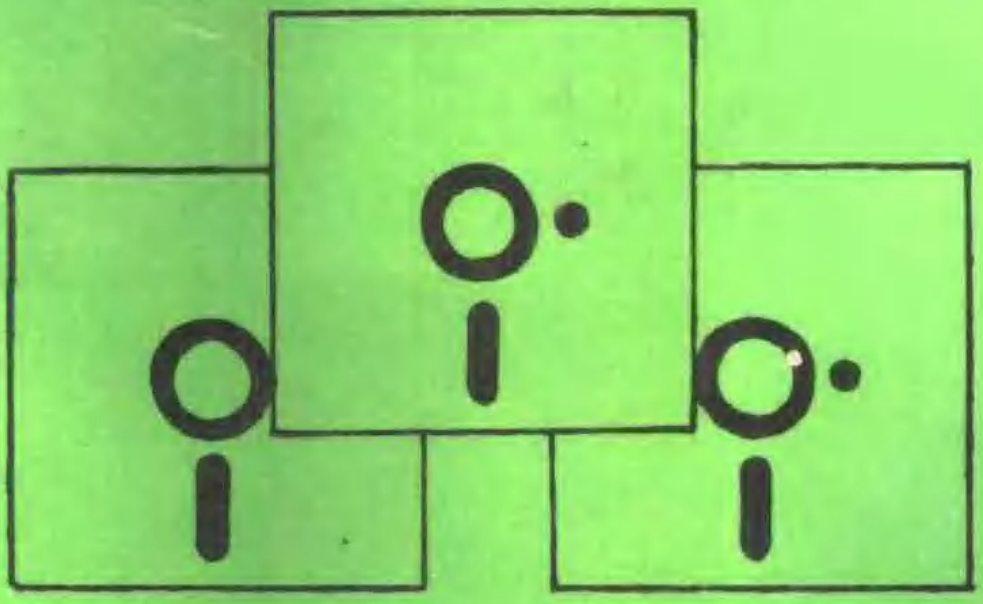
DBASE Ⅲ

程序设计技巧集锦



陕西电子编辑部

前　言

目前，汉字 dBASEⅢ在我国广为流行，是微机管理和开发应用程序的主要应用软件之一，它为建立数据库、汉字应用、数据处理等，提供了强有力的工具。

如何更好地使用 dBASEⅢ，使其发挥更大的作用，是每个程序设计者所关心的问题，为此，陕西电子编辑部，根据国内外大量的资料，编辑了此书，旨在解决教科书和应用手册中没有提及的问题。不失为 dBASEⅢ使用者的良师益友!

由于编者水平有限，本书中难免有误，希望广大读者专家斧正，谢谢!

目　　录

解决高分辨率 dBASEⅢ长语句隐藏问题

在长城 0520 及其它高分辨率机上，用 25 行 dBASEⅢ编辑程序时，如果语句太长，则经常出现假现象（例如：用户清楚地看到将光标置至本行，要删除，修改或插入。但结果修改的不是它的前一行，就是它的下一行），使用起来非常不便。解决这个问题的方法是

一、预防为主

如果在编辑程序时，这种现象还没有发生，那就要预防它的发生。这里谈几种预防方法，供同志们选择采用。

第一种方法：换用其它的编辑软件对它进行编辑。如行编辑软件、字处理软件，及 10 行 dBASEⅢ软件。由于高分辨率的计算机大部分都配有汉卡，这样供用记使用的内存空间就比较多。用户勿需退出 dBASEⅢ就可以用 RUN 命令运行外部命令，对程序进行编辑。这此编辑软件都不会发生转行隐藏现象。

第二种方法：这种方法的特点是铁需退出 dBASEⅢ和运行其它外部软件。而是通过自身解决。它的基本思想是用户在输入或编辑程序时，稍加注意，遇到特别长的语句时，不要一行都输入完，而是分多行进行输入，每行大约 50 个字符。输入编辑好之后，依次从最后一行开始，在行首打入"←"键，这样下一行的内容就被合并到上一行去了。依次往上，就可以将分成多行输入的语句合并成一行。同样要对一条长误句进行修改编辑时，可将"TNSERT"键打开，从上面开始，在需要分行处打入回车键。这样回车以后的内容就显示在下一行上，依次类推，将一条长语句分成多行，然后按多行进行编辑正确后，再从最后一行开始，用"←"键将它们合并成一行可执行语句。

