BASIC 语言
计算机程序设计导引

（下册）

Richard Forsyth著

卢泽愚译

中国科学院自然资源综合考察委员会
系统分析训练班

一九八〇年八月

目　录

第六章　字符行

字符行是一序列字符。至今我们只用到字符常数，即正常包括在引号(″)之内的字符序列。因此，我们已经能写这样的语句

```
400  PRINT "THE  ANSWER  IS" ; A
```

但是还不能以任何别的方式处理字符行。无论如何，非数值信息的处理是一重要的计算领域，Basic 对它提供了充分的方便。

在我们这里，一个字符是计算机能够识别的128种符号之一。它们包括字母A到Z，数字 0 到 9，及各种标点符号。Basic中可用的字符表在附录A中给出。注意每个字符有从0—127的一个数字编码。

注意，在DEC10以外的机器上Basic的字符行处理可能与这里讲的显着不同。除了文件处理（看第七章）以外，文字处理在Basic的不同文本中是最不相同的部分。这是因为原来Dartmouth的Basic既无字符行也无文件处理。它们几乎都是不同情况下作为补救而后移植的。Basic最近代的文本都有实现本章所述运算的手段，但可能用别的方式。

6.1　字符(行)变量和表达式

字符(行)变量可用来保存字符常数，这种变量的名字与数值变量相同但附加一美元号，如A$，A5$，Q2$等。如下语句

```
10  LET  B$ = "BASIC  AS  SHE  IS  SPOKE"
```

表示字符变量B$的当前值置为字符表达式 BASIC AS SHE IS SPOKE，此时是包括空格在内的21个字符的常数。也容许一维的字符数组，它是字符行的向量。于是

```
100  DIM  Q$(60)
```

建立了一个从Q$(0)—Q$(60)有61个元素的字符向量，同时

```
200  LET  A7$ = Q$(7)
```

是把Q$的元素 7 赋给字符变量A7$。在DEC10的Basic中不允许二维的字符数组（矩阵），而且字符数组不能出现在MAT语句中。

显然，在用字符行时乘法、除法等运算是没意义的；但字符行可以用加法联接。于是，当B$中包含 "STR" 时，语句 LET A$ = B$ + "ING" 就使 "STRING" 置入A$。

字符行可以用IF语句比较。比较取决于它们在ASCII（美国标准信息交换代码）字符编码中的排列次序（参看附录A）。下面是一些例子：

```
100  IF  A$ = "YES"  THEN  50
200  IF  B$ < "A"  THEN  650
202  IF  B$ > "Z"  THEN  650
```

后两个语句（200行和202行）事实上可以检查B$中第一个字符是不是字母，如果

不是就转到650行。在字符行比较时，尾部空格可以省去，因此“NO” = “NO ”；但前导空格不能省，故“ NO” < “NO”，对于全是字母组成的字符行来说：“<”意指“在字母表中早于”；“>”意指“在字母表中后于”。对包含其它字符的字符行而言，字符行比较的实际结果必须知道ASCII的编码才能确定。

在DATA语句或相应的INPUT语句中，以字母开头又不包括逗号，空格或标记号的字符常数不一定必须在引号内。其它情况它们必须包含在引号内。（所有字符行均可在引号内）。因此

500 DATA “DATE OF BIRTH”, “2ND”, OCTOBER, “1948”

包含了4个字符行：其中三个必须在引号内，因为第一个包括空格，另外两个以数字开头。注意：“1948”不能用在算术表达式中，虽然它可以被印出或做字符行处理，例如关系“1948” < “1950”在IF语句中应当是正确的。

