第四章　扩展BASIC语句

PC—1500计算机除配有基本BASIC语言以外，还配有一部分扩展的BASIC，主要的有下面几个方面的扩展功能：

1.字符串变量以及有关的函数。

2.控制转向语句（即开关语句）及错误处理语句。

3.格式输出，X—Y座标绘图功能及有关的一些语句。

4.图形、文字编制与显示。

5.程序调试检查语句。

6.在磁带机配合下，对“文件”进行处理的功能。

在第二章和第三章中，有些功能已经作了介绍，如自选格式的USING语句，调试程序的TRON与TROFF语句。对字符串变量也作过一些简单的介绍。

本章将着重讨论字符串变量及有关的函数运算、开关语句、显示格式、简单的图形与文字编制等。

为了讲解方便，对打印机的格式输出打印及X—Y座标绘图留在下一章有关打印机及其功能语句中去讨论。而有关“文件”处理功能留在第六章中去介绍。

第一节　字符串变量

（一）字符串变量的概念

在前面我们已介绍过，我们如果想打印或显示字符串，就要在PRINT或LPRINT语句中用引号把字符串括起来，如：

```
120  PRINT  "BEJING", "SHANGHAI"
```

或者 120 LPRINT"ABC"; "DE—F"; "GH—3"

在语句执行时，引号中的字符串（包括空格）全部照原样打印或显示。如果要多次打印，则要多次写，十分麻烦。若用字符串变量将方便多了。

1.字符串变量的形式：

〈标识符〉$

标识符可以是一个或二个英文字母，或一个字母后跟一个数字，如：A$，BH$，WQ$等都是可以的。

$本是美元符号（['dɔla]）的缩写，在这里作为字符串变量的标志，以区别于数值变量。

有了这个变量，我们就可以对它“赋值”，赋值的内容必须是字符串，如下面程序：

```
10  A$ ="BEJING"
20  B$ ="SHANGHAI"
30  PRINT  A$, B$
RUN〔BEJING  SHANGHAI〕
```

PC—1500机规定每个字符串变量不加扩充说明时最多可以容纳16个字符，如超过16个字符机器自动截除。

例（1）
```
10  A$ ="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
20  PRINT  A$
RUN 〔ABCDEFGHIJKLMNOP    〕
```

因超过16个字符，所以截除一部分，只留下前面16个字符。

2. 字符串变量的数组说明语句：

与简单变量一样，字符串变量也可用数组说明语句说明，如：

```
DIM  A$ (3)
```

表示A$字符串变量数组，共有4串（0～3）。

```
DIM  A$ (4, 3)
```

表示A$字符串变量数组是二维的，共有5行4列。

3. 字符串变量的扩充：

前面已说了，每一个字符串变量不管是否有定维说明，它最多只能容纳的字符只有16个，如果超个16个的话，则可按下法处理，以至可以扩充到每个字符串变量达80个字符，如：

① DIM B$ (5) *20
② DIM X3 $ (52) *18
③ DIM NM$ (1) *80
④ DIM A$ (4, 5) *18
⑤ DIM R4 $ (0) *26

上5例中表示：①B$变量界偶为5，每个字符串变量元素的字符长最多为20个字符；②X3 $变量界偶为52，每个变量长最多为18个字符；④A$变量是二维的，其元素有5行6列，每个变量字长最多为18个字符；⑤一个字符变量R4 $（0），其字符长为26个字符。

4. 字符串变量的联接合并：

字符串变量可以用下列方式联接合并。

例（2）
```
10  A$ ="EFG"
20  B$ ="ABCD"
30  PRINT  B$ +A$
RUN 〔ABCDEFG〕
```

注意：30语句中的+号的含义不是相加，而是连接合并的意思。

例（3）
```
10  DIM  S$ (0) *25
20  B$ ="AND", A$ ="ER"
30  C$ ="MOTH"
40  D$ ="GR"
50  S$ (0) ="WRITE  YOUR" +D$ +B$
60  PRINT  S$ (0) +C$ +A$
```

RUN 〔WRITE YOUR GRANDMOTHER〕

5. 几点说明：

①字符串变量，在未予定维扩充时，最大长度为16个字符。

②赋给字符串变量值，必须用引号括起来，否则算出错，显示〔ERROR 7 〕，数字也可赋给字符串变量，也应加引号，此由数字符号组成的字符串变量，是不能参加数值运算的。

③在PRINT语句中，字符串变量如有多个的话，也要用分号与逗号分开。标点符号使用规则与简单变量一样。

④字符串变量的清除，可用CLEAR指令。对单个的字符串变量清除，可用下述办法清除：A$="。即在赋值号后写半边引号送入机器。

⑤A$～Z$，也可用@$（1）～@$（26）表示，类似固定数值变量。

（二）在输入语句中使用字符串变量

在前面我们已讲了用LET语句给字符串变量赋值。下面讲另两种输入语句的赋值法：

1. 在READ～DATA语句中使用字符串变量：

先见下例：

例(4)
```
10 READ A$, B$, C$
20 PRINT A$; B$; C$
30 DATA "ABC", "DEF", "1 2 3"
40 END
```
RUN 〔ABCDEF 1 2 3〕

用DATA语句给READ语句提供“字符串数据”的形式与简单变量的形式一样，只是字符串数据必须用引号括起来。

我们在DATA语句中，还可以混合使用字符串变量、简单变量和下标变量的数据，但是必须注意，在DATA语句中，“数据”的类型必须和READ中相应的变量类型一致。

如：
```
10 READ A$, B$, C, D, X(1)
20 DATA"CHINA", "BEJING", 1982, 36479, 53.4
```
2. 在INPUT语句中使用字符串变量：

同样，我们可以通过键盘用INPUT语句给字符串变量赋值，也可以用INPUT语句给字符串变量、简变、下标变量混合赋值。

例(5)
```
5 DIM A$(0)*20
10 INPUT "WHAT IS YOUR NAME?"; A$(0)
20 PRINT "POUR NAME IS"
30 PRINT A$(0)
40 END
```
RUN 〔WHAT IS YOUR NAME?〕

由键盘输入： CHANG HAO FU

按执行键显示：〔YOUR NAME IS 〕

再按执行键：〔CHANG HAO FU 〕

例(6) 混合输入的例子

```
10  INPUT  A$, C, B$, D
20  PRINT  C; D; A$; B$
30  END
```

可以自己假设一些字符串数据与数值数据由键盘输入，注意不要将字符串数据与数值数据输入顺序搞错。如果搞错了，若将数值数据送入了字符串变量，那么该数值变量无法进行运算，而只能作为文字符内容的数符号。若将字符数据输入数值变量，则数值变量不接受，显示该数据单元里仍是零。

