第二篇　程序设计篇

第一章　BASIC 语言

§1.1　BASIC 语言的特点及其发展

BASIC 是Beginner 's All－Purpose Symbolic Instruction Code 的缩写，原义为"初学者通用符号指令代码"。它诞生于1964 年，其创始人是美国达特茅斯学院(Dartmouth College)的约翰·凯梅尼(John Kemeny)博士和托马斯·库尔茨(Thomas Kurtz)博士。

BASIC 语言有以下几个主要特点：

(1)小巧灵活，简单易学、使用方便，特别适合初学者的需要。

(2)具有"人机对话"功能、是一种会话式语言。使用者通过终端设备和计算机进行"对话"，以检查和修改错误、运行程序，得到结果。

(3)具有"直接工作方式"，因而能实现台式计算机的功能。

(4)具有字符串操作和与外部设备通信的特殊功能，从而使 BASIC 语言能进行数据处理和实时控制。

(5)语言本身分为单用户、分时和扩充三类，用户选择时比较灵活。

(6)具有较强的接口能力、可以实现与其他各种软件的联用。

BASIC 语言最初是为了便于教学而设计的，而现在的应用已远远超出了教学的范围，国内国际的现实充分表明，BASIC 是一种颇受欢迎、流行甚广，而且能解决实际问题的程序设计语言。由于版本的不断改进，使得 BASIC 的功能越来越强。近几年推出的BASIC 的最新版本——True BASIC 和 Quick BASIC，使 BASIC 成了真正的结构化程序设计语言。

§1.2　BASIC 语言基础

BASIC 语言版本繁杂，而且没有统一的标准。下面将主要介绍IBM PC 及其兼容机上使用的 MS－BASIC (Microsoft BASIC)，即 BASICA 。

1.2.1 程序结构

BASIC 程序由一系列的程序行组成，程序行的一般格式如下：

行号 BASIC 语句 [:BASIC 语句 …] [注释]〈CR〉

其中：

行号：1 — 5 位整数，每个 BASIC 程序行都必须以行号开始，它表示各程序行在内存中的顺序，也可作为控制转移和编辑的参考点，行号的范围为 0 — 65529。在一些命令(如 LIST，DELETE 及 EDIT 等)中，可以用句点代替当前行号。

BASIC 语句：是一个可执行(executable)语句或非执行(non－executable)语句，通常由语句定义符和语句体两部分组成。语句定义符规定计算机执行哪一种功能、语句体则是需要执行的具体内容。例如，在

PRINT X1＋X2

中，PRINT 是语句定义符，X1＋X2 是语句体。

一个程序行可以含有多个语句，各个语句之间用冒号隔开；每个程序行中的字符数不得超过 255。

注释：放在程序行的末尾，并用单引号与行中其他成分隔开(也可单独作为一行，以单引号或 REM 开头)。

CR：回车键，每个程序行必须以回车键结束。

1.2.2 字符集

BASIC 字符集由英文字母、数字和专用字符组成。

英文字母：包括大、小写英文字母各26个。

数字：0 — 9。

专用字符：共 25 个，其含义如下：

␣	空格
=	等号或赋值符号
+	加号或字符串连接符号
−	减号
*	星号或乘号
/	斜杠或除号
\	反斜杠或整数除法符号或路径定界符
∧	指数符号
(	左括号

) 右括号
% 百分号或整数类型说明符
# 磅号或双精度类型说明符
$ 美元符号或字符串类型说明符
! 感叹号或单精度类型说明符
& 和号(ampersand)
, 逗号
. 句点或小数点
' 单引号(撇号)，注释(REM)的简写形式
; 分号
: 冒号或语句分隔符
? 问号，PRINT 的简写形式
< 小于
> 大于
" 双引号或字符串定界符
_ 下划符

1.2.3 常量

常量是在编写 BASIC 程序时使用的，并且在程序执行期间不改变的实际值。常量有两种类型：字符串常量和数值常量。

1. 字符串常量

字符串常量是放在双引号中的字符序列，字符可以是字母、数字及其他符号，一个字符串的最大长度为255 个字符。下面是一些字符串的例子。

"HELLO"，"$25,000.00"，" Number of Employees "

