BASIC 语言入门

国防工业出版社

BASIC 语言入门

高惟龙 赵国林 谢燮坚 编

国防工业出版社

内 容 简 介

本书主要介绍分时BASIC语言及其解释执行方法。

全书共分十章，前七章介绍分时BASIC语言的语法、使用、系统生成、上机操作及应用实例。后三章介绍分时BASIC语言的解释执行方法。

本书可供有关专业的工人、科技人员学习和使用，也可作为非计算机专业的中学文化水平以上的读者学习BASIC语言的入门书。

BASIC语言入门

高惟龙　赵国林　谢燮坚　编

*

国防工业出版社 出版

北京市书刊出版业营业许可证出字第074号

新华书店北京发行所发行　　各地新华书店经售

国防工业出版社印刷厂印装

*

787×1092 1/16　印张5 5/8　124千字

1979年8月第一版　1979年8月第一次印刷　印数：00,001—95,000册

统一书号：15034·1828　定价：0.52元

前　言

BASIC 语言是一种易学、易写，而且具有会话性能的语言。对于具备中等以上文化水平的读者来说，只要经过短期学习就能掌握和使用这种语言。因此，采用这种语言有利于电子计算机的推广应用。采用 BASIC 语言编写程序可以减轻许多繁琐的工作，节省编写程序和查错、改错的时间，从而大大提高工作效率。

为了适应我国电子计算机事业的迅速发展，我们在 BASIC 语言的解释程序的编制 和推广方面做了一点工作。这里，我们将其编写成书，仅供参考。

本书介绍的是分时 BASIC 语言及其解释执行方法。全书共分十 章，前七 章介绍分时 BASIC 语言的语法、使用、系统生成、上机操作和应用实例。 后三章介绍分时 BASIC 语言的解释执行方法，取材于 DJS-154 机分时 BASIC 语言的解释程序。

本书可供有关专业的工人、科技人员使用时参考，并且可以作为非计算机专业的读者学习 BASIC 语言的入门书。

参加该书编写工作的有高惟龙、赵国林、谢燮坚同志，主要执笔人是高惟龙同志。除上述人员外，参加编制程序的人员还有张明珍、贾国忠、于秀柱等同志。在编写该书的过程中，受到了有关领导和同志们的大力支持和帮助，在此，谨致谢意。

由于我们经验不足，水平有限，书中可能会存在一些错误和缺点，恳请读者批评指正。

目　录

第一章　BASIC 语言概述

1.1　BASIC 语言的特点

BASIC 语言是一种具有会话性能的语言。程序员可以用键盘命令来中断一个正在执行中的程序，以便检查各变量的当前值，也可以修改变量的值。程序员还可以检查自己的程序（全部或一部分），并且可删除、插入或修改若干语句。当结束修改后，程序员可以在中断点或其它某个语句恢复程序的执行。由程序员所做的各项修改值和在中断过程中的未变化的变量值将是恢复运算时的初值。这就是 BASIC 语言具有的重要特点——会话性能。

BASIC 语言具有台式计算机运算的命令，能实现台式计算机的功能。

BASIC 语言的另一特点是便于掌握，使用方便。对于具有中学文化水平以上的工人、技术员只需经过几天的学习、训练就能掌握 BASIC 语言的基本要领，并解算自己的问题。这就是 BASIC 语言便于推广的重要原因。它适用于各种中、小工程计算和科学计算。

BASIC 语言从其执行方式来看是一种解释性语言，它可以在输入程序时检查并打印出某些程序的差错信息，同时还可在运算期间检查并打印出一些程序的差错信息。

BASIC 语言分为单用户、分时、扩充等三种类型。本书主要介绍分时 BASIC 语言。

1.2　分时 BASIC 语言的特点

分时 BASIC 语言，除了具有一般 BASIC 语言的功能外，还具有全部矩阵处理和字符行处理的能力。这在数据处理方面是很有用的。

分时 BASIC 语言的特点是允许每个用户同时使用中央处理机（实际上是分享中央处理机时间），这就可大大提高中央处理机的利用效率。每个用户使用情况如图 1-1 所示：

