普通高等教育医药类院校“十二五”规划教材　信息技术类

# Visual FoxPro 程序设计教程

王延红　肖　峰　主编

科学出版社

北京

## 内 容 简 介

本书依据全国计算机等级考试（二级）Visual FoxPro 程序设计的最新大纲及教育部高等学校非计算机专业基础课程教学指导委员会对计算机基础课程教学的基本要求，在多年的教学实践基础上编写。

为方便读者自学及教师教学，本书还配有《Visual FoxPro 程序设计实践教程》作为配套教材，书中还包含针对各章内容的实例、详细的操作步骤及习题。

本书可作为各类高等学校非计算机专业 Visual FoxPro 程序设计课程的教材，也适合作为全国计算机等级考试（二级）Visual FoxPro 程序设计科目的培训教材。

**图书在版编目（CIP）数据**

Visual FoxPro 程序设计教程/王延红，肖峰主编. —北京：科学出版社，2011
（普通高等教育医药类院校“十二五”规划教材·信息技术类）
ISBN 978-7-03-031482-6

Ⅰ. ①V… Ⅱ. ①王… ②肖… Ⅲ. ①关系数据库—数据库管理系统，Visual FoxPro—程序设计—高等教育—教材 Ⅳ. ①TP311.138

中国版本图书馆 CIP 数据核字（2011）第 109634 号

责任编辑：宋 丽 陈晓萍／责任校对：王万红
责任印制：吕春珉／封面设计：东方人华平面设计部

科学出版社出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com

新蕾印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2011年9月第 一 版 开本：787×1092 1/16
2011年9月第一次印刷 印张：20 1/4
印数：1–4 000 字数：474 000

**定价：34.00 元**

（如有印装质量问题，我社负责调换〈新蕾〉）
销售部电话 010-62142126 编辑部电话 010-62134021

# 目　录

前言

## 第 1 篇　Visual FoxPro 程序设计基础

第 1 章　Visual FoxPro 6.0 基础……3
1.1　数据类型……3
1.2　常量与变量……4
1.2.1　常量……4
1.2.2　变量……6
1.3　常用标准函数……9
1.3.1　数值函数……9
1.3.2　字符函数……12
1.3.3　日期和时间函数……14
1.3.4　数据转换函数……15
1.3.5　测试函数……17
1.4　表达式……20
1.4.1　数值表达式……20
1.4.2　字符表达式……21
1.4.3　日期时间表达式……21
1.4.4　关系表达式……21
1.4.5　逻辑表达式……24
1.4.6　名称表达式与宏替换……25
1.4.7　空值……25
1.4.8　运算的优先级……26
1.5　Visual FoxPro 6.0 使用基础……26
1.5.1　Visual FoxPro 6.0 的启动和退出……27
1.5.2　工作方式……27
1.5.3　命令格式……28
1.5.4　命令书写规则……29
思考题……29
第 2 章　数据表的操作……30
2.1　表的创建……30
2.1.1　表结构的设计……30
2.1.2　表的创建……31

2.2 表的基本操作……33
2.2.1 表的打开与关闭……33
2.2.2 表结构的显示与修改……34
2.2.3 表记录的显示……36
2.2.4 记录指针的定位……38
2.2.5 表记录的修改……39
2.2.6 表记录的添加……42
2.2.7 表记录的删除和恢复……44
2.2.8 表的复制……46
2.2.9 表与数组间的数据传递……47
2.3 表的排序与索引……49
2.3.1 排序……49
2.3.2 索引……50
2.4 表的查询……57
2.4.1 顺序查询……57
2.4.2 索引查询……57
2.5 表的统计与计算……59
2.5.1 统计记录个数……60
2.5.2 求和……60
2.5.3 求平均值……61
2.5.4 综合计算……61
2.5.5 分类汇总……62
2.6 多表操作……62
2.6.1 工作区……62
2.6.2 表间临时关联……64
2.6.3 表的联接……66
思考题……67
**第 3 章 数据库及其操作**……68
3.1 数据库基本操作……68
3.1.1 创建与打开数据库……68
3.1.2 数据库其他简单操作……71
3.2 数据库表的操作……73
3.2.1 数据库表与自由表……73
3.2.2 数据库表的其他操作……82
3.2.3 数据库表的索引……84
3.3 数据库多表操作……86
3.3.1 数据工作期的应用……86
3.3.2 创建数据库表间的永久关系……89