2. 数值常量

数值常量可以是正数、0 或负数。正数的正号(+)可要可不要，负数必须有负号(-)。BASIC 中的常量不含有逗号。数值常量共有五种表示方式，即整数、定点数、浮点数以及十六进制数和八进制数。

- 整数 没有小数点和指数符号(D 或 E)，其取值范围为 -32768 — +32767。
- 定点数 含有小数点的数，即正或负的实数。
- 浮点数 用指数形式(科学记数法)表示的正数或负数，其范围为 $2.9*10^{-39}$ — $1.7*10^{38}$ (正数或负数)。浮点数由尾数、指数符号和指数三部分组成，其中尾数是带符号(正数可省)的整数或定点数，指数符号为E(单精度数)或 D(双精度数)，指数是带符号(正数可省)的整数。指数符号 E 或 D 的含义为："乘上 10 的幂次"。 例如：

23E－2

在这里，23 是尾数，E 是指数符号，－2 是指数，它表示一个单精度数，读作“23乘以 10 的负 2 次方”。再例如：

235.988E－7 和 2359D6

分别为单精度数和双精度数，其值分别为 0.0000235988 和 2359000000。

• 十六进制数　由十六进制数字 0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F 组成，最多为4 位数，以前缀 &H 为标志，例如：

&H76　&H32F

• 八进制数　由数字 0 — 7 组成，最多可达 6 位数，以前缀 &O 或 & 为标志，例如：

&O34　& 1235

3. 数值精度

数值常量在内存中以三种形式存储，即整数、单精度数和双精度数。其中以整数、十六进制数和八进制数格式输入的常数在内存中占 2 个字节，都被看作是整数。

单精度数在内存中以 4 个字节存放，存储和输出可达 7 位十进制数，但只有 6 位是精确的。单精度数形式可以是，7 位或少于 7 位的数，或用 E 表示的指数形式，或末尾带有类型说明符(“!”)。例如：

87243，46.8，－1.09E－06，3489.0，22.5!

双精度数在内存中用 8 个字节存放，存储精度可达 17 位，输出只有 16 位。双精度数形式可以是，8 位或超过 8 位的数，或用 D 表示的指数形式，或末尾带有类型说明符(“#”)。例如：

345692811，－1．90432D－06，3489．0#，7654321．1234

数值常量的范围和精度如表 1.1 所示。

表1.1　数值常量的范围和精度

类　　型	范　　围	精度	存储
整　数	－32768.— 32767		2 字节
单精度浮点数	10E－38 — 10E＋38	6 位	4 字节
双精度浮点数	10D－38 — 10D＋38	16 位	8 字节

1.2.4　变量

变量表示给定的数据项的名字，在程序执行中，它的值是可以改变的。BASIC 中的变量也有两种类型，即数值变量和字符串变量。数值变量的值总是数值，字符串变量

的值总是字符串值。

字符串变量的长度是可以改变的，其范围为 0 — 255 个字符，变量的长度由所赋予的字符串变量值的长度决定。

在程序中，可以将常量赋给变量，也可以把计算结果或程序中数据输入语句提供的常量作为变量的值。在上述情况下，变量类型必须和指定的数据类型相匹配。

如果在赋值之前使用一个变量，则该变量的值为 0；对于字符串变量，则表示无字符，变量的长度为 0。

1. 变量的构成

BASIC 变量名的长度没有限制，但只有前40 个字符有效。

变量名由英文字母、数字和小数点组成，必须以字母开头。变量名的最后一个字符也可以是标识其类型的字符(见后)。

变量名不能是保留字，但可以是嵌入保留字的字符串；同时，变量名也不能是末尾带有类型说明符的保留字。例如：

```
10  DATA=100
```

或 `10  DATA$ = "ABC "`

是非法的，因为 DATA 是保留字，而

```
10  DATAAREA=100
```

是合法的，因为 DATA 是变量名的一部分。

2. 变量类型说明

变量的类型由类型说明符来标识。其中 $ 表示字符串型，% 表示整型，!表示单精度型，# 表示双精度型。类型说明符放在变量名的末尾。不同类型的变量在内存中所占的字节数也不一样，如表 1.2 所示。

表 1.2 变量类型

变量类型	整 型	单精度型	双精度型	字符串型
说明符	%	!	#	$
示 例	LIMIT%, A%	MINIMUM!	PI#	PLA$
占内存字节数	2	4	8	0 — 255

