4n+1)!}$$

为方便与清晰，我们可将上式改写成：

$$Y=\left(\frac{X}{1}+\frac{X^5}{5!}+\frac{X^9}{9!}+\frac{X^{13}}{13!}+\cdots\cdots+\frac{X^{4n+1}}{(4n+1)!}\right)-\left(\frac{X^3}{3!}+\frac{X^7}{7!}+\frac{X^{11}}{11!}+\cdots\cdots+\frac{X^{4n-1}}{(4n-1)!}\right)$$

这样，可以看出两个括号内的结构是一样的，我们可以较为容易地用循环语句来编写程序，不过要注意循环变量的初值与终值的选定，还要注意变量的赋初值问题。程序如下：

```
10  N=10,  A=0,  B=0
20  INPUT  X
30  FOR  I=1  TO(4*N+1)  STEP  4
40  S=1,  T=1
```

```
 50  FOR  J = 1  TO  I
 60  T = T * J,  S = S * X
 70  NEXT  J
 80  A = A + S/T : NEXT  I
 90  FOR  I = 3  TO  (4 * N - 1)  STEP  4
100  P = 1,  Q = 1
110  FOR  J = 1  TO  I
120  P = P * J,  Q = Q * X
130  NEXT  J
140  B = B + Q/P : NEXT  I
150  Y = A - B : PRINT  "Y = ";  Y
160  END
```

运行结果读者可自己在机器上算出。

例（11）编写一个求1～100自然数中的质数的程序。

```
 10  INPUT "N = ";  N
 20  LPRINT 2;  3;
 30  FOR  I = 3 TO  N  STEP  2
 40  FOR  J = 3  TO (I - 2) STEP  2
 50  A = INT  (I/J)
 60  IF  A = I/J  THEN  90
 70  NEXT  J
 80  LPRINT  I;
 90  NEXT  I
100  BEEP  3 : END
```

例（12）求定积分$\int_0^1 \mathrm{Sin}\ (x)\ dx$

从数学上知道，定积分$\int_a^b \sin(x)dx$的几何意义是求曲线Sin(x)在x = a与x = b 之间与 X 轴所围成的面积。（见图 3 —20）。我们可以将该面积近似地分成若干个小梯形，各小梯形面积之和即为$\int_a^b \sin(x)dx$的近似值。小梯形的面积分得越多越密，则求出的近似值愈精确。根据题意，令a = 0，b = 1，那么每个小梯形的高为 $h = \frac{1-0}{n}$，（n 为分的小梯形个数），而第一个梯形的面积为 $\frac{(\sin(0)+\sin(h))h}{2}$；

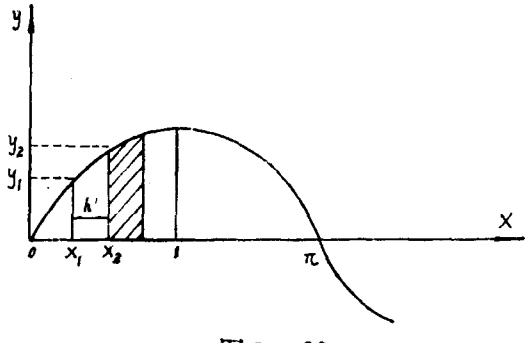


图 3 —20

第i小形梯面积为$\frac{[\sin((i-1)h)+\sin(i\cdot h)]h}{2}$。将几个小梯形面积加起来便可求出其近似值了。

程序如下，N为分成的小梯形个数。

```
 5  INPUT"N="; N
10  A=0, B=1, S=0
20  H= (B-A) /N
30  FQR  I=1 TO  N
40  T= (SIN ((I-1)*H) +SIN (I*H)) *H/2
50  S=S+T : NEXT  I
60  PRINT"S="; S
70  END
```

（本题在运算时，要注意机器的角度选择方式应为RAD）

运算结果：当N=4时；S=4.573009375E-01

当N=10时；S=4.593145488E-01

习　题　三

一、绘出下列各题的流程框图。

1. 根据$A=\pi^2$，计算园面积。r为半径。

2. 我国人均国民生产总值1980年为257美元，若每年梯增7%，问到哪一年可超过1000，美元？

3. $$Y=\begin{cases}\cos(X+3) & \text{当：}\ 0\leqslant X<10\\ \cos(X+7.5) & \text{当：}\ 10\leqslant X<20\\ \cos(X+4.5) & \text{当：}\ 20\leqslant X<30\end{cases}$$

二、指出下面的BASIC语句有什么错误，并改正之。

1. `10  IF  X>=  THEN  70`

2. `20  IF  X+Y≒4 * 3  THEN  90`

3. `20  IF  A-B≥COS40 THEN  70`

4.
```
10  LET  X=3
20  IF  X>0  THEN  60
30  PRINT  X
40  END
```

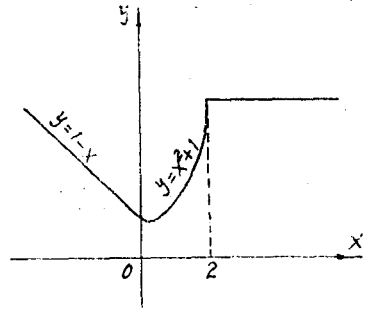
三、写一个程序把下面数据中的负数打印出来，然后求它们正数与负数的和各为多少。

-5，-13，56，-3.2，0，64，12，-2.5，8，-15.3

四、写一个程序，求下式中的Y值，（要求X的值由INPUT语句输入）

$$Y=\begin{cases}\dfrac{\sin x+\cos x}{2} & x\geqslant 0\\ \dfrac{\sin x-\cos x}{2} & x<0\end{cases}$$

五、用条件语句描述下面曲线



六、指出下列程序有否错误，应如何修改

1.
```
10  INPUT"N=";N
20  K=1
30  FOR  I=1  TO  N
40  K=K*I
50  IF  I>12  THEN  80
60  NEXT  I
70  PRINT  K
80  END
```

2.
```
10  A=0
20  FOR  I=1  TO  10
30  A=A+I
40  FOR  B=1  TO  5  STEP  2
50  A=A*B
60  NEXT  I
70  NEXT  B
80  PRINT  A
90  END
```

3.
```
10  P=0
20  FOR  K=1  TO  9  STEP  1.5
30  P=P+K∧2
40  PRINT  P
50  NEXT  P
60  END
```

七、用循环语句编写计算 $1+2+2^2+2^3+\cdots\cdots+2^{63}$ 的程序。

八、有一个由50个数组成的数列，它的头二个数是0和1，第三个数是第一个与第二个之和，第四个数是第二个数与第三个数之和，以后每一个数是其前两个数之和。即此数列为：

0，1，1，2，3，5，8，……

编写程序计算并打印出这个数列。（第一、二个数由INPUT输入）

九、计算：$C_j^k = \frac{j!}{(j-k)!}$（j、k由键盘输入）

〔例如C_{30}^{10}的意思是计算$C_{30}^{10} = \frac{30!}{(30-10)!}$〕

第八节　保留函数与自定义标号、随机函数

一、保留函数及其在程序中的运用

第二章常规方式运算中，已介绍过保留函数在常规方式的使用方法，保留函数的容量分配等，在这里进一步介绍它在编制程序中如何使用。

保留函数在编制程序时是一种非常有用和方便的方法，它可将在程序中要多次重复出现的算式（函数），保留在机器中，随时调出编入程序，这样不仅使书写方便，而且在使用程序时也很方便。

例（1）　计算一个多边形（图3—21）面积。将多边形划分成若干三角形后，各边的长为：a＝4，b＝2，c＝5，d＝2.5，e＝6，f＝5，g＝4.5

我们已知，求三角形面积S的公式为：

$S = \sqrt{P(P-X)(P-Y)(P-Z)}$

其中：P＝（X＋Y＋Z）／2

X，Y，Z为三角形的三边

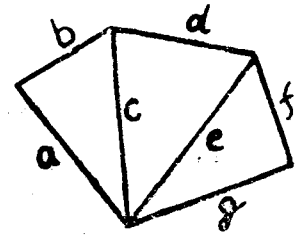


图3—21

下面用保留函数编入程序，先选择机器使用方式到〔RESERVE〕

在第一个保留键F1中存入 P＝（X＋Y＋Z）／2

在第二个保留键F2中存入 $\sqrt{\ }$（P＊（P－X）＊（P－Y）＊（P－Z））

为方便起见，用下列书写方式表示：（今后都是这样）

〔RESERVE〕：[F1] P＝（X＋Y＋Z）／2

[F2] $\sqrt{\ }$（P＊（P－X）＊（P－Y）＊（P－Z））

编写程序如下：

```
10  A = 4, B = 2, C = 5, D = 2.5, E = 6, F = 5, G = 4.5
20  X = A, Y = B, Z = C, Q = 0, S = 0
30  [F1], Q = [F2], S = S + Q
40  X = D, Y = E
50  [F1], Q = [F2], S = S + Q
60  X = F, Z = G
```

```
70 [F1], Q=[F2], S=S+Q
80 PRINT S:END
RUN[                    20.94558289]
```

以上程序是按键操作情况，实际源程序的第30、50、70句中并没有[F1]，程序中的[F1]表示按第一个保留键，将P=（X+Y+Z）/2这个函数调入程序之中来，[F2]表示将函数$\sqrt{\ }$（P*（P-X）*(P-Y)*（P-Z））调入程序之中“Q=”的后面。这样就很方便地将函数调入程序，减少了很多书写程序及按键操作。