二、妙法修改

在实际应用中，也会发生这种现象，解决的办法亦有两种：

第一，如果发生了这种现象，但未退出编辑状态，这时，光标还可以移到行尾，在最后打入几个删除键，就可以达到修改的目的。

第二，如果已是编辑状态，解决的最好办法是退出 25 行 dBASEⅢ，而进入 10 行 dBASEⅢ，调出该程序进行编辑，因为低分辨率 dBASEⅢ不会发生隐藏现象。编辑时，还可以使用插入键，回车键及删除键。编辑正确以后，在高分辨率下可正确运行。

dBAESⅢ中栈的实现

栈是一种应用广泛的数据结构。编译程序里的表达式求值、高级语言里的子程序调用、递归的实现等都使用了栈结构。本人用编译 dBASE—Ⅲ开发的一个软件系统中，在树遍历模块和算术表达式的词法、语法分析模块中也都使用了栈结构。用计算机解决有关后进先处理的问题时，很自然地会使用栈结构。本文给出了 dBASE—Ⅲ中栈的一种实现方法。

在 dBASE—Ⅲ中，可以采用字符串变量来实现栈的各种操作。实现原理是：进栈时经过处理的字符型数据项附加在字符串变量的值的尾端，出栈时截掉字符串变量的值的尾

端数据项，算法如下：

设 S 是一个字符串变量，表示栈，i 为整型变量，表示栈指针，num 为整型变量，表示数据项。

栈的初始化：S=" "；i<0

栈中数据项的长度：4

栈的最大长度：5

栈为空的条件：S=" "或 i=0

栈为满的条件：LET（S）=20 或 i>5

进栈过程：

```
procedure pushs
if i> = 5
    ? "栈满"
else
    i=i+1
    s=s+str (num, 4)
endif
return
```

出栈过程

```
procedure pops
if i< = 0
    ? "栈空"
else
i=i-1
s=substr (s, 1, i*4)
endif
return
```

举一个例子来说明上面的算法。

设有 4 个整数：20、9、250、3116，这 4 个整数的进栈、出栈的过程如下：

程序	栈值的变化
num=20	
do pushs	进栈：S="20"
num=9	
do pushs	进栈："20 9"
num=250	
do pushs	进栈：S="20 9 250"
num=3116	
do pushs	出栈：S="20 9 250"
pops	出栈："20 9"
pops	出栈："20"
pops	出栈：" "

上面介绍了用字符串变量来实现栈的各种操作，当然还有其它方法（比如用单字段数据库文件来实现），这里就不讨论了。

勿忘执行 QUIT 命令

中文 C—dBASE 数据库以其使用灵活方便受到人们的普遍欢迎。但由于人们对某些操作上的忽视，如退机时忘记执行 QUIT 命令，常常致使一些不该发生的事情发生。

众所周知 dBASE 数据库是一种面向办公室自动化管理的工具软件，由于其特点是直接面向操作者和对计算机了解甚少的企、事业管理人员，许多实施操作均是由机器内部自动管理的，人工不能干预，比如 ctrl-w 的修改存盘，有时并不马上执行操作，而是参照内存储器 RAM 的使用等情况由系统随机管理。为此 dBASE 设计了退出系统时用 QUIT 命令对系统环境进行一次清理，该存盘的存盘；对原有信息还要进行一次反存处理。不了解这一点，忽略操作，往往就会丢失数据。

三年前曾发生过一件事：程序员 A 与 B 用一台计算机开发 C-dBASE—Ⅱ（2.41 版）用户程序。一日，程序员 A 在进行程序编辑修改后没有执行 QUIT 退机操作，便把软磁盘从驱动器中取走，机器的屏幕显示仍然停留在 dBASE 的系统提示符"·"状态下。之后程序员 B 考虑是应用模版本软件编制应用程序，没有例行启动程序，插入程序盘即行操作，刚刚敲入一条命令，意外情况发生了。程序员 B 原来编制的程序被机器"冲"的七零八碎，整个磁盘空间见不到完整的文件。更使人奇怪的是，在列磁盘目录时，有相当一部分的文件名均被程序员 A 盘的文件名所替代。由于这个单位的磁盘曾一度统一管理，为此还造成一场不小的误会……

