第一部分　数据输入

1. 在编译 dBASE Ⅲ 中实现数据输入和修改

编译 dBASE Ⅲ 不支持解释 dBASE Ⅲ 中 APPEND、CHANGE、EDIT 等直接对数据库记录进行输入和修改的命令。由于不同的数据库其库结构长度、宽度不同，在实现屏幕编辑时带来很多不方便。下面一段程序，可实现任意数据库记录的输入和修改。其中，A 区打开需进行输入或修改数据的主库，B 区打开的是一临时库。

该程序实现了记录的输入。只要将第四行语句改为记录指针定位语句（如 GO、FIND、LOCATE 等），即可实现特定记录的修改。

```
SELE A
USE ZK
COPY TO JK STRU EXTE
APPE BLANK
SELE B
USE JK
N1=1
M1=1
DO WHILE.NOT.EOF()
CLEA
N2=1
SELE A
@ 0,5 SAY '记录号:'+STR(RECNO(),4)
DO WHIL EN2<10
SELE B
D='ZD'-STR(N1,M1)&D=TRIM(FIELD NAME)
DD=&D
SKIP
SELE A
@ N2,5 SAY &D
@ ROW(),COL() SAY ':' GET &DD
N2=N2+1
N1=N1+1
IF N1>=10
M1=2
ENDIF
ENDDO
READ
SELE B
ENDDO
```

```
CLOS DATA
RETU
```

在 PC286 机上运行通过。

赵 屏

2. 加快数据输入的方法和技巧

一个数据库文件的结构建好以后，接着要做的工作就是输入数据，特别是在汉字信息较多的情况下，输入速度慢的问题较为突出。通过实践，加快数据库文件中的数据输入可以从以下几个方面着手：

(1)正确使用 SET CARRY ON 命令，避免相同数据的重复输入

在为数据库文件输入记录时，常常遇到相邻记录的某些数据项相同的情况，对于这些相同的数据项实际是作了重复的输入操作，对于某些数据项是汉字的就更增加了输入负担和时间，如果能正确使用 SET CARRY ON 命令，就可避免重复操作和节省输入数据的时间。

该命令是在为数据文件追加(APPEND)或插入(INSERT)一个记录时，将当前文件的前一个记录复制到当前记录中，这样当前记录与前一个记录的相同之处可不再输入，只需要修改不同之处，一般情况下系统设置该命令为 OFF 状态。

(2)利用编码代替汉字的输入方法尔后使用替换命令 REPLACE

在许多情况下，输入的数据是有规律的，而对汉字的处理要比字母及数字麻烦得多，尤其在输入汉字时更为突出。如果用数字或字母组成的代码代替汉字串的输入，不仅可以大大减少击键次数，提高输入效率，而且还可以减少出错率。所以使用 REPLACE 命令修改一批有规律的数据是很有效的，在用编码代替汉字后，库文件相应的数据项是编码，而在实际应用中我们所需要的是汉字信息，这时，可用 REPLACE 命令以汉字替换编码。

(3)使用 SET FUNCTION 命令

该命令是用来定义键盘上 F2～F10 的功能，除 F1 不能定义外，F2～F10 均可由用户用 SET FUNCTION 命令来定义。每个功能键的功能可定义为含 1－35 个字符的序列，这些字符可以是数字、汉字和各种符号，所需字符串可以预先分别定义在 9 个功能键上，在需要时按下定义后的某功能键，便可直接输入该字符串，而且，功能键所定义的内容也可根据实际需要进行更换。

孙 鹏

3. 利用词典库实现 C-dBASE 的窗口输入

对于较大的应用系统，有时有必要建立较大的词典库，对于这样的词典库采用编码输入法可实现窗口输入。比如对于籍贯项目可建立全国市县名词典库，并将其内容以拼音字头进行编码放置在对应的字段中。当输入籍贯时，只要键入市县名的拼音字头(比如北京市 BJ)程序即可在窗口中列出所有的重码市县名，编号从零到 kk(＜10)供选择。若窗口中没有所需要的内容，您可键入数字 kk＋1，程序将脱离词典库接受用户的键盘输入。