保留函数在程序之中使用应注意以下几点：

（1）如第二章中所介绍的那样，保留函数的总容量不得超过机器规定的保留函数容量，（共不得超过108个字节，每个函数不得超过80字节。）最多可保留18个。

（2）保留函数在程序之中可多次被调入，次数不限。

（3）保留函数中的变量等，应与程序之中的变量符合。实际上，调入保留函数就是减少书写按键而已。

（4）保留函数中变量都是虚假变量，在执行以前必须是程序中已经赋过值的变量，如上例中的X、Y、Z在运算之前已赋值。

二、保留函数保留指令及其用法

保留函数除前面已经介绍过的两种用法以外，还有一个很重要的用途，即用它保留一些常用的键盘操作指令与语句指令。例如：程序启动运行命令RUN，数据（区）清除命令CLEAR、程序与数据一同清除（文本区清除）命令NEW等。

对于这些常用的指令，不光是其命令的定义符可存入，我们还可连同其执行功能一并存入保留函数之中，也就是将执行键的功能一并存入，下面以RUN命令的存入为例来介绍。

1.RUN与[ENTER]键功能一起保留：

选择机器的运行方式在RESERVE方式：

在[F6]中存入：RUN@

2.RUN命令的使用：

有了此保留命令后，使用时只需在RUN方式下，按所保留了的键[F6]一次，则RUN命令就被执行了。可以看出@在存入了保留函数时，它代表执行键的功能。

对于其他的一些常用指令，我们均可用这种方法存入与使用。这种指令的保留方法，非常方便和实用，我们可根据需要自己编制保留一些。

三、用户自定义标号

在输入语句一节介绍自动读入时，曾讲到过用户自定义标号，现在再详细探讨它的用途用法。

用户自定义标号可以给整个程序的段落规定名称，用一个标号键代表，像“自动读入”一节的例子就属于这一类。有了用户标号键，使用程序时有许多方便，只用这个标号就可以起动程序。例如在编制一个比较大的程序时，可以把输入部分定一个标号，把计算部分定一个标号，再把输出部分定一个标号，全部程序从头到尾执行，或者从中间标号处分段执行，而且还可以随意选择其中的一段检查执行结果。这不仅给启动程序创造许多方便，减少误操

作，还能用它来辅助调试程序。

例（2） 编制一个复数加、减、乘、除运算的程序

第一步，使用保留函数，在下列各保留键中分别保留所列函数：

[F1] 中存入：R=√(X*X+Y*Y)

[F2] 中存入：A=(Y=0)+SGN Y

[F3] 中存入：C=ACS (X/R) *A

[F4] 中存入：X=R*COS C

[F5] 中存入：Y=R*SIN C

在 [F2] 中，A=(Y=0)+SGN Y的作用是判断矢量在第几象限，A为正时，矢量在第一或第二象限，A为负时，矢量在第三或第四象限。A不能为0，为此在表达式中增加了一项逻辑比较（Y=0），当Y=0时，括号比较的结果是1，所以表达式虽SGN Y为0，A还是等于1。

矢量的幅角C由ACS (X/R) 决定大小，由A决定它在第几象限。

第二步，使用用户自定义标号编制程序：

在编写时，我们规定：

标号A为复数加法

标号S为复数减法

标号D为复数乘法

标号F为复数除法

源程序如下：

```
 10  INPUT "X=", X, "Y=", Y
 20  [F1], [F2], [F3]
 30  P=X, Q=Y, U=R, V=C:END
 40  "A"INPUT "X=", X, "Y=", Y
 50  X=P+X, Y=Q+Y
 60  [F1], [F2], [F3]
 70  P=X, Q=Y, U=R, V=C
 80  PRINT P, Q:END
 90  "S"INPUT "X=", X, "Y=", Y
100  X=P-X, Y=Q-Y
110  [F1], [F2], [F3]
120  P=X, Q=Y, U=R, V=C
130  PRINT P, Q:END
140  "D"INPUT "X=", X, "Y=", Y
150  [F1], [F2], [F3]
160  R=U*R, C=V+C
170  [F4], [F5], P=X, Q=Y, U=R, V=C
180  PRINT P, Q:END
```

```
190  "F"INPUT "X=";X,"Y=";Y
200  [F1],[F2],[F3]
210  R=U/R,C=V-C
220  [F4],[F5],P=X,Q=Y,U=R,V=C
230  PRINT P,Q:END
```

以上程序共分五段，第一段是输入第一个复数的程序，不分加、减、乘、除都需要输入。然后可根据题目的要求选用不同段落，加法用标号A；减法用标号S；乘法用标号D；除法用标号F。这个标号不是按英文字母顺序来选用，而是按键盘顺序排列的，用起来方便，不容易出错。每一段程序的结束都用了END语句，这一点很重要，否则，程序执行到用户标号时不会因有用户标号而停止，将继续执行下去，导致整个运算混乱，所以用户标号键只启动程序，而不能中断程序。

必须注意，使用了用户自定义标号后，在程序运行时，一定要先按DEF键，再按所选定的标号字母键。

对上面程序我们用下面例子来进行运算（注意角度方式选择RAD方式）。

求：((5+j7)/(38-j21)+(-21+j7.5))*(-6+j10)-(77-j251)=?

操作过程：

1、先输入（5+j7）：　按RUN以后，显示"X="，然后由键盘输入5
再按[ENTER]显示"Y="，然后由键盘输入7

2、用标号F，输入（38-j21）
[DEF] [F]　将X=38，Y=-21输入
显示〔2.281167E-02　1.968169E-01〕

3、用标号A，输入（-21+j7.5）
[DEF] [A]　输入X=-21，Y=7.5
显示〔-20.97718833　7.6968　16976〕

4、用标号D，输入（-6+j10）
[DEF] [D]　输入X=-6，Y=10
显示〔48.89496022　-255.9527851〕

5、用标号S，输入（77-j251）
[DEF] [S]　输入X=77，Y=-251
显示〔-28.10503978　-4.9527851〕

所以本题计算结果为：

-28.10503978-j4.9527851

使用自定义标号的几点说明：

1、在PC—1500机上可供使用的标号键共有18个，它们是：英文字母键盘的第二排（A～L）共9个，第三排（Z～M共7个再加SPACE键和等号键。

英文字母键的第一排已被机器定义了下列语句指令：

标号Q定义了　INPUT指令
标号W定义了　PRINT指令

标号E 定义了　USING指令
标号R 定义了　GOTO 指令
标号T 定义了　GOSUB指令
标号Y 定义了　RETURN指令
标号U 定义了　CSAVE指令
标号 I 定义了　CLOAD指令
标号O 定义了　MERGE指令
标号 P定义了　LIST指令

这些被定义了的指令，可以象调保留函数那样调入程序之中，这样就可以在用键盘输入程序时，减少按键次数，也不易错。

2、在一个程序中不要使用同一标号来定义不同的程序段。如果用了相同的标号，在启动这个标号的程序时，程序从语句号小的那段开始运行。

四、随机函数RND与RANDOM语句：

在生活与生产中，常有一些随机现象发生。如常见的掷骰子的结果；气象与水利方面的雨量，水流量等。PC—1500计算机为我们提供有产生随机数的函数与语句。我们可利用这些函数生成一定数量的随机数，加上一些特定的边界约束条件，用来模拟某些生产生活之中的随机过程，以供我们生产中分析、计算之用。

1、RND　随机函数

RND函数的一般格式：　　RND(I)

其中：I 可以是表达式、变量与数。

PC—1500计算机规定，(I 的取值范围是：

$-9.999999999E99 \leqslant I < 10E99$

如(I)为大于 1 的小数，则将舍去小数部分；

如 I)为小于 1 的小数，则作为零处理。

RND函数的功能：产生一个不大于 I 的随机数，若：

(I ＞0　RND I)产生一个不大于 I)的正整数

(I ＝0　RND I 产生一个小于 1 的正小数

(I ＜0　RNO I 产生一个小于1的正小数

例（3）　产生 5 个 1 ～10的随机数的程序

```
10  FOR  I = 1  TO  5
20  X = RND  10
30  PRINT  X
40  NEXT  I  : END
```

例（4）　产生 5 个100～150之间的随机数的程序

```
10  FOR  I = 1  TO  5
20  X = 99 + RND  50
30  PRINT  X
40  NEXT  I  : END
```

例（5） 产生200个小于1的随机小数，并统计小于0.5的有多少个，等于大于0.5的有多少个？

```
10  FOR  I=1  TO  200
20  S=RND  0
30  IF  S<0.5  LET  A=A+1 :GOTO  50
40  B=B+1
50  NEXT  I
60  PRINT  A, B
70  END
```

此例产生200个小于1的随机数，通过统计可以发现它们在0.5两边的数的分布大致是差不多的。如果产生的随机数愈多，（比如不是200个而是1000个），则0.5两边分布的数的个数愈接近。

2、RANDOM语句：

RANDOM是恢复随机数发生器初始状态的语句命令。

RANDOM的使用格式：

语句号 RANDOM

RANDOM的功能：恢复随机数发生器的初始状态，它作为语句放在RND函数之前，使得RND函数产生随机数之前先恢复随机数发生器的初始状态，在每关机再开机后产生的随机数不致重复。