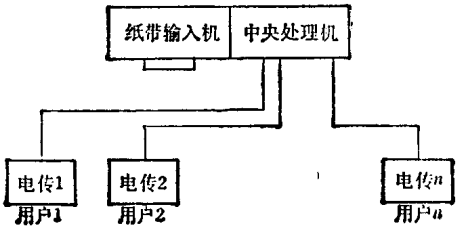


图1-1　分时BASIC 语言中各用户的使用情况

在分时 BASIC 语言中，可根据内存容量和需要建立具有若干个分时用户的系统。

1.3　BASIC 程序示例

BASIC 程序由一系列语句构成。每个语句前面有一个标号，标号是 1～9999之间的任

一整数，机器按标号由小到大的顺序执行各语句（当然也可以用某些语句来改变执行的顺序）。一般编程序时标号之间留一些间隔，便于补充语句。每个语句占一行（即以"回车"键为结束）。

下面以牛顿迭代法求解代数方程为例，说明如何用 BASIC 语言来编写程序。

求解　$f(x)=0$

设　$x=x_0+\Delta$

则　$f(x)=f(x_0)+\Delta\times f'(x_0)+\cdots=0$

$\Delta=-f(x_0)/f'(x_0)$

逐次迭代

$x_1=x_0-f(x_0)/f'(x_0)$

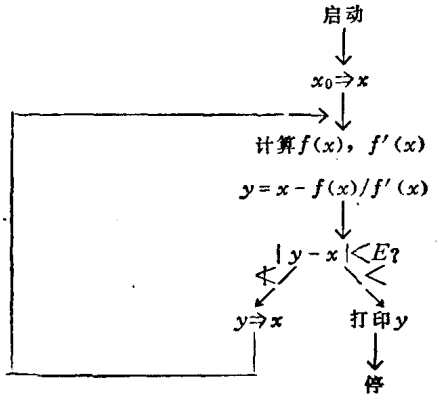
$x_2=x_1-f(x_1)/f'(x_1)$

……

$x_n=x_{n-1}-f(x_{n-1})/f'(x_{n-1})$

当两次迭代结果之差的绝对值 $|x_n-x_{n-1}|$ 满足要求误差，即求解完毕。

计算流程如下所示：



例如：用牛顿迭代法求方程 $6-5x+4x^2-3x^3=0$ 的一个实根，要求误差不大于 0.0001。

设初值 $x_0=1$

这时求得根的 BASIC 程序如下：

```
10  LET    X = 1
20  LET    A = 6 - 5 * X + 4 * X * X - 3 * X * X * X
30  LET    B = - 5 + 8 * X - 9 * X * X
40  LET    Y = X - A / B
50  IF     ABS( Y - X ). LT. .0001GOTO  80
60  LET    X = Y
70  GOTO   20
80  PRINT  Y
90  END
```

1.4 BASIC基本词法

BASIC语句用英文单词、运算符和运算对象组成。

1. 数

BASIC语言中一律采用十进制数。

输入的数允许7位有效数字。

输出的数只有6位有效数字。

数的范围为-3.5×10^{38}至$+3.5\times10^{38}$。

在输出时，任意可以用6位数字表示的数值直接用不带阶的形式表示，不能用6位数字表示的数则用带阶的形式表示（6位有效数字加上十的方次，即〈数符〉n.nnnnnE±r，其中，n为有效数字，r为阶码，可用1至2位数字表示。〈数符〉在正数时为空格，负数为－）。

例：

数	输出格式
－2000000	－2.00000E＋6
20000000000	2.00000E＋10
108.999	108.999
0.000001	0.000001
－0.0000256789	－2.56789E－5
5E＋3	5000

2. 简单变量

简单变量的名字可用一个字母，或一个字母后缀一个数字来表示。因此最多可以有286个简单变量。例如A，A3，X，Z9等是简单变量的名字，而6A，CD，Y15等不是简单变量的名字。

