小型多功能计算机语言

第Ⅱ部份　BASIC语言

西　北　工　业　大　学

一九七八年五月

前　言

现代科学技术，以原子能的利用，电子计算机技术和空间科学技术的发展为主要标志，正在经历着一场伟大的革命，引起一系列新兴工业的诞生，广泛推动生产技术的飞跃发展。

当代，电子计算机正朝着巨型、微型、网络、智能模拟的方向发展。而电子计算机之科学技术水平的高低、生产规模的大小、应用程度的深浅等方面，已经成为衡量一个国家现代化水平的显著标志。

为了尽快地实现社会主义的四个现代化，我国的电子计算机科学技术、控制论和自动化技术必将出现一个新的飞跃，这将极大地推动生产自动化程度的提高、劳动生产率的提高和社会生产力的提高。

622计算机（在软件系统上与国内的100系列机和国外的NOVA系列机兼容）是一种集成电路小型多用电子数字计算机，它可用于科学计算、信息(数据)处理和过程(实时)控制等方面。如果把它当作通用机使用时，则可采用BASIC语言、FORTRAN语言和ALGOL语言等（其中，后2种尚待配置）进行科学计算；如果把它当作专用机使用时，则可采用基本汇编、扩展汇编和操作系统等（其中，后2种尚待配置）进行信息（数据）处理和过程（实时）控制等。

随着计算机硬件的迅速发展，需要了解和掌握计算机软件的人员也就越来越多了。

《小型多功能计算机语言》是为我校有关专业学习使用该类计算机而编写的，为了便于初学者能更快地掌握该机种的程序设计方法，所以将着重介绍汇编语言和BASIC语言，且尽量只从软件的角度出发，而把硬件的内容压缩到最低限度。另外，在相应章节中还根据专用控制机的需要，做了必要的说明和举例。

《小型多功能计算机语言》包括如下两部分内容：

第Ⅰ部分　汇编语言：介绍基本汇编的真、伪指令系统，编写程序的一般方法，基本汇编的使用，某些常用的典型程序，歌曲播送程序，扩展汇编与浮动引导简介，练习题及附录（如：真伪指令系统表，常用近似计算公式表，常用常数表，整数与小数的$2^{\#}$、$8^{\#}$、$10^{\#}$换算表等等）。为了便于查找和使用，将该附录顺放在第Ⅰ部分内容之后边。

第Ⅱ部分　BASIC语言：介绍本语言的符号系统，结合算式程序、分支程序、循环程序、子程序等的编制方法，详细地叙述了各种语句的格式、用法及在机上解题之步骤，某些典型程序的编写实例，练习题及附录（如：BASIC语言符号系统表、运算对象表、语句格式表、键盘操作表、语法错误表等等）。为了便于查找和使用，将该附录顺放在第Ⅱ部分内容之后边。

《小型多功能计算机原理》是《小型多功能计算机语言》的姐妹篇，前者侧重于硬件，后者侧重于软件。需要了解硬件的同志，可参阅前者。

《小型多功能计算机语言》是在《622计算机程序设计》的基础上，做了部分删改和补充之后而形成的。本书由804教研室的陆宗亨同志主编，607教研室的翁湘英同志参加了第Ⅰ部分的第6、7章和第Ⅱ部分内容之编写工作，602教研室的李永锡同志参加了第Ⅰ部分的第6、8章内容之编写工作。

在本书编写的过程中，曾得到许多同志的热情支持，特别是李明钧、王小莉、王秉业、霍秀芳、陈行健、严恒元、林树根、刘丰辛等同志在校对书中有关程序与图表绘制等方面都给予了大力协助。此外，在本书编辑出版的过程中，校印刷厂给予了大力支持，使本书能在较短的时间内完成出版任务。在此，谨向他们表示诚挚的感谢！

因时间紧促，书中难免存在错误和不足之处，希望读者及时指正，并提出宝贵建议。

最后，让我们共同积极、大力推广与普及电子数字计算机的应用，为在本世纪内，把我国建设成具有四个现代化的伟大社会主义强国而努力奋斗！

编　者

1978年4月8日

目　　录

第Ⅱ部份　BASIC 语言

第一章 基 本 概 念

1—1 引 言

BASIC 是 Beginner's All-purpose Symbolic Instruction Code 的缩写，意即“初学者通用符号指令代码”。它是1965年左右出现并开始使用的一种会话式程序设计语言。