例（6）

```
10  FOR  I=1  TO  5
20  X=RND  50
30  PRINT  X;
40  NEXT  I
RUN   〔   10  19  33  40  7   〕
```

关机再开机 RUN 〔 10 19 33 40 7 〕

可以看出，第一次运行后关机再开机运行，产生的随机数的结果一样。我们加上一句5 RANDOM，使程序成为：

```
5  RANDOM
10  FOR  I=1  TO  5
20  X=RND  50
30  PRINT  X;
40  NEXT  I:END
RUN   〔   29  8  23  17  48   〕
```

关机后再开机 RUN 〔 33 43 31 12 9 〕

有了第5语句以后，两次运行的结果就不会相同了。所以，随机函数RND的使用，往往应在其前加上一条RANDOM语句。

第九节　子　程　序

在一个程序中，如果需要多次进行某种计算，我们就必须多次重复书写执行这一运算功能的程序段，但是，这样做在程序中不仅占用较多的程序步，而且书写烦琐，容易出错。为此我们可以使用上节所讲的使用保留函数的办法。但是一个保留函数只允许一个语句，而且每个保留函数还不能超过80步程序步。为了解决这个问题，在BASIC语言中设计了子程序。

子程序，是一个相对独立的程序。当执行主程序过程中。需要用它时，通过调用指令把它调出来使用，使用完以后又通过返回指令让程序返回到主程序中去运行。这样，我们可以对一个或几个子程序进行使用。（因为引入子程序的概念，我们把原程序称之为主程序）。

调用子程序的语句称为转子语句，它通过GOSUB命令来命令计算机自动地转去执行子程序。转子语句的形式如下：

〈语句标号〉　GOSUB　〈子程序的第一条语句号〉

子程序最后一条语句是（必须是）返回语句RETURN。当计算机执行到这条语句时，又自动返回到主程序中GOSUB语句的下一个语句接着向下运行。

我们用下例来说明转子语句GOSUB和返回语句RETURN的功能。

例（1）　用随机函数产生5个10～20之间的整数，并求它们的阶乘与阶乘的和，

```
10  CLEAR
20  FOR  I=1  TO  5
30  RANDOM
40  A=10+RND  10
50  GOSUB  200
60  S=S+T
70  NEXT  I
80  PRRINT"S=", S
90  END
200 T=1                    ┐
210 FOR  J=1  TO  A        ├ 计算阶乘的子程序
220 T=T*J                  │
230 NEXT  J                │
240 RETURN                 ┘
```

当程序运行时，第20句与70句之间的循环语句共循环5次，由第40语句产生5个随机数。每产生一个以后，便由50语句转子语句转到子程序进行阶乘计算。（第200～240句为计算阶乘的子程序）然后由240句RETURN命令程序返回到第60句继续运行，进行阶乘值的累加。

从例子中可以看出，子程序实际是可以并入主程序的。但是用子程序的办法来编写程序主要是可以简化程序的结构，并且使程序更加明了直观。

我们将上节复数的加减乘除运算的程序，改用子程序来编写；

例（2）用子程序编写复数加、减、乘、除的程序。

```
 10  GOSUB  500
 20  GOSUB  600
 30  GOSUB  800 : END
 40  "A"GOSUB  500
 50  X=P+X, Y=Q+Y
 60  GOSUB  600
 70  GOSUB  800
 80  PRINT  P, Q : END
 90  "S"GOSUB  500
100  X=P-X, Y=Q-Y
110  GOSUB  600
120  GOSUB  800
130  PRINT  P, Q : END
140  "D"GOSUB  500
150  GOSUB  600
160  R=U*R, C=V+C
170  GOSUB  700
180  GOSUB  800
190  PRINT  P, Q : END
200  "F"GOSUB  500
210  GOSUB  600
220  R=U/R, C=V-C
230  GOSUB  700
240  GOSUB  800
250  PRINT  P, Q : END
500  INPUT  "X=";  X, "Y=";  Y : RETURN
600  R=√  (X*X+Y*Y) , A=  (Y=0) +SGN  Y,
     C=ACS (X/R) *A
610  RETURN
700  X=R*COS  C, Y=R*SIN  C : RETURN
800  P=X, Q=Y, U=R, V=C : RETURN
      〔STATUS  1  389〕
```

用上节的例子的数据代入，计算结果相同。

用STATUS 1 指令调看占用了389步程序步，而用自定义标号方式编写的程序要占有545步，说明子程序可以大大简化程序

调用子程序的规则：

1、调用子程序时，必须在调用的地方写一条GOSUB语句。

2、子程序的最后一条语句必须是RETURN（返回）语句。

3、子程序返回主程序时，返回到本次转子程序语句的下一个语句（而不是下一行）并继续执行。

4、子程序可以象循环语句那样进行嵌套，在PC—1500机上，每一子程序转子语句占用运算功能堆栈为6字节，所以其最大的可能嵌套重次不得超过32重。（在实际上，也不可能全部堆栈功能都让转子语句占用，因为那样的程序是无实际意义的）

图3—22表示嵌套的方法。

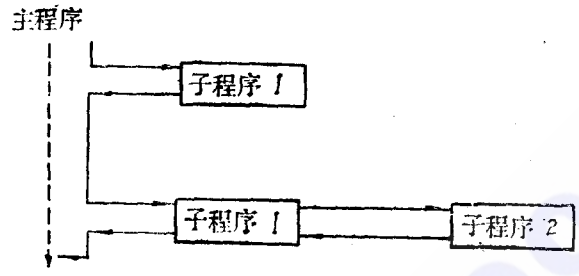


图3—22

5、同一子程序可以有几个入口，也可以有几个返回口，如下图3—23所示：

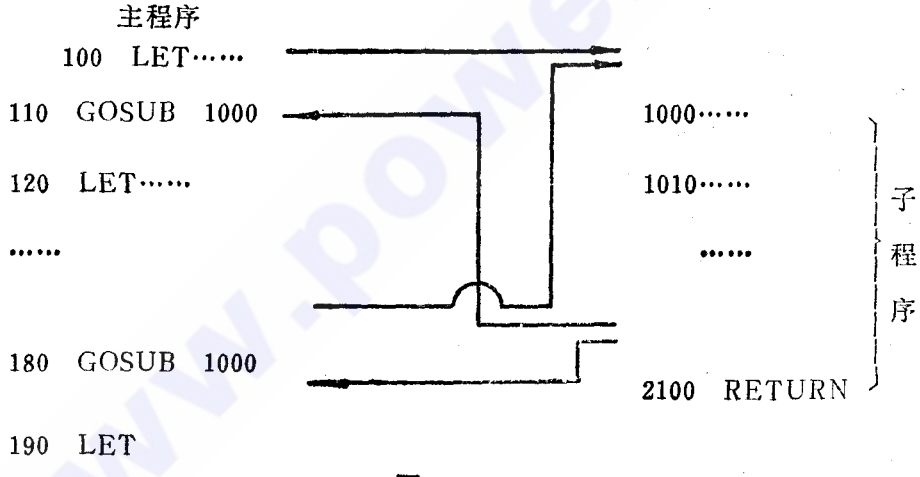


图3—23

6、子程序的行号，一般都采用比较大的号数与主程序隔的远一些，一来看得清楚，不易与主程序混淆，二来可以给主程序留下增添语句的位置。如例（1）中的子程序首句行号用的200号。

下面我们把保留函数，自定义标号及子程序列表比较：

表 3—3

比较项目	保留函数	用户自定义标号	子程序
主要用途	简化书写手续	分段启动程序	简化程序
控制范围	一个程序语句	只能启动用	一段程序
可否套用	不 能	不 能	可嵌套
可省程序步否	不 能	不 能	能
可否返回	不 能	不 能	能自动返回
与主程序关系	在主程序内的语句	主程序分段	主程序以外的独立程序
调用灵活性	不存在调用问题	能用程序调用	可由程序自由调用，有几个入口，也可有几处返回
计算机使用方式	保留时用RESERVE方式，运行时用RUN方式	运行用RUN方式	运行用RUN方式

习 题 四

一、用保留函数与自定义标号方式，计算与编制求园周长，园面积和球体积的程序。

半径设为$r_1=5$，$r_2=13$，$r_3=17$，用INPUT输入半径。

二、以下程序的打印结果是什么？

```
10  FOR  I=1  TO  5
20  GOSUB  50
30  NEXT  I
40  GOTO  70
50  LPRINT  I;
60  RETURN
70  END
```

三、求三组随机数，每组三个数，用转子程序打印出这三组数，用子程序算出每组中最大的数。

四、用子程序方法编制求下式的程序，并根据给出的变量值求M＝？

$$M=A\left(\frac{\sin x}{\sin(x-y)\sin(x-z)}+\frac{\sin y}{\sin(y-x)\sin(y-z)}+\frac{\sin z}{\sin(z-x)\sin(z-y)}\right)$$