变量名相同而类型声明符不同时，所表示的是不同的变量。例如，SMALL%，SMALL!和 SMALL# 是三个不同的变量。

当一个变量没有类型说明符时，将被隐含地说明为单精度变量。如 OLDER 与 OLDER! 等效 。

在三种数值变量中，双精度型的精度最高，占内存字节较多，计算速度较慢。在实际应用中，应根据需要说明变量的类型。当所要求的精度不高时，通常使用单精度变量；如果处理的数据都是整数，则应使用整型变量。这样不仅能节省内存空间，而且可以提高处理速度。

除了用类型说明符说明变量外，在 BASIC 中还可以用类型说明语句来说明变量，其一般格式为：

DEFtype　字母[－字母]　[，字母[－字母]]…

其中DEF 是保留字，type 是类型标志，可以是INT、SNG、DBL 和 STR，分别表示整型、单精度型、双精度型和字符串型，在 DEF 和类型标志之间不要有空格。字母可以是 A 到 Z 中的任一个。例如：

10　DEFDBL　L－P

20　DEFSTR　A

30　DEFINT　D－ H

40　DEFSGN　B

类型说明语句中所定义的字母可以作为那种类型的变量名，而且以该字母开头的变量名也是那种类型的变量。在上面的例子中，L，M，N，O，P 都可以作为双精度变量，而且以这些字母开头的变量，如 LIMIT，NUMBER，POINTER 等也是双精度变量名。

类型说明语句通常放在程序的开头。当整个程序中的变量都使用整型数时，可以这样来说明：

10　DEFINT　A－ Z

3. 数组

数组是类型相同并按一定格式存放的一组数据项，一个数组中数据项的个数是固定的。BASIC 中的数组分为数值数组和字符串数组。

每个数组都有一个名字，数组中的每个数据项称为数组元素。数组元素也是一种变量，叫做带下标的变量，简称下标变量，可以和普通变量一样使用。

说明一个数组的名字和类型，确定其元素的个数及在该数组中排列顺序的操作称为定义数组，通常由 DIM 语句来实现。数组元素只有一个下标的数组叫做一维数组，而有两个下标的数组叫做二维数组。例如：

10　DIM　ARR1$ (5)

定义了一个一维数组，名字为 ARR1$，类型为字符串型，一般含有 6 个(0 － 5)元素，其初始值为空。再例如：

20　DIM　ARR2% (2,3)

定义了一个二维数组，名字为 ARR2%，类型为整型，含有 12 (3×4)个元素。

下标表示元素在数组中的位置，在一般情况下，下标从 0 开始。如在前面的例子

中，数组 ARR1$ 中各元素的排列顺序为：

ARR1$ (0)，ARR1 $ (1)， ARR1$(2)，ARR1$ (3)，ARR1 $(4)，ARR1$ (5)

它可以存放 6 个字符串，其中第一个字符串的名字叫做 ARR1 $ (0)，最后一个叫做 ARR1$ (5)。

数组 ARR2% 中各元素的排列为：

ARR2% (0,0)	ARR2% (0,1)	ARR2% (0,2)	ARR2%(0,3)
ARR2%(1,0)	ARR2% (1,1)	ARR2%(1,2)	ARR2%(1,3)
ARR2%(2,0)	ARR2%(2,1)	ARR2%(2,2)	ARR2%(2,3)

上面是一维和二维数组的例子。BASIC 允许数组维数的最大值为 255，每一维中元素个数的最大值为 32767，这两个值都受到存储器容量的限制。

数组可以隐含说明，即不定义而直接使用。隐含说明数组的下标的最大值为 10，例如：

```
50  SIS (3)= 500
```

由于 SIS 未被定义，隐含地认为它是一维数组，含有 11 个元素，从 SIS(0)到 SIS (10)。

有时候，需要下标从 1 开始而不是从 0 开始，这可以通过 OPTION BASE 语句来实现，其格式为：

```
10  OPTION  BASE  1
```

4. 类型转换

在实际应用中，常常需要把数值从一种精度转换为另一种精度。这种转换根据下述规则进行。

(1)当把一种精度的数值赋给另一种精度类型的数值变量时，数值将服从变量的精度类型，并以这种精度类型存储。例如，在语句

```
10  A% = 23.42
```

