



学生应知信息知识

Quick Basic 简明教程(下)

秋登峰 主编

目 录

第十章 字符处理	1
10.1 字符串变量	2
10.1.1 字符	2
10.1.2 字符串	2
10.1.3 字符串长度	2
10.1.4 求字符串长度函数 LEN	3
10.1.5 字符串常量的定义	4
10.2 字符串变量和数组	5
10.2.1 字符串变量的定义	5
10.2.2 字符串数组	8
10.3 字符串变量的赋值	10
10.3.1 用 LET 语句赋值	10
10.3.2 用 INPUT 语句赋值	10
10.3.3 用 READ...DATA 语句赋值	11
10.3.4 用 LINE INPUT 语句赋值	13
10.4 字符串表达式及字符串的比较	14
10.4.1 字符串表达式	14
10.4.2 字符关系表达式	15
10.4.3 两个字符串大小的比较	16
10.4.4 字符串的检索	18
10.5 取子字符串	21
10.5.1 LEFT\$函数	21
10.5.2 RIGHT\$函数	22
10.5.4 删除字符串的首尾空格	24
10.6 字符串的生成	25
10.6.1 STRING\$函数	26
10.6.2 SPACE\$函数	26

10.6.3	字符串中大小写字母之间的转换	27
10.6.4	改变字符串中的字符语句 MID\$	28
10.7	字符串与数值的相互转换	30
10.7.1	ASCII 码与字符的相互转换	30
10.7.2	数值与字符串的相互转换	31
10.7.3	数制与数制之间转换	33
10.8	自选输出格式	34
10.8.1	屏幕定位语句 LOCATE	34
10.8.2	屏幕格式输出语句 PRINT USING	35
10.9	本章小结	40
	习题	42
第十一章	屏幕控制和做图	43
11.1	屏幕坐标系	44
11.1.1	文本方式与字符坐标系	44
11.1.2	图形方式与点坐标系	45
11.2	屏幕方式及颜色的设置	47
11.2.1	设置屏幕方式语句 SCREEN	47
11.2.2	屏幕颜色设置语句 COLOR	49
11.3	基本绘图语句	52
11.3.1	画点语句 PSET 及 PRESET	52
11.3.2	画直线和矩形框语句 LINE	54
11.3.3	画圆、椭圆和画弧语句 CIRCLE	57
11.3.4	连续画线语句 DRAW	61
11.4	填涂颜色语句 PAINT	65
11.5	图形的窗口操作	68
11.5.1	窗口语句 WINDOW	68
11.5.2	视窗语句 VIEW	70
11.6	本章小结	71

习题.....	72
第十二章 文件.....	73
12.1 文件的概念.....	73
12.1.1 文件的分类.....	74
12.1.2 程序文件与数据文件的区别.....	75
12.1.3 文件属性.....	76
12.2 顺序文件.....	77
12.2.1 数据文件的基本概念.....	77
12.2.2 记录.....	79
12.2.3 用户类型定义语句 TYPE.....	79
12.2.4 定义和使用记录变量.....	81
12.2.6 嵌套记录.....	84
12.3 顺序文件.....	86
12.3.1 顺序文件的存放格式和特点.....	86
12.3.2 建立和打开顺序文件语句 OPEN.....	88
12.3.3 关闭文件语句 CLOSE.....	90
12.3.4 把数据存储到文件中.....	91
12.3.5 读取顺序文件中的数据.....	97
12.4 随机文件.....	106
12.4.1 随机文件的和格式及特点.....	106
12.4.2 建立和打开随机文件 OPEN.....	107
12.4.3 定义随机文件缓冲区中的字段 FIELD.....	107
12.4.4 把数据存储到随机文件中.....	108
12.4.5 读取随机文件中的数据.....	111
12.4.6 用记录类型处理随机文件.....	112
12.5 文件与目录维护语句.....	115
12.6 本章小结.....	117
习题.....	118

第十三章 综合程序设计概要	119
13.1 结构化程序设计方法.....	120
13.1.1 自顶向下设计	120
13.1.2 模块化	122
13.1.3 结构化编码	124
13.1.4 软件工程的概​​念	125
13.2 程序设计综合举例.....	126
13.2.1 用高斯消元求逆矩阵	127
13.2.2 打印万年历	134
13.2.3 “菜单”技术	139
13.2.4 陷阱技术	148
13.2.5 快速排序	153
13.2.6 模拟技术	158
习题.....	162

