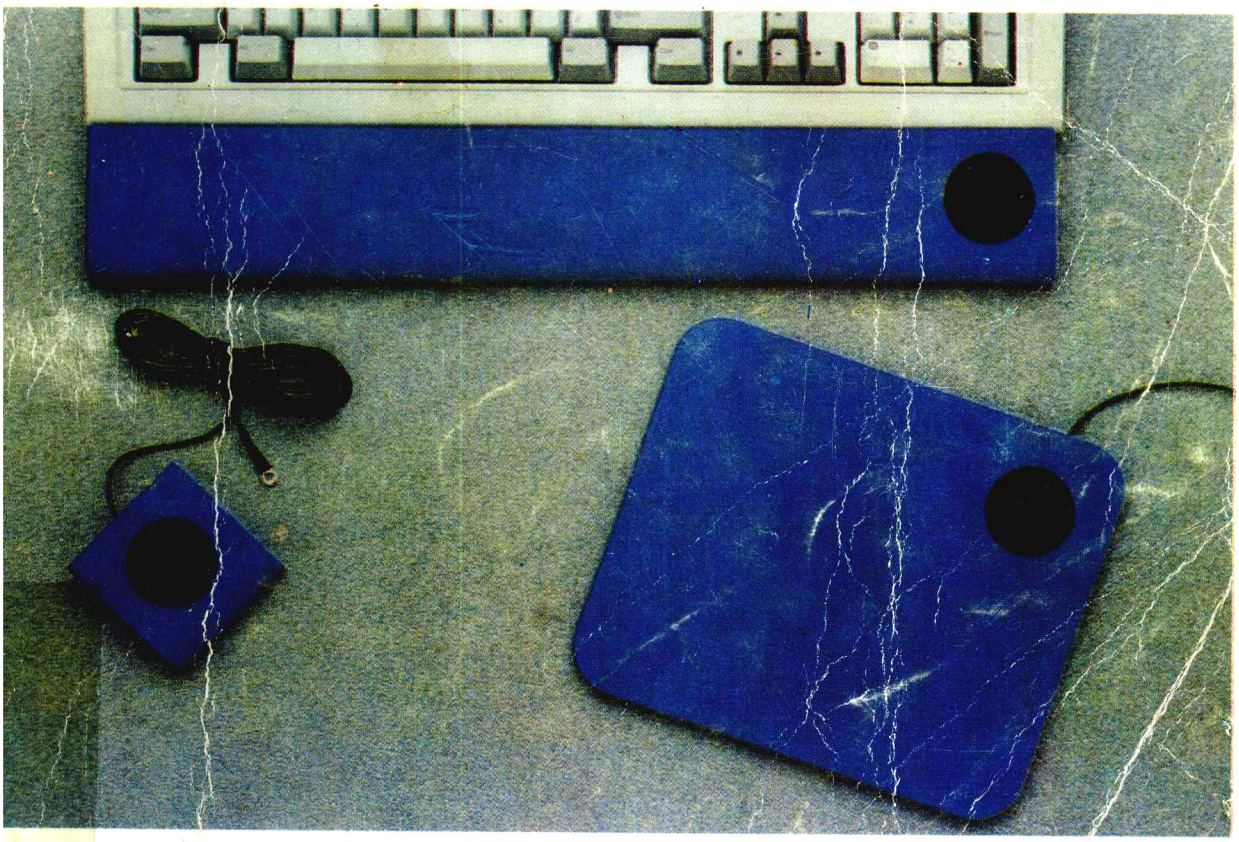
《现代电子技术》增刊

Foxbase⁺/Foxpro高级编程实用技术

俞一彪 编著



陕西电子杂志社

FoxBASE+／FoxPro
高级编程实用技术

俞一彪　编著

陕西电子杂志社

前　言

FoxBASE+是继 dBASEⅢ（dBASEⅢ+）之后由 Fox Software 公司研制的数据库管理系统软件。由于其卓越的数据处理功能和更高的运行速度，并且具备多用户管理功能而成为替代 dBASEⅢ（dBASEⅢ+）的新一代 DBMS 软件，赢得了最广泛的用户。但是，除了基本的数据处理功能之外，FoxBASE+在诸如高级菜单、输入模块及查询模块等设计方面没有提供足够的手段，更缺乏灵活方便的窗口管理、图形、图象处理、文件和应用系统加密、语声处理及定时程序控制等方面的功能。而这些对于实际应用系统开发设计来说是相当重要的，也是任何一个希望开发高质量应用系统的人员所必需具备的知识和技能。虽然在 FoxBASE+2.1 版本中提供了下拉式菜单设计命令和一个商用图形软件包 FoxGRAPH，但实践经验告诉我们，这些命令或软件包汉字操作系统并不支持，所以无法运用。而对于国内用户来说，不用汉字操作系统是不现实的。这些情况同样也发生在较 FoxBASE+后推出的 FoxPro 中。本书的写作目的就是旨在解决以上这些问题，并向广大 FoxBASE+应用系统开发人员和用户介绍其实现方法和具体的编程技术。

本书主要有两大部分组成。从第一章至第四章为第一部分。这一部分从应用系统开发设计的角度来编写，分别介绍了 FoxBASE+基本编程技术、菜单设计技术、输入模块设计和查询模块技术。这些内容中大部分是作者在具体应用系统开发过程中积累的经验总结，对各种流行菜单的编程设计、各种输入方式和查询方式及编程实现都给予了详细和系统的介绍。读者不难结合自己所从事的工作加以运用。从第五章开始为第二部分。在这一部分中，以每章独立的方式介绍在 FoxBASE+ DBMS 中增加图形、图象处理功能、语声处理和时钟控制功能、文件及系统加密功能的原理及编程实现方法。书中对这些高级功能都给出了相应的功能模块程序，读者即使不领会其实际内容也可以方便地在应用系统程序中直接调用这些模块而实现相应的功能。对于熟悉汇编语言的读者来说，可以参考这些功能模块的设计方法作进一步开发设计。