3.4　数据完整性……91
思考题……93
第 4 章　结构化程序设计……94
4.1　程序设计基础……94
4.1.1　程序的概念……94
4.1.2　程序文件的建立……95
4.1.3　程序文件的修改……96
4.1.4　程序文件的执行……97
4.1.5　程序的书写规则……98
4.1.6　程序的基本结构和算法……98
4.1.7　程序的调试……100
4.2　程序文件中的常用命令……101
4.2.1　程序中的辅助命令……101
4.2.2　非格式化输入和输出命令……102
4.2.3　格式化输入和输出命令……106
4.3　顺序结构程序设计……107
4.4　选择结构程序设计……108
4.4.1　简单分支结构……108
4.4.2　双分支选择语句……109
4.4.3　分支的嵌套……111
4.4.4　多分支选择语句……112
4.5　循环结构程序设计……113
4.5.1　DO WHILE 循环语句……113
4.5.2　FOR 循环语句……116
4.5.3　SCAN 循环语句……117
4.5.4　循环的嵌套……118
4.6　多模块程序设计……119
4.6.1　子程序和子程序调用……120
4.6.2　过程和过程文件……122
4.6.3　参数传递……126
4.6.4　自定义函数……129
4.6.5　变量的作用域……131
思考题……135
第 5 章　关系数据库标准语言 SQL……136
5.1　SQL 概述……136
5.2　SQL 的数据查询功能……137
5.2.1　SELECT 语法格式……137
5.2.2　简单查询……137

5.2.3 嵌套查询……140
5.2.4 联接查询……142
5.2.5 排序……144
5.2.6 计算查询与分组查询……145
5.2.7 集合的并运算……146
5.2.8 查询去向……147
5.3 SQL 的数据定义功能……148
5.3.1 表的创建……148
5.3.2 表的删除……149
5.3.3 表结构的修改……149
5.3.4 视图的定义……150
5.4 SQL 的数据操作功能……151
5.4.1 插入记录……151
5.4.2 更新记录……152
5.4.3 删除记录……152
思考题……152
**第 6 章 查询与视图**……153
6.1 查询……153
6.1.1 查询的概念……153
6.1.2 建立查询……153
6.1.3 使用查询……165
6.1.4 查询设计器的局限性……165
6.2 视图……166
6.2.1 视图的概念……166
6.2.2 创建本地视图……166
6.2.3 创建远程视图……172
6.2.4 使用视图……173
6.3 查询文件和视图的特点……173
6.3.1 查询文件与视图的区别……173
6.3.2 视图的优点……174
思考题……174
**第 7 章 表单设计与应用**……175
7.1 面向对象的概念……175
7.1.1 对象与类……175
7.1.2 子类与继承……176
7.2 Visual FoxPro 基类简介……176
7.2.1 Visual FoxPro 基类……176
7.2.2 容器类与控件类……177

7.3 创建与运行表单……179
7.3.1 用表单向导创建表单……179
7.3.2 用表单设计器创建表单……181
7.3.3 保存、运行和修改表单……182
7.4 表单设计器及应用……183
7.4.1 表单设计器……183
7.4.2 数据环境……187
7.5 属性、事件和方法……188
7.5.1 常用属性……188
7.5.2 表单及控件的常用事件和方法……188
7.5.3 为事件（或方法）编写代码……190
7.5.4 添加新的属性和方法……190
7.6 表单控件……191
7.6.1 控件的通用属性……191
7.6.2 常用控件的使用……191
思考题……207
**第 8 章 菜单的设计与应用**……208
8.1 Visual FoxPro 系统菜单……208
8.2 下拉式菜单设计……210
8.2.1 菜单设计的基本步骤……210
8.2.2 为顶层表单添加菜单……218
8.3 快捷菜单设计……219
思考题……221
**第 9 章 报表的设计与应用**……222
9.1 利用报表向导创建报表……222
9.1.1 报表的概念……222
9.1.2 报表向导……222
9.2 利用快速报表创建报表……227
9.3 利用报表设计器创建报表……229
9.3.1 报表设计器……229
9.3.2 报表设计工具……231
9.3.3 报表控件的使用……231
9.4 数据分组和多栏报表……235
9.4.1 设计分组报表……235
9.4.2 设计多栏报表……238
9.5 标签的设计……238
思考题……241

**第 10 章 Visual FoxPro 系统开发实例**……242
10.1 Visual FoxPro 系统开发经历的阶段及软件开发的系统组成……242
10.2 系统规划与主要功能模块设计……243
10.2.1 总体设计……243
10.2.2 主要功能模块……244
10.3 项目管理器的应用……248
10.3.1 用项目管理器组织学籍管理系统……248
10.3.2 “学籍管理系统”部件的组成……249
10.4 应用系统的主程序设计、项目连编及发行……250
10.4.1 应用系统的主程序设计……250
10.4.2 设置项目信息……252
10.4.3 应用系统的连编……252
10.4.4 应用系统的发布……253
思考题……254