字符行的长度没有固定上限，虽然超过132个字符的字符行实际上存在一些问题。

6.2 CHANGE（信息更换）语句

在字符行中存取单个字符的最简单且往往是最方便的办法是用CHANGE语句。CHANGE A$ TO A是将A$的字符转换成一系列ASCII的代码数，并把它们置于向量A中。CHANGE A TO A$做相反的工作。CHANGE语句的两种形式都用向量的元素0保存字符行的字符个数。下面例子是利用CHANGE语句去计算并印出一输入字符行中每个字母的频数。（非字母的符号已计数但不输出）。

```
LETTER  14:20  26—APP—76
 1  REM  CHANGE语句的例子
 10  DIM  V(64), C(127), X(1)
 100  PRINT “请给出一字符行”
 110  INPUT  S$
 120  IF  LEN(S$)<65  THEN  150
 130  PRINT “太长!”
 140  GOTO  100
 150  CHANGE  S$  TO  V
 160  MAT  C=ZER  '向量C置0
 170  FOR  I=1  TO  V(0)
 180  J=V(I)          'ASCII代码将是脚标
 190  C(J)=C(J)+1          '字符计数加1
 200  NEXT  I
 250  PRINT
 260  PRINT  “代码”, “字母”, “频数”
 275  LET  X(0)=1          '一个元素的数组
 300  FOR  I=64  TO  90          '只对字母
 310  IF  C(I)<1  THEN  350          '若无则跳过
```

```
320  X(1)=I
330  CHANGE  X  TO  S$              '单个字母
340  PRINT  I, S$, C(I)
350  NEXT  I
999  END
READY
RUN
LETTER  14:22  26—APR—76
请给出一字符行
? "TO  BE, OR  NOT  TO  BE? THAT  IS  THE  QUESTION."
    代码        字母        频数
     65          A           1
     66          B           2
     69          E           4
     72          H           2
     73          I           2
     78          N           2
     79          O           5
     81          Q           1
     82          R           1
     83          S           2
     84          T           7
     85          U           1
TIME:    0.4SECS.
```

所用的方法是在向量C的相应位置上记上每字符代码的出现数。因此，在170—200行的循环执行以后，C(65)中含有代码为65（字母A）的字符出现个数；C(66)等于B的个数，等等。

从110行可以看到，INPUT语句可以从用户那里像得到数一样得到字符行。同样用READ和DATA语句也能得到字符行以及数。此例在120行用到LEN（长度）函数，它给出字符行的字符个数。150行用的CHANGE语句使S$中字符的ASCII数值编码置于向量V，而字符数在V(0)。还在330行用到CHANGE语句，将向量X的第1元素的单个字符代码转换成字符行（只一个字母）。在160行还用了一种MAT（矩阵）语句给数组C充0（看附录E）。

从这个程序应该清楚，字母A到Z在ASCII编码中相邻地表成整数65到90，数字0到9表为48到57。虽然道理上讲可用从0到127共128个字符，但对Basic程序员来说大体上只有32（空格）到95（反箭头）的字符是有用的。

6.3 字符行函数

在Baaic中用户不能自己定义以字符行为自变量或结果的函数，但已提供了许多有用的予定字符行函数。它们最有助于处理子行：亦即把一字符行分成部分，在另一字符行中找出一字符行，等等。

上节例中已用到LEN（长度）函数，它给出字符行的字符个数。一些别的字符行函数下面讲述。（附录E中有完整的字符行函数表）。

CHR$(N)　给出其ASCII编码数为N的字符。例如CHR$(90)="Z"，CHR(64)="@"。N应为14到127的数值，若非整数则被截断。

INSTR(A$, B$)　在A$内找出B$，如找到，则返回的数值是在A$中B$开始的字符位置；如找不到，则返回的值为0。例如：INSTR("YESTERDAY", "YES")=1，INSTR("YESTERDAY","DAY")=7，而INSTR("YESTERDAY", "TODAY")=0。

还可以用一任选的（即可有可无的）数值自变量来指定找寻的开始位置：例如INSTR(8, "CATS EAT MEAT","AT")=12，但是INSTR("CATS EAT MEAT", "AT")=2。如有这个自变量应放在最前面。

LEFT$(A$, N)　给出A$中前N个字符的子字符行。例如，LEFT$("YE-STERDAY", 3)="YES"。

MID$(A$, I, N)　给出A$中从I位开始含有N个字符的子字符行。如果略去最后一个自变量，则给出从I位开始的全部。例如：MID$("TOMORROW",4,2)="OR"，MID$("TOMORROW", 3)="MORROW"。

RIGHT$(A$, N)　给出字符行中最后N个字符。例如RIGHT$("STRING", 4)="RING"。

SPACE$(N)　给出N个空格的字符行。如SPACE$(3)="　　"。

虽然Basic主要是入门性的语言，但其字符行处理功能是非常强的。用Basic能够进行某些相当复杂的非数值计算，使它成为不只是数学而有广泛用途的程序设计语言。

6.4 程序例子

为了介绍字符行处理，下面例子包括字符行和数两者。程序NUMBER.XXX把输入的数转换成表示该数的文字。例如8505变成EIGHT THOUSAND FIVE HUNDR-ED AND FIVE。