利用词典库窗口输入比使用 DOS 词组节省内存且可提高输入速度，保证数据输入的准确，参见程序。

```
use zjk
appe blank
@2,17 say "籍贯"
```

```
sele 2
use JG inde ijg
bm=‘    ’
do while .t.
@2,1 say ‘输入编码’ get;
bm pict ‘!!!!’
read
bm=trim(bm)
bm1=“‘”+bm+“’”
find &bm1
if eof()
loop
else
jlh=recn()
exit
endi
endd
yy=0
do while .t.
tt=0
do while tt<9
@tt+1,50 say str(tt+1)+JG
tt=tt+1
skip
if hzbmd<>bm
exit
endi
endd
cb=tt+2
@8,1 say “请输入编号,回车;选择” get cb rang 1,tt+2
read
if cb=tt+1
@2,17 say “籍贯” get jg
read
exit
endi
if cb<>tt+1
aa=yy*9+cb-1
go jlh
skip aa
sele 1
repl JG with jg->JG
@2,22 say JG
exit
```

```
endi
@0,50 clea
yy=yy+1
if hzbmd<>bm
go jlh
yy=0
endi
endd
```

杨根令

4. 利用“$”进行自校

利用 C-dBASE Ⅱ 编写应用程序时，输入数据是否正确的自校一般采用以下两种方法：

(1)采用 RANGE 短语自校；

(2)在循环程序 DO WHILE .T. 中用条件控制 IF 和 LOOP 语句配合进行自校。

方法 1 适合有规律的数字型数据的输入；方法 2 既适用于数字型，又适用于字符型。但在可供选择的字符型数据较多时，方法 2 组合条件复杂，程序较长。如果采用包含运算符“$”进行自校能弥补其不足之处。下面结合一个通用建库模块说明其应用方法。在建立库结构时，字段类型代码有五种可供选择，即 C、N、D、L、M。若选此范围以外的字母时，条件.NOT、FT $“CNDLM”成立，等待重新输入数据。否则，.NOT、FT $“CNDLM”不成立，即输入正确，退出循环，继续执行后面程序，从而起到了自校功能。

说明：程序中 JG.DBF 是一个结构库文件，它的四个字段就是一般数据库文件中每个字段的四个属性。

程序清单

```
* JG.PRG 建立库结构模块
SET TALK OFF
SET SAFE OFF
CLEAR
ACCE “请输入待建立结构的数据库名：” TO KM
IF .NOT. FILE(“&KM.DBF”)
   USE JGK
   COPY STRU EXTE TO &KM.JG
ENDI
@5,10 SAY “请输入数据库结构：”
@6,10 SAY “说明：1——字段名必须以字母开头”
@7,10 SAY “      2——字段类型代码：”
@8,10 SAY “C—字符  N—数字  D—日期  L—逻辑  M—备注”
USE &KM.JG
APPE BLAN
X=1
FT=“ ”
DO WHILE .T.
   XS=STR(X)
   @10,10 SAY “结构‘+XS+’：”
```

```
    @11,10 SAY "字段名:" GET FIELD-NAME
    READ
    DO WHILE .NOT. FT $ "CNDLM"
       @12,10 SAY "字段类型:" GET FT PICT "A"
       READ
    ENDDO
    FIELD TYPE=FT
     ⋮
ENDDO
CREAT & KM FROM & KM. JG
CLOSE DATA
RETU
```

数据库结构　:A:JGK.dbf
记录个数　　:　1
最后更新日期:10/17/90

字段	字段名	类型	宽度	小数
1	FIELD-NAME	Character		10
2	FIELD-TYPE	Character		1
3	FIELD-LEN	Character		3
4	FIELD-DEC	Character		3
* * 总计 * *				18

