会计电算化试用教程（三）

会计电算化实用范例

陈宏明主编

湖南省财政厅审定

湖南科学技术出版社

序　言

会计电算化又称计算机辅助财务管理，是会计工作现代化的一个重要标志，也是会计改革的方向。随着改革的不断深化，会计电算化已经在会计领域中广泛推行，它越来越为人们的认识所接受，并取得了良好的社会效益。因此学习会计电算化知识是会计人员知识更新的需要，也是会计工作适应时代发展的需要。可以说，今后随着计算机的普遍推广应用，不懂得和不熟悉会计电算化的会计人员将是不合格，不称职的会计人员。为此，我们特组织编写了《会计电算化试用教程》，以此来提高会计人员的素质，推动会计电算化事业的发展。

本套教程共四本。之一《微型计算机实用技术》讲述计算机的基本组成，DOS3.3 的使用方法，汉字系统 2.13 的操作方法，是会计电算化的入门教材。之二《关系型数据库实用语言与程序设计》讲述会计电算化的工作语言 FoxBASE+的基本原理和一般会计电算化程序的编制方法，是会计电算化的基本教材。之三《会计电算化实用范例》解剖会计电算化系统中最常用的凭证汇总、帐务管理系统、报表管理系统，分析会计电算化实用系统的编制方法，使会计人员能够学以致用，是提高会计人员编写会计电算化系统程序水平的深入教材。之四《上机操作指南》配合前三本教材，讲述如何上机实现会计电算化的全过程，指导电算会计岗位人员进行系统管理，维护及各种基本技能训练。本教程以广大会计人员为读者，深入浅出，通俗易读，内容连贯，自成体系，适合于不同层次的会计人员学习会计电算化知识，是学习会计电算化的系列教材。为了读者能够更好的理解教程内容，我们将教程之一和之二录制了教学录像带，并且将教程之二和之三的程序和数据库录入软盘配合发行。湖南省财政厅指定本教程为全省各种层次和形式的会计电算化岗位培训的统一教材。

全套教程由陈宏明负责总纂并审定。在编写过程中得到了湖南省财政厅、湖南广播电视大学、长沙水利电力师范学院、湖南省计算机专科学校等单位和领导的大力支持，在此一并表示诚挚的感谢。

《会计电算化试用教程》编写组

1992 年 3 月于长沙

前　言

会计电算化推广的一个困难的问题是会计电算化系统软件一般是商品化软件，例如："先锋"，"用友"。这些软件一般价格偏高，另外商品化软件为了保护自己的权益，往往是采用加密的方法。这样用户就无法打开程序，掌握会计电算化程序的编制方法，也无法修改软件以满足本单位的特殊需要。且由于其价格高很多单位买不起，自己又无力从事开发，这样就严重制约了会计电算化的推广和应用。

本书根据作者在开发多项会计电算化系统经验基础上，将其进一步优化完善，编写了会计电算化工作中最常用的系统。例如：凭证汇总、帐务管理系统、报表管理系统。这些范例体现了会计核算软件的总体设计思想，数据库设计方法，程序编制体系，是学习会计电算化知识和软件开发的参考教材。

本书的目的是使会计人员在学会运用 FoxBASE+编写会计电算化应用程序的基础上，进一步掌握编写实用会计电算化系统的方法，使之达到会计电算化程序员水平。

在本书的编写过程中，得到了湖南省财政厅，长沙水利电力师范学院院领导、公共课部和计算中心的大力支持，在此表示诚挚的感谢。

全书所举的实例均在 IBM　PC／XT、286、386 及其兼容机上，在 25 行 FoxBASE+2.10 下运行通过，可以适合于任何采用借贷记帐法的企业使用。为了读者更好地使用，已将本书中的所有程序及数据库录入软盘，配合发行.