本书名为《FoxBASE+／FoxPro 高级编程实用技术》，并以 FoxBASE+ DBMS 软件为背景编写，但由于 FoxBASE+与 dBASEⅢ+的部分兼容性及与 FoxPro 的全兼容性，书中所述大部分内容也适合于 dBASEⅢ+的软件环境，同时百分之百适合于 FoxPro。

本书适合企事业单位从事计算机管理及应用系统开发设计人员学习，也适合各类学校从事 FoxBASE+、FoxPro 和 dBASEⅢ+数据库管理系统教学的老师和学生参考。作者认为，对于应用系统开发设计而言，不可能开发出真正的通用系统适合于各种不同的实际情况，唯有技术才能是通用的。

最后，作者要感谢陕西电子杂志社对本书出版的大力支持，特别是张忠智同志的辛勤工作和帮助。

＊＊读者可同杂志社联系，索取书中所有程序示例的软盘。

作者

1994 年 1 月

目　录

第一章　基本技术

本章作为全书的基础篇，介绍 FoxBASE+程序设计的基本技术。这些基本技术包括对程序控制起关键作用的函数运用以及命令语句运用技巧。在本章最后两节将介绍 FoxBASE+同其它高级语言、文字处理软件等进行数据通讯的方式及 FoxBASE+与 DOS 操作系统的软件接口。特别是 CALL 命令调用由汇编语言构成的二进制文件进行处理的方式将是本章介绍的重点，理解好这一处理方式也是掌握全书其它高级编程技术的重要基础。本章乃至全书都认为读者对 FoxBASE+的基础知识，诸如怎样建立命令文件、运行程序等有一定的了解。因此，本书将不对这些基本内容作介绍。

§1.1　函数及编程设计

同其它高级语言及 dBASEⅢ相比，FoxBASE+数据库管理系统的函数是相当丰富的，功能亦很强。其函数集中共包括七十多个函数，这些函数的功能遍及数据处理、程序控制及磁盘文件管理等各个方面，大大提高了 FoxBASE+这一 DBMS 的综合能力。在应用系统编程设计中发挥了极大的作用。根据函数的功能，可以将函数归结为字符处理函数、数值处理函数、日期处理函数、转换处理函数、库文件测试函数、环境测试函数及输入函数这七类，以下分类列出。

