Visual FoxPro 实用教程

---（NCRE 之VFP全攻略）

宣城市信息工程学校 裴鹏飞工作坊

第5章

Visual FoxPro程序设计

国家二级考试考点

1. 命令文件的建立与运行:

(1) 程序文件的建立。

(2) 简单的交互式输入、输出命令。

(3) 应用程序的调试与执行。

2. 结构化程序设计:

(1) 顺序结构程序设计。

(2) 选择结构程序设计。

(3) 循环结构程序设计。

3. 过程与过程调用:

(1) 子程序设计与调用。

(2) 过程与过程文件。

(3) 局部变量和全局变量,过程调用中的参数传递。

4. 用户定义对话框(MESSAGEBOX)的使用。

导学

一、学习目标

1. 了解过程化程序设计的基本概念及设计方法。

2. 掌握过程化程序设计的语言基础及程序设计的三种基本控制结构。

3. 熟悉模块化程序设计的基本方法，掌握子程序及过程调用的设计与运行方法。

二、重点、难点

1、三种分支结构:

IF-ENDIF

IF-ELSE-ENDIF

DO CASE-ENDCASE

2、三种循环结构:

DO WHILE-ENDDO

FOR-ENDF

SCAN-ENDSCAN

3、多模块程序设计及调用:

子程序、过程、变量作用域及带参数传递的过程调用。

5.1 Visual FoxPro程序与程序设计

5.1.1 程序文件与程序设计

程序是能够完成一定任务的命令的有序集合。

建立程序有如下好处：

（1）可以利用编辑器，方便地输入、修改和保存程序；

（2）可以用多种方式、多次运行程序；

（3）可以在一个程序中调用另一个程序。

Visual FoxPro程序文件的扩展名为.PRG。

5.1 Visual FoxPro程序与程序设计

5.1.2 程序文件的建立与运行

1. 程序的建立

（1）菜单方式

单击【文件】|【新建】菜单→“程序”命令→程序输入→编辑→保存。

（2）命令方式

【格式1】MODIFY COMMAND [路径]<程序名>[.PRG]

【格式2】MODIFY FILE [路径] [< 文件名.PRG>]

5.1 Visual FoxPro程序与程序设计

5.1.2 程序文件的建立与运行

2. 程序的运行

（1）菜单方式

单击【程序】|【运行】

当一个程序处于打开状态时，可以单击工具栏上按钮 

（2）命令方式

【格式】DO [路径]<程序名>[.PRG]

5.1 Visual FoxPro程序与程序设计

5.1.2 程序文件的建立与运行

3.结束程序运行可用下面命令：

① CANCAL：终止程序运行，清除所有的私有变量(私有变量在 5.3.3节介绍)，返回命令窗口；

② RETURN：结束当前程序的执行，返回到调用它的上级程序，若无上级程序则返回到命令窗口；

③ QUIT：退出Visual FoxPro系统，返回到操作系统。

5.1 Visual FoxPro程序与程序设计

5.1.3 三种交互式输入输出语句

（1）ACCEPT 字符串输入命令

【格式】ACCEPT [<提示信息>] TO <内存变量>

【功能】暂停运行，等待用户输入信息。

【说明】只接收字符串且不需要加定界符，否则系统会把定界符作为字符串本身的一部分。如果不输入内容而直接按回车键，系统会把空串赋给指定的内存变量。

【例5-1】从键盘输入“patient.dbf”表的文件名，要求打开该数据表并显示其记录内容。

```
CLEAR
ACCEPT  "请输入数据表名"  TO  sjk
USE  &sjk
LIST
```

运行例题

5.1 Visual FoxPro程序与程序设计

5.1.3 3种交互式输入输出语句

（2）INPUT 表达式输入命令

【格式】INPUT [<提示信息>] TO <内存变量>

【功能】将用户输入的内容赋值给指定的内存变量。

【说明】① INPUT命令可以接收字符型、数值型、逻辑型、日期型和日期时间型等类型的数据；

②输入字符串时必须加定界符，输入逻辑型常量时用圆点定界(如.F.)，输入日期时间型常量时用大括号(如{^2014-11-12})

【例5-2】输入两个边长的值后输出长方形面积结果。

```
INPUT "请输入长方形一条边的边长：" TO a
INPUT "请输入长方形另一条边的长：" TO b
s=a*b
? "该长方形的面积为：", s
```

运行例题

5.1 Visual FoxPro程序与程序设计

5.1.3 3种交互式输入输出语句

（3）WAIT单字符输入命令

【格式】WAIT [<字符表达式>] [TO <内存变量>] [WINDOW [AT <行>,<列>]] [NOWAIT][CLEAR | NOCLEAR][TIMEOUT <数值表达式>]

【功能】暂停程序执行直到用户按任意键或单击鼠标继续执行

【说明】

①如果没有<字符表达式>，则显示“按任意键继续. . .”；

②<内存变量>类型为字符型。若用户按的是Enter键或单击了鼠标，那么<内存变量>中保存的将是空串。

5.1 Visual FoxPro程序与程序设计

（3）WAIT单字符输入命令

【说明】

③如果指定了WINDOW子句，则会出现提示窗口来显示提示信息。提示窗口一般位于主窗口的右上角，也可用AT短语指定其在主窗口的位置；

④选用NOWAIT短语，系统将不等待用户按键，直接往下执行；

⑤若选用NOCLEAR短语，则不关闭提示窗口，直到用户执行下一条WAIT WINDOW命令或WAIT CLEAR命令为止；

⑥TIMEOUT子句用来设定等待时间(秒数)。一旦超时就不再等待用户按键，自动往下执行。

5.1 Visual FoxPro程序与程序设计

（3）WAIT单字符输入命令

【例5-3】键盘输入一个正数，求以此数为半径的圆的面积和体积。在屏幕的右上角提示用户操作结束，且停留5秒钟。

```
CLEAR
INPUT  "请输入圆的半径="  TO r
   s=3.14*r*r
   v=4/3*3.14*r^3
   ? "圆的半径为：",r
   ? "圆的面积为：",s
   ? "球的体积为：",v
WAIT "运行结束，5秒后退出！"  WINDOW TIMEOUT 5
```

运行例题

5.1 Visual FoxPro程序与程序设计

三种输入语句的比较

语句		Accept …… To……	Input…… To……	Wait
相同		暂停程序的运行，显示提示信息，接收从键盘输入的内容，保存到内存变量中		
不同	类型	C	C、N、L、D、T	C
	长度	任意	任意	1
	输入C型常量时	无定界符	有定界符	无定界符