第十章 字符处理

前面主要介绍了计算机在数值计算中的应用，只涉及了一些简单的字符处理。事实上计算机不仅在数值计算上应用广泛，而且在非数值计算中应用范围正在不断扩大，尤其是用于事务管理领域。如图书检索，人事管理，教务管理，经济信息管理等。有关资料表明，计算机在非数值计算中的应用，要远远超过计算机在数值计算中的应用。这就要求计算机具有相当强的文字处理能力。如果一台计算机没有文字处理功能，充其量只能算是一个高级的计算器。Quick BASIC 的一个非常重要的功能就是可以进行字符处理，而且它的字符处理功能十分强大。它提供了字符串变量，字符串数组，字符串语句以及丰富的字符串函数，使用起来直当灵活方便。本章主要介绍 Quick BASIC 对字符串的各种处理功能，涉及的主要内容有：

- 字符串变量
- 字符串变量和数组字符
- 字符串变量的赋值
- 字符串表达式及字符串的比较
- 取子字符串
- 字符串的生成
- 字符串与数值的相互转换
- 自选输出格式

10.1 字符串变量

10.1.1 字符

字符是指单个字母、数字或其他特殊符号。本书附录中的 ASCII 码表中，列出了 IBM-PC 系列微机中的字符集。

10.1.2 字符串

所谓“字符串”就是系统允许使用的若干个字符构成的序列，也称为字符串常量。一个字符串一般是用双引号括起来的一串字符。例如：

"abcdefg" "1234567" "This is a book"

字符串常量与数值常量一样，在程序执行过程中其值是不变的。

Quick BASIC 的字符串中可以使用的字符主要是在键盘上出现的字符，包括：26 个英文字母（大写字母 A~Z、小写字母 a~z），数字 0~9，标点符号，空格，数学符号，专用符号（@、¥、~、& 等）。除了键盘上出现的字符外，还可以包括用 ASCII 码形式给出的图形字符或控制字符。

10.1.3 字符串长度

字符串中的每一个字符在内存中占一个字节。一个字符串占据一段连续的内存单元。例如字符串

“ ABCDEF ”有 7 个字符，因此它在内存中占 7 个单元。

字符串的长度是指字符串常量中的字符的个数。例如：

"This is a book" 该字符串的长度为 14 个字节

引号不是字符串的值，它只是字符串常量的起、止界限，而且在引号中不能再套引号。

例如：

"I say: "This is a book"" 这是 Quick BASIC 中不允许的

一个汉字占两个英文字符的宽度。Quick BASIC 规定一个字符串最多可以容纳 32767 个字符。

10.1.4 求字符串长度函数 LEN

Quick BASIC 提供的 LEN 函数可以方便、准确地测出字符串的长度。

格式：LEN (字符串表达式)

功能：求字符串表达式中字符的个数，即字符串的长度。

说明：字符串表达式 既可以是字符串常量，也可以是字符串变量。

例如：

PRINT LEN ("ABCDEFG") 输出： 7

PRINT LEN (" 10*20 = ") 输出： 6

PRINT LEN (" ") 输出： 1 (空格也为字

符)

10.1.5 字符串常量的定义

字符串常量是指在程序运行过程中始终保持不变的字符串。Quick BASIC 允许使用两种形式的字符串常量。

(1) 显示的字符串常量

显示的字符串常量就是用双引号括起来的一串字符。如："I am a student."

在使用字符串时应注意以下几点：

空格或者间隙也是一个字符，应计在字符串长度之内。

字符串中，大小写字母是有区别的。如"ABCD"与"abcd"被看成是 2 个不同的字符串。

无任何字符的串叫“空串”，用""来表示，空串的长度为零。

应将字符串与数值严格区分开来。例如：12345 是一个数值，而"12345"则是一个字符串，其中每个都是独立的字符。字符串不能用作算术运算。

(2) 符号字符串常量

用一个符号名代表一个字符串常量：

```
CONST xm="Wang Hong"
```

在本程序块中，xm 与"Wang Hong"等价，如果有：

```
PRINT xm
```

则输出 Wang Hong。

为了阅读程序方便，可以在符号常量名后加“\$”，以表明是字符串常量。如：

```
CONST xm$="Wang Hong"
```

注意把符号字符串常量与字符串变量相区别。字符

串常量的值是不能改变的,例如下面的程序段是非法的:

```
CONST xm$="Wang Hong"           '定义了一个符号字符串变
量 xm$="Wang Hong"
xm$="Li Qiang"                   '又要用赋值语句给字符串变
量 xm$赋值
```

