张玉红 著　黄梯云 审

FoxPro 2.x For Windows
管理信息系统程序设计技术

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL: http:/www.phei.co.cn

FoxPro 2.X For Windows

管理信息系统程序设计技术

张玉红　著
黄梯云　审

電子工業出版社

内 容 提 要

本书对FoxPro 2.X For Windows管理信息系统程序设计技术作了系统阐述。全书共分六章，内容着重介绍管理信息系统程序设计的过程组织和程序实现方面的实用技术。以丰富的程序实例详细介绍程序设计思路及方法是本书的一大特色。本书图文并茂、内容实用，是初学者的良师，编程人员的益友。本书可作为各行业管理信息系统计算机应用方面的参考手册，亦可作为大专院校学生、科研人员学习FoxPro 2.X For Windows管理信息系统程序设计技术的教材和参考书。

FoxPro 2.X For Windows
管理信息系统程序设计技术

张玉红 著
黄梯云 审
责任编辑：吕 迈

*

电子工业出版社出版
北京市海淀区万寿路 173 信箱(100036)
电子工业出版社发行 各地新华书店经销
北京牛山世兴印刷厂印刷

*

开本：787×1092 毫米 1/16 印张：18.75 字数：468 千字
1996 年 11 月第一版 1996 年 11 月北京第一次印刷
印数：5000 册 定价：22.00 元
ISBN 7-5053-3717-3/TP・1571

前　言

FoxPro 2.X是继FoxBase X之后的新一代数据库支撑平台，其特点是操作速度非常快，提供更加丰富的命令和标准函数，具有优异的屏幕窗口操作特性，并提供DOS和Windows操作系统下的两种版本，深受程序设计人员的喜爱。

管理信息系统（MIS）是随着系统科学、计算机技术和现代通信技术的发展以及现代化管理的客观需要，逐渐形成的一门边缘科学。基于计算机的管理信息系统有助于提高企业管理的质量和效率，目前它已经成为现代化管理的重要组成部分。

计算机在各行各业、各个领域中的应用越来越普及，导致MIS程序的需求量不断增大。MIS程序的开发必须考虑国情、行业和企业特点，大量、实用的MIS程序中的相当一部分宜由各行各业的计算机应用技术人员自行开发。

中小型的MIS程序很适合用FoxPro 2.X来开发。在开发一项具体MIS程序的过程中，恰当、高效的程序设计（即编写程序）是十分重要的。广大MIS编程人员及FoxPro 2.X程序设计爱好者十分需要编程技术交流和学习方面的书籍，本书正是为满足这样的需要写就的。

本书总结了作者数年来编写MIS程序的经验，提供了大量实例，并给出它们的设计思路，希望以此达到与广大的MIS编程人员及FoxPro 2.X爱好者进行技术交流、互相学习的目的。

全书由黄梯云教授审阅。作者师承黄先生，本书的写作深得黄先生的鼓励和指导，在此表示衷心的感谢。参加本书写作过程的还有肖湘、刘晶珠、安石、王作江、尤宏、陈福明、贾岩等同志，在此表示感谢。杨钰红、马淑华等同志参与了本书的打字、排版工作，在此一并表示衷心的感谢。

由于作者水平有限，本书可能存在这样或那样的错误或不足，诚挚希望读者提出宝贵的意见和建议。

张玉红
于哈尔滨工业大学管理学院
1996年5月

目　录

第 1 章 程序组织及处理技术

管理信息系统(MIS)主要是处理数据的收集、存储转换、加工及信息提取的过程。计算机管理信息系统的程序组织，要鲜明地体现出这些过程的特点。因此，作为MIS程序的软件设计者，首先要在明确任务、详细调查、了解其业务特点的基础上，设计、提出MIS方案，然后编写程序来实现这个方案。

这样说来，MIS软件人员实际是分为两大类的，一类是设计、提出MIS方案的，通常称他们为“系统分析员”；另一类是编写程序的设计人员，通常称他们为“编程人员”。系统分析员一般是MIS应用方面的专家、学者，具有丰富的理论和实践经验，知识广博，因此，他们提出的方案不仅是编程人员的程序设计依据，而且代表了计算机MIS替代手工MIS的现代化管理方向，大、中型计算机MIS系统的实现必须具有优秀的系统分析员队伍。编程人员是最终实现计算机MIS 的“建筑工匠”，MIS系统的质量同他们的技术、劳动息息相关，也就是说，系统设计方案要经他们之手实现。如果把系统设计方案比做“筋骨”的话，则程序是MIS系统的“血肉”，从而使计算机MIS具有生命和活力。