1. 字符处理函数

函数名	结果类型	变量	说明
AT	N	(子串,字符串)	取子串位置
LEFT	C	(字符串,长度)	取左边局部字符串
LOWER	C	(字符串)	字母大到小转换
LTRIM	C	(字符串)	去掉字符串左边空格
REPLICATE	C	(字符串,重复数)	重复字符串指定数
RIGHT	C	(字符串,长度)	取右边局部字符串
SPACE	C	(长度)	生成指定长度空白字符串
STUFF	C	(字串 1,位置,个数,字串 2)	由字串 2 删除插入字串 1
SUBSTR	C	(字串,位置,长度)	取子字符串
TRANSFORM	C	(表达式,格式字串)	格式编辑表达式结果
TRIM	C	(字符串)	去掉字符串尾部空格
RTRIM	C	(字符串)	同 TRIM(.)函数
UPPER	C	(字符串)	字母小到大转换

其中，变量部分各参数中的字符串可为同类型表达式。

2. 数值处理函数

函数名	结果类型	变量	说明
ABS	N	(NL)	求绝对值
EXP	N	(NL)	求自然指数值
LOG	N	(NL)	求自然对数值
INT	N	(NL)	取整,不四舍五入
MAX	N	(NL1,NL2)	求最大值
MIN	N	(NL1,NL2)	求最小值
MOD	N	(NL1,NL2)	取模
ROUND	N	(NL1,NL2)	四舍五入定小数位
SQRT	N	(NL)	求平方根

其中，NL，NL1，NL2 为数字表达式.

3. 日期处理函数

函数名	结果类型	变量	说明
CDOW	C	(日期)	星期字符串
CMONTH	C	(日期)	月份字符串
CTOD	D	(日期字符串)	字符串转换成日期型数据
DATE	D	( )	系统日期
DAY	N	(日期)	日期号数
DOW	N	(日期)	星期值数字(1--7)
DTOC	C	(日期)	日期转换成字符串
MONTH	N	(日期)	日期月份值
TIME	C	( )	系统时间
YEAR	N	(日期)	日期年份值

其中，自变量参数部分的日期可为日期型表达式，即日期型变量加或减某个数值.

4. 转换处理函数

函数名	结果类型	变量	说明
ASC	N	(字符串)	字符串第一字符的 ASCII 码值
CHR	C	(NL)	ASCII 码值与 NL 结果相同的字符
STR	C	(NL,长度,小数位)	数值转换成字符串
VAL	N	(字符串)	字符串第一字符的 ASCII 码值

其中，NL 为数字表达式.

5. 库文件测试函数

函数名	结果类型	变量	说明
BOF	L	( )	开始标志测试
DELETED	L	( )	记录删除标志测试
EOF	L	( )	结束标志测试
FILE	L	('文件名')	文件是否存在测试
FOUND	L	( )	搜索是否成功测试
LUPDATE	D	( )	当前库文件更新日期

函数名	结果类型	变量	说明
RECCOUNT	N	( )	当前库文件的记录数
RECNO	N	( )	当前记录号
RECSIZE	N	( )	当前库文件记录长度
DBF	C	( )	当前库文件名
FIELD	C	(NL)	根据 NL 序号值求字段名
NDX	C	(NL)	求序号为 NL 的索引文件名
ALIAS	C	(NL)	求 NL 为工作区号的库文件名
FCOUNT	N	(NL)	工作区 NL 库文件的字段数

其中，NL 为数字表达式.