10.2 字符串变量和数组

10.2.1 字符串变量的定义

存放数值的变量称为数值变量,存放字符串的变量就称为字符串变量。数值型变量的值是一个数值,而字符串变量的值则是一个字符串,并且,程序执行的过程中,这个值是可以改变的。

Quick BASIC 规定了两类字符串变量:变长字符串变量和定长字符串变量。

1. 变长字符串变量

在程序执行过程中,变长字符串变量长度可在 0~32767 范围内增加或减短,这种变量使用起来比较方便灵活。

定义变长字符串变量类型的方法有 3 种:

(1) 用变量名加上类型申明符

格式:变量名\$

变量名的命名规则与数值变量相同。例如:al\$,b\$,erx\$,dap\$等都是合法的变量名。例如:

```
jtdz$"kaifeng"
```

其中 jtdz\$ 就是一个字符串变量,现在已将字符串

"kaifeng" 赋给了变量 jtdz\$, 以后需要输出字符串 "kaifeng" 时 , 就和使用数值变量一样 , 直接在 PRINT 语句中输出变量 jtdz\$ 即可。

例如 :

```
PRINT jtdz$
```

输出结果为

```
kaifeng
```

(2) 用 DEFSTR 类型说明语句

用类型说明语句 (DEFSTR 语句) 来定义以某个字母开头的变量为字符串变量。

格式 : DEFSTR 字母表

其中 , 字母表是单个字母或一段连续的字母范围。

例如 :

```
DEFSTR a,m-p
```

```
ma="ABCD"
```

```
PRINT ma
```

上例中 ,DEFSTR 语句是说明以 a 和 m,n,o,p 开头的所有变量为字符型变量 ,ma 在的说明的范围内 , 所以是字符串变量。

(3) 在 DIM 语句中使用 AS STRING

格式 : DIM 变量名 AS STRING

例如 :

```
DIM xm AS STRING
```

```
xm="Li Qiang"
```

```
PRINT xm
```

应注意 , 用 DIM...AS STRING 语句定义字符串变量时 , 变量名不应包含类型申明符 , 例如下面的用法是非法的 :

```
DIM xm$ AS STRING
```

将 xm 写成了 xm\$

2. 定长字符串变量

变长字符串虽然灵活方便，但人们有时希望在某些字符串取固定的长度，尤其是在打印输出时使上下对齐。

定长字符串是指它在程序执行过程中，始终保持其长度不变的字符串。定长字符串变量的长度要用 DIM 语句进行说明。

格式：DIM 变量名 AS STRING*n

例如：

```
DIM aa AS STRING*8
```

在定义了 aa 为定长字符串变量之后，可以用赋值语句为其赋值。上面的语句说明了 aa 的字符长度为 8，如果赋的值长度超过了 8，则多余的字符被截去，如果少于 8，则在其后补足空格。例如：

```
aa$="ABCDEF12345"      '其值为："ABCDEF12"
```

```
aa#="ABC"              '其值为："ABC    "(后面补 5 个空  
格)
```

串变量在未赋值之前的初值为空字符串，对于可变长字符串，其长度为 0；而对定长字符串，其他长度为 n，即使用空格为其赋值，如 aa\$="" 其长度不变，也为 8。

使用定长字符串，可以实现定格输出。

[例 10-1] 阅读下面的程序。

```
EXAMPLE 1
CLS
DIM a AS STRING *6
FOR i=1 TO 4
READ a
PRINT i,a
NEXT i
```

```
DATA ABCDEFGHIJKLMN,ABC,XYZOP ,&&&&&&&&
END
```

运行结果为

```
1   ABCDEF
2   ABC
3   XYZOP
4   & & & & &
```

从上例的输出结果可以看出，字符串是以左边对齐，右边补空格的形式输出的。

10.2.2 字符串数组

字符串数组的概念与前介绍的数组相似，即字符串数组是一组有序的字符串变量的集合。字符串数组中的元素称为“串下标变量”。根据字符串数组的维数，同样可分为：一维、二维、三维等。例如，

```
a$( 28) ,b$( 2,3) ,c$( 10,20,30)
```

其中 a\$,b\$,c\$均为字符串数组名，其下标书写规则同数组下标变量。在同一程序中，字符串数组名与字符串变量名可以相同。当程序中使用字符串数组时，同样使用 DIM 语句进行说明。一个 DIM 语句可同时对字符串数组和数值数组进行说明。例如：

```
DIM a( 8) ,b( 2,3) ,a$( 5) ,cb$( 4,5)
```

