| | |
|---|---|
| 章节标题 | 第二章 数据与数据运算 |
| 授课时数 | 10+4 |
| 教学目标 | 了解：程序的概念，应用程序的调试与执行<br>理解：变量的种类和类型，表达式的组成和分类及各种表达式的表示方法<br>简单的交互式输入输出命令，结构化程序的三种基本结构，<br>过程及变量的作用域的概念，参数的定义<br>掌握：VF 的各种类型常量的书写格式,内存变量常用命令,表达式和常用函数的使用<br>程序文件的建立与执行，三种基本结构语句的编程，<br>过程、过程文件的建立、过程的调用及调用中参数的使用 |
| 主要知识点 | ➢ Visual FoxPro 的各种类型常量的书写格式变量的种类和类型，内存变量常用命令<br>➢ 数值、字符与日期时间表达式，关系表达式，逻辑表达式<br>➢ 常用函数：字符处理函数、数值计算函数、日期时间函数、数据类型转换函数和测试函数<br>➢ 命令文件的建立与运行：程序文件的建立、简单的交互式输入输出命令、应用程序的调试与执行<br>➢ 结构化程序的调试与执行：顺序结构程序设计、选择结构程序设计、循环结构程序设计<br>➢ 过程与过程调用：子程序设计与调用、过程与过程文件、局部变量和全局变量、过程调用中的参数传递 |
| 教学重点 | 内存变量常用命令，字符处理函数、数据类型转换函数<br>程序文件的建立、选择结构程序设计、循环结构程序设计，<br>子程序设计与调用、过程与过程文件、过程调用中的参数传递 |
| 教学难点 | 逻辑表达式、关系表达式，数据类型转换函数、测试函数<br>循环结构程序设计，过程与过程文件、<br>局部变量和全局变量、过程调用中的参数传递 |
| 教学方式(教学方法，教学技术手段的运用等) | 在第一章已经介绍了数据处理的概念，本章先从简单的常量入手介绍了六种常见的数据类型，通过实例讲述了各类型常量的用法，接着介绍变量的使用方法，并讲述了内存变量的常用命令。然后介绍各类型的表达式的概念及使用方法、注意事项、语法结构、运算符优先级等。最后分类型介绍一些常用的函数的使用方法，尤其字符处理函数、数据类型转换函数要求学生们熟练掌握。本章针对每一个知识点都有相关的实例讲述其用法，因此主要是通过例子的解析来理解内容。 |
| 必要说明 | |

# 第二章 数据与数据运算

计算机在处理数据的时候，除了需要处理表中的数据之外，还经常要处理其他的数据。根据计算机系统处理数据的形式来划分，VF 有常量、变量、表达式和函数四种形式的数据。常量和变量是数据运算和处理的基本对象，而表达式和函数则体现了语言对数据进行运算和处理的能力及功能。本章将通过逐一介绍这四种数据使大家掌握这些数据的用法用途。

将各种形式的数据经过数字化存入计算机，都须确定其：

数据 { 型：类型
　　 { 值：取值

在 VFP 中,常量、变量、函数和表达式的类型包括 6 种:字符型、数值型、货币型、逻辑型、日期型、日期时间型。

## 2.1 常量和变量

### 2.1.1 常量

定义：在程序运行过程中保持不变的量 。不同类型的常量有不同 的书写格式。

1、字符型（Character）

表示方法是用半角单引号、双引号或方括号[]扩起来的内容。

（’ ’,“ ”和［］称为定界符）

例 2.1：显示几个字符型常量

2、数值型常量（Numeric）

组成：由 0~9、小数点和正负号。如：12、2.45、-6.78。

3、货币型（Currency）

用来表示货币值，其格式与数值型常量类似，但要加上一个前置的（$）符。

4、日期型（Date）

日期型常量的定界符是一对花括号。花括号内包括年、月、日三部分内容，各部分内容之间用分隔符分隔。系统默认斜杠（/）分隔符。常用的其他日期分隔符有连字号（-）、句点（.）和空格。日期型常量有两种格式：

①传统的日期格式：

（系统默认的日期数据为美国日期格式“月/日/年”（mm/dd/yy），其中月、日、年各为两位数字。）

②严格的日期格式

{^yyyy-mm-dd},用此格式写的日期常量能表达一个确切的日期，它不受 SET DATE 等语句设置的影响。

注：1）花括号内第一个字符必须是(^)

2）年份必须用 4 位

3）年月日的次序不能颠倒、不能缺省。{^2003-12-03}

日期型数据用 8 个字节表示，取值范围是：{^0001-01-01}～{^9999-12-31}

①、②的区别：

严格的日期格式可以在任何情况下使用，而传统的日期格式只能在 SET STRICTDATE TO 0 状态下使用。所以使用严格的日期格式十分方便。

③影响日期格式的设置命令

A）SET MARK TO[日期分隔符]　　&&用于指定日期分隔符 “-””.”,系统默认 “/”

B）SET DATE TO[AMERICAN|ANSI|BRITISH|FRENCH|　　&&设置日期显示的格式

GERMAN|ITALIAN|JAPAN|USA|MDY|DMY|YMD]

C）SET CENTURY ON/OFF　　//设置年份的位数 ON：4 位，OFF：2 位(系统默认)

D）SET STRICTDATE TO[0丨1丨2]　　//设置是否对日期格式进行检查（传统日期格式）

0 表示不进行严格的日期格式检查，目的是与早期 Visual Foxpro 兼容。

1 表示进行严格的日期格式检查，它是系统默认的设置。

2 表示进行严格的日期格式检查，并且对 CTOD()和 CTOT()函数的格式也有效。

5.日期时间型（Date Time）

包括日期和时间两部分内容:{<日期>,<时间>}

其中<日期>与日期型常量相似,也有传统和严格两种格式.