5.1 Visual FoxPro程序与程序设计

3. Visual FoxPro输出命令

（1）格式化输入输出命令 @

【格式】@〈行号，列号〉 SAY〈表达式〉 [GET〈变量〉] [RANGE〈表达式1〉〈,表达式2〉] [VALID〈条件〉]

【功能】指定在屏幕上的行、列号处，输出表达式的值。

【说明】

① SAY〈表达式〉用来输出表达式的值。

② GET〈变量〉在屏幕指定的位置接收用户输入的数据，并赋值给指定的变量，且必须与READ命令配套使用。注意这里的变量必须事先存在，程序运行到READ命令语句时激活GET子句。

③ RANGE 子句规定GET输入数值的上、下界。

④ VALID 子句规定GET输入的变量值所符合的条件。

5.1 Visual FoxPro程序与程序设计

（2）文本输出命令TEXT / ENDTEXT

【格式】 TEXT

<若干文本行>

ENDTEXT

【功能】显示输出TEXT与ENDTEXT之间<文本行>的内容。

【说明】① TEXT是命令入口， ENDTEXT 是命令出口，它们必须成对出现。

② 该命令功能必须书写在程序中才执行，在命令窗口中无法执行。

【例5-4】制作一个简单的菜单程序界面。

```
    TEXT
**************************************************************
                        住院信息查询系统
                        ~~~~~~~~~~~~~~~~~~
1-患者信息查询    2-床位分配查询    3-费用查询    4-退出查询
**************************************************************
    ENDTEXT
    WAIT  “ 请输入功能代码1-4: ”   TO  dm
```

运行例题

5.1 Visual FoxPro程序与程序设计

（3）Visual FoxPro的计算与输出命令：？/ ??

【格式1】？〈表达式〉

【格式2】??〈表达式〉

【说明】

① ?单问号格式，先计算表达式并在屏幕当前行的下一行显示表达式结果。

② ??双问号格式，先计算表达式并在屏幕当前行、当前列的后面接续显示表达式结果。

③ ?单问号后面不加任何表达式，程序运行时起到屏幕换行作用

5.1 Visual FoxPro程序与程序设计

4.Visual FoxPro的注释命令

【格式1】NOTE / * <注释字符串>

【格式2】&& <注释字符串>

【说明】

① 上述命令不作任何操作，只是注释标记。

② 注释内容换行时，行首也必须再写注释命令。

③ NOTE或*是用于整行注释的命令；而&&命令是用于注释同一行命令内容的，因此它写在本行的后面。

```
Clea                          &&清屏
x= “程序设计”
@8,10 say x
          *在第8行、第10列显示X值。
```

5.2 Visual FoxPro程序的基本结构设计

Visual FoxPro有3种基本结构：顺序结构、分支结构和循环结构

5.2.1 顺序结构程序设计

以程序文件中命令的书写顺序为执行的先后次序，逐条执行。

【例5-5】 编写一个按病人姓名查询的程序，该程序为顺序结构设计。

```
CLEAR
USE patient
ACCE “请输入查询患者的姓名：” TO xm
LOCATE FOR姓名=ALLTRIM(xm)
DISP 住院号,姓名,性别,出生日期,入院日期,住院科室
USE
RETURN
```

运行例题

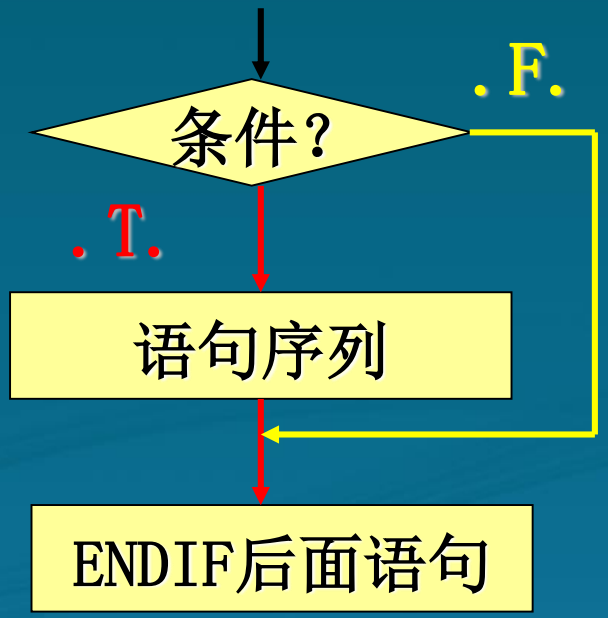
5.2 Visual FoxPro程序的基本结构设计

5.2.2 分支（选择）结构程序设计

1. IF简单条件转向分支结构

【格式】

```
IF  <条件表达式>
    <语句序列>
ENDIF
```



【功能】程序运行到IF语句时，判断条件表达式是否成立，如果表达式为真值（.T.），则执行IF后面的语句序列，如果表达式为假值（.F.），则程序跳转到ENDIF语句后面的语句继续执行。

5.2 Visual FoxPro程序的基本结构设计

1. IF简单条件分支结构

【例5-6】编程在表“patibed.dbf”中查询“张显明”医生负责的患者床位号。

```
CLEAR                          &&清屏幕
USE  patibed                   &&开数据表
LOCATE FOR  医生姓名＝"张显明"
IF  FOUND()                    &&分支语句开头
?"医生姓名："＋医生姓名
? "床位号："＋床位号
ENDIF                          &&分支语句结尾
USE                            &&关闭数据表
RETURN                         && 程序结束
```

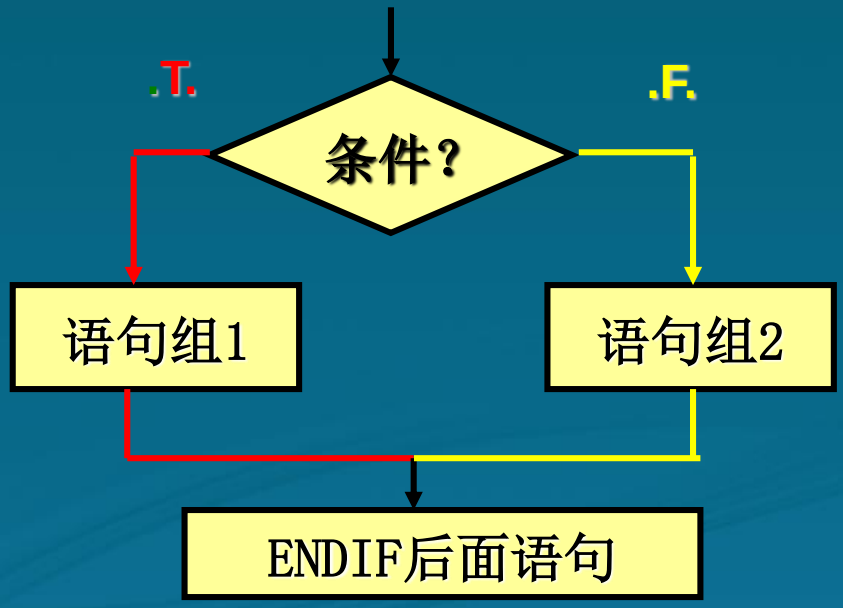
运行例题

5.2 Visual FoxPro程序的基本结构设计

2. IF带否定条件分支结构

【格式】

```
IF <条件表达式>
   <语句序列1>
ELSE
   <语句序列2>
ENDIF
```



【功能】程序运行到IF语句时，判断条件表达式是否成立，如果表达式的值为真值,则执行句序列1后则跳转到ENDIF语句后面继续执行。如果表达式的值为，则程序执行语句序列2后跳转到ENDIF语句后面继续执行。