由于作者水平有限，如有错误之处，欢迎读者批评指正。

作者

1992 年 2 月于长沙

内容提要

本书以会计电算化系统最常用的凭证汇总、帐务管理系统和报表管理作为范例，讲述会计电算化系统的编制方法。

本书中范例是根据作者在开发多项会计电算化系统经验基础上，经过优化完善编写的。该范例结构合理，分析详细，读者可以通过它学习会计电算化系统实用系统的编写方法。本书是在掌握了一般会计电算化编写程序方法的基础上，进一步提高编写水平的指导书。

本书所有范例均在 286，386 机上，FoxBASE+2.10 支持下调试通过。可以适合于任何采用借贷记帐法的企业使用。

为了使读者深入掌握全书的理论和实例，已将本书中的所有程序及数据库录入软盘，配合发行。

本书可作为大专院校财经、会计、会计电算化等专业及各种财会电算化和管理培训班的教材，亦可作为财会人员和会计电算人员的自学参考书。

本书配套软盘一套：150 元
需要者：请汇款到湖南科技出版社
开户行：长沙工商行中山路办事处
帐　号：02004657268
联系人：湖南科技出版社工业室　古华

目　录

第一章　各实用程序的共用模块

根据当前软件工程的观点，软件设计基本采用软件组合方式，即将一些各个应用系统均需使用的程序，做成通用模块，这样在程序中需要使用该功能时仅需要调用即可。本章介绍常用的通用菜单、通用口令校验、通用制表、通用查询模块，这些模块在以下各章的实用程序中均需使用。

§1.1　通用菜单模块

在本教程的之二中，已经介绍了一般菜单的编制方法。但当会计电算化系统中菜单下面又有子菜单时，菜单数目较多，我们对每一菜单均需要编制一段程序；同时为了菜单美观，还必须对于每一个菜单设计屏幕格式，为了解决这个问题，设计一个通用菜单程序。

首先设计一个菜单数据库，将系统的各级菜单的各项功能的汉字及其选择放入库中，建立 TYCD.DBF，结构如下：

```
. USE TYCD INDEX TYCD
. LIST STRUCTURE
```

Field	Field Name	Type	Width	Dec
1	XZ	Character	8	
2	HZ	Character	30	
3	MS	Character	80	
* * Total * *			119	

将各功能模块名、汉字菜单信息及对应菜单的提示信息放入库中，例如：

```
. LIST NEXT 8 TRIM(XZ),TRIM(HZ),TRIM(MS)
```

Record#	TRIM(XZ)	TRIM(HZ)	TRIM(MS)
1	GZA	修改固定项	适用于增加或调出人员，可修改工资的全部数据
2	GZB	修改活动项	每月均需修改的扣款栏等
3	GZC	打印工资发放表	该表供个人签名后，财务科保存
4	GZD	打印工资条	随工资发给个人的工资明细纸条
5	GZE	打印工资汇总表	部门计算工资的小计数，以便记帐时使用
6	GZF	打印钞票统计表	统计各种面值钞票所需的张数，以便发放工资
7	GZG	请查询工资信息	提供查询任何人员，按各种已知的条件查找工资状况
8	GZZ	退　　出	