经调查分析，原来 dBASE 数据库软件的许多命令并不是立即执行的，尤其是涉及外部设备的命令。为此，dBASE—Ⅱ还专门设计了一条环境设置命令 RESET（dBASEⅡ版本），用以通知系统更换磁盘，否则将会出现不可收拾的错误。

通过以上事例的分析，提醒人们切不可为了一时的方便不执行退机，或是直接把机器的电源关掉，都可能丢失数据。请注意，应用 dBASE 数据库后千万不要忘了用 QUIT 退回系统环境。

dBASEⅢ模糊检索技术

一般情况下，对数据库进行检索时需要输入完整的定位条件，有时因输入字符容易出错，且效率低。例如：要检索题目为《模拟计算机和数字计算机及其系统》的刊物时这就要求用户一字不差的输入定位刊物名称，否则便会检索失败，有时因记不清或遗忘便无法进行检索。

为了解决上述 dBASEⅢ检索数据库中存在的问题，就需要寻找新的检索方法，我们经过实践摸索找到一种方法，即模糊检索方法。

所谓模糊检索，是指用户输入不完整或一些词语作为检索条件，然后检索出库中满足给定条件的一些记录，再从这些记录中得到所需找的记录。

实现模糊检索的指导思想是：分解检索长词组为短小词语，或者再用逻辑符

(AND，OR）连接组合这些词语作为检索定位条件进行检索定位．通过dBASEⅢ的字符运算函数$来实现．

设A，B变量赋值如下：

A="abcdefgh"

B="cdef"

则字符运算函数$运算结果：

.? A$B

.F.

.? B$A

.T.

字符运算函数$首先测试前一变量的字符是否包含在后面变量的字符中，若包含（成立）运算结果得到逻辑真（T）值，否则得到逻辑假（F）值．

为进一步说明问题，下面举例．

若有数据库名为BKK.DBF，其结构如下：

字段名	类型	宽度
BKDH（刊物代号）	C	6
BKM（刊物代号）	C	20
BNJ（价　　格）	N	5.2
* * Total * *		32

若用户欲检索名称为：《无线电电子学与自动化》刊物，采用以下模糊检索程序，便能很快得到检索结果．这一小程序就是采用了模糊检索技术．

```
* * 程序: JS88/7/20
set exact off
set talk off
use bkk.dbf
do while.t.
    clear
    store space (10) to bm
    @4, 10 say"请输入模糊刊物名称:"get bm
    read
    stor trim (bm) to bm
    locate for bm $ BKM
    stor 1 to n.h
    clear
    do while.not.eof()
       if n>8.and.h=1
          stor 1 to n
```

```
            stor  40  to h
         endif
         if n>8.and.h=40
            wait"看清后按一键继续…"
            stor  1  to h，n
         clear
         endif
         @n，h say"|"+BKDH+""+BKM+""+str（BNJ，5，2）
         stor n+1  to n
         continue
      enddo
      wait"查看完含有("+bm+")词语的刊物，按键!"、、、
      clear
      stor space（2）to sf
      @ 4，10 say"是否继续模糊检索?
         是（Y）否（N)"get sf
      read
      if upper（sf）="Y"
         loop
      endif
      exit
   edddo
```

当你运行这一程序，屏幕上出现:"请输入模糊刊物名:"这一提示时，你输入"电子"一词语，并按回车键后便会出现如下检索结果:（本文未给出 BKK，DBF 内容）

BKDH	BKM	BNJ
2-888	电子与电脑	2.76
2-889	电子技术应用	3.00
2-890	电子科学技术	3.30
2-708	无线电电子学与自动化	12.60
22-52	电子工艺技术与自动化	2.70