5.2 Visual FoxPro程序的基本结构设计

2. IF带否定条件分支结构

【例5-7】编写密码校验程序(假设密码为XYZ)

```
CLEAR
SET  EXACT  ON        &&设置字符串严格比较

ACCEPT "请输入密码：" TO  mm
IF  mm= “XYZ”
  CLEAR
  @ 15,20 SAY  "欢迎登入，继续！"
ELSE
  ?  "密码错误！"
  WAIT  "按任意键结束"
  QUIT
ENDIF
```

运行例题

5.2 Visual FoxPro程序的基本结构设计

课堂测验：任意输入一个整数，判断其奇偶性并输出结果。

```
*文件名:奇数偶数判断.PRG
CLEA
INPUT "请输入要判断的数：" TO K
IF K/2=INT(K/2)
     ?STR(K,3)+"为偶数"
ELSE
     ?STR(K,3)+"为奇数"
ENDIF
?  “判断完毕！”
```

运行例题

5.2 Visual FoxPro程序的基本结构设计

3. DO CASE多分支选择语句结构

【格式】

```
DO  CASE
    CASE  <条件表达式1>
        <语句序列1>
    CASE   <条件表达式2>
        <语句序列2>
      …
   [OTHERWISE
        <语句序列N>]
ENDCASE
```

【功能】该结构可依次判断每个CASE条件表达式是否成立，如表达式为，则运行其下面的语句序列，然后跳转到ENDCASE后面语句继续运行;如果所有的条件都不成立，则执行OTHERWISE与ENDCASE之间的语句序列，然后跳转到ENDCASE后面语句继续执行。

5.2 Visual FoxPro程序的基本结构设计

3. DO CASE多分支选择语句结构

【例5-8】以学生成绩为例等级为：分数大于等于85分为“优秀”，大于等于70小于85分的为“良好”，大于等于60小于70分为及格，小于60分为不及格。

```
CLEAR
INPUT SPACE（10）+"请您输入分数:"  TO  score
DO CASE
   CASE  score >=85
        ? SPACE(5)+"您的等级评定是"+"优秀"
CASE score>=70
        ?  SPACE(5)+"您的等级评定是"+"良好"
CASE score >=60
        ?  SPACE(5)+"您的等级评定是"+"及格"
OTHERWISE
        ?  SPACE(5)+"您的等级评定是"+"不及格"
ENDCASE
RETURN
```

运行例题

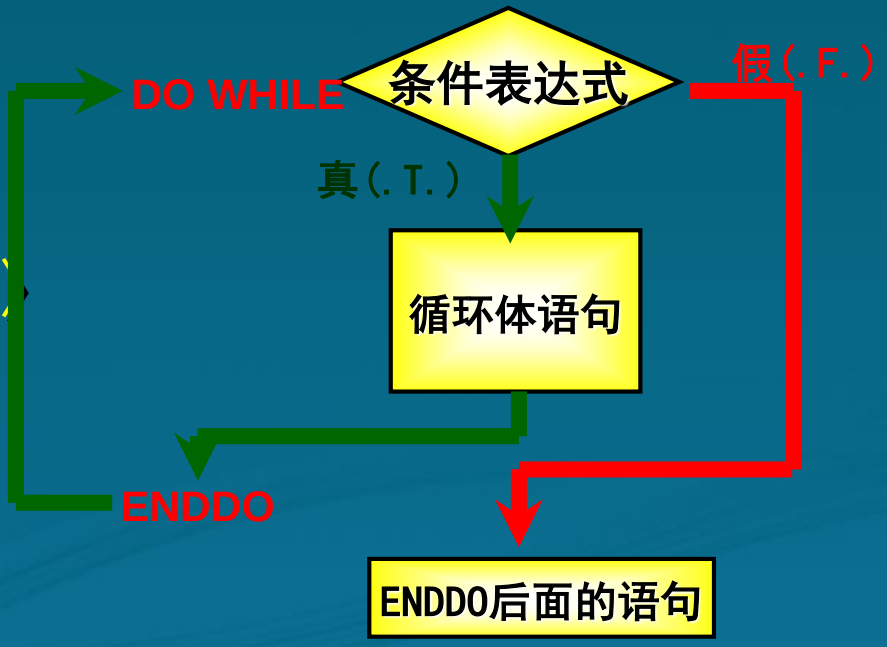
5.2 Visual FoxPro程序的基本结构设计

5.2.3 循环结构程序设计

1. 条件循环结构

【结构1】条件循环结构

【格式】DO WHILE〈条件表达式〉
〈语句序列〉
ENDDO



【功能】每次运行到DO WHILE <条件>语句时，先要判断条件表达式是否成立（真假），为真值时，则运行循环体内的语句序列，遇到ENDDO语句时，程序自动跳转回到DO WHILE语句，再次判断条件表达式的真假状态，往复运行循环体语句序列，直到表达式为假值时结束循环，程序跳转到ENDDO语句后面继续运行。

5.2 Visual FoxPro程序的基本结构设计

1. 条件循环结构

【例5-9】试编程查找性别为“男”的住院患者。

```
CLEAR
USE patient
LOCATE FOR 性别="男"
DO WHILE NOT EOF()          &&循环开头
    DISPLAY 姓名,性别
    CONTINUE
ENDDO                       &&循环结尾
USE
RETURN
```