<时间>部分的格式为 [hh[:mm[:ss]][a丨p]]，其中 hh、mm、ss 分别代表时、分和秒，默认值分别为 12、0 和 0。a 和 p 分别代表上午和下午，默认值为 a。如果指定的时间大于等于 12，则自然为下午的时间。

6.逻辑型（Logical）

只有两个值：真和假。 真的表示形式：.T. .t. .Y. .y. ，假的表示形式：.F. .f. .N. .n.

### 2.1.2 变量

变量是在程序运行过程中其值可以发生改变的量。VF 分为字段变量和内存变量。

注：VF 变量命名规则是：

1）以字母、数字、下划线组成，中文版 VF 可以使用汉字做变量名

2）以字母或下划线开始，中文 VF 可以汉字开始

3）长度为 1~128 个字符

4）不能使用 VF 的保留字

由于表中的各条记录对同一个字段名可能取值不同，因此，表中的字段名就是变量，称为字段变量。

内存变量是内存中一个存储区域，变量值就是存放在这个存储区域里的数据，变量的类型取决于变量值的类型。内存变量的数据类型包括字符型（C）、数值型（N）、货币型（Y）、逻辑型（L）、日期型（D）和日期时间型（T）。

1.简单内存变量

每一个变量都有一个名字，可以通过变量名访问变量。如果当前表中存在一个同名的字段变量，则在访问内存变量时，必须在内存变量名前加上前缀M.（或M　）

变量的赋值命令有两种格式：

<内存变量名>=<表达式>　赋值号

STORE <表达式> TO <内存变量名表>

功能注释：

①等号一次只能给一个内存变量赋值。STORE 命令可以同时给若干个变量赋予相同的值，各内存变量名之间必须用逗号分开。

②在 VF 中，一个变量在使用之前不需要特别的声明或定义。当用命令给变量赋值时，如果该变量并不存在，那么系统会自动建立它。

③可以通过对内存变量重新赋值来改变其内容和类型。

2.数组

数组是内存中连续的一片存储区域，它由一系列元素组成，每个数组元素可通过数组名及相应的下标来访问。每个数组元素相当于一个简单变量，可以给各元素分别赋值。在 VF 中，一个数组中各元素的数据类型可以不同。与简单内存变量不同的是，数组在使用之前一般要用 DIMENSION 或 DECLARE 命令来创建，规定数组的维数，数组名和数组大小。数组大

小由下标值的上下限来决定，下限规定为 1 。
创建数组的命令格式为：
DIMENSION<数组名>（<下标上限 1 >[,<下标上限 2 >]）[,……]
DECLARE<数组名>（<下标上限 1 >[,<下标上限 2 >]）[,……]
以上两种格式的功能完全相同。数组创建后，系统自动给每个数组元素赋以逻辑假.F.
整个数组的数据类型为 A（Array）,而各个数组元素可以分别存放同类型的数据。

### 2.1.3 内存变量常用命令

1.内存变量的赋值
格式 1 STORE<表达式> TO <变量名表>
格式 2 <内存变量名>=<表达式>
2.表达式值的显示
格式 1: ?[<表达式>] 区别
格式 2: ??<表达式>
3.内存变量的显示
格式 1 ： LIST MEMORY[LIKE<通配符>][TO PRINTER|TO FILE<文件名>]
格式 2 ： DISPLAY MEMORY[LIKE<通配符>][TO PRINTER|TO FILE<文件名>]
LIKE 只显示与通配符相匹配的内存变量。通配符包括*和？。
区别：LIST MEMORY 一次显示与通配符匹配的所有内存变量，如果内存变量多，一屏显示不下，则自动向上滚动。DISP MEMORY 分屏显示与通配符匹配的所有内存变量，如果内存变量多，显示一屏后暂停，按任意键之后再继续显示下一屏。
4.内存变量的清除
格式 1 ： CLEAR MEMORY
格式 2 ： RELEASE<内存变量名表>
格式 3 ： RELEASE ALL[EXTENDED]
格式 4 ： RELEASE ALL[LIKE<通配符>|EXCEPT<通配符>]
功能：
格式 1 清除所有内存变量。
格式 2 清除指定的内存变量。
格式 3 清除所有内存变量。一般情况下作用与格式 1 相同。如果出现在程序中则应该加上短语 EXTENDED,否则不能删除公共内存变量。
格式 4 选用 LIKE 短语清除与通配符相匹配的内存变量，选用 EXCEPT 短语清除与通配符不相匹配的内存变量。

* 5 .表中数据与数组数据之间的交换
①将表的当前记录复制到数组
格式 1 ： SCATTER[FIELDS<字段名表>][MEMO] TO <数组名>[BLANK]
格式 2 ： SCATTER[FIELDS LIKE<通配符>|FIELDS EXCEPT<通配符>][MEMO] TO <数组名>[BLANK]
②将数组数据复制到表的当前记录
格式 1 ： GATHER FROM <数组名>[FIELDS<字段名表>][MEMO]
格式 2 ： GATHER FROM <数组名>[FIELDS LIKE<通配符>|FIELDSEXCEPT<通配符>][MEMO]