简单变量的值是一个数。

3. 数组和下标变量

数组的名字用一个字母表示。数组只有一维和二维两种。

下标变量是由数组名字和下标组成。

例：A(5)，B(3,6)

数组A是一维数组，数组B是二维数组。

数组的下标必须大于等于0。例如A(－2)，C(5，－3）等均是不允许的。

二维数组第一下标表示行，第二下标表示列，数组元素在内存中按行的次序排列。

例：数组C(2,2)的各数组元素的排列次序是：C(0，0)，C(0，1)，C(0，2)，C(1，0)，…，C(2，2)。

数组的下标是从0开始的。在数组定义时，指定了下标的最大值。

例：DIM A(M)，B(5，P)

上例通过DIM语句（定维语句）来定义数组A和数组B，此时数组A的下标为0～M，而数组B的第一下标为0～5，第二下标为0～P。因此，数组A中包含M＋1个数组元素，

而数组B中包含6×(P+1)个数组元素。

一维数组的数组元素个数不能超过256。

二维数组的数组元素个数不能超过1024，并且每个下标不能超过255。

BASIC语言中允许引用未定维的数组。在引用未定维的一维数组时，系统将此数组定维成11个元素的一维数组，而在引用未定维的二维数组时，系统将此数组定维成11行，11列的二维数组（121个元素）。

在BASIC语言中，允许对数组重新定维，但重新定维后的新数组包含的元素个数不能超过原数组。

例：

原来　　100　DIM　A(2, 3)

经100语句定维后，数组A的各元素的分布为：

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

重新定维　　200　DIM　A(3, 2)

经200语句重新定维后，数组A的各元素分布为：

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9
3	10	11	12

重新定维　　300　DIM　A(10)

经300语句再次重新定维后，数组A的各元素分布为：

0	1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10	11

重新定维　　400　DIM　A(3, 4)

由于语句400重新定维后的数组包含4×5=20个元素，而原来的数组只包含3×4=12个元素，因此新数组元素个数超过了原数组，这是不允许的，被认为是错误使用。

经过重新定维后，只改变各数组元素的下标式，并不改变原来各元素的值。重新定维后数组元素个数比原来少时，多余的工作单元不能作其他用，不能按原来方式引用，也不能解除对这些多余工作单元的分配。

4．行常量和行变量

行常量是由引号“′”括起来的除“′”外的任意字符串。例如′ABUD125′。需要注意的是在行常量内的所有空格都是有意义的。

行变量的名字用一个字母尾随一个# 号 来表示。行变量所能包含的最多字符个数由定维语句来定义。例如 DIM R#(5), A#(20), 表示行变量R# 至多能包含 5 个字符，行变量A# 至多能包含 20 个字符。行变量能包含的最多字符个数必须≥1。一旦行变量已在定维语句中定维后，它就不允许重新定维（与数组不同）。

5．数表达式

数表达式是由运算对象（数、简单变量、下标变量、函数等）与运算符（+、－、×、/、↑）组成。数表达式的计算顺序是先括号内，后括号外，运算符乘方↑最优先，乘除其次，加减最后。同一优先级别由左至右依次计算。例如 2－(A＋B)↑D×C，它的运算次序为先算A＋B，再作乘方D，再乘C，最后从 2 中减掉上述结果。

6．行表达式

行表达式是行常量和行变量的任意组合，它们之间用逗号隔开。行表达式的值是字符行。例如：

A#,′YES′

B#(2, 5), C#(3), D#(5)

可以用带有下标的行变量来处理一个字符行的一部分。如果下标是一个单一的数或数表达式，则字符行的引用部分是由下标给出的字符位置开始一直到该字符行的末尾。如上例中的C#(3)，它表示由行变量 C# 中的第 3 个字符至 C# 的末字符。若下标中包含有二个数或数表达式，则第一个数表示开始字符的位置，第二个数表示最后字符的位置，如上例中的B#(2, 5)，它表示行变量 B# 的第 2 个字符至第 5 个字符。