申全洲

5. dBASE Ⅲ 屏幕格式输入的设计方法

dBASE Ⅲ为用户提供了丰富的输入/输出语句，并且都有一定的格式设计能力，其中功能最强的一类输入语句是@ say…get…和 READ 语句的结合。使用它可设计出所需要的屏幕格式。但是，*对于下列问题，是否也能很好地解决呢？*

在一个数据库中有 M 个字段，是用户输入数据的存放区，用户希望在某个时候只输入其中的 X 个字段，另一时候只输入其中的 Y 个字段，要求 M－X 或 M－Y 个字段处于输入范围之外，从而提高输入速度，降低输入的冗余度，特别是减少数据串项的可能性。

如果按照传统的程序设计方法，要解决这个问题，还得为每一种可能的输入方法编制一个程序，那么对于 M 个字段就要编写 2^M 个程序，可见几乎是不可能的。

避开传统的设计方法，采用程序自动生成的原理，较理想地解决了这个问题。

本方法的主要思想是：

(1)针对含有 M 个字段的数据库，建立一个有 M 个记录的数据库，取名为 DBSTRU.DBF。该库中至少含有存放 M 个字段段名的字段 MSTR 和一个标志字段 BZ。除此之外，根据需要可增加输入提示信息字段等其它字段。

(2)增设一个数据库 DBCOMM.DBF。该库只有一个字符型的字段 SAYCOM，用于存放自动生成的@ SAY…GET…命令序列，一个记录存放一条命令。

(3)编写程序一(XGBZ.PRG)，该程序首先列出用于输入的 M 个字段的名字，并各给出一个编号，用户依编号选择本次要输入的和不输入的字段，然后程序根据用户的选择，修改数据库 DBSTRU 中的 BZ 字库。

(4)编写程序二(AUTOPROG.PRG)。该程序根据 DBSTRU 中的 BZ,自动生成@ SAY …GET 命令作为一字符串,放入 DBCOMM.DBF 中,然后用 COPY TO MENU.AUTSDF 命令生成文件 MENU.AUT。

(5)在有选择输入的用户程序段中,用下面的命令串 1:

```
DO MENU.AUT
READ
```

或命令串 2:

```
SET FORMAT TO MENU.AUT
CHANGE * 或 EDIT *
CLOSE FORMAT
```

就可以实现前面所讲的任选 X 或 Y 个字段进行输入或修改。

曹岩松

6. 综合检索、统计时汉字条件的输入

在使用 dBASE Ⅲ 编的管理信息系统中,常有综合检索和综合统计两种功能,且检索及统计条件常含有汉字。由于使用对象是非专业计算机人员且带有各地方言,这样就造成汉字输入困难,而由于工作原因,他们不可能花去大量时间学习拼音及其他汉字输入法。基于上述原因,在程序中采用窗口提示一些固定字段,而非固定汉字取若干个连续关键字作为条件。

采取上述方法,即给使用者醒目提示,又免去输入汉字这一繁琐步骤,后面附程序清单。

在程序 RY2 中不难看出,检索条件中"部门、性别、民族、学历、行政职务及专业职务"为固定汉字字段,则可执行相对应的窗口程序 RY23n(n 即检索条件选号),如 RY235(学历提示)。其中 RYK 为学历库,只要按下相应的数字,汉字会自动填到空格中,其它类似。