可以看出，与DATA不同的是，在INPUT语句执行时，输入机器的字符串变量可以不加引号括起来，不管字符串中有什么字符、标点及数字均可。但是每打入一串字符串以后，必须按执行键送入。

(三) 字符串的比较

1.字符ASCII码与ASC函数。

我们都知道，所有的字符、数字在计算机中，都是以二进制的代码来记忆、存贮和运算的，目前都是用国际上通用的ASCII码表示 (PC—1500的ASCII编码参看附录一)，在机器的操作中，我们可以对应不同的字符，用ASC函数形式调出每个字符的ASCII码。机器输出时，自动转换为十进制码，例如：

```
10  LET  A$ = "H"
20  LET  A = ASC  A$
30  PRINT  A
RUN 〔   72〕 显示H的ASCII码为72 (十进制)
          (H的二进制码为：1001000)
```

ASC函数对一串字符或字符串变量中有多个字符时，它只取第一个字符的ASCII 码。

PC—1500机ASCII码的十进制数码如表 4 — 1 所示：

从表 4 — 1 中可以读出各字符的十进制ASCII数码。

例如：A的代码是65；

m的代码是109；

√ 的代码是91；

……等等。今后为方便我们说字符串变量的ASCII代码都用十进制表示。

2. 字符串变量的大小比较：

字符串变量大小的比较，实际上是它们的ASCII数码的比较。

如：A＜B，因为A的代码是65，B是66。

所以：B＞A

同理：W＞C

3＞#

b = b

对于两个字符以上组成的字符串比较，其原则是：

PC—1500机ASCII十进制数码表　　表 4—1

低位＼高位	3	4	5	6	7	8	9	10	11	12
0		)	2	<	F	P	Z	d	n	x
1		(	3	=	G	Q	√	e	o	y
2		*	4	>	H	R	¥	f	p	z
3	!	+	5	?	I	S	π	g	q	{
4	"	,	6	@	J	T	∧	h	r	¦
5	#	−	7	A	K	U	−	i	s	}
6	$	.	8	B	L	V		j	t	~
7	%	/	9	C	M	W	a	k	u	■
8	&	0	:	D	N	X	b	l	v	
9	□	1	;	E	O	Y	c	m	w	

①先比第一个字符，谁的第一个字符大则谁大，若第一个相等，再比第二个，谁第二个字符大则谁大，依此类推。

②两个字符串长短不同，不是看谁长，而是先以①的原则比较，如前面都相等，才是谁长谁大。例如：

"PENCIL">"PEN"，"YOU">"YOUR"。对于字符串的大小，可以有这么一个简单的记法，即：在英文字典中的顺序，也就是它们大到小的顺序。

例(7) 有一批国家名，用字符串变量的比较法，找出按英文字母顺序在最前面的那个来，各国名放在DATA语句中。编程序如下：

```
10  CLEAR : READ  A$
20  FOR  I = 1  TO  8
30  READ  B$
40  IF  A$ >= B$  LET  A$ = B$
50  NEXT  I
60  LPRINT  A$
70  END
100  DATA "CHIXA", "JAPAN", "KOREA", "CANADA",
     "AUSTERITY"
110  DATA "ENGLAND", "FRANCE", "AMERICA", "INDIA"
RUN 〔AMERICA 〕
```

对此例题，若让计算机按字母排列顺序把9个国家名全部打印出来。程序应如何编？

（四）子字符串及子字符串函数

我们把一个完整的字符串的一部分调出来，调出来的部分称作“子字符串”。

如在字符串"THE PEOPLES REPUBLIC OF CHINA"中，我们调出"CHINA"，则称"CHINA"为子字符串。

在PC—1500机中，有三个子字符串函数，用它们可以调出字符串的任意一部分。

1. LEFT $函数

形式：LEFT $（〈“字符串”〉，〈数〉）

LEFT $（〈字符串变量〉，〈数〉）

功能：从字符串或字符串变量中，调出从头开始的几个字符。〈字符串〉或〈字符串变量〉给出原字符串，〈数〉中的数值表示调出共有几个字符。

例（8）
```
10 A$=EFT$("DRESSER", 5)
20 PRINT A$
```
RUN 〔DRESS 〕调出前面5个字符。

例（9）
```
10 B$="My name is Lipin"
20 A$=LEFT$(B$, 5)
30 PRINT A$
```
RUN 〔My na 〕

调出前5个字符，空格也算。

2. RIGHT $函数

形式：RIGHT $（〈“字符串”〉，〈数〉）

RIGHT $（〈字符串变量〉，〈数〉）

功能：大致同LEFT $。只是后面的〈数〉表示的含意是，从后面开始数第几个开始调出。或者说调出最后的几个字符。

例（10）
```
5 DIM D$(0)*29
10 D$(0)="THE PEOPLES REPUBLIC OF CHINA"
20 A$=RIGHT$(D$(0), 5)
30 PRINT A$
```
RUN 〔CHINA 〕调出后5个符字。

3. MID $函数

形式：MID $（〈“字符串”〉，〈数1〉，〈数2〉）

MID $（〈字符串变量〉，〈数1〉，〈数2〉）

功能：对字符串（或字符串变量），从数1开始调，共调出〈数2〉个字符。

例（11）
```
10 A$="I NEED HELP"
20 B$=MID$(A$, 3, 4)
30 PRINT B$
```
RUN 〔NEED 〕第3个字符开始，共调4个。

（五）有关字符串的其它字符串函数

1. CHR $函数

CHR$函数是ASC函数的反函数。

形式：CHR$〈ASCII十进码

CHR$〈数字变量〉

功能：将ASCII码还原成字符。它取ASCII十进码从0～127的数。

例（12）
```
10  A=68
20  PRINT  CHR$  A
RUN 〔D          〕
```

例（13）
```
10  PRINT  (CHR$  67)+"OP"
RUN〔COP    〕
```

2.LEN函数

LEN的形式：

LEN〈“字符串”〉

LEN〈字符串变量〉

功能：用LEN函数来测量了解字符串或字符串变量中共有多少个字符数目。

例(14)
```
10  A$="CHINA"
20  B=LEN  A$
30  PRINT  B
RUN〔          5  〕显示5，表示字符串中有5个字符
```