6. 环境测试函数

函数名	结果类型	变量	说明
COL	N	( )	当前光标所在列
ROW	N	( )	当前光标所在行
DISKSPACE	N	( )	当前工作盘自由空间(BYTE)
ERROR	N	( )	出错号
IIF	C,D,N	(条件,式 1,式 2)	根据条件求表达式值
ISALPHA	L	(字符串)	测试字符串是否以字母开始
ISCOLOR	L	( )	测试当前显示器工作方式
ISUPPER	L	(字符串)	测试字符串是否以大写字母开始
ISLOWER	L	(字符串)	测试字符串是否以小写字母开始
LEN	N	(字符串)	字符串长度
MESSAGE	C	( )	出错信息
PROW	N	( )	当前打印头行值
PCOL	N	( )	当前打印头列值
TYPE	C	('表达式')	测试表达式类型
FKLABEL	C	(键号)	功能键名
FKMAX	N	( )	最大功能键号
GETENV	C	( )	当前 DOS 环境参数
OS	C	( )	当前操作系统名
VERSION	C	( )	当前 FoxBASE+版本号
UPDATE	L	( )	测试当前 READ 命令是否修改变量
SELECT	N	( )	当前工作区号
SYS	C	(16,I)	测试当前应用程序名

7. 输入处理函数

函数名	结果类型	变量	说明
INKEY	N	([NL])	测试按键 ASCII 码值
READKEY	N	( )	全屏幕按键测试
&		(字符型变量)	宏代替

其中，NL 为数字表达式.

FoxBASE+的函数如此丰富，功能如此齐全，若能够在编程中以最大效率运用，则必将发挥出极大的效果。从以上所列函数来看，字符处理函数、数值处理函数及转换处理函数与另一数据库管理系统 dBASEⅢ是相差无几的，也是较好掌握和运用的三类函数。FoxBASE+函数的特点主要体现在其它几类函数上。在以下几个小节中，将就以上所列函数中有特点的一些函数加以介绍，说明其编程功能。

1.1.1 ASCII 码值转字符函数的编程运用

CHR 函数是所有高级语言及 DBMS 都有的转换函数，其将数字表达式值表示的 ASCII 码值转换成相应的字符，函数格式如下:

CHR（<数字表达式>）

数字表达式的取值范围为 0～255，即对应标准 ASCII 码表的一字节表示范围。

在 256 个 ASCII 字符中，以 32～127 为普通 ASCII 字符，在任何语言及 DBMS 中均为可显示字符，可以由键盘直接输入，也可以用输出语句显示或打印输出。128～255 为扩展 ASCII 字符，这些字符不能从键盘直接输入，只能运用 CHR 函数间接输入与输出，亦即扩展字符的运用必须借助于转换函数 CHR 来进行。在汉字操作系统支持下，从 161 到 255 的部分被用作汉字机内码，当两个 ASCII 码值大于 161 的字符紧接输出时将不显示原 ASCII 字符，而是与此机内码对应的汉字。ASCII 码 0～31 一般作为控制码运用，相对应的 ASCII 字符既不能由键盘直接输入，并且，即使利用转换函数 CHR 也不能在屏幕和打印机上输出。只有在利用 BIOS 中断或直接向 VRAM 写时才能在屏幕上显示相应的 ASCII 字符。在 FoxBASE+中，无格式输出语句调用的是操作系统的显示功能进行显示输出的，因此，只能显示普通 ASCII 字符及扩展 ASCII 字符，而前 32 个 ASCII 码将在输出时起控制作用，如回车、换行、警报等。格式输出语句调用的是 BIOS 显示中断的功能，因此，可以全部输出 256 个标准 ASCII 字符，前 32 个 ASCII 码在输出时显示字符而不起控制作用。

从以上分析可知，转换函数 CHR 的编程运用可以归结为以下几个方面。

1. 输入功能

由于由键盘直接输入的字符只能是普通的 ASCII 字符，因此，若程序中需要扩展字符及前 32 个控制字符，只有通过此函数来表示。

2. 输出功能

同样地，由于非普通 ASCII 字符不能直接从键盘输入，因此，在输出语句中要输出这类字符的话，只有用此函数来表示输出。无格式输出语句只能利用此函数输出扩展字符，而格式输出命令则可以利用这一函数输出扩展字符和控制字符。当然，普通 ASCII 字符也可以由此函数表示输出，但一般不需要这样处理。

3. 控制功能

控制功能是由无格式输出语句输出由此函数表示的控制字符来实现的。例如，以下的几条命令将分别实现报警、屏幕画线、屏幕拷贝及输出格式控制的功能。

(1)？ CHR(7)+CHR(7)