运行例题

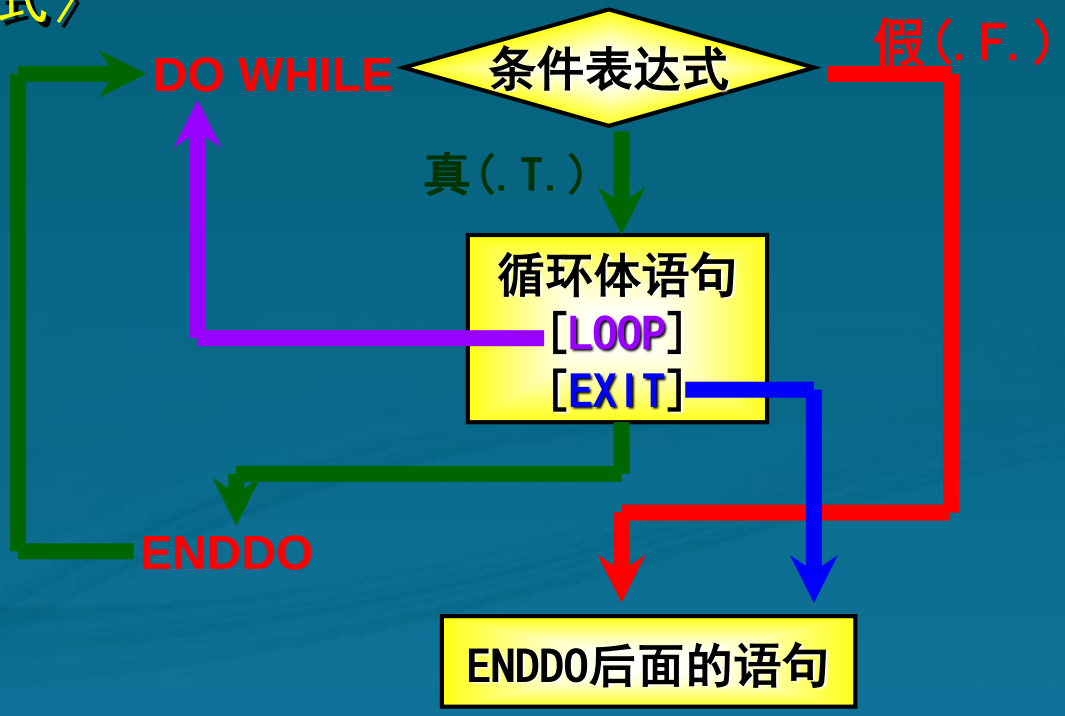
课堂测验：编程求50以内的奇数累加和。（参见:例5-10.PRG）

运行测试题

5.2 Visual FoxPro程序的基本结构设计

1. 条件循环结构

【格式2】DO WHILE 〈条件表达式〉
〈语句序列1〉
[LOOP判断]
〈语句序列2〉
[EXIT判断]
〈语句序列3〉
ENDDO



【功能】

LOOP语句，一般放在循环体内的IF判断语句结构中，根据条件判断为真时，使程序无条件的跳转到循环开头语句 DO WHILE〈条件表达式〉，重复执行循环体操作。

EXIT语句，一般也是放在循环体内的IF判断语句结构中，可使程序无条件的结束循环，跳转到ENDDO语句的后面继续运行。

5.2 Visual FoxPro程序的基本结构设计

1. 条件循环结构

【例5-11】任意输入一个大于3的数，编程判断此数是否是素数

解题思路：

1. 首先用DO 循环、IF 条件语句、LOOP、EXIT语句控制输入一个大于3的数；

2. 一个数如果不能被比自己的平方根小的任何整数除尽，则该数是素数；

3. 用DO 循环、IF 条件语句、LOOP、EXIT语句完成素数的判断；还需定义一个标示变量做跳出循环后是否是素数的标记。

运行例题

5.2 Visual FoxPro程序的基本结构设计

课堂测验（国二真题））：读下面两个程序写出运行结果。

```
*测试题.PRG
CLEAR
STORE  0  TO  X,S
DO WHILE X<20
      X=X+1
    IF MOD(x,3)=0
      S=S+X
     ENDIF
ENDDO
? S
RETURN
```

运行测试题1

```
*测试题2.PRG
CLEAR
S=0
N=5
DO  WHILE  .T.
  S=S+N
  N=N-1
  IF N<0
     EXIT
   ENDIF
ENDDO
?"S="+STR(S,2)
RETURN
```

运行测试题2

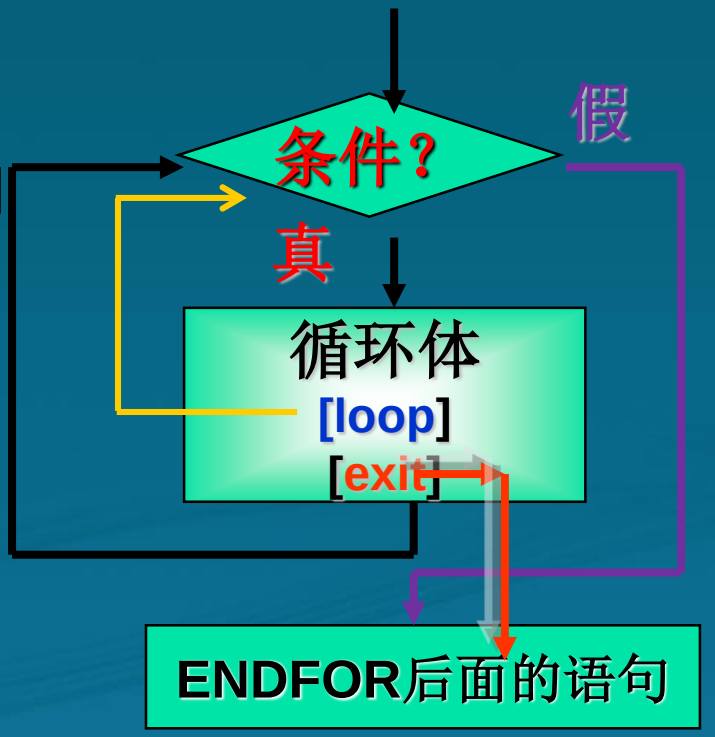
5.2 Visual FoxPro程序的基本结构设计

2. FOR 循环语句

【格式】

```
FOR 〈循环变量〉= 初值 TO 终值 [STEP 步长]
    〈循环体语句序列〉
ENDFOR | [ NEXT]
```

循环次数 =（终值-初值）/ 步长的绝对值+1。步长缺省值为1。



【功能】FOR和ENDFOR构成计数式循环控制结构。当执行FOR语句时，程序自动将初值赋值给循环变量并和终值进行比较，判断循环变量的值是否超过终值，如果没有超过终值执行循环体语句序列，遇到ENDFOR或NEXT语句时，程序自动进行初值加步长值计算，然后将程序跳转回FOR语句，再一次进行上述的循环控制条件的判断比较，当循环变量的值超过终值时，程序直接跳转到ENDFOR后面的语句执行。