## 第 2 篇 计算机公共基础知识

**第 11 章 数据库设计基础**……257
11.1 数据库系统的基础知识……257
11.1.1 数据库系统的基本概念……257
11.1.2 数据库系统的发展……258
11.1.3 数据库系统的内部结构体系……258
11.1.4 数据库系统的基本特点……259
11.2 数据模型……260
11.2.1 数据模型的基本概念……260
11.2.2 E-R 模型……260
11.2.3 逻辑数据模型……262
11.3 关系代数……264
11.3.1 关系代数的运算符……264
11.3.2 关系代数的运算……264
11.4 数据库设计与管理……266
思考题……267
**第 12 章 数据结构与算法**……268
12.1 算法……268
12.1.1 算法的基本概念……268
12.1.2 算法复杂度……270
12.2 数据结构……271
12.2.1 数据结构的基本概念……271
12.2.2 数据结构及其图形表示……271

12.3　线性表及顺序存储结构……273
12.3.1　线性表的基本概念……273
12.3.2　顺序表的运算……274
12.4　栈与队列……276
12.4.1　栈及其基本运算……276
12.4.2　队列及其基本运算……277
12.5　线性链表……279
12.6　树与二叉树……281
12.6.1　树的基本概念……281
12.6.2　二叉树及其性质……282
12.6.3　二叉树的存储结构……284
12.6.4　二叉树的遍历……285
12.7　查找技术……285
12.7.1　顺序查找……286
12.7.2　二分法查找……286
12.8　排序技术……286
12.8.1　交换排序法……287
12.8.2　插入排序法……289
12.8.3　选择排序法……290
思考题……292
**第 13 章　程序设计基础与软件工程**……293
13.1　程序设计方法与风格……293
13.2　结构化程序设计……294
13.3　面向对象的程序设计……295
13.4　软件工程的基本概念……296
13.4.1　软件的概念及特点……296
13.4.2　软件工程的基本概念……297
13.5　结构化分析方法……298
13.5.1　需求分析及其方法……298
13.5.2　结构化分析方法……299
13.5.3　软件需求规格说明书……300
13.6　结构化设计方法……301
13.7　软件测试和调试……303
13.7.1　软件测试的基本概念……303
13.7.2　软件测试的方法和技术……303
13.7.3　软件测试的实施……305
13.7.4　软件调试……306
思考题……307
**参考文献**……308

# 第 1 章　Visual FoxPro 6.0 基础

数据是对客观事物的符号表示。Visual FoxPro 的核心是对数据的处理，掌握数据知识，是学好 Visual FoxPro 的基础。根据计算机系统处理数据形式的不同，Visual FoxPro 将数据分为常量、变量、表达式和函数 4 种形式。常量和变量是数据处理的基本对象；表达式和函数不仅具有数据形式，还具有特定的数据处理功能。掌握 Visual FoxPro 的命令格式和语法规则是使用 Visual FoxPro 的基础。

本章将依次介绍数据的概念、数据的运算与应用以及 Visual FoxPro 的使用基础。

## 1.1　数 据 类 型

数据类型是数据的一种基本属性，表示数据所代表信息的类型，是数据分类的依据。每一个数据都有特定的类型，数据类型决定了数据的存储方式和运算规则。数据值是数据的具体表示，是参与运算的对象。在数据处理时，只有相同类型的数据才能直接运算，否则就会产生数据类型不匹配的错误。

Visual FoxPro 6.0 中定义的数据类型主要有以下几种。

### 1. 数值型

数值型（Numeric）用字母 N 表示，用于存储数据的值。它由数字 0～9、小数点、正负号构成。例如，+15、-2.8、0.618、1.23E+12 等。数值型数据占用 8 字节，其取值范围是-0.9999999999E+19～0.9999999999E+20。

如果对数据的性能和存储空间有特定的要求，Visual FoxPro 还提供了与数值型数据相兼容的数据类型，分别是浮点型（Float）、双精度型（Double）和整型（Integer）。

### 2. 字符型

字符型（Character）用字母 C 表示，由字母、数字、空格等任意的 ASCⅡ码以及汉字和非汉字图形符号（包括俄文字母、日文假名、制表符号等）构成。字符型数据的宽度为 0～254 字节。1 个 ASCⅡ码字符占用 1 字节，1 个汉字占用 2 字节。