此程序用两个子程序写成：一个从1000行开始，把三位数分离成百位、十位和个位的三个数字，其做法是用10反复去除并保存余数；另一个从2000行开始，把三位数转换成文字。分成两个子程序并非因为在程序中不同地方要用到它们，而是这种做法容易把问题分解成好处理的部分。

数名文字存放在101—105行（直到一百）和110行（大于一百）的DATA语句中。总共只需要30个字。它们在程序开始（200—240行）被读到字符行向量A$和B$中。

转换子程序（2000—2222行）用到从1000行开始的例程由已知数N所得到三个分离数字。百位数字在D(3)，十位在D(2)，个位在D(1)中。再用它们作为A$(　)中适

当数名文字的地址，然后与X$中的当前内容连接而汇集成输出的字符行。让我们假定输入数N=125，于是D(3)=1，D(2)=2，D(3)=5，下面是此例程（它是整个程序的“关键”）所经历过程的简略踪迹：

行	动作	X$的内容
2000	进入，跳到2005	?
2005	清X$	（0字符行）
2010	因N非0跳到2020	同上
2020	调用在1000行的子程序而得到百位、十位和个位的数字	同上
2030	D(3)=1故到2035	同上
2035	D(3)=1，故A$(D(3))=A$(1)=“ONE”加上“ONE”+“HUNDRED”到X$	ONE OEN HUNDRED
2040	检查是否需要“AND”：需要	ONE HUNDRED AND
2060	D(2)=2，故跳到2080	同上
2080	加上A$(18+D(2))=A$(20)=“TWENTY”到X$	ONE HUNDRED AND TWENTY
2085	D(1)=5，故到2090	同上
2090	A$(5)=“FIVE”，加它到X$	ONE HUNDRED AND TWENTY FIVE
2095	出口	

需要了解此例程而从刚才讲的它如何工作的追踪还不清楚的读者，建议用一些别的三位数往下试以解释别的通路。

由部分解（能处理0—999的例程）到更一般的解（能处理0—999 999 999 999的程序，虽然只有8位有效数字）是一种捷径，因为这个范围内的数可以加进文字如

'0—999' BILLION（十亿）
'0—999' MILLION（百万）
'0—999' THOUSAND（千）
(AND) '0—999'