本书主要是写给编程人员或有志于成为编程人员的人的。

1.1 数据库支撑平台FoxPro 2.X

计算机MIS是基于数据库技术的应用程序系统。就目前来讲，一般的MIS都是在数据库系统软件（称"数据库支撑平台"）的基础上开发的，它们本身提供了丰富的基础过程（如：排序、求和过程）供编程人员采用。本书介绍的内容是针对FoxPro 2.X数据库关系系统。FoxPro 2.X是微机上快速的数据库系统。FoxPro 2.X有MS-DOS版和Windows版两种，前者可以说是文本模式(Text Mode)的典范，后者更能充分发挥图形界面的特色。

关于FoxPro 2.X所提供的命令和函数的详细情况，请参考其它有关书籍。

FoxPro 2.X所提供的命令、函数有数百条，要想详细地全部记住它们是不现实的。因此，编程人员手头备有一本FoxPro 2.X命令和函数手册是必须的。目前，计算机商店一般都有这样的手册出售。

FoxPro 2.X与dBase X、FoxBase X兼容，因此，过去对dBase X、FoxBase X有所了解的话，学习和掌握FoxPro 2.X会更容易一些。FoxPro 2.X同dBase X 、FoxBase X一样，其命令和函数名字大多是用英文意义明显的单词命名的，书写时，写其前四个字母或是全部都可以。

1.2 管理信息系统的程序结构

MIS程序系统一般包含多种程序过程，编程人员要考虑这些过程的组织问题，即：过程的

任务、数据信息传递及怎样让用户（即操作人员）适时调用有关的过程。实际编程时，一般都把有某种关联或操作意义(比如：输出报表)相近的过程编成一组，因此，MIS程序系统是由程序组组成的。这种程序组通常又称“模块”。MIS程序系统的程序结构一般如图1-1所示。

这种结构通常具有“菜单引导，逐级退返”的特点。这就是说，程序从作用上来看分两类，一类是专门处理同用户对话以确定后续程序操作的对象(称流向)，这种程序实质是功能调度程序，英文称“Menu”。由于“Menu”有餐厅菜谱、菜单的含义，所以国内习惯上把功能调度程序称为“菜单”。