中，把一个单精度数赋给一个整型变量，所存储的是整数 23，而不是 23.42。

(2)当把较高精度的数值赋给较低精度的变量时，进行舍入运算。例如，语句

```
10  C = 423.834685##
```

要把双精度类型的数值赋给单精度变量，其结果为 423.8347，即在双精度数的第八位四舍五入，把一个有 7 位精度的单精度数赋给单精度变量 C。

(3)当把较低精度的数转换为较高精度的数时，转换后的结果不会比原来的低精度数更精确。它有可能等于原来的数，但也可能大于或小于原来的数。设 A 为转换前的单精度数，B## 为转换后的双精度数，则转换后的双精度数和原单精度数差的绝对值小于 6.3E－8 乘以原来的单精度数，即误差

d＝|B## －A|＜6.3E－8＊A

例如，执行语句

10　A＝2.04

20　B## ＝A

则对 B##所赋的值为 2.039999961853027。再例如:

10　A＝1.5

20　B## ＝A

B## 的值也为 1.5。有的赋值后大于原来的值，例如:

10　A＝1.6

20　B## ＝A

赋值的结果，B## 等于 1.600000023841858。

(4)当对一个表达式求值时，算术运算或逻辑运算的所有变量都必须具有相同的精度，通常要转换为表达式中最高精度运算数的精度，算术运算的结果也与该精度相同。也就是说，在算术表达式或关系表达式中，如果含有不同精度的数值，则较低精度的量要向较高精度的量看齐，算术运算的结果也是一个较高精度的量。例如，在

10　D## ＝6 ## /7

中，表达式 6## /7 的结果是一个双精度型，把它赋给变量 D##，则 D## 等于 0.8571428571428571。但是，如果 D 不是一个双精度变量，而是一个单精度变量，则根据上述第二条规则，尽管 6## /7 是一个双精度值，给变量 D 赋值的结果仍然是一个单精度值。例如，执行语句

10　D＝6## /7

D 被赋值为 0.8571429。

(5)当进行逻辑运算时，总是把表达式中的操作数一律转换为整型值，并且结果也是整数。操作数的范围为 －32768 — 32767，否则会产生"溢出"错误 。

5．内部变量

BASIC 中设置了 5 个内部变量，这里介绍其中的 2 个，另外 3 个在以后有关章节中介绍。

(1)DATE$

格式:

① V$ ＝DATE$

② DATE$ ＝X$

功能: 设置或检索系统的日期。

说明:

① 在第一种格式中，DATE$ 返回系统的当前日期，格式为"mm－dd－yyyy"，

其中 mm 是代表月份的两位数字，dd 是代表日的两位数字，yyyy 是代表年份的四位数字。这个日期可以在 DOS 状态下通过 DATE 命令设置。

② 在第二种格式中，X$ 是用来设置当前日期的字符串表达式，它可以按下列格式中的任一种输入：

mm－dd－yy

mm/dd/yy

mm－dd－yyyy

mm/dd/yyyy

其中年(yyyy)的范围为 1980－2099。如果月或日只使用一个数字，则系统假定在该数字前有一个 0。如果只使用一个数字表示年份，则附加一个 0 组成两位数字，如果用两位数字表示年份，则所表示的年份为 19yy。例如：

```
10 DATE$ = "7-2-89"
20 PRINT DATE$
```

的结果为 07－02－1989。

(2)TIME$

格式：

① V$ = TIME$

② TIME$ = X$

功能：设置或检索系统时间。

说明：

① 在第一种格式中，TIME$ 返回系统的当前时间，其值是一个由 8 个字符组成的字符串，格式为 hh:mm:ss 。这里 hh 是小时(00—23)，mm 是分(00—59)，ss 是秒(00—59)。系统时间可以在 DOS 状态下由 TIME 命令设置。

② 第二种格式可用来设置时间。此时 X$ 为字符串表达式，表示将要设置的时间，可以为下列格式之一。

hh 设置小时，范围为0—23，分和秒均省略为 00。

hh:mm 设置小时和分，分在0—59 范围内，秒省略为 00。

hh:mm:ss 设置小时、分和秒，秒在0—59 范围内。

③ 当小时或分为 0 时，至少应含有一个数位。例如，0 点 30 分可以设置为 TIME$ = "0:30"或 TIME$ = "00:30"，但不能写作 TIME$ = ":30"。