BASIC 是面向多入口计算机系统的，在国外，已广泛地采用远距离终端打字机将程序送入计算机内进行运算，并将计算出来的结果送至该终端设备进行输出打印。其最鲜明的特点是能够通过电传打字机实现人和机器之间的信息交流，即所谓“人机对话”，以便于计算者能对整个运算过程进行监督与控制。因此，BASIC 是一种交互式的会话语言。

尽管 NOVA 1200 机配有 FORTRAN Ⅳ、ALGOL 60、BASIC 等语言，但由于目前622机只配有 BASIC 语言，因此，下面我们将主要以 NOVA 1200 机单用户 BASIC 语言为依据，详细地介绍一下 BASIC 语言的基本内容。

1—2 符号系统

1. 大写英文字母——26个

A、B、C、D、E、F、G、H、I、J、K、L、M、
N、O、P、Q、R、S、T、U、V、W、X、Y、Z。

2. 阿拉伯数字——10个

0、1、2、3、4、5、6、7、8、9。

3. 算术运算符——5个

+(加)、-(减)、*(乘)、/(除)、↑(乘幂)。

注意：在 BASIC 沅程序中，用*代替乘号，且规定在沅程序中的乘号*均不能省略。例如：不能把 $a * b$ 写成 $a \times b$ 或 $a \cdot b$ 或 ab。

4. 比较运算符——6个

<(小于)	<=(小于等于)	= (等于)
>(大于)	>=(大于等于)	<> (不等)

5. 分割符、括号、引号——6个

.〈小数点〉	,〈逗号〉	; 〈分号〉
(〈元开〉	)〈元闭〉	“ ”〈引号〉

6. 语句定义符——22个

① 说明符

DIM	DEF	FN	DATA	REM
维数	定义	函数	数据	注释

② 输入符

LET	READ	RESTORE	INPUT
赋值	读	恢复	输入

③ 输出符

PRINT	TAB
打印	格式

④ 终了符

STOP	END
暂仃	终了

⑤ 分支符

GOTO	IF	THEN
转向	如果	则

⑤ 循环符

FOR	TO	STEP
对于	至	步长

⑦ 转子符

GOSUB	RETURN
转子	返回

（注 意： DIM=DIMENSION、 DEF=DEFINITION、 FN=FUNCTION、 REM=REMARK、GOSUB=GO SUBROUTINE）

7. 键盘操作（命令）符——5个

NEW	CTRL	LIST	RUN	ESC
清除	控制	清单	执行	换码

（注 意： CTRL=CONTROL、ESC=ESCAPE）

8. 标志符3个

READY	TYPE	ERR
准备	类型	错误

（注 意： ERR=ERROR）

1—3 运算对象

BASIC 程序的运算对象可以是常数、变量、数组、函数（标准函数、自定义函数）及表达式。下面将分别介绍这些运算对象的基本概念，以及它们在沅程序中的使用方法。

1-3-1 常 数

在BASIC 语言中，不论输入或输出都采用 10^* 数， 它可以用不记阶的自然形式或用记阶的浮点形式来表示。

例 1-1. 数据的 2 种表示法。

无阶码的自然形式	有阶码的浮点形式
3.14159	0.314159E+1 ⟹ 0.314159×10^{1}
57.2958	5729.58 E−2 ⟹ 5729.58×10^{-2}
−0.0001	−1E−4 ⟹ -1×10^{-4}

这里的E表示阶的底数为 10，且规定E后面的数为正整数或负整数。

在进行数据输入时，最多可写 7 位有效数字；输出时，则最多可打印 6 位有效数字。当数据的位数≤6 位时，以不记阶的自然格式输出；当数据的位数＞6 位时，以记阶的浮点格式输出，即