$A=5$，$x=45°$，$y=60°$，$z=90°$

第十节 单下标变量及DIM语句

在前面章节中，已多次介绍并使用了BASIC语言所规定的简单变量，从这节开始，引入一个新的变量——下标变量（包括单下标变量与双下标变量）。下标变量是十分重要的，

它为程序的编写带来极大的方便。下面将对它进行详细的介绍。

一、下标变量

例（1） 下面程序是求一组数2，4，3，8，9，2的和。

```
20 INPUT "N"; N
30 DIM A (N)
40 FOR K = 1 TO N
50 INPUT A (K)
60 PRINT"A ("; K; ") ="; A (K)
70 NEXT K
80 LET S = 0
100 FOR K = 1 TO N
110 S = S + A (K)
120 NEXT K
130 PRINT"S ="; S
140 END
RUN〔S = 28 〕
```

N为数的个数，N = 6

我们在这个程序中，引入一个新的变量，即下标变量，下标变量的形式是：

A(I)

A——为带下标的变量

I——为下标

PC—1500机规定：

1、下标变量可以由一个大写字母组成，也可以由两个大写字母组成，如A(I)、AA(I)、AH (I) 等。下标变量还可以由大写字母带数字组成，如：A 3 (I)；B 1 (I)等。

2、下标只能放在圆括号内。

3、下标可以是数值，也可以是变量，如：A (I)，A (A 1) ……等。

4、下标可以是算术表达式，如：A (1 * 3 - J)。

5、下标的数值必须是正整数。如下标带有小数，则机器自动取整。下标不允许是负数。

二、一维数组

在例（1）中，我们引入一个数组A (K)，K = 0 ～ 9。在实际应用中，我们总是把一些具有相同性质的数放在同一数组中，对每个不同的数，只是用不同的下标来区别它们。

例如，某河的月平均流量，我们可以用Q (I) 来表示，I为1时，表示是一月份的月平均流量，I = 12时，表示是十二月份的流量，这样，即清楚又简单。

我们将具有一个下标的数组称为一维数组。

在计算机中，每一个下标变量（即数组中的每一个数组元素）将占有一个内存单元。PC—1500机为每一一维数组准备了256个单元，也就是说：在PC—1500 机上，一维数组最

多不能超过256个数组元素。

三、数组说明语句——定维语句

对带下标的数组，在程序中必须加以说明，（或者说必须定维），数组只有在说明以后，机器才能为它准备足够的单元来贮存它。

1、数组说明语句的形式是：

〈语句号〉DIM　A（N）

N是数组A的界偶，如上所述，0≤N≤255。

2、数组说明语句必须放在数组使用之前（见上例（1）中的第30语句），在同一程序中如果有几个数组可以在同一数组说明语句内说明，每个被说明的数组之间用逗号分开。如：

10　DIM　A（17），B（18），KK（50），E7(20)

在这句语句之后，A数组的下标最大可用到17(从0～17)，所以A数组可以有18个数。又如E 7数组可以有21个数，等等。

3、在同一数组之中，不得对数组重新说明，或者说，不得使用两个及以上的相同标识符的数组。否则机器显示错误标志：〔ERROR　5　　〕。

4、注意的几个问题：

（1）在程序中使用数组时，都应根据数组元素的个数写出数组说明语句，如果不写数组说明语句，机器在执行到有下标的变量的运算时，显示〔ERROR　6　　〕；一维数组的界偶超过255时，则显示〔ERROR　19　　〕。

（2）已说明的数组在程序中使用时，其数组元素的下标值不得超过所规定的界偶，否则机器显示〔ERROR　9　　〕。

（3）有些机器的数组可以重新被说明，有些机器的数组可以不说明。这要根据各自机器的特点，参考其说明使用。

四、程序举例

例（2）　学生成绩统计的程序。

此程序统计学生成绩：100分的几人，90～99分几人，80～89分几人……即每一分数段几人。全班32人成绩放在DATA语句之中。

用一维数组S（I）记录各分段人数，S(0)记0～9分的人数，S(1)记10～19分人数，S(2)记20～29分人数……S(10)记录100分者人数。

如果学生成绩是83分，则Q/10=8.3，取整后R=8，第80语句使得下标变量S（8）的数值增加1；如果成绩是76，70语句使得R=7，80语句使得下标变量S(7)的数值增加1，如果成绩是76，70语句使得R=7，80语句使得下标变量S（7）的数增加1，依此类推。

DATA　语句的最后一个数值是－1，这是一个结束标志，当Q=－1时，60语句使得程序结束读数。

```
10  DIM  S (10)
20  FOR  I = 0  TO  10
30  S (I)  = 0
40  NEXT  I
50  READ  Q
```

此段为给S（I）清零，可以用一个语句

5　CLEAR　　代替

```
60  IF  Q=-1  GOTO  100
70  R=INT  (Q/10)
80  S (R) =S(R)+1
90  GOTO  50
100 FOR  I=0  TO  10
110 IF I=10  GOTO  140
120 PRINT  10*I; "--"; I*10+9; "        "; S (I)
130 GOTO  150
140 PRINT  I*10; "    "; S (I)
150 NEXT  I
200 DATA  62, 57, 71, 75, 80, 90, 91, 88, 78, 82, 70
210 DATA  77, 86, 45, 38, 44, 46, 83, 82, 79, 85, 68
220 DATA  93, 92, 100, 97, 85, 73, 80, 78, 83, 57, -1
230  END
```

执行结果如下：

```
RUN:   0--9   0
      10--19  0
      20--29  0
      30--39  1
      40--49  3
      50--59  2
      60--69  2
      70--79  8
      80--89  10
      90--99  5
      100     1
```

例（3） 利用随机函数产生一组10个20～30之间的随机整数，并选出其中最大的数。程序如下：

```
1   CLEAR
10  DIM  A (10)
20  FOR  I=1  TO  10
30  RANDOM
40  A (I) =20+RND  10
50  PRINT "A (";  I;  ") =";  A (I)
60  NEXT  I
70  C=A (1)
80  FOR  I=2  TO  10
```

```
90  IF  A (I) <C  THEN  100
95  C=A (I)
100 NEXT  I
110 PRINT  C
120 END
RUN:  A (1) =26
      A (2) =24
      A (3) =24
      A (4) =26
      A (5) =28
      A (6) =26
      A (7) =29
      A (8) =25
      A (9) =24
      A (10) =27
        29
```

如果重新运行一次，可得到不同的结果。

第十一节　双下标变量

在第十节例（2）统计学生的成绩中，只统计了某一门功课的成绩，如果不是一门功课，而是几门功课，例如：语文、数学、物理三门功课，用一维数组就不太方便，而用二维组数就要简单方便得多。

我们列出8个学生的成绩表：

	语文	数学	物理
1号学生	83	92	79
2号学生	54	85	78
3号学生	86	90	82
4号学生	76	82	67
5号学生	77	89	92
6号学生	95	83	90
7号学生	100	87	92
8号学生	76	65	58

如果用二维数组来表示，可以很方便而且清楚得多。

将语文作为第一列的成绩，将数学和物理分别作为第2列与第3列成绩。把学生号作为行号，第3行第二列的成绩90分就是3号学生的数学成绩，即A(3，2)=90，由此，类推，A(4，1)=76，A(6，2)=83等等。

这里，下标变量带有两个下标，如A（6，2），这种带有两个下标的变量我们称为双下标变量。

一、双下标变量的形式及使用规则

双下标变量的形式：

A (I, J)

A是双下标变量，它和单下标变量一样，可由26个字母中的一个组成，也可由两个组成，也可由一个字母带一个数字组成。

I与J是下标，两个下标之间一定要用逗号分开。下标可以用数值，也可用变量及算术表达示。

第一个下标表示二维数组的行数，第二个下标表示数组的列数，如A（2，3），表示A数组第二行第三列的元素，A (I, J) 表示第I行第J列的元素，A（0，4）表示第0行第四列的元素。

二、二维数组

我们把带两个下标的变量所组成的数组称之为二维数组，如上所述，第一个下标表示行数，第二个表示列数。

例(1) 我们写出下面一个程序，使得程序按方程$X=\sqrt{Y+\sqrt{A}}$（其中Y = 0，1，2，3；A = 0，1，2，3，4）得到一张数表，让它们按4行5列排列。（PC—1500的打印机是窄行打印机，所以让它们按每行排成一段来打印）。

```
10  DIM  S (3, 4)
20  FOR  Y = 0  TO  3
30  FOR  A = 0  TO  4
40  S (Y, A)  =√(Y+√A)
50  NEXT  A : NEXT Y
60  FOR  Y = 0  TO  3
70  LPRINT "Y =", Y
80  FOR  A = 0  TO  4
90  S (Y, A) = INT(S(Y, A) * 100 + 0.5)/100
100 LPRINT "A =", A; "S(", Y; A; ") =", S (Y, A)
110 NEXTA : NEXT  Y
120 END
```

运行结果打印如下：

```
RUN: Y = 0
     A = 0   S( 0   0 ) = 0
     A = 1   S( 0   1 ) = 1
     A = 2   S( 0   2 ) = 1.19
     A = 3   S( 0   3 ) = 1.32
     A = 4   S( 0   4 ) = 1.41
```

```
Y = 1
A = 0   S( 1   0 ) = 1
A = 1   S( 1   1 ) = 1.41
A = 2   S( 1   2 ) = 1.55
A = 3   S( 1   3 ) = 1.65
A = 4   S( 1   4 ) = 1.73
Y = 2
A = 0   S( 2   0 ) = 1.41
A = 1   S( 2   1 ) = 1.73
A = 2   S( 2   2 ) = 1.85
A = 3   S( 2   3 ) = 1.93
A = 4   S( 2   4 ) = 2
Y = 3
A = 0   S( 3   0 ) = 1.73
A = 1   S( 3   1 ) = 2
A = 2   S( 3   2 ) = 2.1
A = 3   S( 3   3 ) = 2.18
A = 4   S( 3   4 ) = 2.24
```