5.2 Visual FoxPro程序的基本结构设计

2. FOR 循环语句

【例5-12】编程求任意输入10个数中的最大及最小数。

```
INPUT "请输入一个数：" TO x
STORE  x  TO ma, mi          && ma,mi分别放最大和最小数
FOR i=2 TO 10                && 循环开头
    INPUT "请输入一个数：" TO x
    IF ma<x
       ma=x
    ENDIF
   IF mi>x
       mi=x
   ENDIF
ENDFOR                       &&循环结尾
? "最大数是：" ,ma
? "最小数是：" ,mi
```

运行例题

5.2 Visual FoxPro程序的基本结构设计

2. FOR 循环语句

【例5-13】编程找出100-999之间的“水仙花数”。所谓“水仙花数”是指一个3位数，其各位数字的立方和等于该数本身（如153=1^3+5^3+3^3）。

```
CLEAR
FOR i=100 TO 999
   a=INT(i/100)              &&求出百位数
   b=INT((i-100*a)/10)       &&求出十位数
   c=i-INT(i/10)*10          &&求出个位数
   IF i=a^3+b^3+c^3
      ? "  水仙花数是:",i
   ENDIF
ENDFOR
RETURN
```

运行例题

课堂测验：字符串截取的方法获得个、十、百位数字完成编程。

5.2 Visual FoxPro程序的基本结构设计

2. FOR 循环语句

课堂测验答案：

```
clear
for i=100 to 999
  s=str(i,3)              && 数值I转化成字符型
  a=val(left(s,1))        && 获得百位数
  b=val(subs(s,2,1))      && 获得十位数
  c=val(right(s,1))       && 获得个位数
  if  i=a^3+b^3+c^3
     ? "  水仙花数是:",i
  endif
endfor
```

运行例题

5.2 Visual FoxPro程序的基本结构设计

2. FOR 循环语句

课堂测验（国二真题））：读程序写出运行结果。

```
*FOR测验1.PRG
CLEAR
A=1
FOR X=1 TO 10 STEP 2
    A=A+X
    X=X+1
 NEXT
 ? A,X
 RETURN
```

运行测试题1

运行测试题2

```
*FOR测验2.PRG
CLEAR
FOR X=1 to 5
   FOR Y=1 TO 2*X-1
    @ X+1,20 +Y  SAY STR(X,1)
    @ X+1,40 - Y SAY STR(X,1)
   NEXT
NEXT
RETURN
```

5.2 Visual FoxPro程序的基本结构设计

3. SCAN 数据表循环查询语句

【格式】SCAN [范围] [FOR 条件1]|[WHILE 条件2]

〈循环体语句序列〉

ENDSCAN

【功能】在数据表内查找满足条件的记录，若找到则将指针指向该记录，然后执行循环体，到达ENDSACN时返回循环头。再次查找下一条符合条件的记录，直到无符合条件的记录时结束循环。若用WHILE〈条件2〉，遇到一条不满足条件的记录，则停止循环。

【例5-14】用SCAN语句查询姓名为“张会艳”的患者信息。

```
USE patient
SCAN FOR 姓名="张会艳"
DISPLAY
ENDSCAN
```

运行例题

课堂测验：将“patient.dbf”表中出生日期小于1978年的患者显示出来。

运行测试题

5.2 Visual FoxPro程序的基本结构设计

课堂测验：编程技巧高级训练。

1、键盘任意输入5个数后按降序排序输出。
（参考程序：排序.PRG）

运 行

2、键盘任意输入一个字符串（可以有汉字）后倒叙输出。
（参考程序：字符串倒序.PRG）

运 行

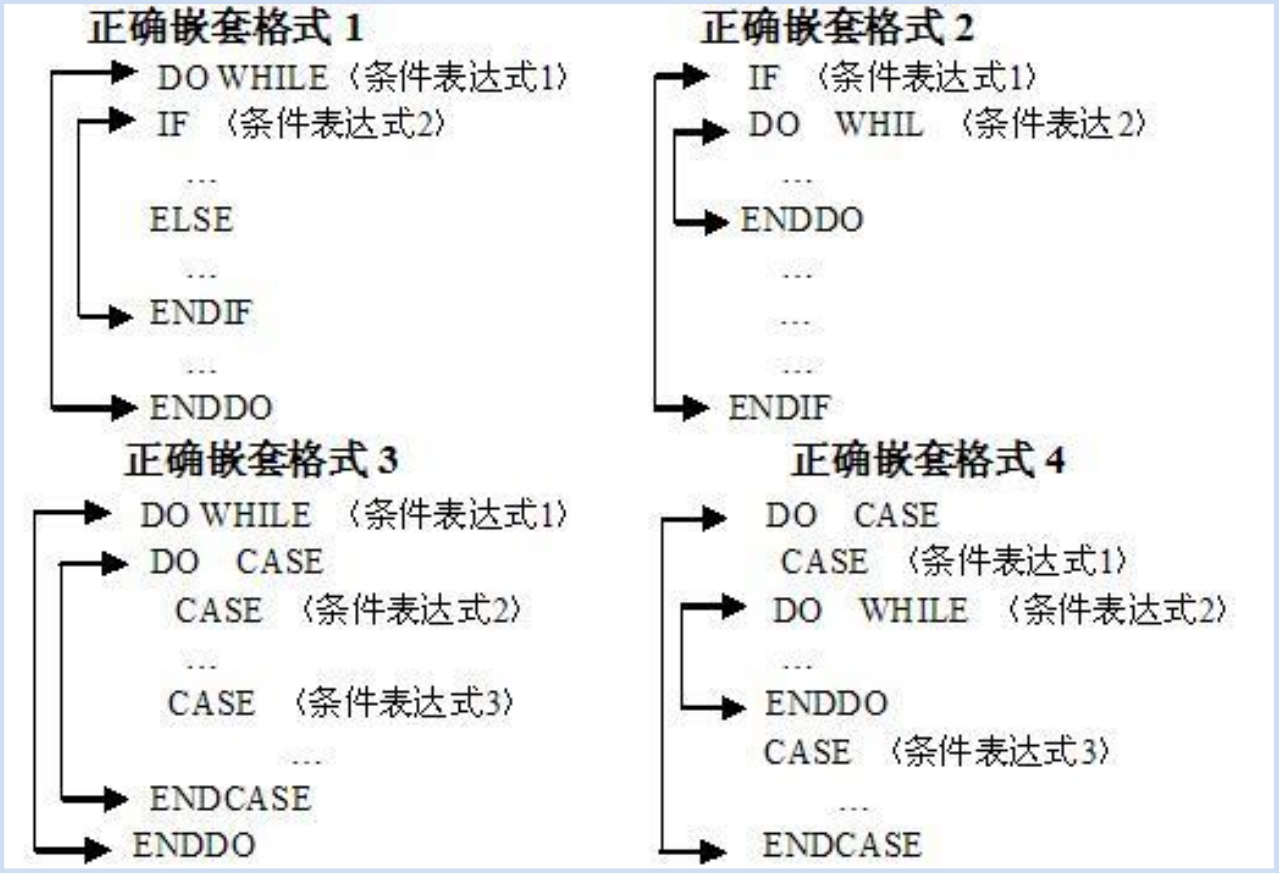
3、键盘输入三角形边长后，打印输出对应的倒三角形。
（参考程序：倒三角.PRG）