在 RY5 中,可输入若干个关键字,其中 117 为判断标志。

程序清单

```
.TYPE RY235.PRG
@ 11,50 SAY '请选择'
S=' '
@ 12,40 SAY '1-文盲;2-小学;3-职初;4-初中;'
@ 13,40 SAY '5-职高;6-高中;7-中技;8-中专;'
@ 14,40 SAY '9-专科;10-本科;11-双学士;12-研究生;'
@ 15,40 SAY '13-硕研;14-博研'
@ 15,56 GET S
READ
S=VAL(S)
IF S>=1 .AND. S<=14
   USE RYK1
   GO S
   STOR C_N TO XL
   USE RYK_1
   GO VAL(IB0)
   REPL DATA WITH '&XL'
   @ 7+IB,23 SAY DATA
```

```
RETU
.TYPE RY2.PRG
SET TALK OFF
CLEAR
IB=1
USE RYK_1
REPL ALL MARK WITH .F.
REPL ALL DATA WITH ' '
REPL ALL SIGN WITH ' '
GO TOP
DO WHILE .NOT. EOF()
   STORE ' ' TO IB0
SET COLO TO 0/2
@ 1,28 SAY '请输入检索条件'
@ 2,5 SAY '1.部门      2.姓名      3.性别      4.民族      5.学历'
@ 3,5 SAY '6.出生年月  7.毕业年月  8.基职工资  9.原学专业  10.从事专业'
@ 4,5 SAY '11.行政职务 12.专业职务 13.聘任时间 14.毕业学校  15.备注'
@ 5,5 SAY '0.检索'
SET COLO TO 2/0
@ 5,55 GET IB0
READ
IF IB0=' '
  RETURN
ENDIF
IF IB0='0'
  DO RY21
  RETU
ENDIF
IB0=VAL(IB0)
IF IB0>=1 .AND. IB0<=15
  IF IB0=2
    DO RY24
  RETU
  ENDIF
  IF IB0=14
    DO RY25
  RETU
  ENDIF
  GO IB0
  REPL MARK WITH .T.
  IF IB0>=6 .AND. IB0<=8
    IF MARK
      IF DATA #' '
        SKIP 10
```

```
            REPL MARK WITH .T.
          ENDIF
        ENDIF
      ENDIF
      @ 7+IB,10 SAY C_NAME
      DO RY22 WITH 7+IB,19
         clear gets
         IF TYPE='N' .OR. IB0=1 .OR. IB0=13 .OR. IB0=15
            @ 7+IB,23 GET DATA
            READ
         ELSE
            SET COLO TO 0/2
            @ 7+IB,23 SAY SPACE(20)
            @ 10,40 CLEAR
         IF IB0=3 .OR. IB0=4 .OR. IB0=5 .OR. IB0=9
            IB0=STR(IB0,1)
         ELSE
            IB0=STR(IB0,2)
         ENDIF
            DO RY23&IB0
            SET COLO TO 2/0
         ENDIF
   IF DATA='  '
      @ 6,10 SAY '数据不能为空'
      @ 7+IB,10 SAY '  '
      REPL MARK WITH .F.
      LOOP
   ENDIF
   IF TYPE # 'N'
      REPL DATA WITH '"'+TRIM(DATA)+'"'
   ENDIF
   IB=IB+1
   ELSE
     LOOP
   ENDIF
   ENDDO
   .
   .TYPE RY215.PRG
   SET TALK OFF
   CLEAR
   USE KJK4
   STORE '          'TO FB1
   @ 10,10 SAY '请输入关键字(1-10个汉字):' GET FB1
   READ
```

```
FB1=TRIM(FB1)
STORE '毕业学校关键字='+FB1 TO IB11
CLEAR
@ 20,20 SAY '正在检索请稍候!'
DO WHILE .NOT. EOF()
FB2=AT('&FB1',18)
REPL 117 WITH FB2
SKIP
ENDDO
COPY TO RYKC FOR 117#0
USE RYKC
IF EOF()
   @ 24,10 SAY '满足该组条件的记录不存在!'
ELSE
   CLEAR
   STORE [N] TO YS
   @ 20,20 SAY '打印吗?' GET YS
   READ
   IF YS=[Y]
      DO RY211
      RETURN
   ELSE
       DO RY212
   ENDIF
ENDIF
RETU
```