## 2.2 表达式

表达式是由常量、变量和函数通过特定的运算符连接直接起来的式子。
表达式可分为数值表达式、字符表达式、日期时间表达式和逻辑表达式

### 2.2.1 数值、字符与日期时间表达式

1.数值表达式

数值表达式由算术运算符将数值型数据连接起来形成，其运算结果仍然是数值型数据。数值型数据可以是数值型常量或者变量。

①算术运算优先级

表 2.2　算术运算符及其优先级

②求余运算

求余运算%和取余函数 MOD()的作用相同。

注：余数的正负号与除数一致，当被除数与除数异号时，结果为余数＋除数。。

2.字符表达式

字符表达式由字符串运算符将字符型数据连接起来形成，其运算结果仍然是字符型数据。

运算符：

＋：前后两个字符串首尾连接形成一个新的字符串

－：连接前后两个字符串，并将前字符串的尾部空格移到合并后的新字符串尾部。

3.日期时间表达式

运算符：＋和－

注：日期时间表达式的格式有一定限制,不能任意组合.如:不能用运算符＋将两个<日期>连接起来。合法的日期时间表达式格式如表 2.3 所示,其中<天数>和<秒数>都是数值表达式。

### 2.2.2 关系表达式

1.关系表达式

由关系运算符将两个运算对象连接起来形成。即<表达式 1><关系运算符><表达式 2>

注：前后两个运算对象的数据类型要一致。但日期和日期时间型数据可以比较。

①数值型和货币型数据比较

②日期或日期时间型数据比较

③逻辑型数据比较

④子串包含测试$

格式：<前字符型表达式>$<后字符型表达式>　若前者是后者的一个子字符串，结果为逻辑真(.T.)，否则为逻辑假(.F.)。

2.设置字符的排序次序

当比较两个字符串时，系统对两个字符串的字符自左向右逐个进行比较，一旦发现两个对应的字符不同，就根据这两个字符的排序序列决定两个字符串的大小。对字符序列的排序设置有人机对话和命令两种方式。

①在人机对话方式（菜单）下设置

工具　选项　数据（选项卡）　排序序列

②命令方式设置

命令：

SET COLLATE TO　“<排序次序名>”

排序次序名有三种:
“Machine”(机器):
按照机内码顺序排序，在微机中，西文字符是按照 ASCII 码值排列的，空格在最前面，大写字母 ABCD 序列在小写字母 abcd 序列的前面。因此，大写字母＜小写字母。对常用的汉字而言，根据它们的拼音顺序决定大小。
“PinYin”(拼音):
按照机内码顺序排序,对西文字符而言,空格在最前面,小写字母 abcd 序列在大写字母 ABCD 序列的前面。因此，空格<小写字母＜大写字母
“Stroke”(笔画):
无论中文、西文，均按照书写笔画的多少排序。
3.字符串精确比较与 EXACT 设置
在用运算符“==”比较两个字符串时,只有当两个字符串完全相同(包括空格以及各字符的位置)时,运算结果才会是逻辑真.T.,否则为逻辑假.F.,在用“=”比较两个字符串时,运算结果与 set exact on|off 设置有关,该命令是设置精确匹配与否的开关.
①系统默认 off 状态.当处于 off 状态时,只要右边的字符串与左边字符串的前面部分内容相匹配,即可得到逻辑真.T.的结果.也就是说,字符串的比较以右面的字符串为目标,右字符串结束即终止比较.
②当处于 on 状态时,比较到两个字符串全部结束,先在较短字符串的尾部加上若干个空格,使两个字符串的长度相等,然后再进行比较.set exact 设置详见下表示例。

### 2.2.3 逻辑表达式

由逻辑运算符将逻辑型数据连接起来而形成，其运算结果仍然是逻辑型数据。
逻辑运算符 ： NOT 或!(逻辑非)、AND(逻辑与)、OR(逻辑或)
逻辑运算符的优先顺序：NOT AND OR
运算符的优先级：
算术运算符、字符串运算符和日期时间运算符→关系运算符→逻辑运算符
高 低

## 2.3 常用函数

用程序来实现的一种数据运算或转换。每一个函数都有特定的数据运算或转换功能，它往往需要若干个变量，即运算对象，但只能有一个运算结果，称为函数值或返回值，函数可以用函数名加一对圆括号加以调用，自变量放在圆括号里，如 LEN(x)
分类：数值函数、字符处理函数、日期类函数、数据类型转换函数、测试函数 5 类。

### 2.3.1 数值运算函数

1.绝对值和符号函数
ABS(<数值表达式>) 返回指定数值表达式的绝对值。
SIGN(<数值表达式>) 返回指定数值表达式的符号。
2.求平方根函数
SQRT(<数值表达式>)
返回指定表达式的平方根。但自变量表达式的值不能为负。
3.圆周率函数
PI()：返回圆周率π(数值型)。该函数没有自变量。
4.求整数函数

INT(<数值表达式>)　　返回指定数值表达式的整数部分。
CEILING(<数值表达式>)　返回大于或等于指定数值表达式的最小整数。
FLOOR(<数值表达式>)　返回小于或等于指定数值表达式的最大整数。
5.四舍五入函数
ROUND(<数值表达式 1>,<数值表达式 2>)　返回指定表达式在指定位置四舍五入的结果。<数值表达式 2>指明四舍五入的位置。若<数值表达式 2>大于等于 0，表示要保留的小数的位数；若<数值表达式 2>小于 0，表示的整数部分的舍入位数。
6.求余数函数
MOD(<数值表达式 1>,<数值表达式 2>)　返回两个数值相除后的余数。<数值表达式 1>是被除数,<数值表达式 2>是除数。注：余数的正负号与除数相同。如果被除数与除数同号，那么函数值即为两数相除的余数。如果被除数与除数异号，则函数值为两数相除的余数再加上除数的值。例：15%-4　3-4=-1　MOD(15,4)
7.求最大值和最小值函数
MAX(<数值表达式 1>,<数值表达式 2>) [<数值表达式 3>…] ）
　　计算各自变量表达式的值，并返回其中的最大值。
MIN(<数值表达式 1>,<数值表达式 2>) [<数值表达式 3>…]　)
注：所有表达式的类型必须相同
例：?MAX(‘2’,’12’,’05’),MIN(‘汽车’,’飞机’,’轮船’)