±n.nnnnn E ± e

↑数符　　↑阶符

其中，n 表示有效数字；e为数的阶码，可用 1～2 位 $10^{\#}$ 数字表示。若数符为正时，则数的“+”号可略去，但阶符为正时，则阶的“+”号不能略去。

例 1-2. 数据的输出格式。

数　　据	输 出 格 式
800000000	8E+8
8848.13	8848.13
173	173
1/32	0.03125
−0.000001	−1E−6

注意：对 NOVA 1200 机来说，为了防止溢出，要求数的绝对值 $\leqslant1.70141\times10^{38}$。

1-3-2 变量与简单变量

所谓变量是指在计算过程中，每次仅取一个数值者，称之为变量。

在 BASIC 语言中，变量仅限于简单变量和下标变量。本节先介绍一下简单变量的表示法，至于下标变量的表示法将放到下节，再作介绍。

本语言规定，用 1 个字母、或 1 个字母后跟 1 个数字作为简单变量的标识符(即名子)，因此，最多可有 286(=26+26×10) 个标识符。简单变量的标识符之一般格式为：

〈一个字母〉 或 〈一个字母〉〈一个数字〉

例 1-3. 简单变量标识符的正确写法与错误写法。

A，B，X1，X2……　　写法正确

COMPUTER，ROOT2，441−B，1500……　　写法错误

在BASIC语言中，简单变量不进行类型区别，一般均作实型处理（即简单变量的取值范围仅为实数），因而，在使用前无需进行类似说明。

1-3-3 数组与下标变量

所谓数组是指在计算过程中，每次可取不同数值者，称之为数组。在BASIC语言中仅限于使用一维数组和二维数组，数组中的每个元素，叫做下标变量。

在使用下标变量之前，必须用数组维数语句来说明数组，以便让解释程序给数组之诸元素——下标变量，分配内存单元，数组维数语句的一般格式为：

```
DIM  〈一维数组名字〉(下标上界)，……，
     〈二维数组名字〉(行标上界，列标上界)，……
```

数组元素——下标变量的一般格式为：

```
〈一维数组名字〉(下标)
〈二维数组名字〉(行标，列标)
```

在使用数组语句时，必须注意以下各点：

① 因BASIC语言规定不论一维数组或二维数组的标识符（即名字）只能用1个字母来表示，故在1个程序中最多只允许有26个不同名之数组。

② 因BASIC语言规定一维数组的下标与二维数组的行标、列标均从0开始，故在数组名字后边的元括号内，对一维数组只需给出下标上界，对二维数组只需给出行标上界、列标上界之后，便可得知一维数组的下标变量个数=下标上界+1，而二维数组的下标变量个数=（行标上界+1）×（列标上界+1）。

③ 本语言规定一维数组的下标上界与二维数组的行标上界、列标上界可以是正整常数、正整简单变量、或正整表达式。

④ 在DIM语句中，行标上界与列标上界之间、以及数组之间必须用“，”隔开，但最后一个数组之末尾不必加用“，”。

⑤ 本语言规定：

i 在引用数组前，若用DIM语句对数组进行说明时，则一维数组的下标变量总个数≤256，譬如DIM A(255)；而二维数组的下标变量总个数≤1024，譬如DIM B(0,1023)或DIM C(15,63)或DIM D(31,31)等。

ii 在引用数组前，若不用DIM语句对数组进行说明时，则解释程序一律把一维数组看作是≤11个元素的数组，而把二维数组看作是≤11×11=121个元素的数组。

⑥ 数组元素——下标变量与DIM语句中的数组之表示方法是完全相似的。二者唯一差别仅在于，一维数组元素中的下标与二维数组元素中的行标、列标可以是一维数组中的下标上界与二维数组中行标上界、列标上界以内的任意一个正整表达式。因此，尽管二者表现形式完全相同，但其含义却不相同。

例1-4. 在计算 $F_{ij}=\frac{x_i y_j}{x_i+y_j}$ 时，若2个一维数组 x_i、y_j 之诸元素为已知，其相应数值如下：

x_i ： x_0 x_1 x_2 x_3 x_4

($i=0\sim4$) 1 3 5 7 9

y_j ： y_0 y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8 y_9

($j=0\sim9$) 2 4 6 8 10 12 14 16 18 20

则二维数组 F_{ij} 之诸元素亦可推算出来，通常用如下矩阵形式来表示：

$$F_{\substack{ij\\(i=0\sim4\\j=0\sim9)}} = \begin{pmatrix} F_{00} & F_{01} & \cdots\cdots & F_{09} \\ F_{10} & F_{11} & \cdots\cdots & F_{19} \\ \vdots & \vdots & & \vdots \\ F_{40} & F_{41} & \cdots\cdots & F_{49} \end{pmatrix}$$