(2)？ CHR(14)+'D20, 20 L100, 100]'

(3)？ CHR(27)+'W'

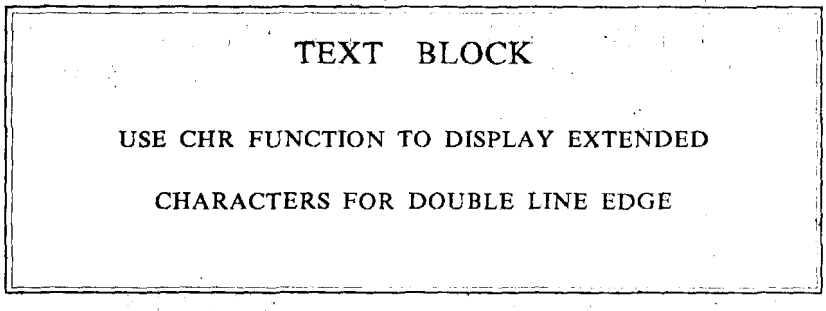
(4)？ '苏州'+CHR(10)+'大学'

以上命令中，(2)和(3)仅在 CCDOS 2.13H 系列汉字操作系统支持下运用才有效。

关于 CHR 函数的一个应用，下面给出由其输出扩展字符来构成屏幕文本区边框的程序例，该程序 CH1_8.PRG 在西文操作系统支持下将在屏幕上画出一矩形双线框，其内容及程序运行结果如下。

```
*CH1_8.PRG
SET TALK OFF
SET STATUS OFF
SET SCOREBOARD OFF
SET COLOR TO 7/
CLEA
SET COLOR TO 6+/0
@ 5,5 CLEAR TO 20,72
@ 5,5 SAY CHR(201)+REPLICATE(CHR(205),65)+CHR(187)
K=6
DO WHILE K<=19
   @ K,5 SAY CHR(186)+SPACE(65)+CHR(186)
   K=K+1
ENDDO
@ 20,5 SAY CHR(200)+REPLICATE(CHR(205),65)+CHR(188)
SET COLOR TO 2/1
@ 5,30 SAY 'TEXT BLOCK'
SET COLOR TO 4/1
@ 10,10 SAY 'USE CHR FUNCTION TO DISPLAY EXTENDED'
@ 13,25 SAY 'CHARACTERS FOR DOUBLE LINE EDGE'
WAIT ''
SET COLOR TO 7/
RETURN
*END
```

运行结果:



上例由双线框将文本区围起的方式在许多商用软件中经常采用，而 FoxBASE+本身的边框设置命令并不能实现这一功能，利用 CHR 函数显示扩展字符的功能却很好地实现了这一点。但是，在汉字操作系统支持下，以上程序的结果将发生变化。因为程序中 CHR 函数转换的是汉字机内码，所以，变框将变成汉字。

CHR 函数的功能除以上所述三点之外，还有一个功能是作为 FoxBASE+和汇编语言程序的传递参数，这将在本章第三节介绍。

1.1.2 程序按键测试函数及编程运用

FoxBASE+中有丰富的输入语句，包括格式输入语句和非格式输入语句两大类。但是，所有这些语句都无法从键盘输入 ASCII 码表中前 32 个和后 128 个字符。实际应用中，由于前 32 个 ASCII 码往往为控制码运用，为了程序、设备的控制作用，还是希望能有某种手段来输入这些控制码值，并用于各种控制中。例如，应用程序中有时需要由↑、↓、←、→这四个键来控制光标的移动，由 Ctrl+C 来结束程序的运行等。所有这些按键对输入语句来说，除←键与→键之外均等同于回车键，不会产生任何控制效果。要在程序中利用这些控制键达到控制的目的，必须利用 FoxBASE+中提供的按键测试函数，其格式如下:

INKEY( ) / INKEY(<数字表达式>)