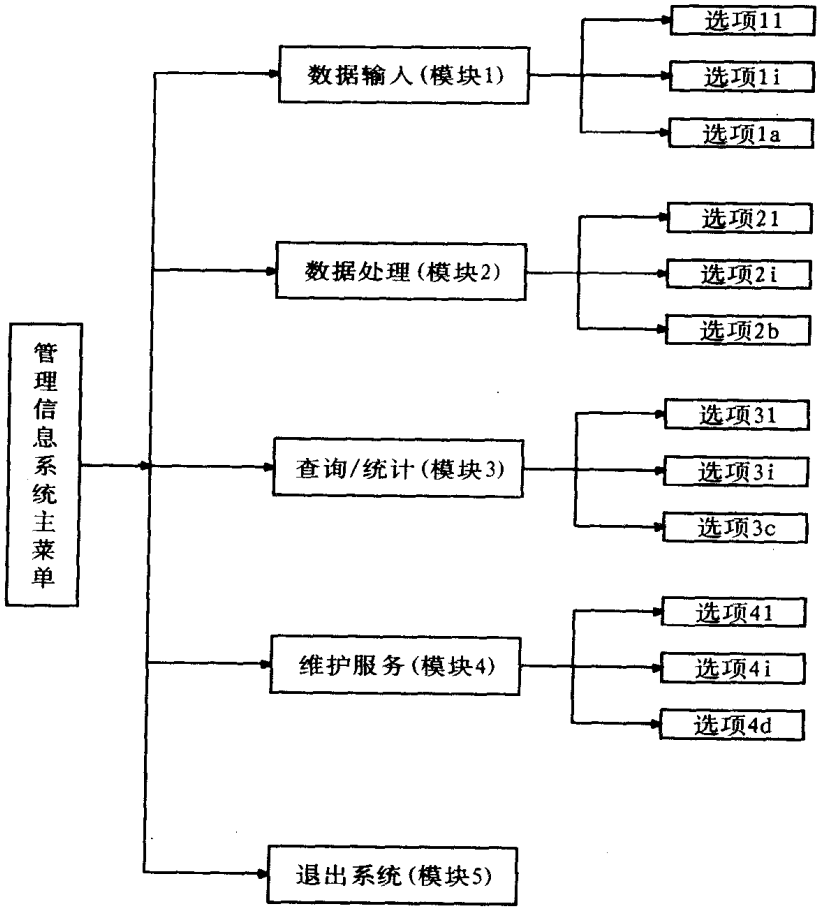

图1-1 管理信息系统程序的结构

最高一级的“菜单”即主调度程序称“主菜单”。另一类程序则是完成某种功能或任务的操作程序，在这里我们将它们称之为“实务”程序，以便与“菜单”程序相区别。

数据输入模块通常包含下面的实务功能：

1. 数据输入：新的原始数据记录追加操作。
2. 数据修改：对已输入过的数据记录进行更改。
3. 数据删除：剔除已输入过的数据记录或废记录。
4. 数据的排序：按某种数据特征（如：购货日期的先后）对数据记录排列顺序。

上述1、2、3、4程序集成(即不明确列示，而允许现场适时调度)时可笼统地称之为“数据编辑”。

查询/统计输出模块则主要完成如下的任务:

1. 数据记录的格式化输出

将数据按照一定的表格形式或屏幕设计格式输出到打印机、屏幕上，有时也可以输出到一个磁盘文件中。

2. 信息输出

将数据按照一定的计算要求，加工整理出新的数据记录并输出它们。

3. 条件查询检索

通过人机对话获取一定的分类、检索条件，并根据这个条件完成数据或信息的输出。

4. 状态数据信息输出

一般是对系统特征数据的输出。典型的系统特征数据包括:

(1) 原始数据库输入对应的有效操作日期。比如: 工资系统中的工资发放月份。

(2) 上一次主处理动作的时间。比如: 帐务系统中上一次汇总结算的时刻。

(3) 主要数据库的记录数目。比如: 文档管理系统中的文档借阅归还记录数目。提供这样的数据信息，目的是使用户能适时调用有关记录剪裁或扩增程序进行有效的结转或汇集处理。

(4) 其它信息。如用户、开发者的名称、版权等必要的信息。

数据处理模块一般是针对原始数据的数据加工、转换等方面的计算、汇总、分类处理，其包括的操作与具体的MIS(如: 帐务系统、工资发放系统、储蓄系统等)系统的特点有关。

维护服务模块通常含有如下几项实务操作:

1. 系统初始化处理。

2. 文件备份处理。

“文件备份”是指将早期的数据记录或当前记录转移到其它存储介质（如磁盘、磁带、光盘等）上去，以便系统因意外损坏数据时，或是需要它们时能再现这些记录。因此“备份”处理包括两种相反的操作，一类是从系统到外存储介质方向的操作，一般称“拷贝”出去，另一类是从外存储介质向系统回传数据的操作，一般称“数据回传”或是“数据恢复”。

3. 不经常运行的程序操作，如“帐务系统中的科目增加操作”，储蓄系统中的利率变更程序、运行口令设置程序等等。

退出系统模块往往是一个程序语句，使MIS程序结束而退返到进入MIS程序的地方(比如DOS状态或Windows状态)。如果对数据有较高的安全、保密性要求，退出系统通常要完成数据库校验标志生成、数据库加密等方面的“自我封闭性”实务操作。上面所述的程序结构是就一般的MIS程序系统而论的，并不是严格的规定。

目前，MIS程序的应用越来越普及，程序的风格(菜单引导方式、人机对话形式、程序过程的执行机制等等的通称)也在不断变化。过去，编程人员习惯于尽可能多地在菜单上罗列较多的选项，其中有相当一部分是为了显示工作量而“硬”添上的。现在，程序逐步向“集成”方向努力，也就是说，尽量使菜单的选项减少，以提高程序自动化运作或加大用户参与程度。事实上，大多数用户已经认识到，“集成”化的程序功能更灵活、强大，更能体现编程人员的水平。