**2.3.2 字符处理函数**
1.求字符串长度函数
LEN(<字符表达式>)　返回指定字符表达式值的长度,即所包含字符个数。函数值为数值型。
2.大小写转换函数
LOWER(<字符表达式>)　将指定表达式值中的大写字母转换成小写字母，其他字符不变。
UPPER(<字符表达式>)　将指定表达式值中的小写字母转换成大写字母，其他字符不变。
3.空格字符串生成函数
SPACE(<数值表达式>)　返回由指定数目的空格组的字符串。
4.删除前后空格函数
TRIM(<字符表达式>)　返回指定字符表达式值去掉尾部空格后形成的字符串。
LTRIM(<字符表达式>)　返回指定字符表达式值去掉前导空格后形成的字符串。
ALLTRIM(<字符表达式>)　返回指定字符表达式值去掉前导和尾部空格后形成的字符串。
5.取子串函数
LEFT(<字符表达式>,<长度>)　从指定表达式值的左端取一个指定长度的子串作为函数值。
RIGHT(<字符表达式>,<长度>)　从指定表达式值的右端取一个指定长度的子串作为函数值。
SUBSTR(<字符表达式>,<起始位置>,<长度>)
　　从指定表达式值的指定起始位置取一个指定长度的子串作为函数值。
注：在 SUBSTR()函数中，若缺省第三个自变量<长度>，则函数从指定位置一直取到最后一个字符。
6.计算子串出现次数函数
　　OCCURS(<字符表达式 1>,<字符表达式 2>)
　　返回第一个字符串在第二个字符串中出现的次数，函数值为数值型。若第一个字符串不是第二个字符串的子串，函数值为 0。
7.求子串位置函数
AT(<字符表达式 1>,<字符表达式 2>[,<数值表达式>])

ATC(<字符表达式 1>,<字符表达式 2>[,<数值表达式>])

功能：函数值为数值型。如果<字符表达式 1>是<字符表达式 2>的子串，则返回字符表达式 1>值的首字符在<字符表达式 2>值中的位置；若不是子串，则返回 0。

ATC 与 AT 功能类似，但在子串比较时字母不区分大小写。

8.子串替换函数

STUFF(<字符表达式 1>,<起始位置>,<长度>,<字符表达式 2>)

功能：用<字符表达式 2>值替换<字符表达式 1>中由<起始位置>和<长度>指明的一个子串。替换和被替换的字符个数不一定相等。如果<长度>值是 0，<字符表达式 2>则插在由<起始位置>指定的字符前面。如果<字符表达式 2>值是空串，那么<字符表达式 1>中由<起始位置>和<长度>指明的子串被删去。

9.字符匹配函数

LIKE(<字符表达式 1>,<字符表达式 2>)

功能：比较两个字符串对应位置上的字符，若所有对应字符都相匹配，返回逻辑真(.T.)，否则返回逻辑假(.F.)。<字符表达式 1>中可以包含通配符*和?。

10.字符替换函数

CHRTRAN(<字符表达式 1>,<字符表达式 2><字符表达式 3>)

功能：该函数的自变量是三个字符表达式。当第一个字符串中的一个或多个字符与第二个字符串中的某个字符相匹配时，就用第三个字符串中的对应字符（相同位置）替换这些字符。如果第三个字符串包含的字符个数少于第二个字符串包含的字符个数，因而没有对应字符，那么第一个字符串中相匹配的各字符将被删除。如果第三个字符串包含的字符个数多于第二个字符串包含的字符个数，多余字符被忽略。

### 2.3.3 日期和时间函数

1.系统日期和时间函数

DATE()　返回当前系统日期，函数值为日期型。

TIME()　以 24 小时制、hh:mm:ss 格式返回当前系统时间，函数值为字符型。

DATETIME()　返回当前系统日期时间，函数值为日期时间型。

2.求年份、月份和天数函数

YEAR(<日期表达式>|<日期时间表达式>)

从指定的日期表达式或日期时间表达式中返回年份。

MONTH(<日期表达式>|<日期时间表达式>)

从指定的日期表达式或日期时间表达式中返回月份。

DAY(<日期表达式>|<日期时间表达式>)

从指定的日期表达式或日期时间表达式中返回月里面的天数。

3.时、分和秒函数

HOUR (<日期时间表达式>)　　从指定的日期表达式中返回小时部分（24 小时制）。

MINUTE(<日期时间表达式>)　从指定的日期表达式中返回分钟部分。

SEC(<日期时间表达式>)　　从指定的日期表达式中返回秒数部分。

### 2.3.4 数据类型转换函数

1.数值转换成字符串

STR(<数值表达式>[,<长度>[,<小数位数>]])

功能：将<数值表达式>的值转换成字符串，转换时根据需要自动进行四舍五入。返回字符串的理想长度 L 应该是<数值表达式>值的整数部分位数加上<小数位数>值，再加上 1 位小

数点。如果<长度>值大于等于<数值表达式>值的整数部分位数（包括负号）但又小于 L，则优先满足整数部分而自动调整小数位数；如果<长度>值小于<数值表达式>值 的整数部分位数，则返回一串星号（*）。<小数位数>的默认值为 0，<长度>的默认值为 10。