例(15)
```
10  A=LEN "␣␣"
20  PRINT  A
RUN〔      2〕两个空格也算2个字符。
```

例(16)
```
5   DIM  B$(0)*20, G$(0)*35
10  INPUT  A$, B$(0)
20  PRINT  LEN  A$, LEN  B$ (0)
30  RRINT  LEN  A$+LEN  B$  (0)
40  C$(0)=A$+B$(0)
50  PRINT  LEN  C$(0)
60  END
RUN  ?  I  AM  A  STUDENT
     ?  YOU  ARE  A  TEACHER
        〔      14      17  〕
        〔              31  〕
```

例(17) 对键盘输入的字符串进行检验，把长度为5的字符串显示出来。
```
10  INPUT  A$
20  IF  LEN  A$<>5 THEN  40
30  PRINT  "A$=";  A$
40  GOTO  10
RUN  ?  CHINA
```

```
        〔A$=CHINA      〕
    ?   CANADA
        〔          〕
    ?   JAPAN
  〔A$=JAPAN     〕
```

3.STR$函数

STR$函数是将数值直接赋给字符串变量的函数，其一般形式为：

STR$〈表达式〉

用法：如要将数值34.6变为字符串变量A$，就可照下面办法赋值：

A$=STR$34.6

实际上，这个式子等价于 A$="34.6"

又如：B$=STR$(32.3*2)也等价于 B$="64.6"

例(18)
```
10 INPUT I
20 A$=STR$(I*2)
30 PRINT "AS=";A$
40 GOTO 10
```

4.VAL函数

VAL函数与STR$函数相反，把字符串变量中的数符恢复成数值。其一般形式为：

VAL〈字符串变量〉

用法：例如，A$="35.7"，把35.7恢复成数值可照下法：

B=VAL A$

例(19)
```
10 A$="389"
20 B=VAL A$
30 PRINT B
RUN 〔      389〕
```

例(20)
```
10 A$=STR$34
20 PRINT A$+"24";
30 M=VAL A$:M=M*2
40 PRINT M
RUN 〔3424 68      〕
```

例(21)
```
10 A$="AB"
20 PRINT A$+"2";
30 D=VAL A$
40 PRINT "A$=";A$;D
RUN 〔AB2 A$=AB  0     〕
```

要注意一点，用VAL函数时，字符串变量中的字符一定要是数符（如例（20）），如果不是数符而是其它符号的话，VAL函数值一律为0（见例（21））。

5. INKEY $ 指令

INKEY $ 指令叫“读键”指令，它是编入程序之中使用的，其使用的形式为：

〈字符串变量 =INKEY $

其功能是：当程序运行时，INKEY $ 使机器一直处于一种所谓“读键”状态，只要我们按键盘上的任一键，该键的ASCII码（也就是该键的字符）就立刻被赋入指定的字符串变量之中，下次再按其他键时，新的键名字符又被赋入该字符串变量之中，下次再按其他键时，新的键名字符又被赋入该字符串变量之中。

下面例(22)是一个说明其功能的程序例子。这个程序一旦运行起来，将一直循环下去，如果不按键，A $之中是“空串”，由40语句显示〔NO KEY〕，表示没有按键，而程序总处于读键状态。若按一下某一键（除ON键以外），机器马上显示该键的名称，也就是该键的名读入了字符串变量A $。如果一松开按键，程序立即返回到20语句，清除A $的内容，重新显示〔NO KEY〕。如果我们按的键不是文字符号类的键，而是一些功能键，虽然A $读入了该键的ASCII码，但不能显示出来。

```
例(22)  10  WAIT  0
        20  A$ =″
        30  A$ =INKEY$
        40  IF  A$ =″ ″PRINT″NO  KEY″ : GOTO 20
        50  PRINT ″A$ =″; A$ : GOTO 20
```

INKEY $ 指令是一个很有用的特殊指令，我们利用它可以作一些控制之用。如用以控制某段程序的开启、停止等等。

例(23)是利用INKEY $ 指令来控制闹钟的止闹的程序，当电子闹钟闹起来后，将一直不停地叫，只要我们按键盘上任一键，立即就止闹。大家可以通过程序的运行来体会。这程序主要是说明INKEY $ 的用法，所以电子闹钟程序尽量简化了。对闹时间由INPUT语句输入，一定要按TIME函数的要求输入。

```
例(23)  10  WAIT 0 : USING″# # # # # # # • # # # # #″
        20  INPUT  T
        30  TA=TIME
        40  PRINT  TA
        50  IF  TA=T GOTO 70
        60  GOTO  30
        70  A$ =″ ″ : BEEP  1
        80  A$ =INKEY$
        90  IF A$ =″ ″GOTO 70
       100  GOTO 30
```

程序运行时应先输入定闹时间T，当到了该时间，机器即发出声响报警。只要我们按任一键即止闹，电子钟继续恢复正常显示。

（六）字符串变量及其函数的应用举例

下面例(24)是一个自动报时钟的程序，程序运行时将在显示屏上分别显示年、月、日及

时、分、秒的数字，而且每逢半小时叫一声，每逢整点按点鸣笛。如五点整时叫五下，八点整时叫八下。在这个程序中综合使用了STR$、VAL、LEFT$、MID$、RIGHT$等子字符串函数来对TIME函数的时间数值进行分解、合成处理，以使时钟按我们要求的格式显示出来，大家可通过操作运行仔细体会各语句中子字符串函数的一些功能与用法。

例(24) 自动报时钟程序

```
 5  CLEAR : WAIT  0
10  YE = 1984
15  IF TIME>93024 LET A$ = STR$ TIME : GOTO 25
20  A$ = "0" + STR$ TIME
25  M$ = LEFT$(A$, 2)
30  D$ = MID$ (A$, 3, 2)
35  D = VAL(M$ + D$ + "00")
40  IF VAL M$<10 LET M$ = "␣" + RIGHT$(M$, 1)
45  IF VAL D$<10 LET D$ = "␣" + RIGHT$(D$, 1)
50  T = TIME - D
55  IF T>24  THEN 15
60  H = INT T, M = (T - H) * 100, N = INT M
65  S = (M - N) * 100, N$ = STR$ N, S$ = STR$ S
70  IF N<10 LET N$ = "0" + N$
75  IF S<10 LET S$ = "0" + S$
80  IF T>=24 THEN 15
85  IF T<0  LET YE = YE + 1 : GOTO 15
90  PRINT YE; "."; M$; "."; D$; "␣␣␣"; H;
    ":"; N; "␣␣␣"; S$
95  IF H = 0  LET H = 12
100 IF H>12 LET H = H - 12
105 IF N = 0  AND S = 0  BEEP H, 50, 1000 - H * 20
110 IF N = 30 AND S = 0  BEEP 1, 120, 2000
115 GOTO 50
```