孔小芳

7. 如何加快录入速度

(1)使用快速传递命令 set carry on

在使用 append、insert 命令之前,打入 set carry on 命令,可以完成记录的快速传递,即把上一记录的内容传递到下一记录。如果下一记录中的有些字段内容和上一记录相同或相似,那么只要对下一记录的有些字段稍作修改,即可完成该记录的输入。如要建立一个通讯录的数据库,该库中的相邻记录或一组记录中的通信地址、电话等字段内容可能相同或相差无几。在这种情况下,如果每一条记录都一个字段一个字段地输入,必然会降低输入速度。如果用 set carry on 命令实现记录的传递,那么只要把传递下来的有关字段稍作修改,就完成了新记录的输入,这样可大大提高记录的输入速度。

(2)灵活运用 Replace 命令

要建立的数据库中有些记录的某个字段内容可能是完全相同的。在输入这些记录的该字段内容时,可先输入一个简单的字母或变量,当所有的记录输入完后,再用真正要输入的内容去替换该字段,以实现记录的快速输入。以人事档案库为例,库中可能有若干个记录的职称字段的内容为"工程师"。在输入过程中,遇到职称字段为工程师时可先输入字母 G,显然这比输

入工程师三个汉字要省时的多。当所有的记录输入完后，使用 Replace 命令进行字段内容的替换，即用“工程师”去替换字母 G。即 Replace all 职称 with“工程师”For 职称二“G”。

宋建恒

8. 数据录入正确性的程序检测

在 dBASE Ⅲ 中已有录入数据有效性的检验功能，但对录入数据的正确性缺乏检测。如果用人工进行查错的话，无疑既费时间又费人力，而且心中始终没有把握，难以确保录入数据的正确率。可以用程序来实现自动查错。

针对这样的数据库结构（库结构附于后），根据数据项的实际意义，我们知道有这样一条规则，即小计的值就是各数据项值的和，在实际输入时，既要输入各数据项的值，又要输入小计的值。有了这条规则，我们就可以据此编制录入数据性检测的程序了（附程序于后）。

下面分析一下这条规则为什么能够检查录入数据的正确与否：

（1）如果各数据项之和不等于小计的值，显然这条记录肯定录入错了；

（2）如果各数据项之和等于小计的值，在这种情况下，仍有两种出错可能：一是各数据项内容录入时顺序颠倒；二是数据项内容和小计值都错了，但巧合相等。实际上，数据录入错的主要来源在于手写体原始数据没印刷体好认，录入时敲错数字键、漏掉或多敲了数据。录入员的标准是万分之五的差错，第一种错一般不会发生，且发生后不影响大的统计（因小计依然正确），仅对各数据项的统计略有出入；第二种纯粹巧合，发生概率太小，不足影响统计分析结果。至此，可以认为在各数据项之和与小计值相等时，即表明录入数据正确。

程序具有的功能与作用：

- 检测出数据录入错，确保录入数据正确；
- 检测出数据登帐时是否抄错；
- 省却人工检错的麻烦，节省人力和时间，提高工作效率；
- 显示出错记录号，以供修改用。

象所附库结构这样庞大的数据库，含 1251 个记录，运行该程序查错，仅用了 15 分钟即查错完毕。

在运行检测程序时，请记录下显示出来的出错记录号，或者用 Shift－prtsc 硬拷贝下来。接下来，打开库文件，按出错记录号，用 EDIT 命令进行修改。

用此程序检测，可确保库文件录入数据错误率降至最低限度，而且我们才有把握确认录入的原始数据正确可靠。对于有类似库结构的，根据那条规则，只要在程序中修改一下数据项名，就可拥有一个查错程序了，从而能够确保录入数据的正确性、可靠性。