其中十亿是1E9。所以如果我们会转换0—999，就容易4次应用此例程去处理12位的数。（还有些别的问题，包括需要AND时的判别，但这是解法的精髓。）

```
NUMBER.XXX  14:23  28—APR—76
 1  REM  把数转换成文字的程序
 2  REM  可接收直到12位的数
 3  REM  精度直到8位有效数字
10  DIM  A$(27), B$(4), S(4)
100  REM  数据文字
```

```
101 DATA ONE, TWO, THREE, FOUR, FIVE
102 DATA SIX, SEVEN, EIGHT, NINE, TEN
103 DATA ELEVEN, TWELVE, THIRTEEN, FOURTEEN, FIFTEEN
104 DATA SIXTEEN, SEVENTEEN, EIGHTEEN, NINETEEN, TWENTY
105 DATA THIRTY, FORTY, FIFTY, SIXTY, SEVENTY, EIGHTY, NIN
    -ETY
110 DATA THOUSAND, MILLION, BILLION
150
200 REM 取数据
210 FOR I=1 TO 27
220 READ A$(I)
225 NEXT I
230 LET B$(1)=" "
240 READ B$(2), B$(3), B$(4)
245
250 REM 转换
260 PRINT "请给出一个数"
270 INPUT N
280 IF N<0 THEN 9999          '负数停机
290 LET Q=N-INT(N)            '小数部分送到Q
295 IF N>=1 THEN 315
300 PRINT "ZERO"              '0是特殊情况
310 GOTO 480
315 LET N=INT(N)              '截断N (即取整)
320 LET Q1=N-100              '是否大于100
325 FOR K=1 TO 4
330 REM S( ) 取得十亿, 百万, 千, 个
335 X9=N/1000-INT (N/1000)
340 S(K)=INT (X9*1000+.5)
350 N=INT (N/1000)
360 NEXT K
370 FOR K=4 TO 1 BY-1
380 N=S(K)                    '三位数字
390 IF N<1 THEN 470           '略去0
400 GOSUB 2000                '数转成字符
410 IF K>1 THEN 450
420 IF N>99 THEN 450
430 IF Q1<1 THEN 450
```

```
435  REM  检查需要“AND”
440  PRINT “AND”
450  PRINT  X$; “  ”;   '印字
460  PRINT  B$(K)          'B$(1)是空白
470  NEXT  K
472
475  REM  如有的话，印小数部分
480  IF  Q<=0  THEN  500
490  PRINT “AND  A  BIT”
500  GOTO  250          '下次输入
505
1000  REM  分解3位数成三个数字的例程
1001  REM  输入是N，输出是D(1-3)
1002  REM  用变量X和D9，J做计数
1005  IF  N<0  THEN  1066
1006  IF  N>999  THEN  1066          '超界
1010  LET  X=N                  '作N的副本
1015  FOR  J=1  TO  3
1020  D9=X/10-INT(X/10)
1025  D(J)=INT (D9*10+0.5)
1030  X=INT (X/10)
1035  NEXT  J
1050  REM  输出在D (1—3)
1060  RETURN          'N没有改变
1066  PRINT  N; “超出0—999的界限”
1077  STOP
1100
2000  REM  数字转成文字的例程：
2001  REM  N是输入；X$是输出
2002  REM  数名文字在A$(1—27) 中
2005  LET  X$=“  ”          '清X$
2010  IF  N<>0  THEN  2020
2012  X$=“NOUGHT”          '印0
2015  RETURN
2020  GOSUB  1000          '分N成数字
2030  IF  D(3)<1  THEN  2050          '无百位
2035  X$=A$(D(3)) + “HUNDRED”
2040  GOSUB  2200                  '检查AND
```

```
2050  IF  D(2)<1  THEN  2085          '无十位
2060  IF  D(2)>1  THEN  2080
2066  REM  这里10—19之间
2070  X$=X$+A$(D(2)*10+D(1))
2075  RETURN
2080  X$=X$+A$(18+D(2))+"  "
2085  IF  D(1)<1  THEN  2075
2090  X$=X$+A$(D(1))
2095  RETURN
2200  REM  子子程序
2202  IF  D(1)+D(2)<1  THEN  2222
2220  X$=X$+"AND"
2222  RETURN
9999  END
READY
RUN
NUMBER.XXX  14:28  28—APR—76
请给出一个数? 1976
ONE THOUSAND NINE HUNDRED AND SEVENTY SIX
请给出一个数? 777.7
SEVEN HUNDRED AND SEVENTY SEVEN AND A BIT
请给出一个? 1234567
ONE MILLION TWO HUNDRED AN THIRTY FOUR THOUSAND
FIVE HONDRED AND SIXTY SEVEN
请给出一个数? 12000000000
TWELVE BILLION
请给出一个数? 1234567.89
ONE MILLION TWO HUNDRED AND THIRTY FOUR THOUSAND
FIVE HUNDRED AND SIXTY SEVEN AND A BIT
请给出一个数? 9876543210
NINE BILLION EIGHT HUNDRED AND SEVENTY SIX MILLION
FIVE HUNDRED AND FORTY THREE THOUSAND TWO HUNDR
-ED AND FIFTY
请给出一个数? 16094
SIXTEEN THOUSAND AND NINETY FOUR
请给出一个数? 13
THIRTEEN
请给出一个数? -1
```

TIME: 0.56 SECS

READY

从2000行开始的子程序广泛利用联结（用＋号）以建立X$中的短语。

注意超过8位的数字精度不保，但保证最多8个有效位。

6.5练习

（1）写出一个对字符行向量进行冒泡分类（按字母次序）的程序。

（2）写出一个程序，它产生Basic中所用的字符编码（从32—95）及相应字符的表格。输出应在一页按两列印表（不像附录A有4列）。

（3）修改6,2节的LETTER程序：从DATA语句中读若干字符行并对它们全体集中记数。本章正文（指英文原文）前15行可用做试验数据，一行算一字符行。你的修改程序应对全体15行印出字母频数。字母E和T应最普遍。最后将程序扩充到还可计算百分比。

（4）扩充NUMBTR.XXX程序，使它也能处理小数。于是77.851应该印出SEVEN-TY SEVEN POINT EIGHT FIVE ONE以代替SEVENTY SEVEN AND A BIT。这是不太难的，因为小数部分每次只分一位。