三、数组说明语句

与一维数组一样，对二维数组也必须进行说明，以便使机器开辟足够的单元来贮存各个数组元素。

二维数组的说明语句形式是：

〈语句号〉 DIM A(I, J)

式中的各字母的含义前面已介绍，在此不复述。

在PC－1500机上，I与J的最大界偶均为255。（须注意：I与J的界偶虽然均为255，并不是说机器有256×256个数据单元供用户使用，至于在什么情况下I与J的范围是多少还决定于机器内存容量的多少。如PC—1500机装有8K模块的，它的内存共有10042字节供用户用，每个数据单元占用8个字节，共有10042÷8＝1255个单元。也就是说I与J的乘积，不得大于1255，否则内存溢出。这还只是一种极端情况，实际上，程序还要占用内存，只有在扣去程序占有的内存数后，计算余下的内存，才能较为准确地算出I与J的取值大小。）

二维数组的说明语句的使用规则与一维数组一样，这里不重复。

四、程序举例

例（2）

设矩阵：$A=\begin{vmatrix} 7 & 3 \\ 8 & 5 \\ 6 & 4 \\ 2 & 8 \\ 4 & 5 \end{vmatrix}$，$B=\begin{vmatrix} 5 & 4 & 9 & 7 \\ 3 & 8 & 1 & 4 \end{vmatrix}$，求矩阵A与矩阵B乘积。

根据矩阵乘法定义：A×B=C

若：A为 N1×M1矩阵，B为 N2×M2矩阵，且N2=M1，

则：C矩阵为 N1×M2矩阵

$$C_{ij}=\sum_{k=1}^{M1}a_{ik}\cdot b_{kj} \qquad (1)$$

式中：i=1，2，……N1

j=1，2，……N2

对于本题的A和B，N1=5，M1=2

N2=2，M2=4

我们可以对式（1）展开如下，（按列排列）

$$C_{11}=a_{11}\times b_{11}+a_{12}\times b_{21}$$
$$C_{21}=a_{21}\times b_{11}+a_{22}\times b_{21}$$
$$C_{31}=a_{31}\times b_{11}+a_{32}\times b_{21}$$
$$C_{41}=a_{41}\times b_{11}+a_{42}\times b_{21}$$
$$C_{51}=a_{51}\times b_{11}+a_{52}\times b_{21}$$

$$C_{12}=a_{11}\times b_{12}+a_{12}\times b_{22}$$
$$C_{22}=a_{21}\times b_{12}+a_{22}\times b_{22}$$
$$C_{32}=a_{31}\times b_{12}+a_{32}\times b_{22}$$
$$C_{42}=a_{41}\times b_{12}+a_{42}\times b_{22}$$
$$C_{52}=a_{51}\times b_{12}+a_{52}\times b_{22}$$

$$C_{13}=a_{11}\times b_{13}+a_{12}\times b_{23}$$
$$C_{23}=a_{21}\times b_{13}+a_{22}\times b_{23}$$
$$C_{33}=a_{31}\times b_{13}+a_{32}\times b_{23}$$
$$C_{43}=a_{41}\times b_{13}+a_{42}\times b_{23}$$
$$C_{53}=a_{51}\times b_{13}+a_{52}\times b_{23}$$

$$C_{14}=a_{11}\times b_{14}+a_{12}\times b_{24}$$
$$C_{24}=a_{21}\times b_{14}+a_{22}\times b_{24}$$
$$C_{34}=a_{31}\times b_{14}+a_{32}\times b_{24}$$
$$C_{44}=a_{41}\times b_{14}+a_{42}\times b_{24}$$
$$C_{54}=a_{51}\times b_{14}+a_{52}\times b_{24}$$

根据以上数学分析，编制A×B的程序如下：

```
1  CLEAR
5  INPUT"N1=";  N1
6  INPUT"M1=";  M1
```

```
 8  INPUT"M2=", M2
 9  N2=M1
10  DIN A(N1, M1), B(N2, M2), C(N1, M2)
20  FOR  I=1  TO  N1
30  FOR  J=1  TO  M1
40  READ  A(I, J)
45  LPRINT"A(", I, J, ")=", A(I, J)
50  NEXT  J: NEXT  I
55  LPRINT
60  FOR  J=1  TO  N2
70  FOR  I=1  TO  M2
80  READ  B(J, I)
85  LPRINT"B(", I, J, ")=", B(I, J)
90  NEXT  I: NEXT  J
95  LPRINT
100 FOR  I=1  TO  N1
110 FOR  F=1  TO  M2
120 FOR  J=1  TO  N2
130 R=A(I, J)*B(J, F)
140 C(I, F)=C(I, F)+R
150 NEXT  J:NEXT  F:NEXT  I
160 FOR  E=1  TO  M2
170 FOR  K=1  TO  N1
180 LPRINT"C(", K, E, ")=", C(K, E)
190 NEXT  K
200 LPRINT:NEXT  E
300 DATA  7, 3, 8, 5, 6, 4, 2, 8, 4, 5
305 DATA  5, 4, 9, 7, 3, 9, 1, 4
310 END
          STATUS  1  465
```

说明：（1）语句1为CLEAR，为了对机器清零

（2）5～8语句，输入两矩阵的行数与列数

N1，M1为A的行与列数

N2，M2为B的行与列数

（3）10语句是数组说明语句，请注意C矩阵的行与列与A及B的关系

（4）20～50语句，输入A矩阵各元素数值

60～90语句，输入B矩阵各元素数值

45与85语句是打印A，B的元素，这两句是为了了解输入单元的情况的。

此两句也可以不要，不影响计算。

（5）A矩阵的数据放在300DATA之中。

B矩阵的数据放在305DATA之中。

对矩阵各元素在DATA的排列顺序是先行后列，千万不能排错，否则计算要出错。所以前面的45语句与85语句的设置还是很有用的。

（6）100～150语句计算语句，160～200打印C的成果。

注意，计算结果打印是按C的列分段打印的。如果用宽行打印机，也可按行排印，程序只须小作修改。

例（3） 已知某电站3年中各月平均出力如下所列，编出将其大小按顺序排列的程序

第一年	第二年	第三年
($N_1=12$)	($N_2=12$)	($N_3=12$)
75.7	108.5	87.3
85.8	161.4	128.5
91.1	160.3	140.3
87.3	189.3	158.7
54.6	167.5	146.4
40.2	67.4	66.8
51.7	60.3	57.6
56.3	64.7	63.3
54.2	62.7	61.4
52.1	59.0	58.1
47.6	52.0	53.0
40.0	52.0	56.1

$N=12+12+12=36$

方法（1）：用交换分类法编程序。交换分类法是将相邻的两个数比较，如顺序不是递减的，将两数交换，若是递减的，则不动。然后第二个和第三个再比，如此类推，直至最大数换到最前面即可。程序如下，N为需排的数的个数。

```
 10  INPUT"N="; N
 20  DIM  A (N)
 30  FOR I=1  TO N
 40  INPUT  A  (I)
 50  PAUSE"A ("; I; ") ="; A(I)
 60  NEXT  I
 70  C=1
120  FOR  D=1  TO  N-1
130  E=A  (C)
```

```
140  FOR  F=1  TO  N-1
150  C=C+1
160  IF  E>A (C)  THEN  180
170  A (C-1) =A (C), A (C) =E
180  E=A (C)
190  NEXT  F
200  C=1 :NEXT  D
210  FOR  C=1  TO  N
220  LPRINT"NO:"; C; "---"; A (C)
230  NEXT  C
240  BEEP  3 :END
```

方法（2）：用定位法编程序：

先假设第一个数是最大数，然后用其他各数与它相比较，哪个大就和它交换位置，继续比较，再发现大的数再和它交换位置，循环一次后，第一个位置上的数必是最大的数，反复循环，直至把数从大到小排列出来（从小到大排列方法也如此）。

```
 5  INPUT"N="; N
10  DIM  A (N)
30  FOR  I=1  TO  N
40  INPUT  A (I)
50  PRINT"A ("; I; ") ="; A (I)
60  NEXT  I
120  FOR  C=1  TO  N-1
120  FOR  D=C+1  TO  N
140  IF  A (C) >A (D)  THEN  160
150  E=A (C), A (C) =A (D), A (D) =E
160  NEXT  D :NEXT  C
170  FOR  C=1  TO  N
180  LPRINT  C; "--"; A (C)
190  NEXT  C
200  BEEP  5 :END
```