以上的例子很简单，旨在说明模糊检索技术的应用，还可以推广到多字段的复合条件模糊检索。

dBASEⅢ与 BASICA 互用探讨

用 dBASEⅢ语言来编写各种事务性管理系统已是司空见惯的事情了，现在几乎所有的事务管理系统都用来编制。但如何科学地利用各种高级语言之优点参与 dBASEⅢ的编

写工作却是鲜为人知的。用 BASIC、FORTRAN、PASCAL 及 COBOL 等高级语言来参与 dBASEⅢ的编程工作，不但可以提高程序的运算和处理能力，而且可以增强管理系统的灵活性。本文将重点阐述 dBASEⅢ与 BASICA 语言的互用问题，而对其它语言概不作叙述。现将对 dBASEⅢ与 BASICA 语言的互用性进行探讨：

一、用批处理方法实现从 BASICA 到 dBASEⅢ的转换

假定运行 BASICA 文件 ZK.BAS 后就必须转入运行 dBASEⅢ中的 ZKM.PRG 命令文件，这时我们可以先在 BASICA（或 EDLIN）状态下，在 ZK.BAS 文件中加入如下语句：

CLS: LOCATE 5, 6: PRINT"请插入 dBASE 软盘,": LOCATE 6, 18: PRINT "后按 RETURN 键"SYSTEM.

然后建一个批处理文件 ZK.BAT，建法如下：

```
C>COPY CON ZK.BAT
    ECHO OFF
    IF EXIST ZK.BAS BASICA ZK
    ECHO OFF
    IF EXIST ZKM.PRG DBASE ZKM
    ECHO ON
    ∧Z（或按 F6 键）
```

这样每次运行这个系统时，只要在 C>状态下敲一下 ZK 的后敲回车键便可以实现从 BASICA 状态到 dBASEⅢ状态下的转变过程。

二、dBASEⅢ状态下运行 BASICA 程序

只要在 dBASEⅢ命令文件中加入如下命令则可：

RUN BASICA 文件名·后缀（若后缀为·BAS，则可剩 T 后缀）。但 BASICA 文件必须有"SYSTEM"这个语句，才能自动返回 dBASE 原来所在之状态。

三、数据库互换

这方面可直接翻阅（《中国计算机用户》1987.第 13、14 期）"dBASEⅢ与 BASICA 联用探讨"一文，在 dBASEⅢ中具体可用这个命令：

COPY TO<系统数据格式文件名>〔范围〕〔FOR<逻辑表达式>〔FIELE<数据项名>〕〔SDF/DELIMITES〔WITH<界限符>〕〕

通过对 dBASEⅢ与 BASICA 互用的探讨，将对事务管理系统的编写与应用增强了很多灵活性，同时还能起到 dBASEⅢ及 BASICA 自身所不能起到的作用。

CdBASEⅢ程序调试技巧

在调试中文 CdBASEⅢ程序时会出现各种问题，系统可以把各种错误信息提示给你。但是对于刚刚开始用 CdBASEⅢ编程的人来说，怎样解决这些问题，往往会耗费许多时间。经过本人的工作实践，弄清了一些错误问题的解决方法。下面就本人体会较深、花费时间、精力较多的问题浅谈几点解决方法，供大家参考。

(1) 在调试程序中，屏幕上若有错误信息"无效的函数变量"提示出现，且在某变量

的上方有"?"出现时，解决该问题的方法是：你可用"LISTMEMORY"命令先来查看该内字变量的类型，看它是否与你确定的该变量的类型一致．若不一致，你就应该检查该变量是什么原因未被赋值，解决了此矛盾，该错误就消除了．出现该错误的原因大都是该变量在程序中未被赋值，系统自动将该变量赋予逻辑值，造成数据类型不匹配的原因所致．

(2) 若有"打开的文件个数太多"的错误信息提示出现时，解决该问题的方法是：首先查看你所打开的文件个数是否符合 CdBASEⅢ允许打开的文件个数的要求（即可同时打开 15 个所有类型的文件或 10 个数据库文件），若不符合则修改程序使之符合．若符合，你下一步就应该查看 config.sys 文件的设置，对于 IBM PC／XT 内存为 640KB 的主机，config.sys 文件的内容应为如下命令：