为了加快查找，本库设有索引文件 TYCD.IDX，关键字为 STR（LEN（TRIM（XZ）），1）+XZ

接着对应此库编制程序，设计方法是，先在菜单数据库中，统计出本类菜单的个数。利用循环从数据库中取出汉字生成光带菜单，根据用户的合法选择，转相应模块执行或退出。

程序清单如下：

```
* * * * * * * * * * * * * * * * * * * * * * * * *
*                    TYCD                       *
* * * * * * * * * * * * * * * * * * * * * * * * *
*通用菜单模块
PARAMETER XZCD
SELECT 8
USE TYCD INDEX TYCD &&打开通用菜单库
L=STR(LEN(XZCD)+1,1)
XZA=L+XZCD+"A"
SEEK XZA &&找到本类菜单的第一条记录
COUNT REST TO N WHILE XZ=XZCD.AND.LEN(TRIM(XZ))=VAL(L) &&统计
本类菜单个数
SEEK XZA &&找到本类菜单的第一条记录
OP=1
N1=1
K=IIF(N<=9,2,1) && 如本类菜单选择数不大于9，为隔行显示
CLEAR
SET COLOR TO GR+/B+,G/R
@ 1,18 SAY "*******功 能 选 择*******"
SET MESSAGE TO 23
DO WHILE N1<=N .AND. (.NOT. EOF()) &&循环从库中取出汉字，生成菜单
  @ROW()+K,30 PROM IIF(N1=N,"Z",CHR(N1+64))+" ——— "+TRIM(HZ) ;
  MESSAGE SPACE((80-LEN(TRIM(MS)))/2)+TRIM(MS) && 菜单选择
  SKIP 1
  N1=N1+1
ENDDO
MENU TO OP
IF OP#N && 如不选择退出
  XZ1=L+XZCD+CHR(OP+64)
  SEEK XZ1 &&指针指向该选择
  XZ1=TRIM(XZ)
  USE
ENDIF
SET COLOR TO
RETURN
```

当需要使用菜单的地方，加上以下一段程序，调用菜单即可。

```
DO WHILE .T.
 XZ1=""
 DO TYCD WITH "GZ"&&执行菜单选择程序
 IF ""#XZ1
```

```
 DO &XZ1 && 执行该程序
ELSE
 EXIT
ENDIF
ENDDO
RETURN
```

如需调用2级菜单，例如"GZA"的功能，仅需将"DO TYCD WITH GZ"，改为"DO TYCD WITH GZA"即可。这样大大减少编制菜单程序的时间，同时当菜单汉字需要变动时，也不需要修改程序，仅需修改数据库中的记录就可以了。我们可以将各个应用系统的所有菜单均放在菜单库中，分别调用。

§1.2 通用口令校验模块

口令的一般设计方法在本教程之二中已经介绍，但是这种方法是将口令预先放在程序中，这样口令是死的，如果操作人员变动，需要更换口令，必须修改程序。如果应用程序中有多处需要设置口令的地方，则需要编制多个口令程序。另外口令没有经过变换保密性差。为此本节介绍加密口令的通用校验模块的设计方法。

通用口令校验模块，是将口令号和口令名存放在内存文件KL.MEM中，为了保密，将它更名为KL.DBF，这样口令是活的，可以增加、修改、删除，而不必修改程序。但需将模块分为两部分，即修改口令和校验口令。

一、修改口令

当操作人员变动时，口令则必须修改，有修改口令特权的人员，即可运行本模块修改口令。另外当模块第一次使用时，口令的初值，也由本模块置入。

由于口令是存放在内存文件中，虽然换名，但保密性不强，在实际应用中，我们存放的口令是经过变换的，并非原口令，这样即使看到口令，也不能知道真正的口令。大大增加了程序的安全保密性。

具体程序清单如下：

```
* * * * * * * * * * * * * * * * * * * * * * *
*                  KLXG                       *
* * * * * * * * * * * * * * * * * * * * * * *
*口令修改模块
SET TALK OFF
SET SAFETY OFF
CLEAR
IF FILE("KL.DBF")
  RESTORE FROM KL.DBF ADDI
ENDIF
K=0
JX=.Y.
```

```
DO WHILE JX && 继续修改
  KL="KL"+LTRIM(STR(K,3)) && 取口令号
  KLM="KLM"+LTRIM(STR(K,3)) && 取口令名号
  IF TYPE("&KLM")=UPPER("U") && 未定义该口令
    STORE UPPER("AAA") TO &KL && 定义口令初值
    STORE SPACE(30) TO &KLM && 定义口令名初值
  ELSE && 已定义口令
    STORE &KLM+SPAC(30-LEN(&KLM)) TO &KLM && 取口令名
  ENDI
  STORE CHR(ASC(SUBS(&KL,1,1))+50)+CHR(ASC(SUBS(&KL,2,1))+45);
  +CHR(ASC(SUBS(&KL,3,1))+40) TO &KL && 转换口令为原口令
  CLEAR
  @ 1,30 SAY "修  改  口  令 "+STR(K,2)
  @ 3,5 SAY "原口令名为:  "+&KLM
  @ 3,50 SAY "原口令为:  " +&KL
  @ 5,5 SAY "口令名改为: " GET &KLM
  @ 5,50 SAY "改口令为:  " GET &KL
  @ 8,32 SAY "继 续 修 改 ? " GET JX PICTURE "Y"
  READ
  STORE TRIM(&KLM) TO &KLM
  STORE CHR(ASC(SUBS(&KL,1,1))-50)+CHR(ASC(SUBS(&KL,2,1))-45);
  +CHR(ASC(SUBS(&KL,3,1))-40) TO &KL && 转换口令为原口令
  K=K+1
ENDDO
SAVE TO KL.DBF ALL LIKE KL* && 将修改后，经变换后口令保存
RETURN
```