按照 BASIC 语言的规定，需把上面 3 个数组及其元素——下标变量改写成如下形式：

X(I)： X(0) X(1) X(2) X(3) X(4)

1 3 5 7 9

Y(J)： Y(0) Y(1) Y(2) Y(3) Y(4) Y(5) Y(6) Y(7) Y(8) Y(9)

2 4 6 8 10 12 14 16 18 20

$$F(I,J) = \begin{pmatrix} F(0,0) & F(0,1) & \cdots\cdots & F(0,9) \\ F(1,0) & F(1,1) & \cdots\cdots & F(1,9) \\ \vdots & & & \vdots \\ F(4,0) & F(4,1) & \cdots\cdots & F(4,9) \end{pmatrix}$$

若用数组维数语句来说明这 3 个数组时，则有

50 DIM X(4), Y(9), F(4,9)

这就表明 X(4) 为含有 5 个下标变量的一维数组，Y(9)为含有 10 个下标变量的一维数组，F(4,9) 为含有 5×10＝50 个下标变量的二维数组。

⑦ 如果原来定义的旧数组不再需要时，可用重新定义的办法来改变数组维数的大小，以适应现时需要。但重新定义后，新数组元素总个数必须≤旧数组元素总个数。

例 1-5. 内存中有一组数据 1、2、3、4、5、6、7、8、9、10、11、12。

若第 1 次数组说明为

100 DIM A(2,3)

则该数组之下标变量 A(M,N) 所对应的数据呈如下关系：

M 行 \ N 列	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

其中数据“4”所对应的下标变量为 A(0,3)。

若第 2 次数组说明为

200 DIM B(3,2)

则该数组之下标变量B(M,N)所对应的数据呈如下关系：

M 列 \ N 列	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9
3	10	11	12

其中数据“4”所对应的下标变量为 B(1,0)。

若第3次数组说明为

```
300    DIM    C(5)
```

则该数组之下标变量C(L)所对应的数据呈如下关系：

L	0	1	2	3	4	5
C(L)	1	2	3	4	5	6

其中数据“4”所对应的下标变量为C(3)。

由此可见，尽管一组数据在内存中的存放顺序与位置没有变，但由于数组经过不同的定义之后，用来表示同一个数据“4”的下标变量却不相同。经过3次定义后，它们分别是A(0,3)、B(1,0)、C(3)。需要注意的是，数组重新定义后，其新数组名可与旧数组名相同，但旧数组之下标变量绝不允许在沅程序的后续语句中再次出现，而只准许出现新数组之下标变量。