1.3 程序语句处理技术

程序是由语句组成的，因此编程人员必须要学会如何用FoxPro 2.X提供的函数、命令组成语句。语句是计算机执行一项任务的表达单体，一般而言，语句的编写质量、水平通常与编程人员的数学知识运用(公式化、逻辑化组织)有很大的关系。本节例举常见的几种应用场合或情景提供示范，以达到抛砖引玉的作用。从本节开始，读者必须要了解数据库创建、使用、字段赋值、运算等基本概念，如果读者对这些内容尚不了解的话，请先查阅有关书籍进行入门知识的“武装”。

1.3.1数据库的创建

数据库的创建有程序化、手工化两种．前者是在对原始数据库进行处理时进行的中间文件制作过程。手工化是在FoxPro 2.X的Command窗口用Crea命令、Copy命令等初始创建数据库的操作过程．手工化创建数据库属于编程人员的入门知识之列，本书不去叙述它们。

【例1.3.1】 某原始数据库的编码字段CODE需要处理，但其中编码的实际文字意义并不在本库中。现在要创建一个中间文件，使之含有CODE字段，另外再具有一个NAME字段，以便后续程序使用它，从编码定义库中查出CODE文字意义，并填写到NAMR字段。假设CODE是C型八个字符长(以后简记为C8)，而NAME为C24，要创建的中间文件名称为 $ CODE. DBF。

实现例1.3.1的语句有两种写法:

1.利用CREA TABL直接创建

```
CREA TABL  $ CODE;
(CODE C(8), NAME C(24))
```

在本句中首行 $ CODE之后有一个分号（ “; ” ）这是FoxPro 2.X编程书写约定，表示下面一行与本行是紧接着的，实际上可以把它们写成:

```
CREA TABL  $ CODE (CODE C(8), NAME C(24))
```

这种“续行”特点适合于所有的语句编写，但要注意，所有的“续行”与首行被视作一句，每一句最多能容纳2048个字符(即2k)。

注意本句的特点，创建文件的结构要描述在一对()之内，在()之内是字段描述，字段描述两两之间用逗号（ “, ” ）分开(最后一个字段描述后不得加 “, ”)。每一个字段描述分为字段名、类型及长度描述，字段名与类型之间要有一个以上的空格，类型的取值是FoxPro 2.X数据库字段类型的首字母(即可取C、D、N、L等等)，长度是字段长度和小数点位数的描述，并用小括号括住及用 “, ” 区分。无小数位部分时可省略小数位数的描述。比如: 字段AB为N型，其含有2位小数，长度为12，则字段描述为AB N(12, 2)。

2.通过文件结构文件来创建

假定原始数据库的名称为SBMX. DBF，可以先写两句获取文件结构的语句:

```
USE SBMX
COPY STRU EXTE FIEL CODE TO $CODE
```

这时，CODE不是最终的文件结果，它仅含有SBMX.DBF中CODE字段的描述。注意用EXTE子句来指定数据库结构的获取，用Fiel子句指明要索取的字段名称(字段名称，两两之间用“，”分隔)。然后，在$CODE.DBF中加入NAME字段的定义，并创建出CODE.DBF。

```
USE $CODE
APPE BLAN
REPL FIEL_NAME WITH [NAME], FIELD_TYPE WITH [C],;
FIELD_LEN WITH 24
CREA $CODE FROM $CODE
```

似乎第二种创建方法比第一种创建文件的方法麻烦，但是有些时候它比第一种方法优越，比如：要创建一个用户任意指定的数据库。

1.3.2数值型、字符型变量的互相转化

【例1.3.2】 某数据库有一个字符型的数字编码字段，长度八位，请将内存变量NBM(整数型)所具有的数值转换成该字段的“字符化”值(设该字段名称为CBM)。要求CBM的值必须满八位，即：如NBM为1时，CBM为“00000001”，如NBM为1023时，CBM为“00001023”。

实现例1.3.2的语句如下：

```
REPL CBM WITH RIGH(REPL("0",7)+LTRI(STR(NBM,8,0)),8)
```

写出该语句的思路是这样的：把一串0(七个字符长)添加在NBM的字符化值前面，再从这个长串中自右向左取八位，显然NBM的字符化值不能含有空格，因此要用LTRI()函数剔除其前导空格，这是因STR()函数将NBM字符化时用空格补位，且是右对齐方式。