二、口令校验

口令校验模块与口令程序一样，也是放在应用程序前面，校验口令，口令对允许进入应用，三次口令错，退出。不同的地方是口令从内存文件中取出，并经过变换转换为原口令后进行校验，并且由于应用程序中可能有多个口令，此时是校验哪个口令，所以采用带参调用，即调用时给出口令号参数，指出校验何口令。

具体程序清单如下:

```
* * * * * * * * * * * * * * * * * * * * * * *
*                  KLXY                     *
* * * * * * * * * * * * * * * * * * * * * * *
PARAMETER KK
*口令校验模块
SET ESCAPE OFF
SET TALK OFF
```

```
CLEAR
RESTORE FROM KL.DBF ADDITIVE
CLEAR
KL="KL"+LTRIM(STR(KK,3)) && 取该口令号
STORE CHR(ASC(SUBS(&KL,1,1))+50)+CHR(ASC(SUBS(&KL,2,1))+45);
+CHR(ASC(SUBS(&KL,3,1))+40) TO &KL && 转换口令为原口令
KLM="KLM"+LTRIM(STR(KK,3)) && 取该口令名
SR=.N.
N=1
DO WHILE N<=3 && 仅允许输入三次口令，校验
  @ 3,10 SAY "请输入为"+&KLM+"而设置的口令"
  SET CONSOLE OFF
  ACCEPT TO KLS && 输入口令
  SET CONSOLE ON
  IF KLS==&KL && 口令对
    SR=.Y. && 置口令对标志
    EXIT
  ENDIF
  @ 5,10 SAY "口令错，请重新输入"
  N=N+1
ENDDO
CLEAR
IF .NOT.SR && 口令错标志
  @ 3,26 SAY "口令错，您不能进入以下操作！"
  @ 4,0
  WAIT SPACE(33)+"按任意键，退出"
  CANCEL
ENDIF
SET ESCAPE ON
RETURN
```

在以下各节中，我们在需要口令的地方反复调用本模块，并且我们将各个应用系统的所有口令均放在 KL.DBF 中。

§1.3 通用制表模块

在本教程之二中，我们介绍了分页打印报表程序。在应用中读者可能已感到表框线比较难画，又容易错，修改也比较麻烦；另外由于表格线在程序中，当表变动时必须修改程序，这样报表是死的。为此本节介绍通用制表程序。

一、设计原理

首先设计一个 TYBT.DBF 数据库，本库存放预先生成的应用程序中所有报表的表头及表头格式栏，这样打印报表时，仅需从该库中取出该表的表头及格式栏信息，即可。报表变动时，仅需重新生成报表格线。

数据库结构如下：

```
. USE TYBT
. LIST STRUCTURE
Field   Field Name  Type        Width    Dec
    1   BZ          Character       6
    2   DY          Character     254
* * Total * *                     261
```

为了加快查找打印速度，本库有一索引文件，TYBT.IDX，索引关键字为：BZ

另外，为了生成以上 TYBT.DBF 数据库，需要一辅助数据库 TYBC.DBF，用来存放表头的所有栏目，以便生成和修改表头栏目时使用。

数据库结构如下：

```
. USE TYBC
. LIST STRUCTURE
Field   Field Name  Type        Width    Dec
    1   LD          Character       6
    2   LC          Numeric         3
    3   LM          Character     254
* * Total * *                     264
```

本数据库索引文件 TYBC.IDX，关键字为 LD.