此程序启动运行后将永远不会停止，如果要中止只能按ON键中断。运行要注意，必须要预先给TIME函数以正确的时间，那么显示出的时间，日期方是正确的。另外，此程序的第10语句YF = 1984是指定了在1984年运行。如果年号不是1984，则需作相应的修改。

下面例(25)，是用字符串变量进行检索的一个简单例子。

例(25) 用计算机对一批人的姓名、地址、电话号码等信息进行检索。这些信息用DATA语句置入程序之中，下面的表列出了五个人的情况，每人的姓名、地址等均用汉语拼音书写。

姓　　名	地　　址	电　　话
FANG YIN LAN 方　英　岚	Pen mim lu—287 人　民　路 287 号	32465
LIU YU HUA 刘　玉　华	Gulhua hutong—78 桂 花 胡 同 78 号	22433
WANG JIAN NAN 王　建　南	Chengbei jie — 156 城　北　街 156 号	47851
CAI QING CHUAN 蔡　青　川	Deng Sanli — 45 东　三　里 45 号	/
LI PIN 李　平	Hongqi daidao—1425 红 旗 大 道 1425号	23517
⋮	⋮	⋮

程序如下：

```
10  CLEAR
20  DIM D$(0)*25
30  INPUT"NAME IS ? "; X$
40  READ M$, D$(0), P
50  IF M$="NO" BEEP 5 :GOTO 70
60  IF X$=M$  THEN 80
70  GOTO 40
75  PRINT "NO BODY" :GOTO 110
80  LPRINT M$
85  LPRINT
90  LPRINT D$(0) : LPRINT
100 LPRINT "TELEPHON--"; P
110 END
501 DATA "FANG YING LAN", "Ren mim lu—287", 32465
502 DATA "LIU YU HUA", "Gulhua hutong—78", 22433
503 DATA "WANG JIAN NAN", "Chengbei jie—156", 47851
504 DATA "CAI QING CHUAN", "Dengsan li—45", 9E99
```

```
505  DATA "LI PIN", "Hongqi daidao—1425", 23517
 ⋮
1000 DATA "NO", "␣", E
```

程序运行时，输入要询问的人名，机器就会自动找出并打印出要找的人的地址、电话号码。如果无此人(或输入姓名错了)，则计算机发声提醒并显示〔NO BODY 〕，表示在这批名单之中无此人。

关于这个程序的几点说明：

(1)当机器显示〔NAME IS?　　〕询问姓名时，应由键盘输入要找的人名的汉语拼音，并且不能错任一个字符，特别要注意空格也不能少。

(2) DATA语句每一条放置一个人的“信息”，这样显得清晰，也不易错。每人的电话号码是作为数值变量来处理的，如果该人没有电话，其电话号码就用一个特殊的数来代替，如504语句中的电话号码用9E99来代替。千万注意不能不写这个号码，否则整个程序将会把“信息”搞混搞错。

(3)最后加有一条额外的DATA语句，如此例中的第1000语句。这是给出的结束标志，不能少。

(4)这个程序仅仅是一个最简的检索程序，目的是想说明一下字符串变量的实用，(其实例(7)也是一个检索程序)。在这个程序中(如机器装的是8 K模块)约可放置200人的数据。这一批200人的数据可随程序存入磁带保存，也可用置换DATA语句的办法调入一批新的数据来检索。这些方法将在第六章音频磁带机的使用中介绍，在这里就不详述了。

第二节　控制转向与错误处理语句

我们在第三章中已介绍过无条件转向与条件转向语句，那只是一种基本控制语句。GOTO与IF～THEN只为我们提供了一个流向，也就是说，它只能转到某一个所指定的语句，而不能有几个不同的流向。为了使程序编写更为方便和实用，PC—1500机为我们提供了控制转向语句。

1.控制转向句语(开关语句)

控制转向语句的形式为：

$$\text{ON}\langle\text{算术表达式}\rangle\begin{cases}\text{GOTO}\langle\text{第1个语句号}\rangle,\ \cdots\cdots\langle\text{第n个语句号}\rangle\\ \text{GOSUB}\langle\text{第1个语句号}\rangle,\ \cdots\cdots\langle\text{第n个语句号}\rangle\end{cases}$$

当算术表达式的整数部分为1时，转向第一个语句号，当算术表达式的整数部分为2时，转向第二个语句号，……当算术表达式的整数部分为n时，转向第n个语句号。若算术表达式的值大于N或小于1时，则执行ON语句的下一条语句。

从下面例(1)，我们可以看出控制转向语句的功能。

例(1)　有24个学生的成绩(放在DATA语句中)用控制转向语句来统计：90分以上有多少人？80～90分多少人？70～79分多少人？60～69分多少人？60以下的多少人？

根据题意，学生数N＝24，我们可以给出如下表达式：

S/10—5

其中S代表学生成绩。程序编制如下：（由键盘输入N=24）

```
10  INPUT "N=";N
20  FOR I=1 TO N
30  READ S
40  ON S/10-5 GOTO 60,70,80,90,90
50  A0=A0+1
55  GOTO 100
60  A6=A6+1
65  GOTO 100
70  A7=A7+1
75  GOTO 100
80  A8=A8+1
85  GOTO 100
90  A9=A9+1
100 NEXT I
110 PRINT "A0=";A0;"␣␣A6=";A6
120 PRINT "A7=";A7;"␣␣A8=";A8
130 PRINT "A9=";A9
140 DATA 76,84,58,32,91,95,94,88,78,83,82,67,63
150 DATA 69,74,77,100,78,81,92,95,67,49,90
160 END
RUN 〔A0=3   A6=4    〕
    〔A7=5   A8=5    〕
    〔A9=7            〕
```

控制转向语句在有的书上称之为**开关**语句，（有的BASIC系统中，其使用功能稍有差异）。它比IF语句更方便灵活，从程序的流程上来说，它们的本质区别在于：IF语句是一个入口两个出口，而ON～GOTO语句是一个入口多个出口。所以开关语句很适合于多个程序分支的情况下使用，而其关键也就在找一个合适的判断流向的算术表达式。

2. ON ERROR GOTO**语句（错误处理语句）：**

形式：ON ERROR GOTO〈语句号〉

功能：当程序运算中出现某种错误或者超过某规定误差范围，它指令程序转到某一指定语句，以避免或改正所产生的错误。

例(2)

```
10 ON ERROR GOTO 50
20 INPUT "X=";X
30 PRINT "√X";√X
40 GOTO 20
50 PRINT "ERROR"
60 GOTO 20
```

```
70  END
```