该函数从键盘缓冲区中取自上一次回车以来键盘输入的 ASCII 码值，其值域为 0～255。因此，INKEY( )函数可以从键盘接收所有 ASCII 码。如果上一次回车以来未按任何键，则该函数接收空字符，其 ASCII 码值或 INKEY( )函数值为 0。INKEY( )函数的这些特点对于程序中设计优良的人机交互界面是极有帮助的。一些控制键及对应的 INKEY( )函数值如下:

控制键	等价键	函数值	控制键	等价键	函数值
→	Ctrl+D	4	Home	Ctrl+A	1
←	Ctrl+S	19	End	Ctrl+F	6
↑	Ctrl+E	5	PgUp	Ctrl+R	18
↓	Ctrl+X	24	PgDn	Ctrl+C	3
Ctrl+→	Ctrl+B	2	Ctrl+Home	Ctrl+I	29
Ctrl+←	Ctrl+Z	26	Ctrl+End	Ctrl+W	23
Ins	Ctrl+V	22	Ctrl+PgUp	Ctrl+-	31
Del	Ctrl+G	7	Ctrl+PgDn	Ctrl+^	30

需要注意的是，INKEY( )函数虽然可以从键盘接收字符，但由于不是输入语句，只能参加表达式或作为程序语句的一部分。INKEY( )函数测试字符的时间很短，并不会象输入语句一样暂停程序等待键盘输入。利用 INKEY( )函数输入字符的方法一般是结合循环语句进行的，其典型的语句结构如下。

```
KEYV=0
DO WHILE KEYV=0
   KEYV=INKEY( )
ENDDO
```

这一语句段利用循环语句使 INKEY( )函数反复测试键盘缓冲区，直至有一个键被按下，使 KEYV≠0，这种方式称为非限时测试。当然也可以设计程序使其仅测试某一时间段，到了一定时间，即使不按键也不再测试，程序继续执行，这种方式称为限时测试。有两种手段来实现限时测试，一种是改变以上语句段为如下内容。

```
K=1
CT=500
DO WHILE K<=CT
   KEYV=INKEY( )
```

```
    IF KEYV<>0
       EXIT
    ENDIF
    K=K+1
ENDDO
```

以上语句段由 CT 来控制循环次数即测试时间。另一种实现限时测试的方法是利用带自变量的 INKEY(<数字表达式>)。在这一形式的函数中，其自变量表示该函数等待测试时间，以秒为单位。例如，INKEY(5)表示按键测试时间最多为 5 秒，在此期间有按键则马上结束测试，否则一直进行直到满 5 秒时间。

限时测试和非限时测试各有特点，在具体设计时应具体分析，根据实际需要采用不同方法，一般非限时测试用得较多。

设屏幕显示范围为 25× 80 字符，以下程序 CH1_13.PRG 由光标键控制光标的全方位移动，回车定位输出 ASCII 字符，并重复以上过程，直至 Ctrl+C 终止。程序中按键测试采用的是非限时测试方式。

```
* CH1_13.PRG
SET TALK OFF
SET ESCAPE OFF
SET COLOR TO 3/
NUM=0
X=12
Y=35
MAXCOL=79
MAXROW=23
MINCOL=0
MINROW=0
CLEA
SET COLOR TO 5/2
@ 24,0 SAY SPACE(80)
SET COLOR TO 3/4
@ 24,12 SAY ' 移动光标↑ ↓←→选位,
          回车定位,Ctrl-C 终止... '
@ X,Y SAY ''
SET COLOR TO 5+/2
DO WHILE .T.
   K=0
   DO WHILE K=0
      K=INKEY( )
   ENDDO
   @ X,Y SAY ''
   DO CASE
      CASE K=4
         IF COL( )<MAXCOL
            Y=COL( )+1
         ENDIF
      CASE K=19
         IF COL( )>MINCOL
            Y=COL( )-1
         ENDIF
      CASE K=5
         IF ROW( )>MINROW
            X=ROW( )-1
         ENDIF
      CASE K=24
         IF ROW( )<MAXROW
            X=ROW( )+1
         ENDIF
      CASE K=3
         SET COLOR TO 7/
         RETURN
      CASE K=13
         NUM=NUM+1
         @ X,Y SAY CHR(NUM)
         LOOP
   ENDCASE
   @ X,Y SAY ' '
ENDDO
* END
```