本程序设计思想是：根据用户输入表头，每栏长度及汉字名自动生成各种格线，同时，将栏名居中，表头居中。

二、各模块功能介绍

1.主控模块

本模块显示子模块选择菜单。当用户选择某功能后，输入表名，如为修改表，查找是否存在；如为生成表，看是否已存在，已存在删除该表格式栏。然后执行相应功能模块。

程序清单如下：

```
* * * * * * * * * * * * * * * * * * * * * * *
*                    TYZB                     *
* * * * * * * * * * * * * * * * * * * * * * *
*通用制表程序主控模块
CLEAR ALL
SET SAFETY OFF
SET TALK OFF
SELECT 1
USE TYBT INDEX TYBT && 打开通用表头库
```

```
SELECT 2
USE TYBC INDEX TYBC && 打开通用表栏名库
DO WHILE .T.
  SET COLOR TO GR+/B+,G/R
  SELECT 1
  OP=1
  CLEAR
  @ 1,19 SAY "*******定义表栏头程序*******"
  @ 3,30 PROMPT "1 —— 生成表栏头"
  @ 6,30 PROMPT "2 —— 修改表栏目"
  @ 9,30 PROMPT "3 —— 修改表头栏"
  @ 12,30 PROMPT "4 —— 旧表生新表"
  @ 15,30 PROMPT "5 —— 打印表栏头"
  @ 18,30 PROMPT "0 —— 返回主选单"
  MENU TO OP
  SET COLOR TO
  IF OP=6
    EXIT
  ENDIF
  BM=SPAC(3)
  CLEAR
  @ 2,27 SAY "请输入表名代号: (<=3)" GET BM PICTURE "!!!"
  READ
  BM=TRIM(BM)
  SELECT 2
  SEEK BM
  IF OP#1 && 非生成表
    IF EOF()
      @ 4,29 SAY "本表未定义，请重新选择"
      @ 5,0 SAY " "
      WAIT SPACE(34)+"按任一键返回"
      LOOP
    ENDI
  ELSE && 为生成表
    N1=.N.
    IF .NOT.EOF()
      @ 6,20 SAY "本表已生成是否重新生成" GET N1 PICTURE "Y"
      READ
      IF N1 && 重新定义
        SELECT 2 && 选择通用表栏名库
```

```
        DELETE REST WHILE LD=BM && 删除该表
        PACK
        SELECT 1 && 选择通用表头库
        SEEK BM
        DELETE REST WHILE BZ=BM && 删除该表
        PACK
      ELSE
        LOOP
      ENDIF
    ENDIF
  ENDIF
  SELECT 1
  OP1="TYZB"+STR(OP,1)
  DO &OP1 && 执行相应模块
ENDDO
SET COLOR TO
CLOSE DATABASE
RETURN
```

2.生成表栏头模块

本模块根据用户选择何种点阵，置最大栏头宽度，然后根据栏数逐栏输入栏长及栏名到TYBC.DBF 中.同时，在屏幕提示表剩余长度。注意最后一栏标志为表名+"99"，根据TYBC 库内容，生成表头格式栏到 TYBT.DBF 中。