数据库结构　　数据库：C：dzdata.dbf

数据库中的数据记录个数：1251

数据库的最后更新日期　：04/18/91

字段	字段名	类型	宽度	小数
1	省份	字符型	6	
2	单位	字符型	40	
3	购买日期	日期型	8	
4	单号	字符型	5	
5	SG－2000	数字型	10	3
6	SG－2504B	数字型	10	3
7	SG－250b	数字型	10	3

```
  8    SG－2507A      数字型      10     3
  9    SG－2507B      数字型      10     3
 10    SG－3201       数字型      10     3
 11    SG－3202       数字型      10     3
 12    SG－3203       数字型      10     3
 13    SG－3204       数字型      10     3
 14    SG－3205       数字型      10     3
 15    SG－3208       数字型      10     3
 16    SG－3201Q      数字型      10     3
按任一键继续……
 17    SG－3202Q      数字型      10     3
 18    SG－3205Q      数字型      10     3
 19    SG－3211S      数字型      10     3
 20    SG－6810       数字型      10     3
 21    SG－6810       数字型      10     3
 22    SG－6810       数字型      10     3
 23    SG－6810       数字型      10     3
 24    SG－6810       数字型      10     3
 25    SG－6810       数字型      10     3
 26    SG－6810       数字型      10     3
 27    小计           数字型      10     3
* *总计* *                        200
USE dzdata
  Clear
  set talk off
  * * * * * * * * * * * * * * * * * * * * * * *
  *   数据录入的正确性检测程序(Ipicture.prg)   *
  * * * * * * * * * * * * * * * * * * * * * * *
  @ 1,1 say "数据录入错的记录号:"
  store 0 to j
  do while .not. eof()
     store 0 to p
     p=sg_2009+sg_2504b+sg_2506+sg_2507a+sg_2507b
     p=p+sg_3201+sg_3202+sg_3203+sg_3204+sg_3205
     p=p+sg_3208+sg_3201q+sg_3202q+sg_3205q
     p=p+sg_3211s+sg_3507a+sg_3507b+sg_4001
     p=p+sg_4002+sg_4401+sg_4402+sg_6810
     e=str(p,10,3)
     w=str(小计,10,3)
     if e<>W
       @ row(),col()+j say recno()
       j=j+1
       if j>=5
         @ row()+1,1 say "  "
         j=0
       endif
     endif
     skip
```

```
enddo
use
retu
```

陆亚飞

9. 代码录入正确性的判断程序

在用 dBASE Ⅲ 进行管理软件编程时，经常碰到的问题是大量的代码输入。而代码本身却是由一些枯燥无味的数字(或字母)组成，极难保证数据输入的一次性正确。一般的做法不外乎用菜单提示方式由用户选择，或用代码直接输入。不论那一种，都需要由用户看到其汉字名称后来确定正误。通常情况下，当汉字给出后，程序已转到下一个命令行处理。当发现错误时，只有等数据输入完成后再进行修改，给用户带来极大的不便。设计如下程序，可以方便地实现认可方式的程序设计。

程序 1 提供了认可方式功能。当用户输入代码后，该程序显示该代码的汉字名称，然后光标仍停留在代码输入行上，当用户输入回车键后，该代码被认可，若用户认为该代码不正确时，可重新输入代码。

程序 2 提供了一个菜单提示功能，当输入错误代码时，给出全部的代码、名称的菜单提示；当输入不完整代码时，给出该代码状态下的代码、名称的菜单提示。这样做的目的是用户不用去查找代码表。

在程序设计时，为了尽量用少的参数传递，将代码库中的代码及名称字段都分别定义为 CODE、NAME(若不这样，可再用两个参数，分别传递代码及名称字段名)。

程序 1(cinput. prg)

```
public vx1,vy1,vx2,vy2,vbegin,vend,vcode
public vcondi
set talk off
vcode1="—————————"
vcode2="—————————"
go top
vrecno=recno()
do while .t.
   @ vx1,vy1 get vcode
   read
   if vcode="    "
     if vcode<>vcode1
       vcode1=vcode
       loop
     else
        exit
     endif
   endif
   find & vcode
   if eof() .and. vbegin<=24
     go vrecno
```

```
        vcondi=".t."
        do cmenu
        if eof()
          go top
        endif
        vrecno=recno()
        loop
      endif
      if len(code)<>len(trim(code)) .and. vbegin<=24
        vcondi="code=trim(vcode)"
        if vcode<>vcode2
          vcode2=vcode
        else
          go vrecno
        endif
        do cmenu
        if .not. (&condi)
          vcode2="------"
        endif
        loop
      endif
      if vcode<>vcode1
        if vx2<=24
          @ vx2,vy2 say name
        endif
        vcode1=vcode
        loop
        endif
        exit
enddo
retu
```