【例1.3.3】 将上例数据库CBM字段的字值还原为数值。

实现例1.3.3的参考语句如下：

```
NPB=INT(VAL(CBM)+0.00001)
```

注意该句将CBM的数值化值加上了一个小数0.00001后再取整是必要的，因为INT()函数仅是取某数的整数部分，并不进行四舍五入。为防止计算机可能出现的近似表示(如将“15”表示为14.999999)，加上一个适当的小数来调整是明智的。

1.3.3检索、分类条件的描述

【例1.3.4】 用Brow语句将记录号为奇数、偶数的记录分别列示出来。

实现例1.3.4的参考语句如下:

对列示奇数记录号的记录:

```
Brow For MOD (RecN(),2)=1
```

对列示偶数记录号的记录:

```
Brow For MOD(RecN(),2)=0
```

该例对屏幕输出时剔除报表的行间栏线显示是有用的。一般来说报表总是一行数据后，紧接着跟一条栏线。屏幕空间有限，为了尽可能在一屏更多地显示数据，可以暂时“屏蔽”栏线的显示。

【例1.3.5】某帐务系统的凭证库中有摘要字段，字段名称为ZY.请将摘要中有“李云龙”字样的凭证记录全部列出。

实现例1.3.5的参考语句如下:

对在屏幕上列示:

```
Brow For “李云龙” $ ZY
```

或是

```
DISP ALL For “李云龙” $ ZY
```

及

```
LIST ALL For “李云龙” $ ZY
```

对复制成一个中间文件来说是(假设中间文件名为 $ PZ.DBF):

```
copy to $ PZ For “李云龙” $ ZY
```

因为“李云龙”可能出现在ZY中的任何位置，因此用 $ 去匹配很合适。对此例来讲，凡是出现“李云龙” $ ZY的地方用如下写法来替代同样是有效的: AT(“李云龙”, ZY)>0。函数AT(“李云龙”, ZY)返回“李云龙”在ZY 中第一次出现时的位置。本例说明，FoxPro 2.X中提供的函数或命令有些是能互换使用的，因此书写语句时的写法与编程人员的习惯有关。

1.3.4代码字段的赋值

【例1.3.6】某人事数据库有一个记录性别的字段XB，为数值型，一位长度。如果与记录对应的人员性别为男性时，XB取1，为女性时取2，不能确定时取3，假定人员性别由内存变量PXB标志，其取值有“男”、“女”或空串三种可能，请根据PXB为XB赋值。

实现例1.3.6的语句有四种写法:

1.利用IF…ELSE…ENDI语句

```
IF PXB=[男]
   RepL XB with 1
ELSE
```

```
    IF PXB=[女]
       RepL XB with 2
    ELSE
       RepL XB with 3
    Endi
 EndI
```

2. 利用CASE语句

```
DO CASE
   CASE PXB=[男]
        REPL XB with 1
   CASE PXB=[女]
        REPL XB with 2
   OTHE
        REPL XB with 3
ENDC
```

3. 利用IIF()函数

```
RepL XB With IIF(PXB=[男], 1, IIF(PXB=[女],2,3))
```

4. 利用AT()函数

```
REPL XB WITH 1+(AT(PXB,[男女])-1)/2
```

注意，对字符常量的书写可用双引号、单引号、方括号之一来括住，因此“男”、‘男’或[男]都是字符常量“男”的正常书写方法。FoxPro 2.X的函数可采用自嵌套式写法(即:函数参数中可包括函数本身)。本例中1、2两种写法是常见的写法，但速度响应及程序输入量明显不如3、4两种写法。在3、4两种写法中显然前者是替代IF…ELSE…ENDI的写法，而AT()可用于替代Do CASE…ENDC的写法，读者要认真领会写法4，它能明显地减少源程序长度。为了帮助读者领会，请看下面的例子及分析。

【例1.3.7】 某工件任务单关于工件的材质用CZ字段记录，CZ类型:数字型2位长度，内容为材质代号。可能的材质种类及代号规定见表1-1。试根据内存变量PCZ(材质名称，6个字符长的字串)为CZ字段赋值。

实现例1.3.7的语句显然可以用DO CASE…ENDC语句。下面给出另一种参考写法:

```
REOL CZ WITH 1+(AT(PCZ,;[铁  铜  铝  尼龙  木材  塑料  不锈钢  其它 ])-1)/6
```