2.字符串转换成数值

VAL(<字符表达式>)

功能：将由数字符号（包括正负号、小数点）组成的字符型数据转换成相应的数值型数据。若字符串内出现非数字字符，那么只转换前面部分；若字符串的首字符不是数字符号，则返回数值零，但忽略前导空格。

3.字符串转换成日期或日期时间

CTOD(<字符表达式>)　　将<字符表达式>值转换成日期型数据。

CTOT(<字符表达式>)　　将<字符表达式>值转换成日期时间型数据。

注：如果 SET CENTURY TO 语句指定：小于等于 50 的两位数年份属于 21 世纪（19＋1），而大于 50 的两位年份属于 20 世纪（19）。

4.日期或日期时间转换成字符串

DTOC(<字符表达式>，1)　　将日期型数据或日期时间数据的日期部分转换成字符串。

TTOC(<字符表达式>,1)　　将日期时间数据转换成字符串。

注：字符串中日期部分的格式与 SET DATE TO 语句的设置和 SET CENTURY ON|OFF 语句的设置有关。时间部分的格式受 SET HOURS TO 12|24 语句的设置影响。

对 DTOC()来说，如果使用选项 1，则字符串的格式总是为 YYYYMMDD，共 8 个字符。对 TTOC()来说，如果使用选项 1，则字符串的格式总是为 YYYYMMDDHHMMSS，采用 24 小时制，共 14 个字符。

5.宏替换函数

&<字符型变量>[.]

功能：替换出字符型变量的内容，即＆的值是变量中的字符串。如果该函数与其后的字符无明确分界，则要用“.”作函数结束标识。宏替换可以嵌套使用。

### 2.3.5 测试函数

1.值域测试函数

BETWEEN(<表达式 T>,<表达式 L>,<表达式 H>)

功能：判断一个表达式的值是否介于另外两个表达式的值之间。当<表达式 T>值大于等于<表达式 L>且小于等于<表达式 H>时，函数值为逻辑真（.T.），否则函数值为逻辑假（.F.）。如果<表达式 L>或<表达式 H>有一个是 NULL 值，那么函数值也是 NULL 值。注：三个自变量的数据类型要一致。

2.空值（NULL 值）测试函数

ISNULL(<表达式>)

功能：判断一个表达式的运算结果是否为 NULL 值，若是 NULL 值返回逻辑真（.T.），否则返回逻辑假（.F.）。

3.空值测试函数

EMPTY(<表达式>)

功能：根据指定表达式的运算结果是否为“空”值，返回逻辑真(.T.)或逻辑假(.F.)。

注：这里所指的“空”值与 NULL 值是两个不同的概念。函数 EMPTY(.NULL.)的返回值为逻辑假(.F.)。其次，该函数自变量表达式的类型除了可以是数值型之外，还可以是字符型、逻辑型、日期型等类型。不同类型数据的“空”值，有不同的规定.

4.数据类型测试函数

VARTYPE(<表达式>)[<逻辑表达式>]
功能：测试<表达式>的类型，返回一个大写字母，函数值为字符型。字母的含义如下表所示：
若<表达式>是一个数组，则根据第一个数组元素的类型返回字符串。若<表达式>的运算结果是 NULL 值，则根据<逻辑表达式>值决定是否返回<表达式>的类型：如果<逻辑表达式>值为.T.，就返回<表达式>的原数据类型。如果<逻辑表达式>值为.F.或缺省，则返回 X 以表明<表达式>的运算结果是 NULL 值。
5.表文件尾测试函数
EOF([<工作区号>|<表别名>])
功能：测试指定表文件中的记录指针是否指向文件尾，若是返回逻辑　真.T.，否则返回逻辑假.F.。表文件尾是指最后一条记录的后面位置。若缺省自变量，则测试当前表文件。

Top 记录 1 记录 2…记录 I Bottom

若在指定工作区上没有打开表文件，函数返回逻辑假.F.。或表文件中不包含任何记录，函数返回逻辑真.T.。
6.表文件首测试函数
BOF([<工作区号>|<表别名>])
功能：测试当前表文件(若缺省自变量)或指定表文件中的记录指针是否指向文件首，若是返回逻辑真.T.，否则返回逻辑假.F.。表文件首是指第一条记录的前面位置。若指定工作区上没有打开表文件，函数返回逻辑假.F.。若表文件中不包含任何记录，函数返回逻辑真.T.。
7.记录号测试函数
RECNO([<工作区号>|<表别名>])
功能：返回当前表文件(或缺省自变量)或指定表文件中当前记录(记录指针所指记录)的记录号。如果指定工作区上没有打开表文件，函数值为 0。如果记录指针指向文件尾，函数值为表文件中的记录数加 1。如果记录指针指向文件首，函数值为表文件中第一条记录的记录号。
8.记录个数测试函数
RECCOUNT([<工作区号>|<表别名>])
功能：返回当前表文件(或缺省自变量)或指定表文件中的记录个数。如果指定工作区上没有打开表文件，函数值为 0。返回的是表文件中物理上存放的记录个数。不管记录是否被逻辑删除以及 SET DELETED 的状态如何，也不管记录是否被过滤（SET FILTER），该函数都会把它们考虑在内。
9.条件测试函数
IIF(<逻辑表达式>,<表达式 1>,<表达式 2>)
功能： 测试<逻辑表达式>的值，若为逻辑真.T.，函数返回<表达式 1>的值，若为逻辑假.F.，函数返回<表达式 2>的值。<表达式 1>和<表达式 2>的类型不要求相同。
10.记录删除测试函数
DELETED(<表的别名>|<工作区号>)
功能：测试指定的表，或在指定工作中所打开的表，记录指针所指的当前记录是否有删除标记“*”。若有为真，否则为假。若缺省自变量，则测试当前工作区中所打开的表。