1-3-4 标准函数与自定函数

在BASIC语言中，函数分为标准函数和自定义函数两类。

1. 标准函数

所谓标准函数是指某些常用的初等函数（如 $sinx$、e^a、$\sqrt{x}$……等），为了便于随时调用，通常把它们予先存放在机器中。标准函数的一般格式为：

〈标准函数符〉（自变量）

其中：〈标准函数符〉为由3个大写英文字母组成的初等函数专用名称；（自变量）可以是常数、简单变量、或表达式，且必须用元括号把它括起来。

在 BASIC 语言中，可直接被调用的标准函数有如下 11 种：

序号	标准函数	含　　义	备　　注
1	SIN(X)	x 之正弦 = sin x	$2\times10^{-39}\leqslant x\leqslant1.7\times10^{38}$
2	COS(X)	x 之余弦 = cos x	$2\times10^{-39}\leqslant x\leqslant1.7\times10^{38}$
3	TAN(X)	x 之正切 = tg x	$2\times10^{-39}\leqslant x\leqslant1.7\times10^{38}$ 不允许 x 接近$\frac{\pi}{2}$之奇数倍
4	ATN(X)	x 之反正切的主值 = arctg x	$2\times10^{-39}\leqslant x\leqslant1.7\times10^{38}$
5	EXP(X)	x 之以 e 为底的指数函数 = e^x	$2\times10^{-39}\leqslant e^x\leqslant1.7\times10^{38}$
6	LOG(X)	x 之自然对数 = ln x	$x\geqslant0$
7	SQR(X)	x 之平方根 = $\sqrt{x}$	$x\geqslant0$
8	ABS(X)	x 之绝对值 = $\lvert x\rvert$	
9	SGN(X)	x 之符号函数	
10	INT(X)	x 之不超过 x 的最大整数	INT(1.5) = 1 INT(−1.5) = −2
11	RND(X)	产生一个[0,1]区间中为均匀分布的伪随机数	∵ x 只有形式作用 ∴ x 可为任意值

注意：

① 凡三角函数之自变量都必须用弧度来表示，则

x = 角度/57.295779 = 角度×180/π，其中 π = 3.1415927。

② x 之符号函数 SGN(X) 的含义，如图 1-1 所示，即

$$SGN(X)=\begin{cases}+1 & \text{当 } x>0 \text{ 时}\\ 0 & \text{当 } x=0 \text{ 时}\\ -1 & \text{当 } x<0 \text{ 时。}\end{cases}$$

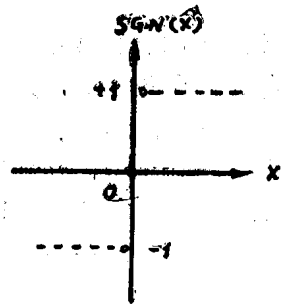

图 1-1

③ x 之整数部分 INT(X) 的含义如下：

INT(+7.25) = 7

INT(−7.25) = −8

INT(12) = 12

又如 x = 14.8时，　　则　INT(x + 2) = 16。

⋮

通常利用 INT（X）对数据进行舍入处理。譬如：

公　　式	含　　义	举　　例
INT（X）	将X舍入到其代数值为最小时之整数	假设 X＝6.24719，则 INT（6.24719） ⟹ 6
$\frac{(INT(X*10))}{10}$	将X舍入到第1位小数	假设 X＝6.24719，则 （INT(6.24719*10))/10 ⟹6.2
$\frac{(INT(X*(1E+D)))}{(1E+D)}$	将X舍入到第D位小数	假设 X＝6.24719，D＝5，则 (INT(6.24719*(1E+5)))/(1E+5) ⟹6.24719

④ 使用标准函数时，必须让自变量X落在相应初等函数之定义域内，以确保函数的计算结果不致超出机器数所能表示的最小、最大范围。

2. 自定函数

所谓自定函数是指由计算者自己所定义的某些一元函数，在沅程序中，自定义函数必须用函数定义语句来加以说明，它是一种说明型语句，其一般格式为

DEF FN〈字母〉（自变量）=〈表达式〉

其中：DEF为函数定义符；FN〈字母〉为自定义函数的名称，它必须以FN开头，其后的字母可以是26个英文字母中的任一个，不同的字母代表不同的函数，因此，在一个程序中最多可定义26个函数，即FNA、FNB、……、FNZ；在函数名称后的元括号内，需填写自变量的名字，称之为虚变量(相当于形式参数)，它可以是任意的简单变量，而且允许与程序中的其它变量同名；符号右边是一个含自变量的算术表达式，它体现了该自定义函数需要计算的内容。

在定义和引用自定义函数时，必须注意以下几点：

① 自定义函数必须先定义后引用，即自定义函数一经定义后，便可像标准函数一样，在后续语句中加以引用。引用时，只需写出自定义函数的名字，并用算术表达式（相当于实际参数）去替换其中的虚变量即可。

例1-6.

```
100  DEF  FNA(X)=3*X↑2+4*X+1
     ……
150  LET  Y=3+FNA(5)
```

标号为150的语句为赋值语句，执行该语句之后，便将如下表达式

3+(3*5↑2+4*5+1)=99

之值赋给变量Y。关于赋值语句的概念，将在后续内容中作介绍。

又如：

```
200  DEF  FNS(X)=(EXP(X)-EXP(-X))/2
```

若在后续语句中引用FNS(X)，则可用来计算虚变量为X时的双曲正弦函数 shx 之函数值。

再如：　　300　DEF　FNV(T)=E*(1-EXP(-T/R/C))