程序清单如下:

```
* * * * * * * * * * * * * * * * * * * * * * *
*                    TYZB1                    *
* * * * * * * * * * * * * * * * * * * * * * *
* 生成表栏头模块
SELECT 2 && 选择通用表栏名库
APPEND BLANK
REPLACE LD WITH BM && 替换表名
L1="24"
CLEAR
@ 1,28 SAY "按 何 种 点 阵 打 印 表 " GET L1
@ 2,10 SAY " (注: 24 点阵每行可打印 204 个字符，16 点阵 254 个字符) "
READ
LN=IIF(L1="16",254,204) && 根据点阵置最大打印字符数
REPLACE LC WITH LN
N=1
HZM=SPACE(50)
@ 3,7 SAY "请输入本表汉字名" GET HZM
```

```
@ 4,29 SAY "本 表 共 多 少 栏 ? " GET N PICTURE "99" RANGE 1,99
READ
N1=1
LN=LN-2 && 最大长度减 2
LN1=LN && 表余长度
T=.Y.
N2=N && 栏数
DO WHIL N1<=N
  IF T .OR. N1>N2 && 非重做或重做时大于已有栏数
    APPEND BLANK && 追加新记录
    REPLACE LD WITH BM+LTRIM(STR(N1+10,2)) && 表名+栏号
  ELSE && 重做
    SKIP 1 && 仅修改
  ENDIF
  CLEAR
  @ 1,2 SAY "表余长度 "+STR(LN1,3)+" ( 注: 必须大于 0 ) "
  @ 2,2 SAY "第 "+STR(N1,2)+" 栏"
  @ 3,2 SAY "本栏长度 ( 注: 必须为偶数 ) " GET LC
  READ
  X=["]+REPLICATE("X",LC)+["] && 相对表栏长度的格式函数"X"
  @ 4,2 SAY "本栏汉字名: " GET LM PICTURE &X
  READ
  LN1=LN1-LC-2 && 剩余长度
  IF LN1<0 && 表太长, 重新做
    CLEAR
    @ 4,26 SAY "表太长, 无法打印, 请重新制做"
    @ 5,0 SAY " "
    WAIT SPACE(32)+"按任一键, 重做! "
    T=.N. && 置重做标志
    SEEK BM && 找本表第一栏
    LN1=LN && 最大长度
    N2=N1 && 已生成栏数
    N1=1 && 从第一栏开始修改
    LOOP
  ENDIF
  N1=N1+1
ENDDO
REPLACE LD WITH BM+"99" && 置最后一栏标志
CLEAR
@ 4,21 SAY "正 在 生 成 表 栏 头 , 请 等 待 … … "
```

```
L1="└"  && 表头格式栏各初始码
L2="┌"
L3="|"
L4="├"
N1=1
SEEK BM
DO WHILE N1<=N && 生成表头格式栏
  SKIP 1
  LA=REPLICATE("-",LC/2) && 根据每栏长度生成
  LN1=LC-LEN(TRIM(LTRIM(LM)))
  LN2=SPACE(INT(LN1/2)) && 栏名居中所需左边空格数
  LB=LN2+TRIM(LTRIM(LM))+LN2 && 生成栏名
  LB=IIF(INT(LN1/2)=LN1/2,LB,LB+" ")
  L1=L1+LA+"┴" && 逐渐生成表头各格式栏
  L2=L2+LA+"┬"
  L3=L3+LB+"|"
  L4=L4+LA+"┼"
  N1=N1+1
ENDDO
LN1=LEN(L1)
L1=SUBSTR(L1,1,LN1-2)+"┘" && 生成完整表头各格式栏
L2=SUBSTR(L2,1,LN1-2)+"┐"
L4=SUBSTR(L4,1,LN1-2)+"┤"
LN2=LEN(TRIM(LTRIM(HZM))) && 表名长度
LN3=SPACE(INT((LN1-LN2*2)/4)) && 表名居中所需左边空格数
HZM1=LN3+TRIM(LTRIM(HZM)) && 完整表名
HZM2=LN3+REPLICATE("-",LN2/2)
SELECT 1 && 选择通用表头库
APPEND BLANK
REPLACE BZ WITH BM+UPPE("B1"),DY WITH HZM1 && 将表名放入通用表头库
APPEND BLANK
REPLACE BZ WITH BM+UPPE("B2"),DY WITH HZM2
N1=0
DO WHILE N1<4 && 将各个格式栏放入库中
  X=UPPER("L")+STR(N1+1,1)
  APPEND BLANK
  REPLACE BZ WITH BM+UPPER("C")+STR(N1,1),DY WITH &X
  N1=N1+1
ENDDO
REPLACE BZ WITH BM+UPPE("C99") && 置最后一栏标志
```