1.1.3 全屏幕编辑按键测试函数及编程运用

前一小节所述程序按键测试函数与输入语句是相互独立的，互不影响。这样，当输入语句接收键盘数据时就无法接收控制键。在实际应用系统中却往往有这样的要求，即在输入语句输入时也可以接收控制键，并根据控制键进行相应的处理或使程序转移执行方向，

但对于真正接收的数据应不受控制键的影响，即一切控制键的输入对整个数据而言相当于回车。满足这种将接收数据和接收控制键既独立又联系地体现在输入语句中的要求可以由全屏幕编辑按键测试函数来实现，其格式如下:

READKEY( )

对于 FoxBASE+中的输入语句，仅仅只有格式输入语句才能与该函数结合使用，非格式输入语句无效。READKEY( )函数测试最近一次格式输入命令 READ 的编辑结束按键，如果 READ 命令没有修改原 GET 变量值，则函数分布在 0～36 这一区域，反之则分布在 256～292。READKEY( )函数可以接收的结束按键及对应函数值如下:

控制键	等价键	函数值	控制键	等价键	函数值
→	Ctrl+D	1(257)	Home	Ctrl+A	2(258)
←	Ctrl+S	0(256)	End	Ctrl+F	3(259)
↑	Ctrl+E	4(260)	PgUp	Ctrl+R	6(262)
↓	Ctrl+X	5(261)	PgDn	Ctrl+C	7(263)
Ctrl+→	Ctrl+B	9(265)	Ctrl+Home	Ctrl+I	33(289)
Ctrl+←	Ctrl+Z	8(264)	Ctrl+End	Ctrl+W	14(270)
Ctrl+U		10(266)	Ctrl+PgUp	Ctrl+—	34(290)
Ctrl+N		11(267)	Ctrl+PgDn	Ctrl+∧	35(291)
Ctrl+Q		12(268)	Ctrl+M		15(271)
✓		16(272)	F1		36(292)

以上列出的控制键在格式输入命令仅作用于一个 GET 变量时，对输入的或修改的数据而言，除 BACKSPACE、←和→以外同回车是一样的，不会影响 GET 变量的值。例如，下面这段语句从键盘传送数据给变量 VREAD，可以输入 50 后回车，也可以输入 50 后按↓键，其结果 VREAD 的值都为 50，但 READKEY( )函数值不一样，前一种情况为 272，后一种情况为 257。

```
VREAD=0
@5, 5 SAY '输入数据' GET VREAD
READ
?VREAD
?READKEY( )
```

对于 READ 命令对应多个 GET 变量的情况，例如 READ 命令激活格式文件的情况，这些控制键即为全屏幕编辑键。如↓将把光标移到下一 GET 变量输入域，READKEY( )函数总是测试最近一次 GET 变量输入的结束按键。

以下为应用全屏幕编辑按键测试函数，在修改检查库文件 STAFF.DBF 记录字段职称和工资的同时，由控制键 Ctrl+PgDn 将该记录数据直接输出到打印机，由 Ctrl+End 结束程序。

```
* CH1_14.PRG
SET TALK OFF
SET ESCAPE OFF
SET COLOR TO 2/
CLEA
USE STAFF
```

```
@ 3,20 SAY '====检查修改职称工资===='
@ 5,0 SAY ''
ACCEPT '输入起始职工编号:' TO STARTN
LOCATE FOR 职工编号=STARTN
IF .NOT.FOUND( )
    ? '库中无职工编号为'+STARTN+'的记录...'
    RETURN
ENDIF
CLEAR
@ 3,5 SAY '职工编号'
@ 3,20 SAY '职工姓名'
@ 3,35 SAY '职称'
@ 3,50 SAY '工资'
@ 4,2 SAY REPLICATE('=',70)
@ 6,2 SAY REPLICATE('=',70)
SET COLOR TO 6+/,/6
DO WHILE .NOT.EOF( )
    @ 5,2 SAY SPACE(70)
    @ 5,5 SAY 职工编号
    @ 5,20 SAY 职工姓名
    @ 5,35 SAY 职称
    @ 5,50 SAY 工资
    @ 5,35 GET 职称
    READ
    DO SUB1_14
    @ 5,50 GET 工资
    READ
    DO SUB1_14
    SKIP
ENDDO
RETURN
*END
* SUB1_14.PRG
SET TALK OFF
SET ESCAPE OFF
IF READKEY( )=35.OR.READKEY( )=291
    SET DEVICE TO PRINT
    @ PROW( )+1,5 SAY 职工编号
    @ PROW( ),20 SAY 职工姓名
    @ PROW( ),35 SAY 职称
    @ PROW( ),50 SAY 工资
    @ PROW( )+1,0 SAY ''
    SET DEVICE TO SCREEN
ENDIF
IF READKEY( )=14.OR.READKEY( )=270
    CANCEL
ENDIF
RETURN
*ENDSUB
```