运 行

5.2 Visual FoxPro程序的基本结构设计

4. 循环结构与分支结构的正确嵌套关系

分支和循环语句都是由开头语句和结尾语句构成的。它们之间允许嵌套使用，但必须遵守各自头尾语句互不交叉的使用原则。



5.3 Visual FoxPro多模块程序设计

5.3.1 子程序的建立与调用

1. 子程序：子程序是一个独立的程序文件，扩展名为 .PRG。

2. 子程序的调用和返回

调用子程序的命令为：

【格式】 DO <子程序名称>

【功能】控制程序转向子程序继续执行。

子程序返回调用程序的命令为：

【格式】RETURN [TO MASTER/TO <子程序名称>]

【功能】返回上一级程序中调用命令的下一条命令继续执行

5.3 Visual FoxPro多模块程序设计

5.3.1 子程序的建立与调用

1. 子程序：子程序是一个独立的程序文件，扩展名为 .PRG。
2. 子程序的调用和返回

调用子程序：

【格式】 DO <子程序名称>

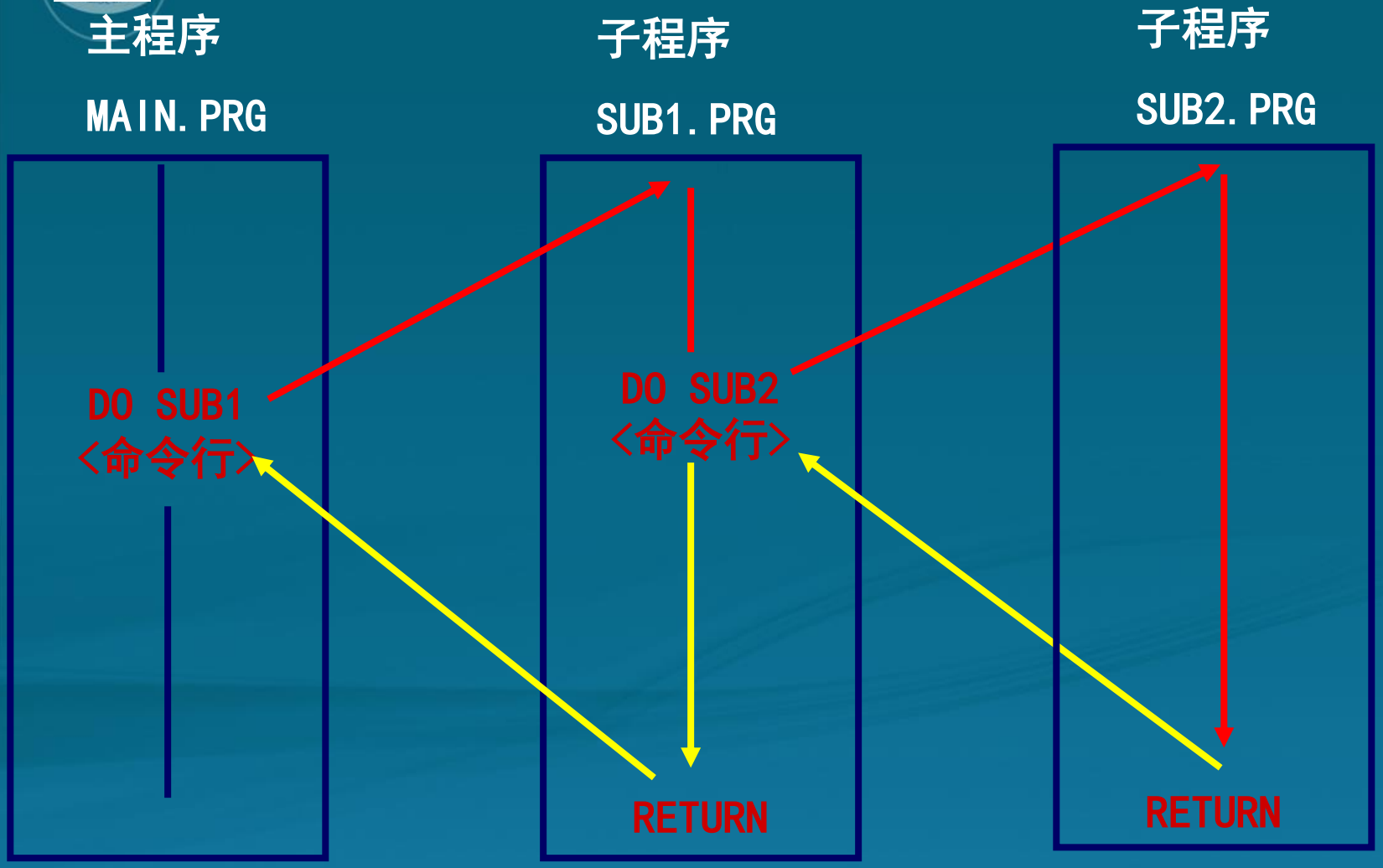
【功能】控制程序转向子程序继续执行。

子程序返回：

【格式】RETURN [TO MASTER/TO <子程序名称>]