④ 在设置时间时，如果值超出范围，则产生"Illegal function call"错误；如果 X$ 不是一个有效的字符串，则产生"Type mismatch"错误。在上述两种情况下，将保留原来的时间。

1.2.5　数值表达式和运算符

数值表达式的含义较广。它既可以是由一个数值常量或变量组成的简单表达式，也可以是用后面讲的各种运算符把常量、变量及函数连接起来的较复杂的表达式。对数值表达式的运算结果是一个数值。数值运算主要是对数值进行算术运算和有关数值方面的逻辑运算，有时也可以是关于字符串值的逻辑运算。BASIC 的数值运算有以下几类：

- 算术运算　对算术表达式求值，结果为数值。
- 关系运算　对关系式求值，结果为逻辑值。
- 逻辑运算　对布尔表达式求值，结果为逻辑值。
- 函数运算　对自变量求值，结果为数值。

1. 算术运算

算术运算的结果是一个数值，包括加、减、乘、除及指数等运算。根据在表达式中的运算顺序，这些运算如表 1.3 所示。从表中可以看出，指数运算的优先级最高，浮点除法运算优先于整型除法运算，加、减运算的优先级最低。

表 1.3　算术运算

运　　算	运算符	表达式例子
指数运算	^	X ^ Y
取负数	－	－X
乘法	*	X* Y
浮点除法	/	X/Y
整数除法	\	X\Y
取模运算	MOD	XMODY
加法	+	X+ Y
减法	－	X－Y

表中的多数运算大家都比较熟悉，下面只对整数除法和取模运算作简单说明。

(1)整数除法。整数除法的运算符是反斜杠“\”，操作数可以是整数，也可以带小数。当带有小数时，首先被四舍五入为整型数，然后再进行整除运算，操作数必须在 －32768 — 32767 范围内，整除运算的结果被截断为整型，即小数点以后的数字全部丢掉，而不是进行舍入处理。例如：

```
10  A=10\4
20  B=25.63\6.78
```

根据上面的运算规则，可以知道运算结果为 A 等于 2，B 等于 3。

(2)取模运算。取模运算的运算符是 MOD，其结果为一整型数值，这个值是整数

除法的余数。例如:

```
10 A=7 MOD 4
```

7 被 4 整除，商为 1，余数为 3，因此 A 的值为 3。再例如:

```
10 PRINT 25.68 MOD 6.99
```

上述语句首先通过四舍五入把 25.68 和 6.99 分别变为整数 26 和 7。26 被 7 整除，商为 3，余数为 5，因此上面的语句打印出整数 5。

2.关系运算

关系运算用来实现两个值的比较操作。被比较的两个值可以都是数值，也可以都是字符串。比较的结果只有两种可能的值，一种是"真"(非 0)，一种是"假" (0)。在程序中，关系运算的结果通常用来判断当前程序的流程。BASIC 中的关系运算符及其意义如表 1.4 所示。

表 1.4 关系运算

运算符	测试关系	表达式例子
=	相等	X=Y
<> 或 ><	不相等	X<>Y或X><Y
<	小于	X<Y
>	大于	X>Y
<= 或 =<	小于或等于	X<=Y 或 X=<Y
>= 或 =>	大于或等于	X>=Y 或 X=>Y

(1)数值比较。 数值比较实际上是对两个算术表达式的值进行比较。当算术运算和关系运算在同一个表达式中一起使用时，先执行算术运算，然后再执行关系运算。例如，在表达式

```
X+Y<(T-1)/Z
```

中，如果(X+Y)的值小于(T-1)/Z 的值，则该表达式的值为"真"，否则为"假"。

比较操作的结果是一个逻辑值，即"真"或"假"。在 BASIC 中，逻辑值不作为"TRUE"或"FALSE"输出，而是输出 -1 或 0。例如，执行

```
PRINT 5<2;5<10
```

结果为 0 和 -1，表明 5<2 为假值，5<10 为真值。

在程序中，关系运算的结果作为判断用，通常放在 IF 语句中。例如:

```
10 X=100
20 IF X<>200 THEN PRINT "NOT EQUAL"ELSE PRINT"EQUAL"
```