（5）写出一个Basic程序：它接受包括多个单词（用一个空格隔开）的字符行，而产生的输出是计算此字符行的单词个数以及由每单词起始字母组成的字符行。已知“B-ASIC AS SHE IS SPOKE”应印出5和BASIC。

*（6）假定向量A$（1：N）和B$（1：M）两者包含有按字母次序的单词。写出一个合并两者而成单一的按字母顺序的向量C$。

（7）写一程序从DATA语句读一段正文，并计算字母对的频数。把所有非字母的字符都当成等值的@。在维数为（26，26）的矩阵C中累计频数。因此C（1，1）是字母对AA的频数，C（26，5）是ZE的频数，C（0，2）是@B的频数（@表示所有非字母字符）等等。只需印出非0的元素，亦即实际出现的字母对。

（8）写出一子程序，用CHANGE语句去颠倒字符行的顺序。已知输入“BASIC”，应输出“CISAB”。

*（9）写出一个不用CHANGE语句的颠倒字符行子程序，再写出一个程序应用它以及上题中用CHANGE的变型，多次比较两种方法的方便性。

（10）写出一个给出Basic语句的程序。它接收从键盘来的Basic关键字（如LET，PR-INT等），要末印出这类语句的标准形式和语句例子；要末印出相应信息：KEY-WORD NOT RECOGNIZED。（不认识此关键字）。

（11）写出一个从文字表示转换成数字表示的程序，例如，ONE HUNDRED AND NINETY转变成190。这是程序NUMBER.XXX的逆转换，不是容易的。

（12）写出一个程序，读进从Ⅰ（1）到M（1000）的罗马数字，而印出它们现代的等值数。例如ⅤⅣ应印出结果104。这个程序是非常困难的！

第七章 数据文件

至今，我们在程序运行期之间永久保存数据的唯一办法是用DATA语句。但是DATA语句的信息不容易为不同程序分享，而且大量的这种数据会使程序过于庞大。

不特别属于某一程序的信息可以长期存贮在“文件”上。文件通常保存在后援存贮器中，在Basic程序中可用通道号去引用它，而通过号是用FILE或FILES（文件）语句赋给命名文件的。

在DEC10的Basic中有两类数据文件：顺序文件和随机存取文件。顺序文件可分为带行号的和不带行号的文件；随机存取文件可分成保存数值的和保存字符行的文件。通道号标识一个文件，如是顺序文件要前置数号(#),如是随机存取文件则用冒号(：)。在任一时刻可以使用9个通道（从1—9）。

顺序文件只能从起始点依连贯顺序处理，而随机存取文件被分为“纪录”，每个纪录都可单独存取。顺序文件除起始点以外，不能从任何地方开始读或写。例如，在顺序文件上没有办法读至某一特定点，再从此开始写进文件。这类处理对随机存取文件来说是可行的。

不幸，已发现的Basic变型中，文件处理是最不相同的部分。其它Basic系统的用户从本章能得到一些基本原则，但其细节还必须去留心制造厂家的手册。（文件处理通常不是无经验程序员使用的范围。）

7.1 顺序文件

7.1.1 行号文件

开始用的最容易的一类文件是有行号的顺序文件，因为它可以像Basic程序那样从终端上建立和编辑。实际上保存的Basic程序就是顺序的行号文件之一例。

下面的对话建立了一个称为DATA.BAS的行号文件（其内容包括：名字，姓氏首字及出身日期）。

```
READY
NEW DATA
READY
10  OLD, MRS. V., 1, JAN, 1900
20  SMITH, DR.J.C., 15, MAR, 1940
30  FORSYTH, MR.R.S., 2., OCT, 1948
SAV   (存贮命令)
READY
```

在一行上数据分项的规则与DATA语句的数据相同，逗号，标记号或空格用来分隔项，并且字符行只有由非字母开头或者包含逗号、空格之类的分隔符时才必须在引号之内。至少必须以一个空格（或标记号）来分开行号与第一项数据。因此，上述文件的每

一行包括三个字符行（名，称谓和月分）及两个数（年和日）。假定此文件已指定通道号1，则读此文件之一行的程序语句可以是：

```
500  READ#1,  N$, I$, D, M$, Y
```