FILES＝24

BUFFERS＝15

若 config.sys 文件内存与上述不同则修改使之相同，若相同仍出现错误，这时，你应特别注意 config.sys 文件不能放在 CdBASEⅢ系统主盘上，而应与 CCDOS 放在一起，否则仍出现错误的信息提示，这是因为启动系统的中西文 CCDOS 没有考虑到 CdBASEⅢ对缓冲区数和文件数的要求，它的 config.sys 文件中仅包含一条设备驱动命令(DEVICE＝ANSI.SYS)，这样 CdASEⅢ就不能同时打开 15 个文件，必须在 config.sys 中添加设置缓冲区数和文件数的命令，然后重新启动 CCDOS，才会使 config.sys 生效，否则仍会出现错误的信息提示．

注意：对于 IBM PC／XT 内存为 640KB 的主机 config.sys 文件中：FILES＝24，BUFFDRS＝15 为宜．

(3) 在利用 str（＜数值表达式＞[，长度＞][，＜小数位＞）函数，将数值表达式的值转换成字符串输出时，若给定的输出字符串的"长度"小于数值中的位数，将输出星号(＊)，若数值表达式的值符合要求，便始终不显示结果，这时你就应该检查 str 函数中的"长度"的设置是否符合数值的最大位数为 19 位的要求，若超出了数值的位数允许范围，则显示不出结果来，此时，你必须将 str 函数中"长度"值改为小于等于 19 的数值，这样就可以显示出你要显示的结果来了．

(4) 在 CdBASEⅢ字处理状态下，输入用 CdBASEⅢ编写的源程序时，若在屏幕上方显示出："file too large data may last"即：系统提示你文件太大，一些可能会被丢失．此时，你不能继续再输入程序了．否则，你再输入的程序不能被存盘，而导致部分程序丢失的现象发生．这时，你应存盘退出 CdBASEⅢ字处理状态，进入行编辑 EDLIN 状态，继续完成程序的输入．造成程序丢失的原因是 CdBASEⅢ"MODIFY COMMAND"命令编辑文件长度限于 4KB 字节，故超过 4KB 字节的程序最好是在 EDLIN 状态或其它状态下进行输入和修改程序．

一种提高分类求和处理速度的方法

dBASEⅢ是目前微型计算机上应用较为广泛的一种数据库管理系统，它提供了较强的管理方面的功能，但也存在着运算速度较慢的不足之处．

在利用 dBASEⅢ进行往辅助管理时，经常会遇到对数据库数据进行分类求和处理的

问题，有些分类求和处理利用建立分类求和文件命令（TOTAL）并不能实现．例如在计算《统物 6 表》时，要对各种材料的增加、减少进行分类统计，而在增加中还要分出：国家合同到货、上级机关调入、自采等，减少要分出：消耗和调出，这种分类在一般情况下用 TOTAL 命令不能实现．如果用几个"SUM 表达式清单 TO 内存变量清单 FOR 条件 I"语句，在原数据中进行计算，由于记录较多，运算速度很慢．

对于这种情况，如果利用"COPY TO 文件名 FOR 条件Ⅱ语句，将满足条件Ⅱ的记录拷贝到一个中间过渡数据库中，然后在另一个工作区打开此过渡数据库，再利用几个 SUM 语句进行分类求和，可提高数据处理的速度．笔者利用这种方法计算《统物 6 表》及其它类似的数据处理问题，速度可提高 5 倍以上．

CdBASEⅢ '@BZ'功能符号的效果

在 C—dBASEⅢ下打印表格时，对于数值为零的字段用空白符打印，可以使得表格清晰美观，很多报刊都有经验介绍．笔者介绍一种最简单而且运行速度快的办法，既可以使整型数字字段为零时打印空白符，也可以使非整型数字字段为零时打印空白符（即小数点也不打印）．利用"@＜行，列＞SAY＜数学型字段名＞PICTURE '@BZ'"命令可达到上述目的．命令中'@Z'只能将整型数的零值用空白符打印，而非整型数的零值（如 *0.0*，*0.00* 等等）只是把"*0*"用空白符表示，小数点却照打不误．'@BZ'使二者结合使用除能完成各自的功能外，还产生一种特殊效果，那就是对于整型数或非整型数字段（变量），完全打印空白符．此方法编程简单，运行速度比用别的方法都快．