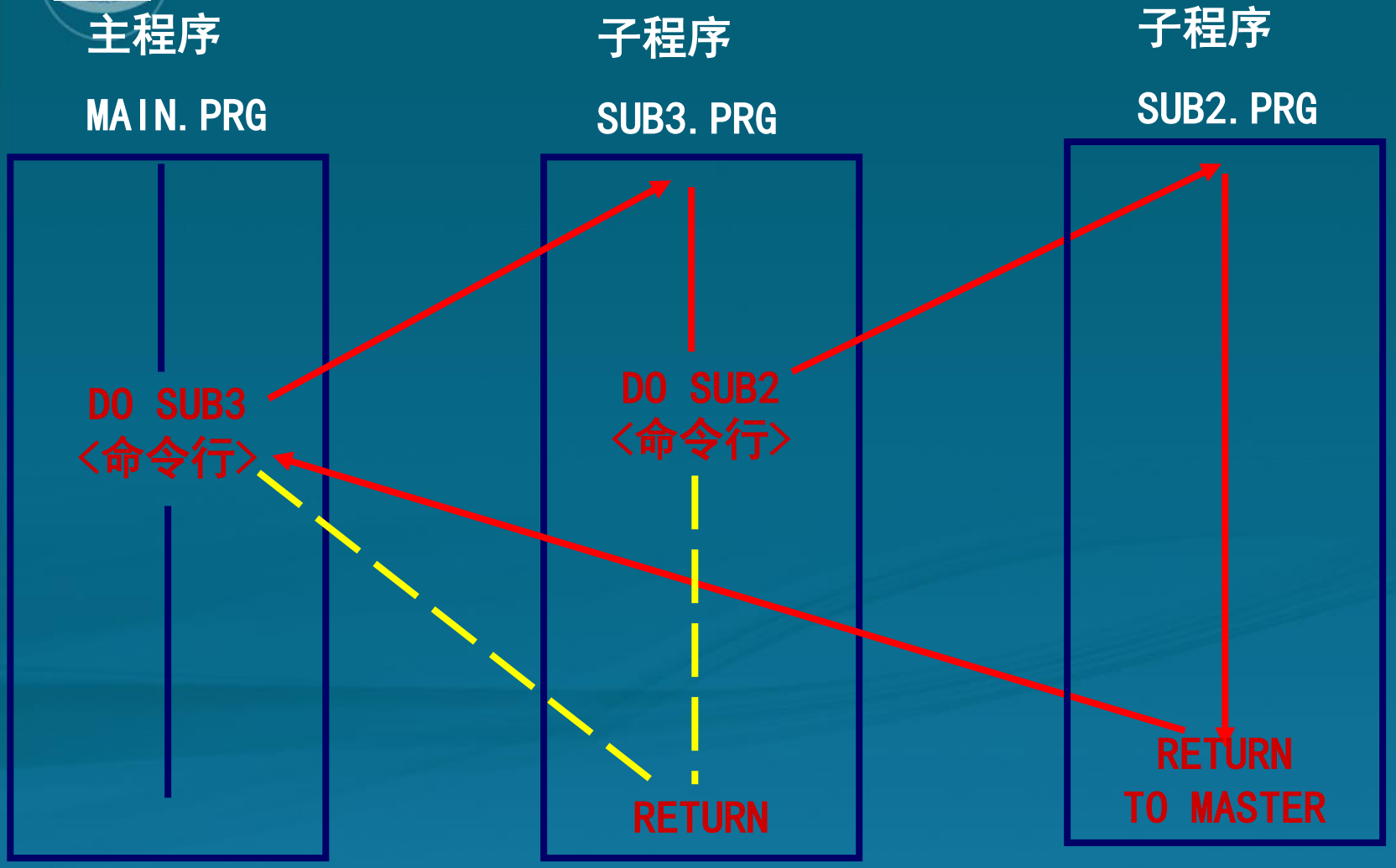
【功能】返回上一级程序中调用命令的下一条命令继续执行.

子程序的调用和返回



嵌套调用，逐级返回

子程序的调用和返回



嵌套调用，越级返回 A

5.3 Visual FoxPro多模块程序设计

5.3.2 过程的建立与调用

1. 过程的概念

过程是为完成一定功能而设计的一组命令序列。过程文件的建立方法与程序文件的建立方法相同，只是每个过程都有明确的首尾标识语句。

2. 过程的建立

过程有内部过程与外部过程之分。内部过程一般放在主程序文件后面，或者单独建立一个过程文件(文件里放若干个内部过程)；而外部过程就是一般的程序文件（子程序）。建立内部过程的语句格式为：

【格式】 PROCEDURE <过程名>
<命令序列>
[RETURN [<表达式>]]
[ENDPROC | ENDFUNC]

5.3.2 过程的建立与调用

【说明】

①内部过程名必须用字母开头，其中可以包含字母、数字、下划线。内部过程也可以用RETURN语句结束返回断点。

② 内部过程的第一条语句必须要用PROCEDURE〈过程名〉命令，声明要使用的过程。

3. 过程的调用

【格式1】DO〈文件名〉|〈过程名〉

【格式2】〈文件名〉|〈过程名〉（）

【说 明】格式2要求文件名或过程名后加一个空的小括号，注意：〈文件名〉后不能加扩张名。如调用外部过程文件名为“CX1.PRG”的文件，其代码为 CX1（）。

4. 内部过程的调用与返回

（1）内部过程的存放与调用---与主程序存放在同一个程序文件中

程序文件格式如下：

```
<命令组>
DO 过程1
<命令组>
DO 过程2
<命令组>
```

主程序

```
PROCEDURE 过程1
<命令组>
RETURN
------

PROCEDURE 过程N
<命令组>
RETURN
```

【例5-16】编程求三角形面积。

```
CLEAR
a=7
b=8
c=9
DO PP1          &&调用内部过程
? 面积
PROCEDURE PP1
      s=(a+b+c)/2
      面积=SQRT(s*(s-a)*(s-b)*(s-
c))
RETURN
```

运行例题

4. 内部过程的调用与返回

课堂测验（国二真题））：读程序写出运行结果。

```
* 5-17.PRG
CLEAR
STORE 3 TO X
STORE 5 TO Y
PLUS((X),Y)          &&调用内部过程
? X,Y
PROCEDURE PLUS
PARAMETERS A1,A2
    A1=A1+A2
    A2=A1+A2
ENDPROC
```

运行例题

4.内部过程的调用与返回

如果过程单独存放在一个过程文件里，调用这些过程时，必须首先打开此过程文件。

【格式】SET PROCEDURE TO [<过程文件1>[,<过程文件2>,…]] [<ADDITIVE>]

【说明】可以打开一个或多个过程文件，如果打开一个过程文件，则该过程文件中的所有过程都可以被调用。如果选用ADDTIVE短语，则在打开过程文件时，不关闭原来已经打开的过程文件；否则当打开一个新的过程文件时，原来已经打开的过程文件将自动关闭。

4.内部过程的调用与返回

（2）内部过程的存放与调用---过程存放在过程文件中

程序文件格式如下：

```
<命令组>

SET PROCEDURE TO<过程文件名>

DO 过程1

<命令组>

DO 过程2

<命令组>

SET PROCEDURE TO
(CLOSE PROCEDURE)
<命令组>
```

过程文件格式如下：

```
PROCEDURE <过程1>
<命令组>
RETURN

      ⋮

PROCEDURE <过程N>
<命令组>
RETURN
```