注意：顺序文件的一行与随机存取文件的一个纪录不是一回事。与DATA语句一样，顺序文件中的所有数据构成一连续的数据流而一个接一个地被读去。每一行的数据分布是随意的（虽然没有超过142个字符的行），往往依方便而定。

7.1.2 通道与文件

文件与通道之间的对应可以用两种办法建立。FILES语句在程序运行前的编译阶段把文件赋给通道。FILES语句取一个或多个以逗号隔开的文件说明（即文件名）作为变化项，并依它们的出现次序被赋给通道号。因此最多可以放置9个文件，并且也可以用一对连接的逗号而跳过一个通道。例如

```
100  FILES  MARX.ISM, DATA.BAS,, F1.DAT
101  FILES  FIVE
```

则使文件MARX.ISM在通道1；DATA.BAS在通道2；通道3空；F1.DAT在通道4；并且FIVE.BAS在通道5。注意扩充名BAS是在没有给出时被指定的，这些文件除了指定为随机存取的以外假定是顺序的。

如果对一给定名和扩充名没有文件存在，则此名的一个新文件将在你的后援存贮器中开放。因此，Basic程序可以建立新文件以及读、修改或扩充旧文件。

FILE语句在执行时才起作用，可用来在程序运行时改变通道配置。它的变化项是前置#（或者随机存取时的：号）的通道号，后跟一逗号，再是命名此文件的字符行公式。因此文件名如果是文字的应该在引号之内。例如：

```
550  FILE#8,  "INPUT"
575  FILE#9,  "A$+".CAT"
```

将把文件INPUT.BAS赋给通道8，并且假若在执行575行时A$中含有"OUTPUT"的话，则使OUPUT.DAT赋给通道9。FILE语句的一种常用方法说明如下：

```
200  PUINT "请问什么文件"
210  INPUT  F$          '输入文件名
220  FILE  #2, F$       '给它分配通道2
225  IF  END#2  THEN  250
230  READ#2, A$, B$, N, X(N), C2$
240  IF  A$="TARGET"  THEN  500        '找到
240  GOTO  225
250  REM  TARGET未找到，程序继续。
```

如果在这段程序到达之前通道2上已有文件开放，那末在通道2上开放由F$指定的文件前它应当关闭（不致损失信息）。

7.1.3 顺序文件的读和写

READ和WRITE语句用来把信息传送到顺序文件和从文件把信息传出。语句中第一个变量是通道区分符，后面是变量表（READ语句）或者公式表（WRITE语句）。例如

```
100  READ#2, A(I), B(I), N$, P1
```

将从通道2的行号顺序文件中读到两个数、一个字符行和另一个数。

```
200  WRITE#J, I, A(I), 2*B, SQR(X/Y)
```

将在通道5（假定J=5）的文件上写上I, A(I), 2*B及X/Y的平方根值。

READ语句要求文件的每行以行号开始，读时被略去。WRITE语句以行号后跟标记号开始每一新行——第一行编号1000，其后的行号增加10。在程序中的每个WRITE语句都在文件上写一行。

INPUT和PRINT语句与READ和WRITE语句有相同的形式，但不要求行号。如果INPUT语句遇到一个行号，将被认为一个数值。INPUT和PRINT是用于非行号文件的。下面给出例子：

```
300  INPUT#1, N, A$(N)
333  PRINT#2, A$(1); I**2
```

333行的PRINT语句说明有关配置的约定：在顺序文件的WRITE和PRINT语句中可以用分号代替逗号去分隔项目以示紧密排列。（紧密输出可节省后援存贮空间）。

对所有文件，要记住数必须读进数值变量（或数组元素），而字符行必须进字符行变量。试图把数读进字符行变量其后果不可预测，而把字符行读进数值变量其后果往往是灾难性的。

7.1.4 RESTORE（复原）与SCRATCH（作废）语句

为了从头开始再读或写已经开放的文件，必须用到RESTORE（用于读）和SCRATCH（用于写）两种语句。于是

```
50  RESTORE#4, #J+2
```

将准备好从头读第4通道和第6通道（假定J=4）上的文件。同样，

```
60  SCRATCH#7, #ABS(A/2)
```

将准备好从头写上在通道7和通道9（假设A=+16或-16）上的文件，当前的文件内容如有的话被擦掉并且重置它们的开始点。（所有文件在程序开始时都处在RESTORE状态下。）