其中E、R、C的值为已知，若在后续语句中引用FNV(T)，则可用来计算虚变量为T时，在RC电路之电容充电过程中，其端电压V(t)的瞬时值。

② 函数定义语句中的右部表达式可以包含标准函数、虚变量、其它参数、及其它自定义函数，但不能自嵌套，即不允许在该函数定义语句的右部表达式中又出现了该自定义函数的名字。

例1-7. 下面的函数定义语句是合法的

```
400  DEF  FNA(X)=3*X↑2+4*X+1
420  DEF  FNB(Y)=(1+Y)*EXP(Y))
```

然而，下面的函数定义语句是非法的

```
450  DEF  FNF(Z)=1+5*Z+SIN(Z)*FNF(Z)
```

③ 函数一旦定义之后，便可以递归地调用，即可以在后续语句中，多重地调用前面已经定义了的函数，换言之，即可以多次地自己调用自己。

例1-8.

```
500  DEF  FNA(X)=1+SIN(X)
     ……
550  LET  Y=7+FNA(FNA(2+6*I))
```

执行了标号为500的语句之后，便将如下表达式

```
7+(1+sin(1+sin(2+6*I)))
```

的值赋给了变量Y。

1-3-5 表 达 式

BASIC语言中的表达式仅限于简单算术表达式，它是由常数、简单变量、下标变量、标准函数、自定函数等通过算术运算符及元括弧等所组成的有确定数学意义的式子。

例1-9.

```
3.1415927
P1
Q[I,J]
E*(1-EXP(-T/R/C))
(-B+SQR(B↑2-4*A*C))/(2*A)
```

等等都是简单算术表达式，它在沅程序中用来对常数、变量及函数进行指定的算术运算。

在BASIC语言中，与通常的习惯一样，规定表达式中运算顺序的优先级为：先括号内，再乘方，随后乘除，最后加减。在同一优先级内的运算符，将从左至右按其出现的先后顺序进行运算。

在使用表达式时，必须注意以下各点：

① 凡出现在表达式中的所有符号都必须书写在同一条横线上。

例1-10. 对于给定的通常数学式子

$$\frac{E}{\sqrt{R^2+\left(2\pi FL-\frac{1}{2\pi FC}\right)^2}}$$

在 BASIC 沅程序中应写成

E/SQR(R↑2+(2*3.14159*F*L-1/(2*3.14159*F*C))↑2)

BASIC 语言定，在规表达式中，可视需要，适当加用元括号，但绝对不允许加用其它形式之括号。

② 在 BASIC 语言中，若要进行 $F=X^N$ 之运算，则必须要求 X>0，且依据 N 取不同的值，可分作两种情况来处理：

第 1 种情况：若 X>0、N = 整数，则按普通乘法来进行计算。

第 2 种情况：若 X>0、N≠整数，则按X↑N = EXP(N*LOG(X))来进行计算。

当X与N取各种不同的值时，其 $F=X^N$ 之处理情况如下：

X	N	$F=X^N$
X>0	N = 整数>0	F = X*X*……*X = X↑N
	= 0	F = 1
	<0	F = 1/(X*X*……*X) = 1/(X↑N)
X>0	N≠整数	F≈EXP(N*LOG(X))，保证 $10^{\#}$ 的前 5 位数准确。
X = 0	N>0	F = 0
	N≤0	F = 无定义
X<0	N = 任意	F = 无定义

1—4 沅程序的组成

为了便于说明 BASIC 语言之沅程序的组成及编写要领，请看下面例题。

例 1-11. 已知 RC 电路之电容充电过程中，其端电压瞬时值 V(t) 的计算公式为

$$V=E(1-e^{-\frac{t}{RC}})$$

其中，变量 E、R、C、t 的值是已知的，电压 V 是待求的。试编写计算 V(t) 之 BASIC 沅程序。

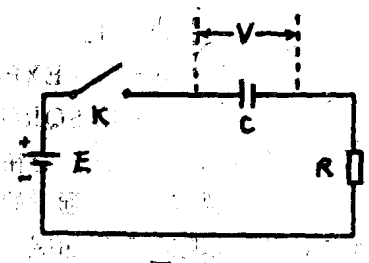

图 1-2

通常可把该计算过程分作如下几步来进行：

① 输入：通过电传机输入 E、R、C、t 的数值，为此，可用如下输入语句

INPUT E, R, C, T

② 计算：按给定公式进行计算，为此，可用如下赋值语句

LET V=E*(1-EXP(-T/R/C))