5. 内部过程文件的关闭

【格式1】CLOSE _PROCEDURE

【格式2】SET_ PROCEDURE TO

【说明】 过程文件运行结束时，应该用上述的两种方式中的任何一个语句来关闭。上述语句将关闭所有打开的过程文件，可以使用下面语句关闭个别过程文件。

RELEASE PROCEDURE TO [<过程文件1>[, <过程文件2>, …]]

【例5-18】利用过程文件完成求两个三角形面积之和。

```
*建立主程序MAIN.PRG，将其存储在磁盘上。
CLEAR
SET PROCEDURE TO  aaa   &&打开过程文件
a=7
b=8
c=9
面积=0
DO  pro                 &&调用过程
面积1=面积
a=17
b=18
c=19
DO  pro                 &&调用过程
面积2=面积
SET PROCEDURE TO        &&关闭过程文件
? ”面积和是:”,面积1+面积2
```

```
*建立过程文件aaa.PRG
PROCEDURE pro
    s=(a+b+b)/2
    面积=SQRT(s*(s-a)*(s-b)*(s-c))
RETURN
```

运行例题

5.3 Visual FoxPro多模块程序设计

5.3.3 内存变量的作用域

有两类内存变量：全局变量和局部变量。

1.全局变量的定义

【格式】PUBLIC <内存变量表>

【功能】定义全局变量。多于2个变量时，用逗号分隔。

【说明】全局变量是指在程序的所有模块中都有效的变量。程序结束后，不会自动释放，只能用RELEASE或CLEAR命令释放。

例如 PUBLIC a,b,c,d

5.3.3 内存变量的作用域

2.局部变量的定义

在程序中未经过PUBLIC定义的内存变量均是局部变量，局部变量是指在当前程序中以及被当前程序调用的程序中有效的变量，程序结束后局部变量会自动释放。

如果希望局部变量只在当前程序中有效而在其调用的下级程序中无效即被释放，可用下面命令定义。

【格式】 LOCAL <内存变量表>

注意：用LOCAL创建的内存变量初始值为“假”（.f.），由于LOCAL和LOCATE前4个字母相同，所以此条命令动词不能缩写。

5.3.3 内存变量的作用域

3.私有变量的使用

在程序中直接使用并且由系统自动隐含建立的变量称为私有变量。私有变量通常用于过程中，其作用范围仅限于此过程和其调用的下层过程中，此类变量在进入此过程中才被定义，离开此过程返回到其上一层后被释放。

【格式1】 PRIVATE 〈内存变量表〉

【格式2】 PRIVATE ALL [LIKE | EXCE PT 〈通配符〉]

例如 PRIVATE x，y

PRIVATE ALL LIKE a*

【说明】该命令并不建立内存变量，它主要是隐藏指定在上层模块中可能存在的内存变量，使这些变量在当前模块程序中暂时无效。

5.3.3 内存变量的作用域

课堂测验（国二真题））：读程序写出运行结果。

```
SET TALK OFF
PRIVATE X,Y
X=“数据库”
Y=“管理系统”
DO  SUB1
? X+Y
RETURN
```

运行例题

```
*子程序：SUB1
PROCEDU SUB1
LOCAL X
X=”应用”
Y=”系统”
X=X+Y
RETURN
```

5.3.4 过程调用中的参数传递

Visual FoxPro中调用过程文件中的过程，可以分为有参数过程和无参数过程。

1. 有参数过程中的形式参数定义

【格式】PARAMETERS <参数表>

【说明】该语句必须放在过程文件中的第一条语句位置。其中<参数表>中的参数称为形式参数，简称形参。形式参数是过程中的局部变量，主要是用来接收过程运算中产生的实际参数的值，形式参数可以用任何有效的内存变量名，如果有多个形式参数，可以用“，”逗号分开，且这些参数必须和过程中的实际参数（变量）相兼容，即个数相同、类型相同、位置相同。

5.3.4 过程调用中的参数传递

2.程序与被调用过程之间的参数传递

过程的调用方法和调用程序文件的方法相同，不同之处在于调用过程文件可以用WITH 〈传递参数表〉实现主程序和过程之间的参数传递工作，且每个过程文件的结束行至少应包含一个RETURN返回语句，以便将控制权交回到主程序调用断点处的下一语句，继续运行。

【格式1】DO 〈文件名〉|〈过程名〉WITH〈实参〉

【格式2】〈文件名〉|〈过程名〉（实参）

【说 明】

① DO〈过程名〉WITH〈参数表〉中的参数为实际参数，简称实参。

② 格式1中如果实参是常量或一般形式的表达式，就按值传递，如果实参是变量，就按地址传递。如果实参是内存变量而又希望进行值的传递，则可以用圆括号把该内存变量括起来，这样可以强制该变量以值传递数据。

③ 格式2的默认情况是按值传递，但是如果实参是变量，可以通过SET UDFPARMS进行设置

SET UDFPARMS TO VALUE：按值传递

SET UDFPARMS TO REFERENCE：按地址传递

2. 程序与被调用过程之间的参数传递

【例5-19】利用参数传递方法计算圆的面积。

主程序为MYMAIN.PRG

```
*MYMAIN.PRG
CLEAR
STORE  0  TO  rr, area
DO WHILE .T.
INPUT " 请输入圆的半径：" TO  rr
DO MYSUB  WITH  rr, area        &&参数传递
   ? "圆的面积是：", area
   WAIT "还要继续计算吗（Y/N）？" TO
answer
   IF UPPER(answer)="Y"
      LOOP
   ELSE
      EXIT
   ENDIF
ENDDO
```

子程序为MYSUB.PRG

```
*MYSUB.PRG
PARAMETERS  r , s
S=PI( )*R^2
RETURN
```

运行例题

2. 程序与被调用过程之间的参数传递

课堂测验（国二真题））：读程序写出运行结果。

```
*5-20.PRG
CLEAR
STORE 100 TO  X1, X2
SET  UDFPARMS TO VALUE &&此语句用在格式1没用
DO P4 WITH X1, (X2)
? X1, X2
*过程P4
PROCEDURE P4
PARAMETERS X1, X2
STORE X1+1 TO X1
STORE X2+1 TO X2
ENDPRO
```

运行例题

本章小结

Visual FoxPro程序设计基础的学习，对于掌握结构化、模块化面向过程的程序设计非常重要。要求熟练的使用顺序、分支、循环3种语句控制结构，完成较为复杂的程序设计，锻炼自己的程序思维方法并按合理的逻辑解决实际遇到的编程问题，使自身具备运用一定的数据库管理技术处理实际问题的能力。