在程序中为了从读一个文件改为写它，必须利用SCRATCH语句；而从写它改为读它，必须利用RESTORE语句。试图去读你正在写的文件而不插入RESTORE语句，或者试图去写你已读的文件而无SCRATCH语句，将是一个错误。

7.1.5 文件结束

在从一顺序文件读信息时，为检查文件的结束采用IF END（如果结束）语句。例

如

```
950  IF  END#3  THEN  9999
```

如果通道3上没有数据可读时控制转移到9999行。IF END语句只在执行时才检查，这不像某些别的Basic系统到一个新的IF END语句以前一直保持着有效的检查。如果你对处于写状态的顺序文件（即已在WRITE，PRINT或SCRATCH语句作用下而没有后继RESTORE语句），应用IF END语句会产生错误。

7.1.6 输出格式

因为顺序文件可以在终端或行式打印机上列出，所以有可能把它们用作打印报告，为使他们易于阅读有一些方便措施。对顺序文件可用PRINT USNIG语句(或行号文件时用的WRITE USING语句）作较大的格式控制。形式是PRINT #N USING I, …… 或者WRITE #N UING I, ……，其中N是通道号，I是图象引用，省略点表示输出表。对顺序文件还可以用QUFTE（引号）和NOQUOTE（非引号）语句（参看附录B）以使得以数字开头或含有分隔符的字符行是在或者不在引号包围之中。QUOTE形式适用于其后读用的文件；NOQUOTE形式适用于那些予定将印成报告之类的文件。NOQUOTE语句在程序开始是不设置的。

还可以用PAGE（页式）语句（看附录D）把输出的顺序文件分成固定长度的页面。注意，顺序文件的一行不能含有多于142个字符，并且对于72个字符的行必须用MARGIN（页边）语句来指定非标准的页面宽度。

7.2 随机存取文件

7.2.1 随机存取文件的开放

在DEC10的Basic中随机存取文件被分成固定长度的记录，每个纪录可以单独存取。与每个通道相结合的是指示当前纪录的“指针”，它可以由程序检查或改变，以便得到任何特定纪录的存取。指针决定了下次将读或写的纪录。纪录可以在随机存取文件的任何位置，按任意次序随便读或写（与顺序文件的规则不同）。一个随机存取文件可以包含字符行或者数值，但不能两者并有。

随机存取文件用FILE或FILFS语句赋给通道：在数字符（#）的地方用冒号(：)，后面是带有类型符的文件名，（类型符有两种；字符行的文件为$，数值文件为%），然后若文件是字符行，则跟上每个纪录的字符数，它不超过132。例如

```
20  FILES  INPUT.DAT$64, RANDY%
25  FILE :4, "STRAND$", :8, "RAX.FIL%"。
```

将在通道1上开放一个随机存取的字符行文件，它叫做INPUT.DAT，其纪录长度为64个字符；在通道2上是称为RANDY.BAS的数值随机存取文件；在通道4是随机存取的字符行文件STRAND.BAS；在通道8上是名为RAX.FIL的数值随机存取文件。STRAND.BAS的纪录长度是34个字符（被指定的缺值）。对于数值文件，纪录长度恒为一个数，因此单个的数可以直接存取，几乎像是一个数组似的。（注意扩充名如没有，就指定为BAS）。

7.2.2 指针的控制

SET（放置）语句加上LOC（当前位置）和LOF（最后位置）两种函数，能使程序员控制文件指针以处理特定的纪录。SET语句的形式取为：SET:N, M, 其中N是通道号，M是新的指针值。两者都可以是任何算术表达式，但如需要的话要向下舍入为整数值，同时N＞9或N＜1将产生错误。例如

```
100  SET:5, J+1
```

对通道5置指针到第J+1纪录。SET语句可以取多于一对的通道号和指针位置的公式，比如

```
120  SET:5, J+2, :N, 20
```

置通道5指针到J+2，通道N指到20。（N应从1—9，所指的通道应已有随机存取文件赋给它。）

预定函数LOC以通道号为自变量并给出当前纪录的编号，亦即该通道文件的当前指针值。因此

```
200  SET:4, LOC(4)+10
```

表示通道4的指针置于比它上次值多10的地方，这可用来处理该文件上逢10的纪录。

LOF函数以通道号为自变量，并给出该通道上保存数据文件的最后纪录编号。因此，如下两个语句

```
250  IF LOC(N) >LOF(N)  THEN 500
```