这两种不同的编写法，可通过机器的运算看看其差异如何，特别是在运算时间方面的差异。

在这两个程序中，输出语句PRINT改用了LPRINT，这是为了让输出数值在打印机打出，以便观察。使用了LPRINT语句，必须要装上打印机才能运行程序。

习　题　五

一、假设已经把数值分配给以下变量：

A	2
A1	3
B	1
C	4

X(1)	57
X(2)	42
X(3)	18
X(4)	98
X(5)	76
X(6)	6
X(7)	14

写出下面下标变量的数值

（1）X（3）=_____　　（2）X(A+A1)=_____

（3）X(X(A+C)-A1)=_____　　（4）X(C)=_____

（5）X(7)=_____　　（6）X(A*A1)=_____

（7）X(X(A1)-X(A+C)-(A*C))=_____

二、判断以下程序执行后将会出现什么结果。（先判断，再上机）

```
（1）10  FOR  I = 1  TO  5
     20  DIM  A(I)
     30  A(I) = I
     40  PRINT  A(I)
     50  NEXT  I : END
```

```
（2）10  DIM  C(20)
     20  FOR  I=10  TO  30
     30  LET  C(I) = I
     40  PRINT  C(I)
     50  NEXT  I : END
```

三、给出以下DATA语句：

DATA　10, 11, 6, 15, 30, 21, 6

利用下标变量写一个程序，使得按以下顺序打印出结果：

6, 21, 30, 15, 6, 11, 10

四、写一个程序，把A数组改写成B数组并打印出来。

A

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

B

$$\begin{pmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \\ 4 & 8 \end{pmatrix}$$

五、利用双下标变量写一个程序，做以下矩阵的相加和相减。

A矩阵

$$\begin{vmatrix} 101 & 102 & 103 & 104 \\ 201 & 202 & 203 & 204 \end{vmatrix}$$

B矩阵

$$\begin{vmatrix} 4 & 5 & 6 & 7 \\ 3 & 2 & 5 & 6 \end{vmatrix}$$

六、求下面矩阵的逆

$$\begin{vmatrix} 3 & 2 & 1 \\ 1 & 1 & 1 \\ 2 & 3 & -1 \end{vmatrix}$$

第十二节　其他功能语句和指令

一、停止程序运行的语句和指令

一个程序在运行之中可以有许多的方法使其停止、暂停或中断，而这些方法又各有其特点和用途。

（一）END结束指令：这种指令是编在程序之中的，前面已作过介绍。在程序之中它不一定在程序的最后。也可能在中间，当程序运行到此时，就停止下来，不再继续执行以后的语句。如果还想执行下面的程序，一定要用启动程序的指令如：RUN（语句号），GOTO（语句号）或DEF方式的标号。

（二）正常中断指令：程序的中断可分成正常中断、强制中断二类，正常中断的语句有INPUT、PRINT。

（1）INPUT：INPUT是问答式输入指令，它在程序之中起接受输入数据和字符串的作用。但它本身有中断功能，只有当输入数据字符串之后再按ENTER键，程序才开始继续运行，（不输入任何内容，仅按ENTER键也可，只相当不作新的输入）。

（2）PRINT：显示屏输出指令，当程序运行到它时，即按程序的要求显示所输出的内容，在显示时程序中断。但程序并没有结束，只须按ENTER键，程序继续运行。

如果在PRINT以下没有其它语句，也可以结束程序。

（三）强制中断：属于强制中断的指令有：

（1）STOP中断：STOP中断指令是编入程序内的指令，当程序运行至STOP时即中断，并显示〔BREAK IN〈语句号〉〕，不显示运算结果，但可以在中断期间检查存贮器内存贮情况。如果使程序继续运行需用CONT“继续运行”指令。

（2）ON键：ON键除开机作用外还有中断功能，它不编入程序。在程序进行运行时，按ON键即可使程序中断并显示〔BREAK IN〈语句号〉〕，但它不能选择中断位置。如需使程序继续运行则同样用CONT指令。

（3）PAUSE暂停指令：PAUSE是编入程序的，它的作用是间断0.85秒显示一个计算结果。我们也可以将它编在程序中间的某部分，让它显示我们所需了解的中间成果。此时程序运行中断0.85秒，然后自动继续运行。

（4）WAIT等待指令：WAIT可以用语句形式编入程序之中，程序运行到WAIT语句后的PAUSE或PRINT指令时，便停止下来，等待一定时间再自动继续运行，等待的时间长短由WAIT所给出的数值所决定。如WAIT64约为一秒，WAIT128为二秒，而WAIT 0为不等待，由此类推。有关WAIT的用法，第四章输入格式一节中还要详细介绍。

二、启动程序的指令与操作

启动程序运行，只能在RUN方式下进行。

（一）RUN：程序运行指令。我们已经很多次地使用过它了。RUN是键盘操作指令，除我们已知的从头运算起以外，也可在RUN后加上语句号，让程序从我们指定的语句开始运行。

（二） ENTER执行键：它除对语句以及指令负输入机器并执行的任务外，在程序正常中断时，按它可以使程序从正常中断处开始继续运行。

（三） CONT继续启动指令：强制中断后用CONT指令继续启动。

（四） GOTO指令：GOTO指令也可以用来启动程序，它作启动指令用时，其用法与RUN指令一样。

（五） ↓键：↓键在机器介绍一章中，我们称之为下行变行键。它也可以用来启动运行程序，不过它不能象RUN或GOTO指令那样从头开始启动程序，只能在程序中断时来启动程序，而且，每按一次，程序向下执行一句，按住不放，程序就连续运行，一松开它程序运行又停止下来。

三、清除存贮器语句和指令

（一）CLEAR清除指令：它专用于清除存贮器中的数值及字符串。CLEAR可以编入程序，也可在程序外手动操作，在PRO与RUN方式下均可。

（二）NEW清除指令：只能在程序外手动操作。它可清除全部存储器、数值、程序或保留函数。它只能在PRO与RESERVE方式执行。

（三）CLS语句指令：CLS指令是作为语句的形式放入程序之中的清除显示屏的指令。类似CLEAR在程序中的用法。它只能清除显示屏上的显示，不能清除存贮器的内容。

CLS清屏语句指令，可利用它编入程序之中，供我们设计一些显示格式。例如在下面一段程序中，结合WAIT命令输入数组时，将数组元素的行号列号显示出来，以作为提示符之用。这是一种常用的方法。

```
10  CLEAR : WAIT  0
20  DIM  A(10)
30  FOR  I = 1  TO 10
40  PRINT "A("; I; ") =";
50  INPUT  A(I) : CLS
60  NEXT  I : WAIT
      ⋮
（其他运算语句）
```

输入数据的程序段

（四） CL 和 ON键。按此二键中的任一个均可清除显示器显示。不能编入程序，也不能清除存贮器内容。

四、检查存储指令： 可（在PRO或RUN方式）键盘手动操作，也可编入程序内，一般作键盘命令使用。

（一）MEM——在显示器中显示未占用的内存字节数。

（二）STATUS 1——在显示器中显示已占用的内存字节数。

（三）STATUS 0——同MEM功能。

五、运算方式锁定指令：

（一）LOCK——运算方式锁定指令。在程序外输入或程序内编入均可。

当LOCK送入机器时，对方式选择MODE键即锁定，不能再改变机器的方式。

（二）UNLOCK——与LOCK相反，解除锁定。

这两个指令的提供，可以防止一些误动作而给操作带来方便。

六、REM注释语句：

REM语句在程序中只起注解作用，没有任何操作功能，在程序运行时遇到注释语句即跳过执行下面的内容，在BASIC中，称之为非执行语句。它也同样被机器存贮，开列清单打印时，原样打印，但它同样要占内存。REM后一般附说明，如：

```
10  REM  "MAIN  PROGRAM"（主程序）
```

七、TIME时间指令（或叫时间函数）

在PC—1500中，有一频率为32.768KHZ的实时钟，我们可以用下列形式送入月、日、时、分、秒，以后可随时调出查时。

送入机器形式：TIME = MMDDHH . mmss ENTER

这就是校正时间的方法。

上式中，MM——月份的两位数。

DD——日子的两位数。

HH——小时的两位数。

mm——分钟的两位数。

ss——秒数的两位数。

调问形式：TIME ENTER

TIME除程序外操作以外，程序内编入也可。

例（1） （1）将日期时间2月17日18时34分16秒送入机器。

TIME = 21718.3416 ENTER

注意，除月可用一位数以外，其余必须是二位数。

（2）查问时间。

TIME ENTER 显示〔 21508.3107〕

表示 2 月15日 8 时31分 7 秒。

本例的（1）的操作，实际上也就是开钟或校时的操作。

TIME时间指令不仅给我们提供了一个计时的电子钟，我们还可以将它编入程序之中用以计算程序的运行时间。例如下面的例子就是一个记录程序开始到结束运行时间的简单方法。

```
 1  CLEAR
 2  T1 = TIME
10  INPUT…
      ⋮          } 某一运算的程序段
      ⋮
500  ……
510  T2 = TIME
520  LPRINT  T1, T2
530  END
```

此示意的程序中的第10～500语句是某一程序，在这里且省略它。而在此前加入了语句：2 T1 = TIME将开始时间记录下来，在500句程序完毕后加入510句记录结束的时间，然后由第520句打印出来。当然还可作一些更多的处理，大家可自己试试。