智力录入程序的方法

在用计算机处理日常事务的过程中，dBASEⅢ自然是最直接、最有效的工具．虽然，dBASEⅢ在 dBASEⅡ的基础上增加了备注型域的概念．但因备注型在数据录入过程中操作起来很不方便，因此，大多数程序设计者在设计程序时碰到一个域变量要描述较长内容的时候，仍喜欢用字符型域变量来描述，并且把它的长度设置成固定的．这种方法当把接收数据的内存变量设置的太短，则所要录入的内容录不下，往往要对原来内容进行压缩．其次，如果把内存变量设置的太长，格式文件又不好写，而且不能合理利用屏幕显示，为了解决这一问题，笔者在长期的实践中，总结了一套有效可行的方法，这种方法既克服了内容长、内存变量短、内容不能全部被接受的缺点，又避免了录入格式文件难写的不足．

这种方法的设计思想如下：

第一、在建立数据库结构时，应全面地分析待录入的内容，充分估计某些项目内容的长度，尽量增大域变量的长度，如果一个域变量还表达不了其内容可用多个域变量共同表达．

第二、把终端显示器上接收的内容首先送给一些内存变量，然后对这些内存变量进行合并处理再传送给域变量．

用这种思想编制的程序在执行时：首先在显示屏上显示等待录入的文字提示，然后根

据文字提示键入内容．一般情况下，较短内容的内存变量可几个占用一行，较长内容的内存变量可根据具体长短占用位置．假如某条记录的某个域变量要接受的内容较长，那么在格式文件中先给分配一行位置，等待接收数据．当这一行占满，紧接着再给它分配一行，它后面变量的文字提示也随着向后推一行．同样，第二行占满时，紧接再给它分配一行，它后面的文字提示也相应地往后推一行，以此类推，直到录完为止．假如某一条记录的这个域变量内容较少，即分配给它的那一行没有占满时，则不再给它分配位置．紧接着去录其它变量的内容．为了较直观地理解这一方法，试举一例，供参考．

例：有一个仅有三个域变量的数据库：KK．dbf，等待录入数期间，只假设 A1 很长有 254 个字符，A2、A3 域变则较短．现设计一程序要实现数据录入过程．程序开始要求录入条数．每录一条处理一下，既简单又明了．

程序清单如下：

```
set talk off
set color to 5
clear
input 请输入记录条数" to ts
use KK
go top
ii = 1
do while ii < = ts
  clear
  store space (74) To S1, S2, S3, S4, S5
  append blank
  @ 1, 30 say"data input"
  @ 2, 26 say ii picture "9999"
  @ 3, 1 say "NAME"get S0
  @ 4, 1 say"XB"
  @ 5, 1 say "MZ"
  read
  if len (trim(S0)) > 70
     @ 4, 1 get S1
     @ 5, 1 say"XB"
     @ 6, 1 say"MZ"
     read
     if len (trim(S1)) > 70
        @ 5, 1 get S2
        @ 6, 1 say"XB"
        @ 7, 1 say"MZ"
        read
        if len (trim (S2)) > 70
```

```
            @ 6, 1 get S3
            @ 7, 1 say"XB"
            @ 8, 1 say"MZ"
            read
            else
            @ 6, 1 say"XB"get S4
            @ 7, 1 say"MZ"get S5
            read
            end if
              @ else
              @ 5, 1 say"XB"get S4
              @ 6, 1 say"MZ"get S5
              read
              endif
            else
              @ 4, 1 say"XB"get S4
              @ 5, 1 say"MZ"get S5
              read
            end if
repl A1 with trim (S0) +trim (S1) +trim (S2),
     A2 with S4, A3 with S5
ii=i+1
ENDDD
CLOSE data
return
```

C—dBASEⅢ中"0.00"字段打印空白的办法

在打印程序中，将值为0.00的字段用空格形式输出，可使表格清晰、美观。具体有四种基本的处理方法，选用时应按其特点，根据用户的要求、记录的多少和字段的类型等来决定。

一、使用@命令：

程序1：

```
SET DEVI TO PRINT
SET PRINT ON
    ......
IF 其它津贴=0.00
  STORE 0 TO ABCD
ELSE
  STORE 其它津贴 TO ABCD
```

```
ENDIF
@PROW ( ), PCOL ( ) SAY ABCD PICT"@Z"
.....
```

注意：(1) IF 判断语句是必要的；(2) @命令中功能函数 Z 表示，当字段的内容为 0（而不是 0.00）时，输出空白字符。样模不能用"999.00"或"####　##"，否则，当字段的内容为 0（或 0.00）时，要在相应的位置输出小数点。