## 2.4 程序与程序文件

以前我们介绍的命令都是以交互式的，即在命令窗口中逐条输入命令或通过选择菜单来执行 VF 命令的。除此之外我们也可以采用程序的方式来调用 VF 系统功能来完成更为复杂的任务。本章我们将介绍程序设计及其相关的内容，如程序与程序文件、程序的基本结构、多模块程序以及程序调试等内容。

### 2.4.1 程序的概念

学习 VF 的目的是什么？

我们要用它的命令来组织和处理数据、完成一些具体任务。而采用程序的方式是解决复杂任务的最好方法。

程序：是能够完成一定任务的命令的有序集合。

程序文件：也称为命令文件，是指将需要执行的一系列命令集中编写并存入指定的文本文件。（程序文件的扩展名为.PRG）当运行程序时，系统会按照一定的次序自动执行包含在程序文件中的命令。

与命令窗口逐条输入命令相比，采用程序方式有如下好处：

- 可以利用编辑器，方便地输入、修改和保存程序
- 可以用多种方式、多次运行程序。
- 可以在一个程序中调用另一个程序。（子程序）

### 2.4.2 程序文件的建立与执行

一．建立

建立：文件→新建→程序

保存：文件→保存（^+w）

命令： MODIFY COMMAND<文件名>

注：若指定文件存在，则打开修改；否则系统认为是要建立一个指定了名字的文件。

二．执行

程序→执行

注：采用此方式运行程序文件时，系统会自动将默认的盘和目录设置为程序文件所在的盘和目录。

命令方式：DO <文件名>

注：该命令既可以在命令窗口发出，也可以出现在某个程序文件中，这样就使得一个程序在执行的过程中还可以调用执行另一个程序。当程序文件被执行时，文件中包含的命令将被依次执行，直到所有的命令被执行完毕，或者执行到以下命令：

(1) CANCEL：终止程序运行，清除所有的私有变量，返回命令窗口。

DO：转去执行另一个程序。

(3) 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。

(4) QUIT：退出 VF 系统，返回到操作系统。

### 2.4.3 简单的输入输出命令

1、INPUT 命令

格式：INPUT<字符表达式> TO <内存变量>

该命令等待用户从键盘输入数据，用户可以输入任意合法的表达式。当用户以回车键结束输入时，系统将表达式的值存入指定的内存变量，程序继续运行。

功能：

（1）若选用<字符表达式>，系统会首先显示该表达式的值，作为提示信息。

（2）输入的数据可以是常量、变量等，但不能不输入任何内容直接按回车键。

（3）输入字符串时必须加定界符，输入逻辑型常量时要用圆点定界（.T.、.F.），输入日期时间型常量时要用大括号。

2、ACCEPT 命令

格式：ACCEPT [<字符表达式>] TO <内存变量>

该命令等待用户从键盘输入字符串。当用户以回车键结束输入时，系统将该字符串存入指定的内存变量，程序继续运行。

功能：

（1）若选用<字符表达式>，那统会首先显示该表达式的值，作为提示信息。

（2）该命令只能接收字符串。用户在输入字符串时不需要加定界符；否则，系统会把定界符作为字符串本身的一部分。

（3）若不输入任何内容而直接按回车键，系统会把空串赋给指定的内存变量。””

3、WAIT 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>]

该命令显示字符表达式的值作为提示信息，暂停程序的执行，直到用户按任意键或单击鼠标时继续程序的执行。

功能：

（1）若<字符表达式>值为空串””，那么不会显示任何提示信息。若没有指定<字符表达式>，则显示默认的提示信息“按任意键继续…”。

（2）<内存变量>用来保存用户键入的字符，其类型为字符型。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。若不选 TO <内存变量>短语，输入的单字符不保留。

（3）一般情况下，提示信息被显示在 VF 主窗口或当前用户自定义窗口里。如果指定了 WINDOW 子句，则会出现一个 WAIT 提示窗口，用以显示提示信息。提示窗口一般定位于主窗口的右上角，也可用 AT 短语指定其在主窗口中的位置。

（4）若选用 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。

（6）TIMEOUT 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。

## 2.5 结构化的程序设计

三大结构：顺序结构、选择结构、循环结构。

1、顺序结构：　按命令在程序中出现的先后次序依次执行。

2、选择结构：　支持选择结构的语句包括条件语句和分支语句。

A）条件语句

格式：IF <条件>

<语句序列 1>

[ELSE

<语句序列 2>]

ENDIF

注：

（1）IF 和 ENDIF 必须成对出现，IF 是本结构的入口， ENDIF 是本结构出口。

（2）条件语句可以嵌套，但不能出现交叉。嵌套时，为了使程序清晰、易于阅读，可按缩进格式书写。

B）分支语句

语句序列 1 条件 1

格式： DO CASE

CASE <条件 1>

<语句序列 2>

CASE <条件 2>

<语句序列 2>

……

CASE <条件 n>

<语句序列 n>

[OTHERWISE

<语句序列>]