例：

```
100 DIM A#(50), B#(50)
110 LET A#='THE␣RECIPROCAL␣OF␣(X+Y)'
120 LET B#=A#(1, 20), '↑2+Y↑2)'
```

当执行 120 语句后，B# 中将包含下列内容：

THE␣RECIPROCAL␣OF␣(X↑2+Y↑2)

7．内部函数

BASIC 语言中提供的内部函数有 13 种，它们是：

SIN(X)—求以弧度表示的 x 的正弦；

COS(X)—求以弧度表示的 x 的余弦；

TAN(X)—求以弧度表示的 x 的正切；

ATN(X)—求 x 的反正切（弧度值）；

$(-\pi/2<\mathrm{ATN}(x)<\pi/2)$

LOG(X)—求 x 的自然对数；

EXP(X)—求 e^x；

SQR(X)—求 x 的平方根；

ABS(X)—求 x 的绝对值；

INT(X)—求不大于 x 的最大整数；

RND(X)—求在0和1之间的一个随机数（这里的 x 是数或数表达式，但它没有具体意义，它并不影响产生的随机数）；

SGN(X)—求 x 的代数符号。

当 $x>0$ 时	SGN(X)=+1
$x=0$ 时	SGN(X)=0
$x<0$ 时	SGN(X)=−1

LEN(S#)—求行变量S# 的当前长度（即行变量S# 中当前已存放的字符个数）；

DET(X)—求当前被求逆的矩阵的行列式值（这里的 x 是数或数表达式，但它没有具体意义，它并不影响行列式的值）。

下面再对某些函数作进一步说明：

INT(X) 函数产生一个小于或等于 x 的最大整数。例如：

INT(7.25)= 7

INT(−7.25)=−8

INT(12)=12

INT(−1)=−1

INT(X)=X(其中$|x|\geqslant 2^{+23}$)

可以用INT(X+0.5) 来求得 x 的最接近的整数（即舍入后的整数）。

LEN函数可求得行变量的当前长度。例如行变量A# 中包含有字符行TOTAL␣SALES，则：

LEN(A#)=11

DET函数可求得当前被求逆的矩阵的行列式值。

例如矩阵B为：

$$\begin{pmatrix} 3 & 1 \\ 7 & 3 \end{pmatrix}$$

矩阵B的行列式值为：$\Delta=2$

则

```
25  MAT   C=INV(B)   B矩阵求逆后存入矩阵C
30  LET   X=DET(1)   当前被求逆的矩阵（矩阵B）的行列式值存入x
35  PRINT X          打印x的值
```

产生的输出将为2。

第二章　BASIC 语句

在本章中介绍分时 BASIC 语言中各语句的格式和使用功能。

BASIC 语句可分为执行语句和非执行语句。

执行语句：

语句	功能
LET	给变量赋值
FOR NEXT	建立程序的循环
GOSUB	转入子程序
RETURN	从子程序返回主程序
GOTO	转移到指定语句
ON	实现选择转移
IF	实现条件转移
RESTORE	恢复或改变读数指针
RANDOM	恢复随机数发生器
OUTFOR	中途退出循环
CHAN	申请占用或释放设备
CALL	调用汇编语言书写的子程序
READ	从数据区中读数
INPUT	从电传打字机读入数据
PRINT	输出数据
STOP	中止程序执行并转入等待命令状态
END	结束程序执行并转入等待命令状态
MAT	矩阵运算（在第三章中介绍）
DIM	给数组和行变量定维