[例 10-2] 有一班级学生，要求按学生姓名的汉语拼音顺序输出。

问题分析：建立数组来存放学生姓名，用字符串比较的方法来实现顺序输出。程序为：

EXAMPLE 2

```
CLS
DIM a$ ( 6 )           '定义字符串数组
FOR i=1 TO 6           '各数组元素读入数据
READ a$ ( i )
NEXT i
FOR i=1 TO 5           '排序
p=i
FOR j=i+1 TO 6
IF a$ ( j ) <a$ ( p ) THEN p=j
NEXT j
IF p<>i THEN SWAP a$ ( p ) ,a$ ( i )   '交换数组元素的值
NEXT i
FOR i=1 TO 6           '排序后输出
PRINT a$ ( i )         '输出
NEXT i
PRINT
DATA "AHANG SAN","LIU FENG","WANG AHI"
DATA "SONG HONG","YANG PING","KANG XIANG"
END
```

程序运行结果为：

```
KANG XIANG
LIU FENG
SONG HONG
WANG HAI
YANG PING
ZHANG SAN
```

10.3 字符串变量的赋值

我们已经知道要把一个数值送给一个数值型变量，可以使用 LET、INPUT、READ...DATA 语句来实现。同理也可以使用这 3 种赋值语句字符型变量赋值，另外还可以用行输入语句 LINE INPUT 语句来为串变量赋值。

10.3.1 用 LET 语句赋值

与数值变量相同，但要注意字符串常量必须用双引号括起来。

例如，下面的程序用 LET 语句对字符型变量赋值。

```
CLS
LET b$="BeiJing"
LET c$="ShangHai"
PRINT b$,c$
END
```

运行结果为

```
BeiJing      ShangHai
```

10.3.2 用 INPUT 语句赋值

INPUT 语句也可以用来从键盘输入字符串数据，其用法与从键盘输入数值数据基本相同。但要注意：

(1) 在键盘上输入字符串变量的值时，字符串可用引号也可不用引号括起来。但在下列情况下必须用引号括起来：

字符串中有首尾空格时，要用引号括起来。否则首尾空格被删去。

字符串中带有逗号时，要用引号括起来。因逗号是两个数的分隔符，否则一个字符串将在逗号处分开。

(2) 在一个 INPUT 语句中可以既有数值型变量又有字符串变量，当从键盘回答时，应键入相应的数字和字符串。否则把一个字符串赋给一个数值型变量时将会出错。

(3) 建议初学者在用 INPUT 语句给变量赋值时，最好用引号将字符串括起来，以避免不必要的出错。

[例 10-3] 阅读下面的程序。

```
EXAMPLE 3  
CLS  
INPUT"姓名：";xm$  
INPUT"学号：";xh$  
PRINT xm$,xh$
```

运行结果为：

```
姓名：?  Wang  Hong  
学号：?  2000102  
Wang Hong           2000102
```

10.3.3 用 READ...DATA 语句赋值

READ...DATA 语句除了可以用来读入数值变量外，还可以读字符串。其用法与数值型数据基本相同。但要注意：

(1) 在一个 READ 语句中既可以出现数值型变量，也可以出现字符串变量，同样在 DATA 语句中也可以出

现这两类数据。

(2) 如果 READ 语句中既有数值型变量,又有字符型变量,那么 DATA 语句中相应的数据类型,必须与 READ 语句中相应的数据类型一致,否则将会出现数据类型不匹配的错误。

(3) DATA 中的字符串可用引号也可不用引号括起来。但在下列情况下必须用引号括起来:

字符串中有首尾空格时,要用引号括起来,否则首尾空格被删去。

字符串中带有逗号时,要用引号括起来。因逗号是两个数的分隔符,否个字符串将在逗号处分开。

[例 10-4] 编制程序,输出学生姓名、学号、年龄、成绩,假设总共有 6 个学生。

程序为:

```
EXAMPLE 4  
CLS  
PRINT"姓名","学号","年龄","总分"  
FOR i=1 TO 6  
READ na$,no$,age,totaol  
PRINT na$,no$,age,totaol  
NEXT i  
DATA "丁红莉","101",15,578  
DATA "李小翔","102",16,532  
DATA "刘大勇","103",15,512  
DATA "杜姗姗","104",17,590  
DATA "孙 红","105",16,580  
DATA "张小华","106",16,587  
END
```