TIME指令有很多有用的用途，它提供的计时也是非常准确的，我们还可用来作一些控制之用，如生活之中的闹钟就是一例。下面例（3）就是一简单的闹钟程序。

八、音BEEP音响指令与BEEP ON和BEEP OFF音响开关指令

（一）BEEP音响指令：

BEEP音响指令为我们提供一种新的方式，引起使用者的注意或对操作者提供一种信息。我们在前面的一些例中，已多次使用过它。

它的形式是：BEEP 〈A〉，〈B〉，〈C〉

〈A〉，〈B〉，〈C〉可为具体数值，也可为表达式。其意义与范围为：

〈A〉——音响次数。 $0 < A \leqslant 65535$

〈B〉——音响频率。 $0 < B \leqslant 255$相应的频率为$230H_z \sim 7KH_z$

〈C〉——音响长短。 $0 < C \leqslant 65279$

BEEP可在程序外手操作，也可编入程序之内，为我们提供某种信息，如计算完毕，循环几次等等。音响频率与长短也可以不写，也就是说〈B〉与〈C〉可省略。BEEP后如只有一个数，机器认为是音响次数；如有两个数，机器认为一个是音响次数，一个是频率。

有趣的是，因为我们可以对音响的频率、次数、长短进行选择也就可以用它来编出一些简单的音乐，下面就是一例。

例（2） 10 CLEAR

```
 20  READ  A
 30  FOR  I=1TO  A
 40  READ  B, C
 50  BEEP  1, B, (60*C)
 60  NEXT  I:END
100  DATA  44, 245, 1, 245, 1, 160, 1, 160, 1
110  DATA  145, 1.2, 145, 1.2, 160, 2
120  DATA  180, 1, 180, 1, 193, 1, 193, 1
130  DATA  220, 1, 220, 1, 245, 2
140  DATA  160, 1, 160, 1, 180, 1, 180, 1,
           193, 1, 193, 1, 220, 2
150  DATA  160, 1, 160, 1, 180, 1, 180, 1,
           193, 1, 193, 1, 220, 2,
160  DATA  245, 0.3, 220, 0.3, 200, 0.5, 180, 0.5,
           160, 1.5, 118, 1, 145, 1.5, 118, 1.5, 160, 3
170  DATA  180, 1, 180, 1, 193, 1, 193, 1
180  DATA  220, 1, 220, 1, 245, 2
```

程序运行时，奏出一段乐曲。

例（3） 结合时间函数TIME编一闹钟程序

```
  5  WAIT  0
 10  INPUT  "ALARM  TIME=?";  TI
 15  TI=TI*1E4
 20  A=TIME:K=INT(A/100)
 30  A=(A-K*100)*1E4
 40  B=TIME:K=1NT(B/100)
 50  B=(B-B*100)*1E4
 60  IF  A=B  GOTO  40
 70  C=INT (B/1E4)
 80  M=INT ((B-C*1E4)/100)
 90  S=INT(B/100):S=B-S*100
100  PRINT "␣␣␣␣";  C;  "h";  M;  "m";  S;  "s"
110  IF  TI=B  THEN  BEEP  30:END
120  A=B:GOTO  40
```

（二）音响开关指令 BEEP ON与BEEP OFF

BEEP ON 打开音响开关指令。它也可作为语句形式放入程序之中使用。机器在出厂时音响开关已经是打开的，因此只有音响开关被关闭以后需要重新打开时才用它。

BEEP OFF 关闭音响开关指令。它被执行后音响开关即被关闭，不能再发声。它同样可作为语句编入程序之中。

九、程序举例

例（4） 求任意解析函数f(x)在x＝a（a为实数）处的微分近似值

我们用公式： 微商＝$\frac{f(x)-f(a)}{x-a}$，其中a为定点

x＝a＋(0.5)^n，n为逐次逼近的次数，取n＝10

$$\left.\frac{df(x)}{dx}\right|_{x=a} \approx 2 \times \text{第十次微商} - \text{第九次微商}$$

以方程f(x)＝x^5＋x^3＋x－5为例，求当x＝1时的微分近似值。

先将方程存在保留函数之中：

〔RESERVE〕方式 FI 中存入 X∧5＋X∧3＋X－5

程序如下：

```
10  REM "PROGRAM OF DERIVATIVE"
20  INPUT "ENTER VALUE OF X="; A
30  X=A:B= FI
40  FOR  N=1  TO  10
50  C=D
60  X=A+0.5∧N
70  E= FI
80  D=(E-B)/(X-A)
90  NEXT  N
100 F=2*D-C
110 PRINT "DERIVATIVE  AT  X="; A
120 PRINT "IS"; F
130 BEEP 3, 100
140 END
```

例中，第10语句使用了REM语句，注释说明本程序是求导源程序。第20句是输入，程序在此处停止等待输入，当输入X的值A后（X＝1），程序即继续运行，直到显示〔DERIVATIVE AT X＝1〕。再按 ENTER 键，显示〔IS 8.999978994〕。

当然，对此数可以用四舍五入法取数，如取四位小数只需将第100句改为：

```
100 F=2*D-C:F=INT(F*10000+0.5)/10000
```

显示结果为〔DERIVATIVE AT X＝1〕，再按执行键显示〔IS 9 〕

此程序经过10次循环（N＝10），如果要检查每次循环的D值（即微商），可在80与90句之间加入强制中断语句：

```
85 STOP
```

此时每计算一次D值，程序中断一次，中断后再用CONT指令继续运行程序。在中断时，可以检查每次微商的D程，（只需按 D ENTER）。

第1次 D＝18.9375

第2次 D＝13.01953125

第 3 次　D = 10.80688476
第 4 次　D = 9.856704704
第 5 次　D = 9.417145728
第 6 次　D = 9.205829696
第 7 次　D = 9.102236288
第 8 次　D = 9.050949376
第 9 次　D = 9.025432576
第10次　D = 9.012705785

中断的方式如用 ON 键，我们发现每次中断所显示的中断地址不一样。

下面各表列出运行程序中断与启动的一些指令的比较情况：

停止指令功能比较　　表 3—4

比较项目 \ 指令名	END	PRINT	INPUT	STOP	ON	PAUSE
1.编在程序内	可以	可以	可以	可以	/	可以
2.程序外键盘操作	/	可以	/	/	可以	/
3.结束程序	可以	/	/	/	/	/
4.正常中断程序	/	可以	可以	/	/	/
5.强制中断程序	/	/	/	可以	可以	/
6.用 ENTER再启动	/	可以	可以	/	/	/
7.用CONT再启动	/	/	/	可以	可以	/
8.停止打印机磁带机	/	/	/	/	可以	/
9.暂停0.85秒	/	/	/	/	/	可以

启动指令功能比较　　表 3—5

比较项目 \ 指令种类	RUN	GOTO	ENTER 键	CONT	自定义标号键
1.从头启动程序	可以	可以	/	/	可以
2.指定语句启动	可以	可以	/	/	可以
3.正常中断启动	/	/	可以	/	/
4.强制中断启动	/	/	/	可以	/
5.启动一段程序	/	/	/	/	可以

不同启动程序方法功能比较　　表3—6

功能 \ 启动方式	RUN	GOTO	DEF方式
显示屏被清除	是	是	否
运行“指针”回到第一句	是	否	否
WAIT被置于∞	是	否	否
调试命令取消	否	否	否
固定存储器被清除	否	否	否
数据区被清除	是	否	否
FOR～NEXT与GOSUB堆栈清除	是	是	是
ON ERROR GOTO被取消	是	否	否
READ用的DATA指针复原	是	否	否
USING 格式取消	是	否	否

第十三节　程序的编辑和调试

我们通过前面的学习，已基本上掌握了BASIC语言在PC—1500机上的用法，并能设计和编写源程序，但是我们也发现，编好的源程序在送入计算机进行运算时，常常不能立即成功，或者计算运行中出错，或者结果不合理。

这类情况，往往是程序中某些地方有不符合计算机规定所造成的。例如，错用了标点符号，错用了标识符，以及一些其他原因造成。所以，在程序设计完，送源程序入计算机之后，必须对程序进行调试。这是必不可少的一步。

调试程序就是检查新编程序能否与是否按我们预期效果运行，如果不能，则要找出为什么不能按编制意图正确工作的原因，改正其中的错误与不合理的地方，使程序完善，达到我们所需的计算目的。

PC—1500机有较强的调试功能，调试程序是较为方便的，我们可通过键盘操作一些指令来进行。

一、调试和编辑的指令及键

1．调出语句的键：　[↓]　[↑]　变行键

变行键一是上行键 [↑]，一是下行键 [↓]。主要用来调出程序，(变行键的其他用途另述)。程序送入机器内存以后，因为某些原因（如修改）需要调出观察时，可在PRO方式之下，按变行键即可。按下行键时，显示屏上显示程序最开始的一句语句；按上行键时，显示最末一句程序。以后每按一次变行键，显示程序语句递增一句或递减一句。如果按住它不

放，则程序的显示约按每秒十句的速度递增或递减地跳动。放开变行键，程序上下变行即行停止。

PC—1500机的显示屏一次只能显示一条语句，所以变行键在调程序时，就显得很重要也很方便。

变行键一般不能任意选定从某句开始，但有时按变行键，显示的语句不是最前一句或最末一句，这往往是程序曾运行过，由于某些原因其运行“指针”停在某句的缘故。

2. 目录指令：LIST

目录指令 LIST 是一键盘命令，它是调出源程序的指令。与变行键的不同之处在于，它可以调出我们指定的某一语句。其形式为：

LIST〈语句号〉

当键盘送入上面形式的命令以后，显示屏就显示出TIST指令后指定的那条语句。

如果在源程序之中无此语句，例如送入了LIST 70，而程序之中无70语句，那么则显示紧跟70语句以后的那一句。如果在LIST命令后无语句号，则显示程序最头一句语句。