非执行语句：

语句	功能
DEF	定义一个用户函数
DATA	给定可供读出的数据
REM	注解

2.1　LET 语句

功能：

将数表达式的值或行表达式的值赋予某个数变量或行变量。

格式：

LET{〈数变量〉=〈数表达式〉|〈行变量〉=〈行表达式〉}$^{1}_{1}$

例如：

```
10  LET  A=4.17+G
40  LET  X=X+Y↑3.5
80  LET  W=(W-X)↑4.3*SQR(Z-A)/B
90  LET  C(I,INT(K/10))=COS(FNA(K+1))
```

上例均是给数变量赋值的语句。等号右端的各变量（或参数）必须是已经被赋值的（通过LET语句，READ语句，INPUT语句等给变量赋值），否则机器认为程序错误，此时在电传打字机上打印出差错信息，并中断程序的执行，转入等待键盘命令状态。程序员可以修改程序后重新启动程序。

例如：

```
20   LET  A#=B#     —A#的内容用B#的内容代替
25   LET  A#='␣'    —A#的内容用空格来代替
30   LET  A#=A#,B#—A#的当前值加上B#的内容
100  LET  A#='ABCDEF'
110  LET  B#='1'
120  LET  A#(3,3)=B#
130  LET  A#(3,6)=B#   } 在A#中都产生同一结果AB1DEF
140  LET  A#(3 )=B#
150  LET  A#(3)=B#,B#,B#—A#的结果为AB111F
```

上例均是给行变量赋值的语句。可以用LET语句把若干个字符赋给一个行变量或行变量的一部分。被赋值的字符个数，确定了将存贮的字符个数。

2.2 FOR语句与NEXT语句

功能：

用于建立程序的循环

格式：

FOR〈循环变量名〉=〈数表达式$_1$〉TO〈数表达式$_2$〉{STEP〈数表达式$_3$〉}$^{1}_{0}$

⋮

NEXT 〈循环变量名〉

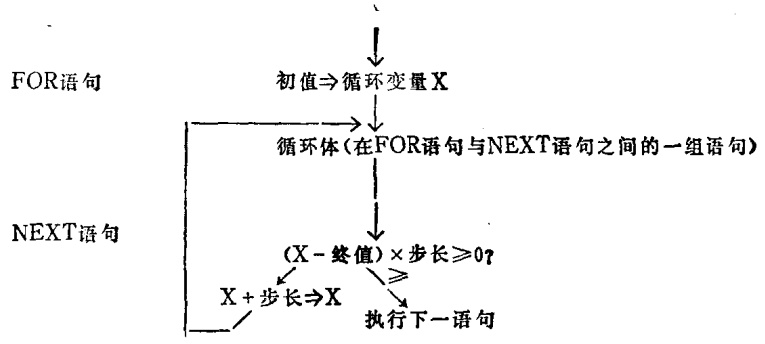
例如：

```
 5  LET  Y=0
10  FOR  X=1  TO  100
15  LET  Y=Y+X
20  NEXT  X
```

其中，X是循环变量，数表达式$_1$为循环变量的初值；数表达式$_2$为终值；数表达式$_3$为步长。步长可为正数（递增）或负数（递减）。NEXT是循环“出口”，每次执行到那里就比较循环变量是否已达到终值，超过了就往下执行，否则继续循环。FOR语句与NEXT

语句之间的语句是循环体，FOR 语句与 NEXT 语句两者使用的循环变量名必须一致。当步长为 1 时，可省略 STEP 1。此时，FOR 语句的格式为：FOR〈循环变量名〉=〈数表达式$_1$〉TO〈数表达式$_2$〉

循环语句执行情况如下所示：



循环可以套循环，但内外循环层次要分清，不能交叉，只允许 4 重循环嵌套。

正确的例子：

```
10    FOR    X=…
       ⋮
30    FOR    Y=…      } 内层循环   } 外层循环
       ⋮
50    NEXT   Y
       ⋮
80    NEXT   X
```

错误的例子：

```
10    FOR    X=…
       ⋮
30    FOR    Y=…      }            } X循环
       ⋮               } Y循环
50    NEXT   X        }
       ⋮
80    NEXT   Y
```