在这里，由于X不等于200，因此关系运算的结果为真，执行 THEN 后面的操作，输出"NOT EQUAL"。

(2)字符串比较。　字符串比较按字符顺序进行，即对相应的 ASCII 码进行比较。具体过程为，从左端第一个字符开始，依次同时从两个字符串中各取出一个字符，比较它们的ASCII 码，如果所有相应的 ASCII 码都相同，则这两个字符串相等；如果在比较中相应的 ASCII 码不同，则代码较小的字符串小于代码较大的字符串；如果在比较过程中一个字符串先结束，则该字符串较小。注意，在字符串中，开始及尾部的空格也有意义。下面所有的关系式均为真：

"AA" < "AB"

"FILENAME" = "FILENAME"

" X&" > "X#"

"WR␣" > " WR"　　("␣"表示空格)

"kg" > "KG"

" SMYTH" < "SMYTHE"

"B$" > "718"

在进行字符串比较时，所有字符串常量都必须用双引号括起来。

用下面的程序可以比较两个字符串的大小。

```
100 INPUT A$,B$
110 IF A$<B$ THEN PRINT A$;"<";B$
120 IF A$=B$ THEN PRINT A$;"=";B$
130 IF A$>B$ THEN PRINT A$;">";B$
140 GOTO 100
```

3．逻辑运算

逻辑运算也叫布尔运算。通常用逻辑运算符连接两个或多个关系式，组成一个布尔表达式，对该表达式的求值过程叫做逻辑运算。布尔表达式的结果也是一个逻辑值。在逻辑运算中使用的逻辑运算符有：

NOT	非
AND	与
OR	或
XOR	异或
IMP	蕴含
EQV	等价

逻辑运算的规则如下(其中"T"表示真，"F"表示假)：

X	Y	NOT X	X AND Y	X OR Y	X XOR Y	X IMP Y	X EQV Y
T	T	F	T	T	F	T	T
T	F	F	F	T	T	F	F
F	T	T	F	T	T	F	T
F	F	T	F	F	F	T	T

和关系运算一样，逻辑运算也用来判断程序的流程。下面举几个例子。

IF HE>60 AND SHE<20 THEN END

在这里，如果 HE 大于 60 并且 SHE 的值小于 20，则布尔表达式的值为真，程序结束。

IF I>10 OR K<0 THEN PRINT I,K

在上述语句中，如果 I>10，或者 K<0，满足这两个条件之一，或者这两个条件都满足，则布尔表达式的结果为真，打印 I 和 K 的值。

IF NOT (P=-1) THEN P=10

在这个语句中，如果 P 不等于 -1，则把 10 赋给 P。

100 FLAG% =NOT FLAG%

该语句将一个值从真到假，从假到真地进行转换。

当对数值进行逻辑运算时，操作数必须在 -32768 到 32767 范围内，并且要转换为整数。如果操作数不在这个范围内，则产生溢出错误；如果操作数是负数，则把它变成相应的补码形式。参加运算的数都要转换成 16 位二进制数，逻辑运算是在这样的数上按位完成的，也就是说，运算结果中的每一位由两个操作数的对应位来确定。用逻辑运算可以检测机器的状态位。下面是一些逻辑运算的例子。

A=63 AND 16

由于 63 的二进制码是 0000000000111111，16 的二进制码是 0000000000010000，63 AND 16 等于0000000000010000，即 16，因此 A 的值为 16。

B=-1 AND 8

-1 的二进制补码为1111111111111111，8 的二进制码是 0000000000001000，两个数相"与"的结果为 0000000000001000 ，因此 B 的值为 8。

C= 4 OR 2

4 的二进制码为 100，2 的二进制码为 010，两个数"或"后为 110，因此 C 的值为 6。

X=2

TWO SCOMP=(NOT X)+1

该例表明了求一个数的补码的方法。X=2，二进制码为 10，NOT X 为1111111111111101，即 -3 的补码形式，加上 1 后等于 1111111111111110，即为 -2 的补码形式。

4. 数值函数

BASIC 提供了一些用于计算的数值函数，如 SQR(开平方)，SIN (求正弦值)等，这些函数将在 1.2.7 节介绍。此外，BASIC 语言也允许用户自己定义函数，并提供了定义函数的语句 DEF FN。利用这个语句，用户可以根据自己的需要在程序中定义函数。