表1-1 材质种类表

材质代号（2位）	材质名称（6个字符长）
1	铁
2	铜
3	铝
4	尼龙
5	木材
6	塑料
7	不锈钢
8	其它

显然本例与例1.3.6的第4种写法出于同一种思想，是程序语句书写上很好的例子，它根据什么写出来，这就要知道AT()的功能是什么，前面已讲过，AT()用于返回PCZ在[铁　　…其它]字串中第一次出现的位置，这个位置是[铁　　……其它]自左向右的字符位置(注意1个汉字占两个字符位置)。假定这个位置是M，显然有下列公式:

M=1+(n-1)*6　　(n是材质代号)

从而，n=1+(M-1)/6

所以，一个好的语句写法通常受益于正确、精炼的数学公式的推导。

1.3.5 数值字段的赋值

【例1.3.8】 某乳品厂“奶源管理”的MILKS.DBF记录了奶户送鲜奶的数据，其中D_CLASS字段记录了某奶户某次送鲜奶的等级，N2型，而SL字段记录了送奶的数量，N8.2型，还有一个BCNZ字段(N10.2型)是应当付给本笔记录的费用(奶支)。费用按“等级价格×数量”核算。等级价格如表1-2所示。

表1-2 鲜奶等级价格表

鲜奶等级	价格（元）
1	0.657
2	0.582
3	0.490
4	0.415
5	0.372

试写出BCNZ字段赋值的语句(对MILKS.DBF的所有记录）。

实现例1.3.8的参考语句写法如下:

```
REPL ALL BCNZ WITH;
```

```
SL*VAL(SUBS([0.657 0.582 0.490 0.415 0.372], 1+(D_CLASS-1)*6, 5))
```

写出本句的思路是：根据等级值(D_CLASS字段的值）计算出对应的单价在价格描述字串的位置，并从此位置开始向右取5位而获得其单价字串，再用VAL()函数将其转换为数值，该数值与数量(SL) 的乘积即是应付给本笔记录的费用(BCNZ)。

SUBS()函数的用法:

```
SUBS(CHAR, N1, N2)
```

是返回CHAR所描述的字串中自N1位置开始(左起算)向右的N2个字符形成的子字串。

实际上，就本例而言，CHAR可以写成[0.657 0.582 0.49 0.415 0.372]，只是易读性不好，不便于程序维护(修改)。因此在编写程序时，使每一单价字串后尾随一个空格以解决这个问题。这样看来，每一个单价字串的长度就变为六位，有效部分是五位。等级为n的单价在CHAR中的起始位置显然符合如下公式:

$$1+(n-1)*6$$

常见的例1.3.8实现方法是写成类似如下的程序段:

```
USE MILKS DBF
GO TOP
DO WHIL NOT EOF()
   DO CASE
      CASE D_CLASS=1
           MDL=0.657
      CASE D_CLASS=2
           MDL=0.582
      CASE D_CLASS=3
           MDL=0.490
      CASE D_CLASS=4
           MDL=0.415
      OTHE
           MDL=0.372
   ENDC
   RRPL BCNZ WITH SL*MDL
   SKIP
ENDD
```

读者不难判断该写法与本例提供的语句写法孰优孰劣！

【例1.3.9】 成绩单数据库CJD.DBF结构如表1-3所示。该库记录的课程分为主课与辅修课两大类，其中“高等数学”至“外语”为主修课课程；自“体育”开始至“品德修养”为辅修课课程。综合成绩计算公式如下：

综合成绩=主修课平均成绩×0.85+辅修课程平均成绩×0.15

根据综合成绩可以评定综合成绩等级：

综合成绩等级=
- 优　综合成绩≥95.00
- 良　综合成绩介于85.00与94.99之间
- 中　综合成绩介于75.00与84.99之间
- 及　综合成绩介于60.00与74.99之间
- 差　综合成绩＜60.00

试写出满足如下要求的语句：

(1)对CJD.DBF全部记录的ZHCJ字段计算赋值。

(2)对CJD.DBF全部记录的ZHDJ字段赋值。

(3)计算班号为“92080301”的班级平均成绩到N0301内存变量。

(4)计算全部学生的不及格率（即综合成绩低于60.00分的人数占全部学生人数的百分比）到NBL(%)。

实现例1.3.9的参考语句如下：

1.对CJD.DBF各记录的综合成绩计算赋值。

```
REPL ALL ZHCJ WITH 0.85*(GDSX+DXWL+WJHX+YJHXC+WY)/5;
        +0.15*(TY+WXJS+JGSX+JT+PDXY)/5
```