RETURN

3.修改表栏目模块

本模块对已生成的表进行修改，可以增加、删除、修改栏目，即对 TYBC.DBF 进行修改。修改后，重新生成表信息到 TYBT.DBF 中。

程序清单如下：

```
* * * * * * * * * * * * * * * * * * * * * * * * *
*                    TYZB2                        *
* * * * * * * * * * * * * * * * * * * * * * * * *
*修改表栏目模块
DO WHILE .T.
  SELECT 2 && 选择通用表栏名库
  LB=LEN(BM)
  X1="  "
  OP1=1
  SET COLOR TO GR+/B+,G/R
  CLEA
  @ 1,24 SAY "* * * * * * * 修 改 栏 目 * * * * * * *"
  @ 4,30 PROMPT "1 ———— 增加栏目"
  @ 7,30 PROMPT "2 ———— 修改栏目"
  @ 10,30 PROMPT "3 ———— 删除栏目"
  @ 13,30 PROMPT "0 ———— 退出修改"
  MENU TO OP1
  SET COLOR TO
  DO CASE
    CASE OP1=1 && 增加栏目
      CLEAR
      SELECT 1 && 选择通用表头库
      SEEK BM+UPPER("C0")
      L=LEN(TRIM(DY)) && 本表宽度
      SELECT 2 && 选择通用表栏名库
      SEEK BM && 找本表第一栏
      L2=SPACE(3)
      L3=SPACE(254)
      @ 2,2 SAY "表余长度 "+STR(LC-L,3)+"  (注: 必须大于 0) "
      @ 3,2 SAY "增加在第几栏之前? (末栏输 99)  " GET X1 PICTURE "99"
      @ 4,2 SAY "本栏长度 (注: 必须为偶数且小于表余长度) ";
      GET L2 PICTURE "999"
      READ
      L2=VAL(LTRIM(L2))
      X=["]+REPLICATE("X",L2)+["]
```

```
  @ 5,2 SAY "本栏汉字名:  " GET L3 PICTURE &X
  READ
  SEEK BM+"99" && 找本表最后一栏
  SKIP -1
  X2=BM+IIF(X1="99",STR(VAL(SUBS(LD,LB+1,2))+1,2),STR(VAL(X1)+10,2))
  DO WHILE LD=BM .AND. LD>=X2 && 将该栏之后，所有栏的栏号+1
    REPLACE LD WITH BM+LTRIM(STR(VAL(SUBSTR(LD,LB+1,2))+1,2))
    SKIP -1
  ENDDO
  APPEND BLANK && 追加一条
  REPLACE LD WITH X2,LC WITH L2,LM WITH L3 && 替换新栏名
CASE OP1=2 && 修改栏名
  CLEAR
  @ 2,2 SAY "修改第几栏? (末栏输 99)  " GET X1 PICTURE "99"
  READ
  X2=BM+IIF(X1="99",X1,STR(VAL(X1)+10,2))
  SEEK X2 && 找到该栏
  @ 3,2 SAY "本栏原有长度为:  "+STR(LC,3)+" 修改为:  " GET LC
  READ
  LM1=SUBSTR(LM,1,LC)
  @ 4,2 SAY "本栏名改为:  " GET LM1
  READ
  REPLACE LM WITH LM1
CASE OP1=3 && 删除栏名
  CLEAR
  @ 2,2 SAY "删除第几栏? (末栏输 99) " GET X1 PICTURE "99"
  READ
  X2=BM+IIF(X1="99",X1,STR(VAL(X1)+10,2))
  SEEK X2
  @ 3,2 SAY "是 本 栏 ?  "+TRIM(LM)
  X3=.N.
  @ 6,2 SAY "删 除 ?  " GET X3 PICTURE "Y"
  READ
  IF X3
    DELETE && 删除本栏
    IF X1#"99" && 非末栏，将该栏下面栏号-1
      REPL REST LD WITH BM+LTRI(STR(VAL(SUBS(LD,LB+1,2))-1,2));
      WHILE  LD=BM.AND.LD#BM+"99"
    ELSE && 删除末栏
      SKIP -1
```