ENDCASE

注：

（1）不管有几个 CASE 条件成立，只有最先成立的那个 CASE 条件的对应命令序列被执行。

（2）若所有 CASE 条件都不成立，且没有 OTHERWISE 子句，则直接跳出本结构。

（3）DO CASE 和 ENDCASE 必须成对出现，DOCASE 是本结构的入口,ENDCASE 是本结构的出口

3、循环结构

循环结构也称为重复结构，是指程序在执行的过程中，其中的某段代码被重复执行若干次。被重复执行的代码段，通常称之为循环体。VF 中的循环结构语句包括：DO WHILE-ENDDO、FOR-ENDFOR 和 SCAN-ENDSCAN 语句。

A）DO WHILE-ENDDO 语句

格式： DO WHILE <条件> <语句序列 1>

[LOOP] <语句序列 2>

[EXIT] <语句序列 3>

ENDDO

执行该语句时，先判断 DO WHILE 处的循环条件是否成立，如果条件为真，则执行 DO WHILE 与 ENDDO 之间的命令序列（循环体）。当执行到 ENDDO 时，返回到 DO WHILE，再次判断循环条件是否为真，以确定是否再次执行循环体。若条件为假，则结束该循环语句，执行 ENDDO 后面的语句。

功能：

（1）若第一次判断条件时，条件即为假，则循环体一次都不执行。

（2）若循环体包含 LOOP 命令，那么当遇到 LOOP 时，就结束循环体的本次执行，不再执行其后面的语句，而是转回 DO WHILE 处重新判断条件。

（3）若循环体包含 EXIT 命令，那么当遇到 EXIT 时，就结束该语句的执行，转去执行 ENDDO 后面的语句

（4）通常 LOOP 或 EXIT 出现在循环体内嵌套的选择语句中，根据条件决定是 LOOP 回去，还是 EXIT 出去。

B)FOR-ENDFOR 语句

该语句通常用于实现循环次数已知情况下的循环结构。

格式：

FOR <循环变量>=<初值>TO <终值> [STEP <步长>]

<循环体>

ENDFOR｜NEXT

功能：

（1）<步长>的默认值为 1。

（2）<初值>、<终值>和<步长>都可以是数值表达式。但这些表达式仅在循环语句执行开始时被计算一次。在循环语句的执行过程中，初值、终值和步长是不会改变的。

（3）可以在循环体内改变循环变量的值，但这会影响循环体的执行次数。

（4）EXIT 和 LOOP 命令同样可以出现在该循环语句的循环体内。当执行到 LOOP 命令时，结束循环体的本次执行，然后循环变量增加一个步长值，并再次判断条件是否成立。

C)SCAN-ENDSCAN 语句

格式：

```
SCAN [<范围>][FOR<条件 1>][WHILE<条件 2>]
        <循环体>
        ENDSCAN
```

执行该语句时，记录指针自动、依次地在当前表的指定范围内满足条件的记录上移动，对每一条记录执行循环体内的命令。

功能：

（1）<范围>的默认值是 ALL。

（2）EXIT 和 LOOP 命令同样可以出现在该循环语句的循环体内。

多重循环结构程序设计

在一个循环体中再包含有循环结构，称之为二重循环结构（或二重循环嵌套）；在二重循环结构的循环体中，若还包含循环结构，称为三重循环结构。二重及其以上的循环结构统称为多重循环结构。这里我们主要介绍二重循环结构。

多重循环的基本结构

```
DO WHILE <条件 1>
            <循环体 1>
      DO WHILE <条件 2>
                  <循环体 2>
      ENDDO
ENDDO

FOR <变量 1>=<初值 1> TO <终值 1> STEP <步长 1>
        <循环体 1>
        FOR <变量 2>=<初值 2> TO <终值 2> STEP <步长 2>
          <循环体 2>
        ENDFOR
        <循环体 3>
ENDFOR
```

执行过程的总体原则是：

外层循环变量每取一次值，内层循环要完整地循环一遍。若<循环体 1>、<循环体 2>或<循环体 3>中包含有 LOOP 或 EXIT 命令，它们会改变循环程序的执行顺序。

## 2.6 多模块程序

模块是一个相对独立的程序段，它可以被其他模块所调用，也可以去调用其他的模块。通常，把被其他模块调用的模块称为子程序，把调用其他模块而没有被其他模块调用的模块称为主

程序。将一个应用程序划分成一个个功能相对简单、单一的模块程序，不仅便于程序的开发，也利于程序的阅读和维护。

### 2.6.1 模块的定义和调用

格式：

PROCEDURE丨FUNCTION <过程名>

<命令序列>

[RETURN <表达式>]

[ENDPROC丨ENDFUNC]

（1）PROCEDURE丨FUNCTION：命令表示一个过程的开始，并命名过程名。过程名必须以字母或下划线开头，可包含字母、数字和下划线。

（2）ENDPROC丨ENDFUNC：命令表示一个过程的结束。如果缺省，那么过程结束于下一条 PROCEDURE丨FUNCTION 命令或文件结尾处。

（3）当过程执行到 RETURN 时，控制将转回到调用程序（或命令窗口），并返回表达式的值。若缺省 RETURN 命令，则在过程结束处自动执行一条隐含的 RETURN 命令。若 RETURN 不带<表达式>，则返回逻辑真.T.。

（4）过程可以放置在程序文件代码的后面，也可以保存在称为过程文件的单独文件里。过程文件建立仍使用 MODIFY COMMAND 命令，文件的默认扩展名还是.prg。过程文件里只包含过程，这些过程能被任何其他程序所调用。但在调用过程文件中的过程之前首先要打开过程文件。打开过程文件的命令格式为：