2.对CJD.DBF各记录进行综合成绩等级赋值。

```
REPL ALL ZHDJ WITH;
         IIF(ZHCJ>=95.00,[优],;
         IIF(BETW(ZHCJ,85.00,94.99),[良],;
         IIF(BETW(ZHCJ,75.00,84.99).[中],;
         IIF(BETW(ZHCJ,60.00,74.99),[及],[差])))
```

实现本句的常见方法是类似下面的程序段：

```
REPL ALL ZHDJ WITH [优] FOR ZHCJ>=95.00
REPL ALL ZHDJ WITH [良] FOR ZHCJ>=85.00 AND ZHCJ<=94.99
REPL ALL ZHDJ WITH [中] FOR ZHCJ>=75.00 AND ZHCJ<=84.99
```

```
REPL ALL ZHDJ WITH [及] FOR ZHCJ>=60.00 AND ZHCJ<=74.99
REPL ALL ZHDJ WITH [差] FOR ZHCJ<60.00
```

本程序段与上面的语句相比，执行速度（即响应速度）要慢一些。

表1-3 成绩单数据库CJD.DBF结构

字段名	类 型	意 义
BH	C8	班号
XM	C8	姓名
XH	C8	学号
GDSX	N6，2	高等数学成绩
DXWL	N6，2	大学物理成绩
WJHX	N6，2	无机化学成绩
YJHX	N6，2	有机化学成绩
WY	N6，2	外语成绩
TY	N6，2	体育成绩
WXJS	N6，2	文献检索成绩
JGSX	N6，2	金工实习成绩
JT	N6，2	军体成绩
PDXY	N6，2	品德修养成绩
XHCJ	N6，2	综合成绩
XHDJ	C2	综合成绩等级（优、良、中、及、差）

3. 计算班号为“92080301”的班级之平均成绩到N0301内存变量。

```
AVER ALL ZHCJ TO N0301 FOR BH=[92080301]
```

注意: 写成下面的程序段影响执行速度:

```
SUM ALL ZHCJ TO N0301 FOR BH=[92080301] &&累加成绩
COUN ALL TO NUM FOR BH=[92080301]       &&统计人数
N0301=N0301/NUM                         &&计算平均成绩
```

4. 计算全部学生的不及格率到NBL内存变量，用百分数表示。

```
SUM ALL 100.00/RECC() TO NBL FOR ZHDJ=[差]
```

或是:

```
SUM ALL 100.00/Recc() TO NBL FOR ZHDJ<60.00
```

对本语句如写成类似下面的程序段来实现也是正确的:

```
COUN ALL TO NBL FOR ZHDJ=[差]
NBL=100.00×NBL/RECC()
```

或是

```
COUN All TO NBL FOR ZHCJ<60.00
NBL=100.00×NBL/RECC()
```

1.3.6其它计算

【例1.3.10】 根据系统日期（机内时钟日期）为CPB 内存变量赋以一个“××××年××月××日”格式的字串值。

实现本例的参考语句如下:

```
CPB=STR(YEAR(DATE()),4,0)+[年]+STR(MONT(DATE()),2,0)+[月];
   +STR(DAY(DATE()),2,0)+[日]
```

在常见的一些写法中，相当一部分程序是通过SUBS()函数与DTOC()函数联合使用来实现的，比如:

```
SET DATE ANSI
CPRQ=DTOC(DATE())
CPB=SUBS(CPRQ,1,2)+[年]+SUBS(CPRQ,4,2)+[月];+SUBS(CPRQ,7,2)+[日]
```

这种写法遇跨纪元年份（如2032年）时，可能发生混淆两个不同纪元的年份(比如1932年与2032年分不清)，希望能引起读者的注意。

【例1.3.11】 根据一个日期型的内存变量SRQ的取值，计算出月末的日期号数到内存变量NLAST。

实现本例的参考写法如下:

```
NLAST=IIF(INLI(MONT(SRQ),1,3,5,7,8,10,12),31,;
      IIF(INLI(MONT(SRQ),4,6,9,11),30,;
      IIF(MOD(YEAR(SRQ),4)>0,28,29)))
```

写出本句的思路: 俗语曰: “一三五七八十腊，三十一天都是大。四六九冬三十天，唯有二月二十八。”这里的“腊”指的是十二月，“冬”指的是十一月。但是“二月二十八”一句