二、生成全字符型字段的打印库：

假定 GZ.DBF 中有"其它津贴"等数值型字段，将要打印的字段和结构拷贝成 GZ1.DBF，然后，修改 GZ1.DBF 中全部字段类型为字符型。

程序 2：.....

```
USE GZ1
SET SAFE OFF
ZAP
SET SAFE ON
USE
USE GZ
COPY TO GZ SDF
USE GZ1
APPE FROM GZ.TXT SDF
REPL ALL 其它津贴 ALL SPAC (8) FOR 其它津贴' 0.00'
?其它津贴.....
```

ZAP 命令不要去掉，否则，程序每运行一次，GZ1.DBF 的记录数，将得到一次追加。

三、直接判断、打印

程序 3：.....

```
IF 其它津贴=0.00
  ? SPAC (8)
ELSE
  ? STR (其它津贴, 8, 2)
ENDIF
  .....
```

四、将字段内容替换：

预先将可能出现值为 0.00 的字段（如"其它津贴"），对应地设置字符型字段（如"津贴其它"），用程序替换并打印字符型字段。

程序 4：.....

```
REPL ALL 津贴其它 WITH SPAC (8) FOR
其它津贴=0.00
REPL ALL 津贴其它 WITH STR
(其它津贴, 8, 2) FOR 其它津贴>0.00
```

? 津贴其它

.....

五、各种方法的特点

在程序 1 和程序 3 中，如果使用 SUBSTR（）或&命令，可使打印程序简洁，但由于 IF 命令在打印过程中判断，因而降低了打印速度。

在程序 2 和程序 4 中，可将打印的命令单独组成一个模块，打印速度可明显地提高，但打印库、增加的字段增大了占据磁盘的空间。也可将四种方法适当组合，灵活地应用。

CdBASEⅢ菜单的彩条驱动

菜单驱动是 CdBASEⅢ应用程序中广泛采用的软件驱动方法，但大多数都是以选择功能项的编码来实现的，虽然其实现及操作都较简便，但不足之处在于缺少变化。如果以彩条作为选择标志，并用光标移动键来移动彩条，将可丰富菜单的选择方式。本文以长城 0520C-H 为背景，介绍一种 CdBASEⅢ应用程序菜单的彩条驱动方法。这种方法将功能项涂上彩色来表明其被选择，使用光标移动键或字符编码来改变选择，然后通过回车键确认并执行。

一、菜单项目的涂色方法

为了实现彩条驱动，在选择菜单的过程中，要将所选的菜单项目涂上彩色，同时还要将前次选取的项目去色。

(1) 屏幕的显示属性

CdBASEⅢ提供了下述语句来设置屏幕的属性：

SET COLOR TO <标准>〔<增强>〕〔边缘〕

系统初始值为：标准显示是黑底白字，增强显示为白底黑字（反像显示），边缘是黑色。

用于指定标准和增强显示的颜色对为：前景／背景。

SET COLOR TO 6／1，7／4，6<或 SET COLOR TO GR／B，W／R，GR

将使屏幕的显示属性成为：标准显示用蓝底黄字，增强显示用红底白字，边缘用黄色。

(2) 菜单的显示

通常@行，列 SAY<字符串>语句和? ／?? <字符串>语句用于菜单显示。

?<字符串>；从下一行开始显示字符串。

?? <字符串>；从光标的当前位置开始显示字符串。

@行，列 SAY<字符串>；从指定的行和列开始显示字符串。

@行，列 SAY<字符串>不受 SET COLOR TO 语句的影响，显示黑底白字，而? ／?? <字符串>则受到 SET COLOR TO 语句的控制，显示带有彩色或闪烁的字串。这样用@行，列 SAY<字符串>语句可在屏的屏幕的任何位置显示彩色字串，而用@行，列 SAY"和?? <字符串>结合可在屏幕的任何位置显示彩色字串。因此，可用@行，列 SAY"语句和?? <字符串>语句为菜单中被选择的项目涂色，而用@行，列 SAY<字符串>语句为菜单中前次被选择的项去色。从而完成彩条在菜单上的移动。