例(2)的程序是求平方根。根据规定$\sqrt{X}$中X不得为负数，若键盘输入了负数，机器出现错误，因为有了错误处理语句，机器不停机而指示程序转向第50句。

ON ERROR语句的使用必须特别慎重。我们一般用它是在为防止一些可预见到的错误而使程序运算出错中断的时候，也就是说要有明确的目的而设置。否则它会掩盖一些程序中我们所没有发现的真正错误而使程序运行混乱，以致产生错误的运算结果。

ON ERROR语句一般放在程序的开头部分或者放在需要进行错误处理的程序段落前面。

ON ERROR语句不影响程序的正常运行，如果程序之中没有错误，它的设置相当于没有设置一样。

下面例(3)也是一使用ON ERROR语句的例子。当I=30，I=50，I=80时，程序将对运算进行处理，（在这个程序中60语句就是处理的具体做法，）使程序运行不致中断。

例(3)
```
10  WAIT  32
20  ON ERROR GOTO 60
30  FOR I=1 TO 100
40  Y=1/(I-50)/(I-30)/(I-80)
45  PRINT  Y
50  NEXT  I
55  IF  I=101 THEN END
60  BEEP  3 :GOTO 50
```

ON ERROR语句不能处理第一类错误（句法错误和格式错误）及一些其他语法方面的错误，而只能处理运算之中的错误。

第三节　显示屏的格式输出

在第二章常规方式计算及第三章的有关章节中，曾详细介绍过自选格式函数语句USING的用法。USING是用得很多也很灵活的，在不同的机器上，USING格式函数有些不同的规定，这一点要注意。

在这一节中，着重介绍CURSOR语句(函数)与GCURSOR语句（函数），而TAB格式函数及LCURSOR语句（函数）放在下一章打印语句中去介绍。另外，WAIT语句（函数）也将在此介绍。

一、CURSOR语句

1．CURSOR函数作为语句指令形式在程序中的作用就是指定输出显示的位置。

在PC—1500机显示屏上共有26个可供显示字符的位置，（从0号位到25号位）。如果要指定在7号位显示，则用CURSOR命令来指定就是：CURSOR　7

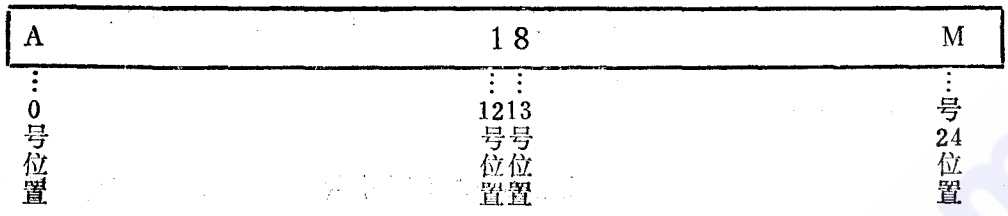
2．CURSOR语句的形式：

〈语句号〉CURSOR〈A〉

其中〈A〉可是数（0≤A≤25），也可是表达式。A的值必须是0～25的正整数，若有

小数，机器自动舍去小数部分。

例（1）在显示屏0号位置显示"A"，在12号位置显示"18"，在24号位置显示"M"，如下图所示：




```
10  PRINT"A"
20  CURSOR  12 : PRINT"18"
30  CURSOR  24 : PRINT"M"
40  END
```

CURSOR命令只与PRINT命令配合，所以它作为语句形式应放在PRINT 语句之前。

CURSOR后的数值不得大于25，否则显示产生19类错误。

二、GCURSOR语句

在PC—1500机的显示屏上，液晶光点共分为7行156列，所以共有156×7个光点组成。在正常显示时，每一个字符的显示是由5列组成（即5×7=35个点），因此156列共可显示26个字符。

CURSOR指令指定的位置，是以一个个字符位置为单位指定。

GCURSOR指令是以点的列位置为单位来指定的，因此其功能类似于CURSOR指令。

GCURSOR的形式：

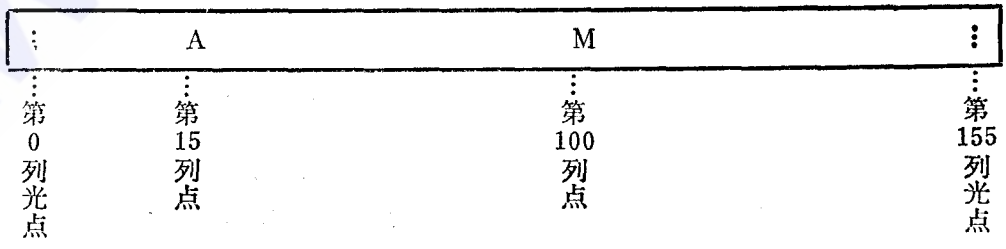
〈语句号〉GCURSOR　A

其中A可为数、表达式、变量。A的取值范围为：0≤A≤155

GCURSOR作为语句形式存在于程序之中，只能放在PRINT语句之前来说明与指定PRINT命令显示的位置。

例（2）从显示点的第15列开始显示"A"，从第100列开始显示"M"。

```
10  GCURSOR 15
20  PRINT"A"
30  GCURSOR 100
40  PRINT"M"
```



GCURSOR后的数值不得大于155，否则机器显示19类错误。GCURSOR的用途主要是在图形显示，详见下一节。

GCURSOR指令在程序中的使用规定同CURSOR，只是其取值范围不同。

三、WAIT语句

我们在第二章已介绍了一种输出语句PAUSE，PAUSE的功能是显示0.85秒就消失，(间歇显示) 并执行下一句语句。在PAUSE语句中，显示的时间是固定的。而WAIT语句提供了可选择的时间。

WAIT语句的形式：

〈语句号〉WAIT〈A〉

其中〈A〉的取值范围为0～65535的正整数，〈A〉若为小数，自动取整。A可为变量与表达式。

WAIT的功能：

由WAIT后的数字，为PRINT语句（或GPRINT语句）提供了一个可选择的时间。而MAIT只对PRINT命令与GPRINT命令有效。（GPRINT语句下节介绍）。

当WAIT 0时，显示很快，无法读出，（或者说，显示语句不予等待）；

当WAIT 64时，显示时间约1分钟；

当WAIT 3840时，显示时间约1分钟。

下面例题通过输入不同的I值，改变时间，观察其运行情况如何。

例（3） 5 INPUT "I=";I
10 A$="AB",B$="CD"
20 WAIT I
30 BEEP 1,5
40 PRINT A$;B$
50 A=12345
60 BEEP 1,5
70 PRINT A
75 BEEP 1,5:GOTO 5