程序 2(cmenu.prg)

```
public vbegin,vend,vcode,vcondi
set talk off
do while .t.
   vx=vbegin
   vy=0
   vlengh=len(code)+len(name)+6
   do while .not.eof() .and. vx<=vend .and. (&vcondi)
      @ vx,vy say code+";"+name
      vy=vy+vlengh
      if vy>=80
        vy=0
        vx=vx+1
```

```
      endif
      skip
    enddo
    do while vx<=vend
      @ vx,vy say space(vlengh)
      vy=vy+vlengh
      if vy>=90
        vy=0
        vx=vx+1
      endif
    enddo
    exit
enddo
retu
```

何昌波

10. 充分利用屏幕方便数据录入

使用数据库 dBASE Ⅲ PLUS 时，利用 APPEND 命令、BROWSE 命令都可以实现数据录入，但是这两条命令都有一定的缺点。

使用 APPEND 命令，一屏只能显示一条记录，当数据项稍多时，一屏只能显示一条记录的部分数据项，而右半边的屏幕却常常被浪费。BROWSE 命令虽然较前者进了一步，一屏可以同时显示多条记录，但是数据项稍多时，同样只能显示计录的部分数据项。如果需要输入或者修改其它数据项时，必须通过使用 CTRL-B，CTRL-Z 进行水平滚动才能实现。

如下一个数据录入程序 YAOSR. PRG(程序附后)，可以将一条记录的所有数据项都布置在同一屏幕上，使数据录入员可以一下子看到需要输入的所有提示项及已输入的所有内容，而不需要象 BROWSE 命令那样的水平滚动。

同时，该程序还可以在同一屏幕上布置多条记录(这里只布置了 5 条完整的记录)，扩大了视野，使数据录入员可以在处理当前记录的同时，能够观察和修改其它相邻记录的任一数据项，即具备全屏幕编辑的功能。

该程序在 SUN 286 及 GREAT WALL 0520CH 上运行通过。

程序中调用的示例数据库 YCH. DBF 结构如下：

```
Structure for database:C:YCH.dbf
Number of data records:      0
Date of last updtae :11/09/90
Field  Field Name  Type        Width  Dec
    1  商品名称    Character            40
    2  发货地点    Character            40
    3  数量        Numeric              7
    4  单价        Numeric     6        2
    5  金额        Numeric     9        2
** Total **                             103
***** YAOSR.PRG
```

```
SET DELI TO "[]"
SET DELI ON
SET TALK OFF
SET COLOR TO B,R
SET DEVI TO SCREEN
CLEAR
USE YCH
GO BOTT
X=1
IF .NOT.EOF()
  SKIP -3
  DO WHILE X<21 .AND. (.NOT.EOF())
     DO YAOSRZ
     SKIP
     X=X+4
     CLEAR GETS
  ENDDO
  SKIP -1
  X=X-4
ELSE
  APPE BLAN
ENDIF
DO WHILE .NOT.EOF()
   SET COLOR TO BG+,R+
   DO YAOSRZ
   READ
   SET COLOR TO B,R
   DO YAOSRZ
   CLEAR GETS
   IF READKEY()=270
     EXIT
   ELSE
     IF READKEY()=12
       LOOP
     ENDIF
   ENDIF
   IF READKEY()/2=INT(READKEY()/2)
     IF RECNO()<>1
       IF X=1
       * * * * * 向下滚动
       SKIP 4
       X=21
       DO WHILE X>5
          SKIP -1
```

```
            X=X-4
            DO YAOSRZ
            CLEAR GETS
          ENDDO
          * * * * *滚动结束
        ENDIF
        SKIP -1
        X=X-4
      ENDIF
      ELSE
      SKIP
      IF EOF()
        YN="  "
        SET COLOR TO BG+,R+
        @ 23,1 SAY "ADD NEW RECORDS?" GET YN PICT "!"
        READ
        SET COLOR TO B,R
        @ 23,1
        IF YN="Y"
          APPE BLAN
        ELSE
          SKIP -1
          X=X-4
        ENDIF
      ENDIF
      SKIP -1
      IF X=17
        * * * * *向上滚动
        SKIP -4
        X=-3
        DO WHILE X<>13
          SKIP
          X=X+4
          DO YAOSRZ
          CLEAR GETS
        ENDDO
        * * * * *滚动结束
        CLEAR GETS
      ENDIF
      SKIP
      X=X+4
  ENDIF
  ENDDO
  USE
```

```
RETURN
* * * * * YAOSRZ.PRG
@ X,1 SAY RECNO()
@ X,15 SAY "商品名称" GET 商品名称
@ X+1,15 SAY "发货地点" GET 发货地点
@ X+2,15 SAY "数量" GET 数量
@ X+2,35 SAY "单价" GET 单价
@ X+2,50 SAY "金额" GET 金额
RETURN
```