这个例子由于X循环与Y循环交叉，因此是不允许的。

FOR 语句的循环体，不是由标号的顺序来确定的，而由程序执行时的情况来决定。可以通过 GOSUB 语句，GOTO 语句等来扩充 FOR 语句的循环体。

例 1

```
100   FOR    I=0  TO  10
            ⋮
150   GOSUB    400
            ⋮
200   NEXT     I
            ⋮
400   LET      X=0     }
            ⋮          } 由 150 语句转入时，就将这段程序作为循环体
500   RETURN           }
```

例 2

```
100   FOR   J=0  TO  20
            ⋮
150   GOTO       500
            ⋮
200   NEXT   J
500   FOR  K=0  TO  10
            ⋮                    } 由 150 语句转入，就将这段程序作为循环体
550   NEXT  K
600   GOTO  200
```

例 3

```
100   FOR  L=0  TO  5  STEP  0.5
            ⋮
120   STOP    ——执行 120 语句后，转入键盘命令状态，此时的键盘命令（除了
                命令 RUN 外）均算为这个 FOR 语句的循环体
            ⋮
200   NEXT  L
```

2.3 GOTO 语句

功能：

用来改变程序执行顺序，从给定标号的语句继续执行。

格式：

GOTO〈语句标号〉

例：

```
100   READ  X
120   FOR  I=1  TO  5
130   LET  Y(I)=X*I
140   NEXT  I
150   GOTO  200
        ⋮
200   DATA  1,2
210   LET  A=5
```

此例中 150 语句实现转移到 200 语句，因 200 语句是非执行语句，所以其功能等价于 GOTO 210。

2.4 IF 语句

功能：

用来实现条件转移。

格式：

IF{〈数表达式〉|〈行表达式〉|〈布尔表达式〉}{GOTO|THEN|GOSUB}〈语句标号〉

当 IF 后的数表达式的值不等于 0、行表达式的值不等于 0 或布尔表达式的值为真时，就转向执行 GOTO、THEN 或 GOSUB 后的标号所指定的语句。若 IF 后的数表达式的值等于 0、行表达式的值等于 0 或布尔表达式的值为假时，则顺序执行 IF 语句后的下一语句。

布尔表达式的格式：

〈数表达式〉〈关系运算符〉〈数表达式〉

或

〈行表达式〉〈关系运算符〉〈行表达式〉

在 BASIC 语言中有下列关系运算符：

.LT.	小于（相当于“<”）
.LQ.	小于等于（相当于“≤”）
.NQ.	不等于（相当于“≠”）
=	等于
.GE.	大于等于（相当于“≥”）
.GT.	大于（相当于“>”）

例如：

```
100    IF  X+Y  THEN  1000
150    IF  A#    GOTO  300
200    IF  0.01.GT.M+N GOSUB  500
250    IF  B#(4,10).LQ.'ABCI' THEN400
```

上例中的 100 语句等价于下面的语句：

```
600    IF X+Y≠0  THEN  1000
```

当行表达式中没有字符时，认为此行表达式的值为 0。

布尔表达式的值的确定规则为：

当比较数表达式的值时，若条件成立则布尔表达式的值为“真”，否则为“假”。例如 M＝2，N＝－5，则M＋N＝－3 所以布尔表达式 0.01.GT.M＋N的值为“真”。

当比较行表达式的值时，先将两个行表达式的第一个字符按机内编码的大小进行比较，若第一个字符不同，则就为比较的结果；若两个行表达式的第一个字符相同，则往下比较第二个字符，直至两个行表达式的字符不相同时或直至比较完两个行表达式的全部字符。只有当两个行表达式具有相同的字符和字符个数时，这两个行表达式才相等。字符在机内的编码按国际七单位字符编码确定（见附录C）。

例如：

若 B#(4,10) 的内容为 ABCD，

则 B#(4,10)>'ABC1'（因为尽管前 3 个字符相同，但第 4 个字符D的编码＞1的编码）；

若 C#(2,7) 的内容为 BX5，

则 C#(2,7)>B#(4,10)（因为第一个字符 B 的编码＞A的编码）；