③ 输出：通过电传机打印计算结果，为此，可用如下输出语句

PRINT V

④ 终了：告诉解释程序全部计算过程已进行完毕，必须用终了语句

END

在上面的每个语句前面，还必须附加一个语句标号，每个标号可以是1～9999中的任意一个正整数，但必须根据语句的执行顺序而依次递增。于是，上述问题的沅程序可写成

```
20  INPUT  E, R, C, T
40  LET    V=E*(1-EXP(-T/R/C))
60  PRINT  V
80  END
```

BASIC沅程序有下列几个特点：

① 每个语句的结尾不须加用标点。

② 在书写沅程序时，必须采用BASIC语言所规定的符号系统和语法规则。由于国产5单位电传打印机每行只允许打印69个字符，所以每个语句的字符及空格的总数不能超过69个。若超过了，则必须把一个语句分成两个语句来书写。

例1-12.

```
20  LET  S=A+…+B+C+…+D
```

假设该语句的字符及空格的总数超过69个，则可分成两个语句

```
20  LET  S1=A+…+B
25  LET  S=S1+C…+D
```

③ 沅程序中的标号排列，一般都不采用按自然数依次递增（如10、11、12、…）的方式进行排列的，而是在相邻两语句的标号之间留出当适的空隙（如例1-11中的标号为20、40、50、80），以便在编写沅程序或从电传机输入沅程序的过程中，一旦发现程序中有遗漏或错误时，可临时插入一些语句。

④ BASIC语言的独特优点是，只要语句标号的大小顺序能正确地体现出给定问题所要求的运算顺序，那么沅程序中各语句的编排次序或输入次序均可为任意。这一功能的设置，为编写或修改沅程序带来很大的灵活性。

例1-13. 可把例1-11中的4个语句按如下次序写出

```
80  END
60  PRINT  V
40  LET    V=E*(1-EXP(-T/R/C))
20  INPUT  E, R, C, T
```

甚至允许各语句的先后次序书写得十分零乱，但只要标号之大小符合所要求的运算次序，那么都不会妨碍运算的正确性。这是因为，当上述各语句输入到机器内存之后，解释程序将自动地从最小标号之语句开始，去逐个地执行每个语句，当且仅当执行到终了语句时，才使整个运算仃止下来。

又如：我们在编写或输入上述沅程序时，如果只编写或输入了其中的3个语句，这时，只需在程序的最后边再添加上被遗漏的语句，并根据运算顺序的要求，给予恰当的语句标号即可。

1—5 基本语句的类型

BASIC语言的语句有两种类型：一类是说明型的，用以指明程序中的某些对象（如数组、自定义函数等等）的性质；一类是执行型的，它指示机器去执行给定的运算或操作。

单用户 BASIC 语言的基本语句共计分为两大类型 14 种：

1. 说明型语句

① 数组（维数）语句 DIM

② 自定（函数）语句 DEF FN

③ DATA（数据）语句 DATA

④ 注释（备考）语句 REM

2. 执行型语句

① 赋值语句 LET

② 输入语句 READ
⋮
DATA
⋮
RESTORE
READ
INPUT

③ 输出语句 PRINT …,
PRINT …;
PRINT … TAB

④ 暂仃语句 STOP

⑤ 终了语句 END

⑥ 转向语句 GOTO

⑦ 条件语句 IF … THEN 或 GOTO

⑧ 循环语句 FOR … TO … STEP
⋮
NEXT

⑨ 转子语句 GOSUB

⑩ 返回语句 RETURN

下面将别介绍这些语句的一般格式、功能、以及如何运用它们来组成 BASIC 语言之沅程序的方法与技巧。