例（4） 10 FOR I=0 TO 102 STEP 2
20 WAIT I
30 BEEP 1,5
40 PRINT "#";
50 NEXT I
60 END

第四节 文字符号与图形的编制显示

PC—1500机主显示屏，是由7行156列共7×156个小液晶点组成的，对于这许多的液晶显示点，都可用GPRINT语句命令调动，从而显示出一些简单的图形，文字。

GPRINT的使用形式有三种：

形式1：GPRINT〈数1〉;〈数2〉;……;〈数n〉

其中：〈数1〉～〈数n〉为十进制的数码，其取值范围为0～127。

形式2： GPRINT 〝〈十六进制编码数串〉〞

其中：〈十六进制编码数串〉的用法写法详见“编码的十六进制方式”中所述。

形式3： GPRINT &〈数1〉; &〈数2〉; ……&〈数n〉

其中：〈数1〉～〈数n〉为十六进制的数，也必须在十进制数0～127之范围内。

GPRINT的功能：由光点的编码（十进制码或十六进制码）指定液晶显示屏光点显示，以构成各种简单图形文字符号。

GPRINT的具体用法详见下面三种编码法的介绍。

GPRINT命令在写入计算机时不能在PRINT命令定义符前加一个“G”字，即使加上去机器也不执行，而显示 ERROR 1。必须一次顺序连贯写出（可以是缩写如GP.）。

一、显示点排列编码的十进制方式

为了能使它们用数学形式的编码来反映各自的位置，机器对每一列（7点）规定了从上至下分别为（十进制）1，2，4，8，16，32，64；也就是2^0，2^1，2^2……2^6。（见图4—1）。

例如我们要显示一个“〈”形状的符号，从图4—1中可以看出，第一列的显示只有“8”点显示，第二列是“4”与“16”显示，以此类推，那么用GPRINT语句来写出就是：

```
10 GRPINT 8; (16+4); (32+2);
     (64+1)
```

或者直接写成：

```
10 GPRINT 8; 20; 34; 65
```

通过运行，它将在显示屏左端上显示一个“〈”形状符号。

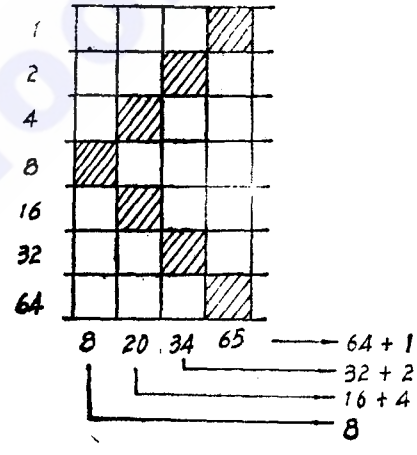


图4—1

这就是十进制编码的方式，写成一般形式为：

〈语句号〉GPRINT数1; 数2; …数n。

其中数1～数n是每列点的十进码。如前所述，它们不得为负数，也应小于1+2+4+8+16+64=127，即：

0≥数≥127

例（1） 用十进制编码法显示汉字“北京”。

首先，应明确“北京”两字应有那些点组成，可在方格上按每列七格进行描绘如图4—2。然后根据光点的组成来进行编码，10语句是选择显示位置从第50列光点位置开始。

程序如下：

```
10 GCURSOR 50
20 GPRINT 68; 68;
     127; 0; 127; 68; 68;
```

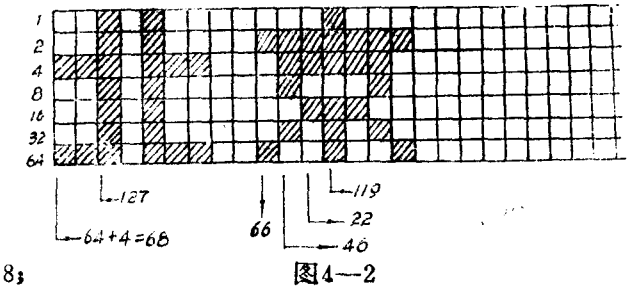


图4—2

```
30  GPRINT 0; 0;
40  GPRINT 66; 46; 22; 119; 22; 46; 66
50  END
```

二、显示排列编码的十六进制方式

1. 十六进制表示数的形式。

在PC—1500机上，用十六进制的数的形式表示十进制数的形式的方法如下：

十 进 制：0，1，2，……9，10，11，12，13，14，15，16，17，……

十六进制：0，1，2，……9，A，B，C，D，E，F，10，11，……

……30，31，32，33，……

……1E，1F，20，21，……

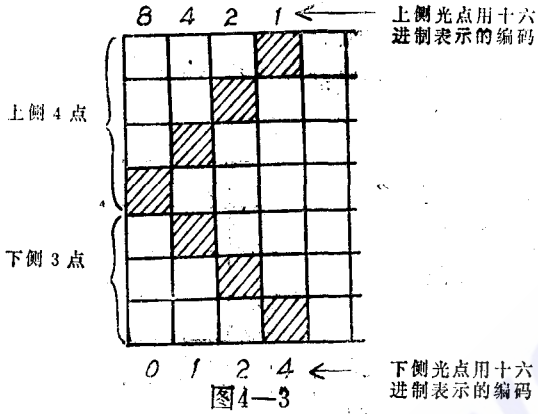


图4—3

可以看出，对十进制的10～15，用十六进制则用 A～F 来表式。逢16进一，所以十进制的16用十六进制则为10。以此类推。

为了在程序中区别于十进制数，在十六进制数前均要加上一个＆符号，如：

＆12表示是十六进制的12（也就是十进制的18）。

＆2A表示是十六进制的2A（也就是十进制的42）。

2. 显示点的十六进制编码显示。

为了方便，将主显示屏每列7点分成上侧4点与下侧3点两组，如图4—3。上侧与下侧的十六进制编码表见图4—4图4—5。

对于图4—3中描绘的“ ”符号用GPRINT语句的十六进制上下各自指定方式如下：

```
        ↓ ↓ ↓ ↓ ← 上侧4点的数码
GPRINT"08142241"
        ↑ ↑ ↑ ↑ ← 下侧3点的数码
```

例（2） 将汉字“北京”用十六进制编码显示。

```
10  GCURSOR 50
20  GPIRNT"44447F007F4444";
30  GPRINT"0000";
40  GPRINT"422E1677162E42"
50  END
```

上侧4点十六进制编码光点图 图4—4

十六进制	光点图	十六进制	光点图	十六进制	光点图	十六进制	光点图
0		1		2		3	
4		5		6		7	
8		9		A		B	
C		D		E		F	