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

可以打开一个或多个过程文件。一旦一个过程文件被打开，那么该过程文件中的所有过程都可以被调用。如果选用 ADDITIVE，那么在打开过程文件时，并不关闭原先打开的过程文件。当使用不带任何文件名的 SET PROCEDURE TO 命令，将关闭所有打开的过程文件。若不想把过程文件全部关闭，而要关闭个别过程文件，可用命令：

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

注：这里的模块主要是指过程和命令文件里的代码。而过程的调用需要它所在的文件处于打开状态。

模块调用的两种格式：

格式 1：使用 DO 命令:　　DO <文件名>丨<过程名>

格式 2：在名字后加一对小括号：　　<文件名>丨<过程名> ( )

注：这两种格式，如果模块是程序文件的代码，用<文件名>否则用<过程名>。格式 2 既可以作为命令使用（返回值被忽略），也可以作为函数出现在表达式里。在这里<文件名>不能包含扩展名。

### 2.6.2 参数传递

模块程序可以接收调用程序传递过来的参数，并能够根据接收到的参数控制程序流程或对接收的参数进行处理，从而大大提供模块程序功能设计的灵活性。

接收参数的命令有 PARAMETERS 和 LPARAMETERS，它们的格式如下：

PARAMETERS <形参变量 1>[<形参变量 2>,…]

LPARAMETERS <形参变量 1>[<形参变量 2>,…]

PARAMETERS 命令声明的形参变量被看作是模块程序中建立的私有变量，LPARAMETERS 命令声明的形参变量被看作是模块程序中建立的局部变量。除此之外，两条命令没有什么不同。

调用模块程序的格式为：
格式 1：DO <文件名>|<过程名> WITH <实参 1>[,<实参 2>,…]
格式 2：<文件名>|<过程名>(<实参 1>[,<实参 2>,…])
注：
（1）实参可以是常量、变量，也可以是一般形式的表达式。调用模块程序时，系统会自动把实参传递给对应的形参。形参的数目不能少于实参的数目，否则系统会产生运行时错误。如果形参的数目多于实参的数目，那么多余的形参取初值逻辑假.F.。
（2）采用格式 1 调用模块程序时，如果实参是常量或一般形式的表达式，系统会计算出实参的值，并把它们赋值给相应的形参变量。这种情形称为按值传递。如果实参是变量，那么传递的将不是变量的值，而是变量的地址。这时形参和实参实际上是同一个变量（尽管它们的名字可能不同），在模块程序中对形参变量值的改变，同样是对实参变量值的改变。这种情形称为按引用传递。
（3）采用格式 2 调用模块程序时，默认情况下都以按值方式传递参数。如果实参是变量，可以通过命令 SET UDFPARMS 命令重新设置参数传递的方式。该命令的格式如下：
SET UDFPARMS TO VALUE|REFERENCE
TO VALUE: 按值传递。形参变量值的改变不会影响实参变量的取值。
TO REFERENCE: 按引用传递。形参变量值改变时，实参变量值也随之改变。

还可以在调用程序和被调用程序之间传递数组。当实参是数组元素时，总是采用按值传递方式传递元素值。当实参是数组名时，若传递方式是按值传递，那么就传递数组的第一个元素值给形参变量，若传递方式是按引用传递，那么传递的将是整个数组。 P5.PRG

### 2.6.3 变量的作用域

程序设计离不开变量。一个变量除了类型和取值之外，还有一个重要的属性就是它的作用域，变量的作用域指的是变量在什么范围内是有效或能够被访问的。在 VF 中，若以变量的作用域来分，内存变量可分为公共变量、私有变量和局部变量三类。
1、公共变量
在任何模块中都可使用的变量称为公共变量。公共变量要先建立后使用，公共变量可用 PUBLIC 命令建立。
PUBLIC <内存变量表>
该命令的功能是建立公共的内存变量，并为它们赋初值逻辑假.F.。
注：公共变量一旦建立就一直有效，即使程序运行结束返回到命令窗口也不会消失。只有执行 CLEAR MEMORY、RELEASE、QUIT 等命令后，公共变量才被释放。在命令窗口中直接使用而由系统自动隐含建立的变量也是公共变量。
2、局部变量
局部变量只能在建立它的模块中使用，不能在上层或下层模块中使用。当建立它的模块程序运行结束时，局部变量自动释放。局部变量用 LOCAL 命令建立：
LOCAL <内存变量表>
该命令建立指定的局部内存变量，并为它们赋初值逻辑假.F.。由于 LOCAL 与 LOCATE 前四个字母相同，所以这条命令的命令动词不能缩写。局部变量也要先建立后使用。
3、私有变量
在程序中直接使用（没有通过 PUBLIC 和 LOCAL 命令事先声明）而由系统自动隐含建立的变量都是私有变量。私有变量的作用域是建立它的模块及其下属的各层模块。一旦建立它的模块程序运行结束，这些私有变量将自动清除。

本章小结：

- 本章前 3 节内容通过大量的实例介绍了各种类型数据的常量、变量、表达式及一些常用的函数使用方法。关键是要掌握各不同类型数据的常量、变量之间使用的语法格式不同，表达式里数据类型是否无误、表达式的关系运算符的优先级顺序，使用函数时参数的选取、函数的返回值类型及使用场合。

- 后 3 节较全面地介绍了关于程序设计的一些内容，包括程序文件的建立调用，程序设计的基本结构，多模块程序以及变量的作用域等。其中关于循环结构和多模块程序的使用问题有大量的程序例题作为讲解使学生能更好地掌握其用法。有了良好的程序设计基础，在今后学习第六章章表单的时候才能知道怎么在表单里编写程序，因此掌握好这章的内容是学习第六章的基础。

作业：P89 习题二