READKEY( )函数的作用使得程序在接收数据的同时也能接收控制键，而

INKEY( )函数则做不到．但是，从程序控制的角度来看，由于 READKEY( )必须与 READ 命令结合使用，因此是不方便的，而 INKEY( )函数却可以很方便地实施控制．

§1.2 命令语句及编程设计

前一节介绍了函数的编程运用，这一节将讨论 FoxBASE+命令语句在编程设计中的运用．这里，我们并不想对 FoxBASE+的所有命令语句一一介绍，而仅仅选择命令语句中对编程设计较有影响及需要充分考虑的部分语句进行论述．

FoxBASE+的程序语句由两大类组成：一类是不仅可以运用于程序，而且在系统状态下可以单独执行的语句，另一类是仅用于程序逻辑结构控制的语句，离开程序在系统状态下执行毫无意义，如循环语句、分支转移语句等．前一类语句称为命令语句，如何在编程中对其充分考虑、优化运用是这一节将要论述的内容．

1.2.1 库文件结构处理命令及编程运用

在编程中常用的库文件结构处理命令有如下三条：

(1) COPY STRUCTURE TO ＜库文件名＞ [FIELDS＜字段表＞]

(2) COPY STRUCTURE TO ＜结构描述库文件＞ EXTENDED

(3) CREATE ＜库文件名＞ FROM ＜结构描述库文件＞

第一条命令将当前库文件的结构按指定字段表拷贝到指定库文件，该库文件的字段及参数与当前库文件结构中的部分或全部字段相同，而库内容为空．第二条命令将当前库文件结构拷贝到指定结构描述库文件，其内容为当前库文件的结构，这一命令完成库结构到库内容的转换．第三条命令依据结构描述库文件内容所示结构建立指定文件，其库结构为结构描述库文件的内容，而库记录内容为空，这命令完成库内容到库结构的转换．

这三条命令在应用系统中经常用来间接生成库文件．一般来说，第一条命令在应用系统有一系列相同结构库文件时运用，而第二、三条命令在应用系统需要生成任意结构库文件时运用，以下分别介绍它们的编程运用．

1. 库文件结构的拷贝

在会计电算化财务软件、银行信贷管理系统及其它应用系统中，对大量的数据往往需要按单位、部门或按时间分别存储于不同库文件中，从而便于数据管理．例如每个月或每年一个库文件．这样，在应用系统中就有一系列的库文件，它们的库结构都相同，所存储的数据都具有相同的特点，并且库文件名往往有相同的前缀．例如，对每月一个库文件的情况，各月库文件名可以如下规定：

DATA1.DBF，DATA2.DBF，...，DATA12.DBF

库文件的这种命名法对系统的处理有极大好处．例如，当需要查询某月数据时，只需输入月份就可以根据库文件名后二位或一位数字找到相应数据所在的库文件．一个需要考虑的问题是当某月结束，系统进入新的月份时，数据应该向该月份库文件输送，但此时该库文件并不存在，这时程序应该自动生成新的库文件来接收数据．例如，当从 9 月份进入 10 月份时，程序应该自动生成 10 月份的库文件 DATA10.DBF．在这种情况下，由于每月库文件特性相同，因而可以采用 COPY STRUCTURE TO 命令来生成．库文件的生