下侧3点十六进制编码光点图 图4—5

十六进制	光点图	十六进制	光点图	十六进制	光点图	十六进制	光点图
0		1		2		3	
4		5		6		7	

第10语句指定从第50列光点开始显示；

第20语句，显示“北”字的语句；

第30语句，空开两列；

第40语句，显示“京”字的语句。

例(3)　将“中国四川”四个汉字编码显示。

先将“中国四川”四个字描出并写出光点的十六进数码，然后编程序。

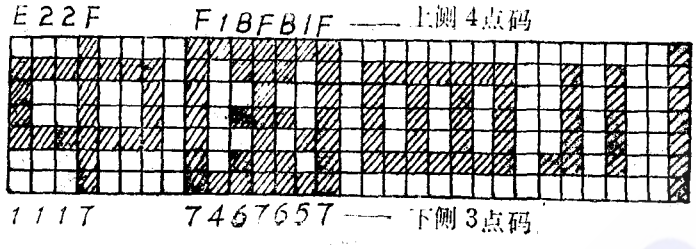


图4—6

```
10  GCURSOR  50
20  GPRINT"1E12127F12121E"; "00";
30  GPRINT"7F416B7F6B517F";
40  GCURSOR  75
50  GPRINT"3E223E223E223E"; "00";
60  GPRINT"203E003E00007F"
70  END
```

例(4)　编制一个坦克战车的活动图形的程序

```
2   A$="205070507C5E7E5E"
4   B$="7C54742040000"
6   C$="040404040404"
10  WAIT  10
20  FOR  I=1  TO  140
30  CLS:GCURSOR  I
35  GPRINT  A$+B$
40  IF I<>INT (I/10)  *10  GOTO  55
45  GCURSOR  I+15
50  GPRINT  C$:BEEP  2, 50, 100
55  NEXT  I
60  END
```

3．　指定十六进制光点编码显示法。

这种方法类似于前面所讲的十进制编码法。只要用十六进制的数码替换十进制数码即可。

如十进制数码显示形式：

```
GPRINT  5；13；54；7；11
```

改为十六进制数码显示形式：

```
GPRINT  &5；&D；&36；&7；&B
```

例（5） 将例（1）的“北京”两字的十进制编码法改为十六进制光点编码法显示。

```
10  GCURSOR  50
20  GPRINT  &44；&44；&7F；00；&7F；&44；&44；
30  GPRINT  0；0；
40  GPRINT  &42；&2E；&16；&77；&16；&2E；&42
50  END
```

从例（4）的程序可以看出，它与例（3）的区别只在于一个是在引号内的，不用分号隔开的十六进制编码。一个是没有引号，而每列的数码用分号隔开，并且每个数码前加了一个&符号。

三、POINT指令

POINT指令可以用来读出光点的任一列的数码数（十进制数码）。如我们想了解第145列光点的十进数码是多少。就可用POINT145来读出（调出）

POINT指令的一般形式：

```
POINT  〈A〉
```

其中：〈A〉可以是数。其取值范围为：0≤A≤155

〈A〉也可以是表达式，变量。

例（6） 在例（1）的程序中，添入45句。

```
10  GCURSOR  50
20  GPRINT  68；68；127；0；127；68；68
30  GPRINT  0；0
40  GPRINT  66；46；22；119；22；49；66
45  K=POINT  51：PRINT  K
50  END
```

通过程序运行，显示出第51列光点的十进制编码是68。

例（7） 字符黑白交递显示的程序

```
10  CLEAR：WAIT  0
20  A$=INKEY$
30  IF  A$=""  THEN  20
40  CLS：WAIT  10
50  FOR  I=8  TO  17
60  CURSOR  I
70  PRINT  A$
80  NEXT  I
90  WAIT  0
100  FOR  I=38  TO  116
```

```
110 GCURSOR  I
120 X=&7F-POINT  I
130 GPRINT  X
140 NEXT  I
150 GOTO  10
```

这个程序启动运行后，显示屏上将出现一条黑色显示带，并且程序处于读键状态。我们只要任意按键盘上的某字符键，显示屏上就交递显示该字符一次。再按另一字符，又交递显示另一字符一次。

我们可通过此例，综合体会一下 POINT、CURSOR、INKEY$、GCURSOR 等的功能和用法。

第五节　逻辑运算

逻辑运算或逻辑代数运算，又叫布尔运算，其基本概念是由布尔（Boole）提出来的。它的实质就是用符号来表达思维和语言的逻辑性。随着生产与自动控制技术特别是计算机技术的发展，逻辑运算得到日益广泛的应用。

逻辑运算与算术运算有其相似之处，也有完全不同之处。算术运算的结果可以是各种不同的数，而逻辑运算的值，只能是“1”与“0”。

在我们日常生活中，有很多现象（或事件）是可以用某种“逻辑值”来表示的。例如电灯的开、关，我们可以将这两种状态分别用“1”和“0”来描述，把电路接通（开灯）叫做“1”状态，那么“0”状态就是关灯状态（电路断开）。这样我们就从具体的事件中抽象出两个元素“0”和“1”来。对于这样两个数“0”和“1”，在逻辑代数中称为逻辑值或者叫逻辑变量的值，也就是说逻辑变量的值只能是“0”和“1”。

在计算机运算中，都是用二进制的数来进行的，而二进制的数字只有“0”和“1”，所以在计算机上进行逻辑运算是很方便的。

逻辑运算的基本运算有三个：逻辑“与”运算、逻辑“或”运算、逻辑“非”运算。其逻辑运算符分别为AND、OR、NOT。

下面我们对PC—1500机的逻辑运算作一简单介绍。

一、逻辑“与”运算。简称“与”运算。

“与”运算是对参加运算的两个逻辑变量进行乘法运算，所以也叫逻辑乘。因为任何逻辑变量的值只能是“0”和“1”，那么我们直接用0和1来表示。“与”运算定义如下：

```
1  AND  1 = 1          0  AND  1 = 0
1  AND  0 = 0          0  AND  0 = 0
```

或者直接写成：

```
1 * 1 = 1              0 * 1 = 0
1 * 0 = 0              0 * 0 = 0
```