LIST指令可以在大量程序之中立即找到我们所选定的语句，所以比变行键有其灵活方便之处。

LIST指令只能在PRO方式之下执行。如果方式用错则显示〔ERROR 26〕。

3. 选定修改位置的光标键（编辑键）：[◀] 和 [▶]

修改程序语句内容可用光标键找到要改的位置进行程序的增添、删除、修改等操作。修改办法在第二章已介绍过，在此不重复了。

4. 调试指令：TRON和TROFF

命令机器进入调试状态或解除调试状态的指令，有下面两种：

TRON——程序调试状态指令。

TROFF——解除调试状态指令。

TRON可在程序外键盘操作。当依次送入TRON与RUN指令后，程序即进入调试状态。此时程序不是连续执行，而是依语句号逐句执行。按 [▼] 一次，执行一句，显示屏上显示被执行的语句号。若按住 [▼] 不放，程序则连续运行，松开运行即停。只能顺序执行，不能逆行。所以，有的书上称之为“跟踪指令”。

TRON也可编入程序，可编在最前面，也可编在所需调试的那段程序前面。

TROFF是解除调试指令，在程序外与程序内均可。只有机器接受了TROFF指令后，才能解除调试状态，恢复正常状态。

有了TRON指令及试调状态运行，我们可以在程序运行到任意的语句时，按需调出变量进行观察，也可以随时改变变量的数值。然后再继续用变行键向下调试程序。要注意的是，在调试过程中，如遇到需要输入数据或字符串时，要按正常运行规定的操作输入，不能例外。

除了调试状态以外，在程序中断状态下，也可以使用同样的方法一步一步检查程序。

二、程序的编辑

1. 删除语句：在PRO方式之下，给出所需删除的语句号，按ENTER键即可。

2. 增添插入语句：在PRO方式之下，给出插入位置语句号，按正常情况编写语句送

入。

3. 交换语句号：在PRO方式之下进行。交换语句号不能把两个语句号直接交换。例如第40句与50句号交换：第一步将第40号改为45号(45号必须是程序之中原没有一条语句号)，第二步将50号改为40号，第三步将45号改为50号。切记最后要将45号删除。否则45号与50号的内容完全相同，并且同时存在。

三、错误信息

PC—1500机规定：当程序在运算或操作中，如果不符合其规定，则自动停机并显示指误信息，显示的格式如下：

〔ERROR 〈错误类号〉 IN 〈语句号〉 〕

或 〔ERROR 〈错误类号〉 〕

这个错误信息中，〈错误类号〉是指示的错误性质。具体是何种错误可查看附录三《错误信息表》。〈语句号〉指示的是错误地址。

例如：〔ERROR 7 IN 20〕表示第20语句有第7类错误。

还可以更进一步查清错误发生的确切位置。方法如下：先用 CL 键清除错误信息显示，再按上行键 ↑ ，于是发生错误的语句显示在显示屏上，并且有一光标在错误处闪烁。

操作类错误不显示地址，光显示错误性质，如：

〔ERROR 26 〕表示第26类错误

四、调试程序举例

用牛顿迭代法求f (X) = 8800X⁵ − 2.31X − 5.115的根。

设X的初值为1，允许误差ε＜0.01

牛顿迭代法求根公式为：

$$X_{n+1} = X_n - \frac{f(X_n)}{f'(X_n)}$$

当 $|X_{n+1} - X_n| \leqslant \varepsilon$ 时，即为 f (x) 的解。

设计框图如图2—24所示。

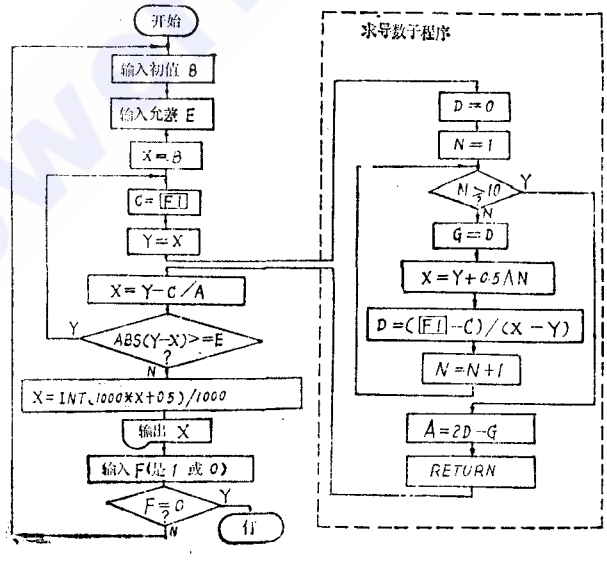


图 3—24

因为利用保留函数键存入原方程，所以框图及程序中F1表示原方程。

〔RESERVE F1 8800 * X ∧ 5 − 2.31 * X − 5.115〕

编写程序如下：

```
10  INPUT "X=";B,"TOERANCE=";E
20  X=B
30  C=F1
40  Y=X
50  GOSUB 500
```

```
 60  X=Y-C/A
 70  IF  ABS  (Y-X) >=E  THEN  30
 80  X=INT (1000*X+0.5) /1000
 90  PRINT "ONE  ROOT  IS"; X
100  INPUT "MORE  ROOT?  (YES=1, NO=0) "; F
110  IF  F=0  THEN  130
120  GOTO  10
130  END
500  D=0
510  FOR  N=1  TO  10
520  G=D
530  X=Y+0.5^N
540  D= ( ([F1]) -C) / (X-Y)
550  NEXT  N
560  A=2*D-G
570  RETURN
```

为了调试方便，在输入程序时，加入下面两句：

```
545  STOP
565  STOP
```

我们通过此例来练习程序的调试，具体调试步骤如下：

1. 由键盘送入TRON调试指令，然后再启动程序，显示〔X= 〕，输入X的初值1，然后按 [↓] 键，显示〔TOLERANCE=〕，输入0.01；再按 [↓] 显示语句号10，表示第10句执行完毕。调出B与E分别为1与0.01，证明输入无误。

2. 继续按 [↓] ，显示20，30。此时可调出C值，为8792.575。为了验证是否正确可作如下计算：调 [F1] ，按执行键得数为8792.575，证明无误。

3. 继续按 [↓] ，显示40，50，500，证明转子程序无误。510句，520句，530句时，检查X：X=1.5。540句，调D为116047.69。验算如下：([F1] -8792.575) / (1.5-1) ，结果为116047.69，证明无误。

4. 续继按 [↓] 显示〔BREAK IN 545〕表示停在545句。再按 [↓] ，550句，520句，说明循环无误，调G为116047.69；再至530句，调X为1.25；540句，545句，D为72219.565，验算D无误。此时证明循环程序正确。下面只要每当显示〔BREAK IN 545〕时，调D检查即可。

5. 继续每一循环，当显示〔BREAK IN 545〕时调D，分别为：

56460.77594	49852.31646
46834.97866	45394.34273
44690.5821	44342.78535
44169.90097	44083.71132

6. 当10次循环完以后，550、560显示〔BREAK IN 565〕表示循环已完，子程序也将

要返回了，此时D＝44083.71132。再向下运行，570子程序返回，再按 [↓] 显示60句，证明子程序返回无误，检查X为8.001574937E－01。继续向下，70句，检查X－Y＝－0.1998425063。再向下80句，30句。程序回到了30句，进行第二轮计算。

7. 程序经过以上检查未发现不正常之处，下一步可以检查整个程序需要经过多少轮运算方能达到绝对误差小于允许误差。此时，调回PRO方式，删去545与565句，加入65 STOP再换回RUN方式，将TROFF解除调试方式指令送入，然后再用RUN30启动程序。

此时，程序运行到65句即停，显示〔BREAK IN 65〕检查X—Y为－0.1596668251。再用CONT指令启动程序，每中断一次检查一次X－Y，其值分别为：

－0.1272484239　　－0.1006624094
－0.0778993205　　－0.0565084905
－0.0340807987　　－0.0126481927
－0.0015675967

当X－Y＝－0.0015675967时，已经小于允许误差0.01，所以只经八轮运算即迭代出X的值来。

8. 再用CONT指令启动，显示〔 ONE ROOT IS 0.23〕表示一个根是0.23。

将0.23代入原方程验算：按 [F1] 显示原方程，再按执行键显示〔1.768184027E－02〕。因为是近似计算，故将0.23代入原方程计算不会为零。

9. 再按ENTER键，显示〔MORE ROOT？(YES＝1，NO＝0 〕问是否求第二个根，求则输入1，不求则输入0。

我们在此输入0，整个程序结束。

10. 最后将插入的65语句删除，正式运算。

以上是一个没有错误的程序的调试，我们可以人为地制造一些错误练习，如将70句的＞＝号改为＜＝号，将510句的1 TO 10改为1 TO 3等等。请大家自己试试。

调试程序，是非常重要而又必须熟练掌握的，上面这个例子还只就一些一般的方法作了一些介绍。其实调试程序还有很多技巧方面的问题。有时，发现并找出程序之中所存在的问题，并不是那么容易的事情，特别是有时既算不了结果，又不显示错误信息，往往使人无从下手进行调试。这时就需要从数学模型，计算方法，编写程序技巧方面去检查。当我们较为熟练地掌握了算法语言并有了一定的经验之后，对调试程序也就会得心应手起来的。