在表达式中，函数可以象变量一样使用。

5. 表达式的执行顺序

一个表达式含有多种运算，计算机按一定的顺序对表达式求值。一般顺序如下：

(1)首先进行函数运算。

(2)接着进行算术运算，其次序为：乘幂(^)—— 一元运算(-)—— 乘法类(*，/，\，MOD)—— 加法类(+，-)。

(3)然后进行关系运算。

(4)最后做逻辑运算，顺序为：NOT — AND — OR — XOR — EQV — IMP

此外，同一级的运算顺序为从左到右。为了改变运算顺序，可以使用括号，因为括号中的运算可以先进行；在同一层括号中，仍按正常的顺序运算。

下面的例子是一些代数式及其相应的 BASIC 表达式：

代数式	BASIC 表达式
$x+2y$	X+2*Y
$x-\frac{y}{z}$	X-Y/Z
$\frac{xy}{z}$	X*Y/Z
$\frac{x+y}{z}$	(X+Y)/Z
$(x')^y$	(X^Z)^Y
x^{yz}	X^(Y*Z)
$x(-y)$	X*(-Y)

在书写表达式时，要注意以下几点：

(1)乘号(*)不能省略，也不能用·代替。

(2)不允许两个运算符相连，必须用括号分开。例如，3*(-2)，不能写成 3*-2。

(3)在表达式中只能使用圆括号，不能使用方括号或花括号。

(4)乘幂^表示自乘。如 A^B 表示 A^B，即 B 个 A 连乘。当 A 和(或)B 不是单个常数或变量时，要用括号括起来，例如(A+B)^(C+2)，不能写成 A+B^C+2。

1.2.6 字符串表达式及其运算

与数值变量不同，对字符串变量不能进行算术、逻辑及其他运算，字符串运算只有两种，即连接和函数。

字符串表达式可以由一个字符串常量或字符串变量组成，也可以是若干个字符串常量或字符串变量的组合。字符串运算的目的是用不同的方法对字符串进行排列。

1. 字符串连接

所谓字符串连接，就是把两个或多个字符串拼接在一起，这个操作由字符串连接

符(即"+")来实现。例如:

```
10 COMPANY$ = " Great Wall "
20 TYPE $ = "␣Personal ␣ "
30 FULLNAME $ =TYPE $ + "Computer "
40 PRINT COMPANY$ + FULLNAME$
```

上述语句的执行结果为:

Great Wall Personal Computer

2. 字符串函数

字符串函数与数值函数的使用方法基本相同，唯一的区别是字符串函数返回字符串，而数值函数返回的是数值。BASIC 中提供了相当数量的内部函数，可以把它们放在表达式中。

如同可以由用户定义自己的数值函数一样，也可以用 DEF FN 语句定义用户自己的字符串函数。

除上述两种运算外，也可以用关系运算符 =、< >、<、>、<= 以及 >= 对两个字符串进行比较运算。不过一般来说，关系运算不是字符串运算，因为由关系运算符对字符串比较所产生的结果是数值，不是字符串值。而字符串运算符用来进行字符串运算时，所产生的结果是一个字符串值。

1.2.7 函数

BASIC 语言中有很多函数，其中有些是通用的，有些则与某种操作有关。下面介绍数值函数和字符串函数，其他函数，诸如与绘图、输入/输出等有关的函数将在以后有关部分中介绍。

函数实际上是 BASIC 解释系统提供的一些标准子程序，它可以减轻用户编制程序的工作量。在执行时返回数值的函数叫做数值函数，返回字符串的函数叫做字符串函数。

1. 数值函数

BASIC 的数值函数大致可分为以下几类:

(1)数值转换函数。 SGN,ABS,INT,CINT,FIX,CDBL,CSGN。

(2)三角函数。 SIN,COS,TAN,ATN。

(3)随机数函数。 RND。

(4)其他函数。 EXP,LOG,SQR,FRE。

上述函数和功能如表 1.5 所示。

表 1.5　BASIC 数值函数

名　称	格　式	功　　能
SGN	V＝SGN (X)	返回表达式 X 的值的符号
ABS	V＝ABS (X)	返回数值表达式的绝对值
INT	V＝INT (X)	返回一个不大于自变量的最大整数
CINT	V＝CINT(X)	把表达式的值转换为整数
FIX	V＝FIX (X)	对自变量的值截断取整
CDBL	V＝CDBL(X)	把一个数值转换为双精度数
CSGN	V＝CSGN(X)	把一个数值转换为单精度数
SIN	V＝SIN (X)	计算 X 的正弦值
COS	V＝COS (X)	计算 X 的余弦值
TAN	V＝TAN (X)	计算 X 的正切值
ATN	V＝ATN (X)	计算 X 的反正切值
EXP	V＝EXP (X)	返回 e^X，e＝2.718282
LOG	V＝LOG (X)	计算 X 的自然对数
SQR	V＝SQR (X)	返回自变量的平方根
FRE	V＝FRE (X)	返回数据空间中未被使用的字节数
RND	V＝RND [(X)]	返回一个 0 — 1 之间的随机数

表中的自变量 X是一个表达式。在使用数值函数时，应注意以下几点：

(1)函数 SGN 返回表达式 X 的值的符号，当 X 是正数、0 和负数时，函数分别返回 1，0 和 －1。

(2)INT，CINT 和 FIX 都返回自变量的整数值，但在对数值取整时具体做法不一样，即

• INT 的结果小于或等于被取整的表达式的值。因此，当X为带有小数的正数时，取整后小于 X；而当 X 为带有小数的负数时，取整后的绝对值大于 X 的绝对值。

• CINT 函数的自变量的值必须在 －32768 — 32767 范围内，否则会产生溢出错误。用该函数把 X 转换为整数时，X 的小数部分四舍五入。

• FIX 函数返回 X 的整数部分，即把小数部分完全舍弃，不进行四舍五入处理。

(3)用 CDBL 和 CSNG 进行类型转换时，按 1.2.4 节中所讲的规则处理。

(4)三角函数的自变量一律以弧度为单位，把角度化为弧度的公式如下：

$$\text{弧度} = \text{角度值} \times \frac{\pi}{180}$$

反正切函数 ATN 的自变量可以是单精度型或双精度型的数值表达式，但 ATN(X)的计算结果总是以单精度表示；其他三角函数的结果也是一个单精度值。

(5)随机数函数 RND 的自变量可任选，它的值将影响该函数所返回的随机数。一般来说，RND 的自变量可以是 0、正数或负数。当自变量为 0 时，产生一个与前面随机数序列的最后一个随机数相同的随机数；当自变量相同时，RND 函数所产生的随机数相同；当自变量不同时，所产生的随机数也不同。

用 RANDOMIZE 语句可以为随机数发生器设置"种子"数，这个数不同，所产生的随机数也不同。其格式为

RANDOMIZE [n]

其中n 可选，它是一个整型表达式，其值作为将要产生的随机数的"种子"。当 n 相同时，由 RND 函数产生的随机数序列相同，如果 n 缺省，则在执行该语句时显示：

Random Number seed (-32768 to 32767)?

要求输入一个"种子"数，其范围在 -32768 — 32767。

(6)FRE 函数的自变量是一个"虚"自变量，可以是任何数值或字符串表达式。所返回的值不包括解释程序工作区保留部分的长度。

2. 字符串函数

一般来说，字符串函数的结果是一个字符串值。但在下面列出的函数中，有些函数的结果为数值，这主要考虑到它们所处理的对象是字符串，因而没有放在数值函数中。和前面一样，作为函数自变量的 X$、Y$、m、n 等一般均代表字符串表达式或数值表达式。

BASIC 中字符串函数的格式和功能如表 1.6 所示。

表 1.6 BASIC 字符串函数

名 称	格 式	功 能
ASC	V=ASC(X$)	返回 X$ 的第一个字符的 ASCII 码
CHR$	V$=CHR$(n)	把 ASCII 码转换成相应的字符
HEX$	V$=HEX$(n)	返回十六进制表示的字符串
INSTR	V=INSTR([n,]X$,Y$,)	在 X$ 中检索第一次出现的 Y$，并返回 Y$ 在 X$ 中的位置
LEFT$	V$=LEFT$(X$,n)	返回 X$ 左端开始的 n 个字符
LEN	V=LEN(X$)	返回 X$ 中字符的个数
MID$	V$=MID$(X$,n[,m])	返回 X$ 的一个子串，其长度为 m 个字符，从第 n 个字符开始
OCT$	V$=OCT$(n)	返回八进制表示的字符串
RIGHT$	V$=RIGHT$(X$,n)	返回 X$ 最右边的 n 个字符