注意，上式中的“0”与“1”并不是一个我们常用的十进制的数，而应理解为代表某种“事件”的成立与否的标志，“0”代表不成立，“1”代表成立。例如我们有如图4—7

所示的一条串联电路，其上有两个开关，只有两个开关都合上时电灯才会亮；只要有一个开关断开，电灯就熄灭。这就是逻辑“与”的概念。

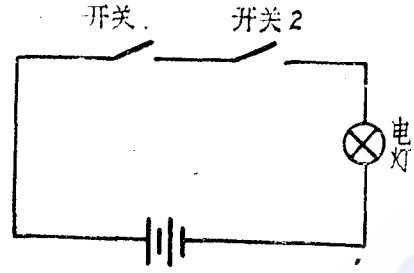


图4—7

对于上面的运算，在机器上是用二进制来进行的，而PC—1500的逻辑运算，我们是用十进制的数进行的，如43和5的逻辑“与”运算又是怎样的呢？在计算机上，十进制的逻辑运算是先将数值化成二进制的数，然后对二进制的数的各个位进行逻辑运算，然后将运算的结果再还原成十进制的数。

例如，十进制的数逻辑运算式：

43 AND 5

先化成二进制：

43十——→101011二

5十——→000101二

再根据“与”定义进行各个位的运算

```
      1 0 1 0 1 1二
AND   0 0 0 1 0 1二  化为十进制
      0 0 0 0 0 1二  ——→1十
```

所以：43 AND 5 = 1

又例如：

18 AND 24

```
18十 ——→  1 0 0 1 0二
AND
24十 ——→  1 1 0 0 0二
          1 0 0 0 0二——→16十
```

所以：

18 AND 24 = 16

二、逻辑“或”运算。简称“或”运算。

逻辑“或”运算是对参加运算的逻辑变量进行加法运算，所以又叫逻辑加运算。“或”运算定义如下：

1 OR 1 = 1　　0 OR 1 = 1

1 OR 0 = 1　　0 OR 0 = 0

或者直接写成：

1 + 1 = 1　　0 + 1 = 1

1 + 0 = 1　　0 + 0 = 0

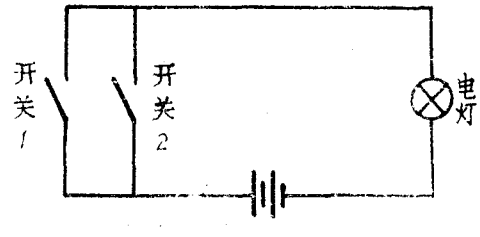


图4—8

可以看出，只有两个逻辑数均为0时，“或”运算的值才为0。只要一个不为0，则运算值就为1。又如我们有图4—8所示的一条电路，并联有两个开关，只要有一个合上电灯就会亮，而要电灯

熄灭，必须两个开关都断开。这就是逻辑“或”的概念。

PC—1500十进制的逻辑“或”运算也是先将十进制的数化为二进制，然后进行逻辑运算，将算得的结果再还原成十进制输出。

例如：　26　OR　42

```
      26十——→0 1 1 0 1 0二
OR    42十——→1 0 1 0 1 0二
──────────────────────
          1 1 1 0 1 0二 ——→58十
```

所以　26　OR　42＝58

对于十六进制数的逻辑运算，也类似于十进制的数的运算，如：

10　OR　&F

```
       10十——→1 0 1 0二
OR     F十六 —→1 1 1 1二
──────────────────
          1 1 1 1二————→16十
```

所以　10　OR　&F＝16

二、逻辑“非”运算。简称“非”运算。

逻辑“非”的概念与我们普通代数是的概念是完全不同的。所谓“非”的含义是“否定”、“不是”的意思。逻辑“非”的定义符是NOT，定义为：

NOT　0 = 1

NOT　1 = 0

注意，这里的0和1不是十进制的数。只是表示“肯定”、“否定”或“是”、“非”而已。实际上，“非”的含义就是求二进制数的反码。“0”的反码是“1”，“1”的反码是“0”。

在PC—1500计算机上，“非”运算是对16位二进制的数的各个位进行反码运算，然后将其值再化成为十进制的数来输出。

下面我们列出十进制的数与二进制的数的对应关系。因为PC—1500机是用16位二进制来进行逻辑运算的，所以二进制数列出16位。

十进制数	16位二进制数
32767	0111111111111111
32766	0111111111111110
⋮	⋮
3	0000000000000011
2	0000000000000010
1	0000000000000001
0	0000000000000000
−1	1111111111111111
−2	1111111111111110
−3	1111111111111101
⋮	⋮

−32767	1000000000000001
−32768	1000000000000000

对十进制数进行“非”运算，就是对其相应的二进制数求反码，例如对2求反码：

因为　　2+　　0000000000000010二

NOT　2+　求反　1111111111111101二

所以　NOT　2 = − 3

可以看出：　2 +NOT　2 =1111111111111111二

16位全为1

这种二进制相加后每一位都为1的关系，叫做补数关系。16位都是1的二进制数，从上面所列对应关系可以知道是十进制的－1。也就是说，十进制数与其逻辑非相加其值是－1。

即：　X+NOT　X= − 1

从而可以导出：　NOT　X= − (X + 1)　这就是求十进制数逻辑非的的一般公式，可以直接运用。如求：NOT　32

NOT　32= − (32 + 1) = −33

PC—1500计算机规定，参加逻辑运算的数或变量，其（十进制）数的取值范围是：

32767～−32768

若超过此范围，将出现　ERROR　19。

习　题　六

一、下面程序有何错误，请改正。

（1）
```
10  A$ =SHARP
20  PRINT  A$
30  END
```

（2）
```
10  READ  A$, B, C$, D$
20  PRINT  C$, A$
30  DATA  ABC, 13.4, 8.5, DEF
40  END
```

（3）
```
10  FOR  I= 1  TO  3
20  READ  A$, B$, C
30  PRINT  A$
40  NEXT  I
50  DATA  "LI", "A", "35", "CHANG", "B", "50"
60  DATA  "WANG", "C", "10"
70  END
```

二、写出下列字符串的比较结果，是为1，否为0。

（1）"WAN">="MEN"

（2）"WHAT"<"WHERE"

(3) ""BALL">"1235"

(4) "BAS">="12%"

(5) "ITS">"ITS"

(6) "CAT"<"DOG"

(7) "X(2)"<"X2"

三、写出下面程序的运行结果。(不上机，直接写出)

```
10 DIM A$(0)*35, B$(0)*22, C$(0)*50
20 A$(0)="THERE ARE TWO BIRDS IN THE TREE。"
30 B$(0)="THIS IS THREE DUCKS ON"
40 C$(0)=MID$(A$(0), 1, 10)+RIGHT$(B$(0), 9)
   +MID$(A$(0)+23, 5)+"STREAM"
50 LPRINT C$(0)
60 END
```

四、有下面这些英文单词，编一程序，将它们按字典中的顺序排列出来：

satellite Saturday pole TV rocket music ISLET house Georgia

五、编一程序使得显示的字符的间隔加大一倍，即隔一个正常字符位显示一个。显示什么字符读者可自行选择。