与

```
275  IF END :N  THEN  500
```

有相同的效果，因为两者都是当通道N的指针在文件最后纪录之后时就跳到500行。

7.2.3 随机存取文件的读和写

RESTORE（复原）和SCRATCH（作废）语句也可用在随机存取文件上，虽然并不是非要不可的。RESTORE :N 是让通道N指向第一个纪录；SCRATCH :M 对通道M上的随机存取文件也是指向第一纪录，并且去掉了它的当前纪录。当心你实际要保留的文件决不可用SCRATCH语句。

READ和INPUT语句用于随机存取文件是等效的(与顺序文件不同)。两者是从指针指示的当前纪录开始读，并依次继续到输入表中的变量被充满为止，离开时指针指到最后一次读后的下一纪录。WRITE和PRINT语句对随机存取文件也是等效的：它们在当前纪录上开始写，并依次继续到输出表中的所有值都已传送到文件上为止，离开时指针指到写的最后一项之后。这说明依次读或写随机存取文件是容易的。事实上它比处理顺序文件要快，除非要求顺序文件的性质（如可印性，一个文件可兼有字符行和数值，易编辑等），都可采用随机存取文件。

对随机存取文件的READ, WRITE, INPUT和PRINT语句与对顺序文件有相同的格式，只是数字符（#）被代为冒号（:），如下面例子

```
808  READ :5, X, J, A(J)
```

888 PRINT #6, A$, LEFT$(Z$, 4)

注意在输入表和输出表中字符行和数不能混合，不能用格式控制符。还要注意在随机存取文件所含数据的最后纪录之后不能再读，当然在它以后还可再写，这样扩充了文件，甚至在文件最后一项之前建立没有数据的“空隙”纪录。当你读到这种空隙纪录时，数值文件的值为0，字符行文件是空白字符行。

7.3 补充

对文件不能用MAT PRINT与MAT READ语句，要在向量或矩阵与文件之间传送信息，必须单个元素地传送，通常用FOR／NEXT循环的方法。

顺序文件容易用程序以外的Basic系统处理，而随机存取文件不适用这种形式，一般只能由Basic程序处理。顺序文件还可在终端上列出。

不幸，在DEC10的Basic中还没有办法去检查从顺序文件上读的下一项是字符行还是数。因此，虽然在一个文件上字符行和数可以混合，但程序员必须知道它们的次序，或者插入记号以告诉程序下次来的数据是什么类型。（另一办法是将所有的数都按字符行存贮，在读出以后用VAL函数转换（看附录F）。

文件处理是个庞大而重要的，有时还是很复杂的课题，本章只包括主要的内容。有关文件处理语句的进一步知识在附录D中给出。

在后援存贮上有你的数据区（在“数据库”中）的最大好处是，一旦建立起来，你作任何工作时，只要想到都可以用它。但是，在你能够用 Basic 在大的文件上做复杂处理时，大概你已准备利用更加“成熟”的程序设计语言了。

7.4 程序例子

下面的程序是从通道3的行号顺序文件中一次读一个单词，并在两个随机存取文件上写出所有不同的单词及它们的频数。两个随机存取文件一个（OUT1）放字符行，一个（OUT2）放数。这里采用的“散列编码”方法。散列编码（或“分散存贮”，或“杂凑法”）是当待存的项目以随意次序来到时分配它到存贮区去的一种方法。应用进来项的某些性质决定其存放位置。在这里的情况下，项是单词，并以其首字母决定所放的地方。于是程序试图在OUT1的纪录1开始存放以A开头的单词，从13开始放B开头的单词，等等。散列函数（或称杂凑函数）由50行定义。显然可能许多单词被分配到同一位置，但当此位置（纪录）已用时，程序逐步通过顺序的纪录直到找出一个空位，新的单词就放于此。

在文件上有没有某一项的问题可以很快确定，因为给出一个单词，程序就知道从那里开始去找寻它；在此例的应用中对每个新来的单词都必须决定以前是否出现过。如果已有单词所生成的文件没有次序，那末每读进一个新词都必须找遍整个文件才能发现它至今还未遇到过，平均来说，要搜索一半的文件才能找出输入的词以前已读过。这大致是低效的。（当散列表（即文件）大半已满时，散列法也变得低效）。

下面是散列存贮法的逻辑步骤梗概。对应于此算法的行号在每步骤后适当的地方用方括号给出。