姚建祥

11. 学会运用@输入/输出格式命令

该命令格式为：@〈行，列〉〔SAY〈表达式〉〔PICTURE〈格式〉〕〕〔GET〈变量名〉〔PICTURE〈格式〉〕〔RANGE〈表达式1，表达式2〉〕〕〔CLEAR〕。本命令中，以〈格式〉理解最为重要，这可以由功能字符(有十种)和匹配符(有十一种)组成，若二者连用时，中间需用一空格隔开。在常用的报表打印时，有关人员往往要求当输出值为零时，不要打出"0.00"字样，而只输出若干个空格；非零时则打出该非零值。不了解本命令的用户编写程序时，往往加上条件判断语句，判定当输出值为零时，打出若干空格，否则打出该非零值。这样做的结果程序显得冗长，执行速度当然也就慢了。其实可用本命令，只要设置功能符为"Z"便行了(大多数C－dBASE Ⅲ说明书中将"Z"功能符描述为：Z—空字符串时显示零，而使人大惑不解，应改为"使为零时显示空字符串")。如设有某数据库中有一字段名为"销售量"，则可写出：@4,2 SAY 销售量 PICT"@Z 999.99"。其中功能符"Z"表示若销售量为零时，显示一串空格，不为零时按匹配符"999.99"格式输出该值。

张华如

12. dBASE Ⅲ数据分屏输入法浅析

对于那些栏目少的数据库来说，屏幕设计时只要采用@…SAY…语句在屏幕上指定所要输入内容的位置即可。但是，在实际工作中，有的数据库的栏目多达几十个甚至更多，对于这类大型数据库数据输入程序的屏幕设计，假如仍用@…SAY…语句，则程序设计比较麻烦，且要花费大量的精力去调整屏幕的显示位置。当前，对这类数据库的数据输入程序的屏幕设计，通常采用以下两种方法：

循环输入法

循环输入法就是采用dBASE Ⅲ的DO WHILE…ENDDO这样的循环语句及一条@…SAY…命令来解决屏幕的显示问题，以简化输入程序的设计。

分库输入法

根据数据库栏目的大小及内容的性质，可把数据库分为基本库、特殊库1、特殊库2等，并在基本库中设定调用各特殊库的标识，以便在数据输入时根据标识决定是否执行特殊库操作。分库输入法的基本原则一般以屏面为准。

但是，上述两种方法都存在着一个致命弱点：一经发现前面的数据输入有误，不能立即返回修改之。这对操作人员来说，无疑是增加了数据输入时的纠错困难，降低了建库速度。对此，可采用另一种编程方法——数据分屏输入法。其方法如下：