如果将数字定义为字符型数据，数字就成了一种符号标识，不再有数学上的数量含义，也不能进行数学运算。如身份证号码和电话号码等，虽是数字形式，却只是一串失去数学意义的标识符号。

### 3. 日期型

日期型（Date）用字母 D 表示，用于存储由年、月、日构成的日期值。年、月、日之间用分隔符分隔，分隔符可以是斜杠（/）、连字号（-）、英文句点（.）。日期型数据的格式取决于 SET DATE、SET MARK TO 和 SET CENTURYD 等的设置。默认设置是美国格式 mm/dd/yy，其中 mm 表示月，dd 表示日，yy 表示年。日期型数据的宽度固定为 8 字节。

定不变。Visual FoxPro 支持的常量类型有数值型、货币型、字符型、逻辑型、日期型和日期时间型。不同数据类型的常量有不同的使用形式，这是各种类型常量的使用要求，也是其能被正确使用的保障。

1. 数值型常量

由数字（0～9）、小数点和正负号组成。使用形式为直接输入。例如，1.23，+999，-1000.88，1.234E10 等。

2. 货币型常量

货币型常量的构成与数值型常量相似。使用时要以前置符号$开头，不能用科学计数法表示。货币型常量在存储和计算时最多保留 4 位小数，多出的部分将被系统自动进行四舍五入处理。例如，$1234.56789 将被存储为$1234.5679。

【例 1.1】　显示几个货币型常量的值。

```
? $500
```

其中，?是输出命令。每条命令都应单独执行，在输入完命令后按 Enter 键确认即执行该命令。

```
? $1234.56789
```

3. 字符型常量

字符型常量也被称为字符串。由 ASCII 字符集中可打印字符和汉字等组成。使用时用定界符括起来。许多常量都有定界符，定界符虽然不是常量本身的内容，但它规定了常量的类型以及常量的起始和终止界限，是常量应用中必不可少的组成部分。字符型常量的定界符包括半角的双引号（" "）、单引号（' '）、方括号（[ ]），如"Visual FoxPro"，'123'，[计算机]等。

字符型常量的定界符必须成对匹配，不能一边用双引号而另一边用单引号。如果某种定界符本身也是字符串的内容，则需要用另一种定界符为此字符串的定界符。

【例 1.2】　显示：'Who is he ?'Anna asked。

```
? "'Who is he ?'Anna asked"
```

**注意**：不包含任何字符的字符串（""）称为空串。空串不是只包含空格的字符串（" "），空格串是有“空格”值的字符型数据。

4. 逻辑型常量

逻辑型常量只有两个值：逻辑真和逻辑假。应用形式是用定界符句点（.）将逻辑值括起来。逻辑型常量真值的表示形式是.T.；逻辑型常量假值的表示形式是.F.。在输入时，可以用.T.、.t.、.Y.或.y.来输入逻辑真（.T.），用.F.、.f.、.N.或.n.来输入逻辑假（.F.）。

在逻辑值前后作为定界符的两个句点是必不可少的，否则会被系统识别为变量名而造成错误。

5. 日期型常量

日期型常量一定要包括年、月、日 3 个值。每两个值之间由一个分隔符（如斜杆“/”）

隔开。书写格式是用定界符花括号（{}）将日期型常量括起来。

传统的日期格式是{mm/dd/yy}（月/日/年）。目的是与早期的 Visual FoxPro 兼容，在 Visual FoxPro 6.0 中使用时受到系统设置的影响。

严格的日期格式是{^yyyy/mm/dd}（年/月/日），花括号内第一个字符必须是脱字符（^），年份必须是 4 位，年月日次序不能颠倒，不能默认，否则会出错。空白的日期可表示为{}或{/}。严格日期格式是 Visual FoxPro 默认设置的日期格式，可以在任何情况下使用。

日期型常量的输出格式可以由用户自己设定。

（1）设置日期格式的命令方式

SET STRICTDATE TO [0|1|2]，用于设置是否对日期格式进行检测。

SET CENTURY ON|OFF|TO [<世纪值>[ROLLOVER<年份参照值>]]，用于设置显示日期型数据时是否显示世纪值。

SET DATE [TO] AMERICAN|ANSI|BRITISH|FRENCH|GERMAN|JAPAN|USA|ITALIAN|MDY|DMY|YMD|SHORT|LONG，用于设置日期型数据显示或输出的格式。

SET MARK TO [日期分隔符]，用于设置显示或输出日期型数据时所使用的分隔符。

**【例 1.3】** 日期设置示例。

```
SET STRICTDATE TO 1
* 设置严格的日期格式检测
SET CENTURY ON
* 设置日期型数据输出时显示世纪值，即年份用 4 位表示
SET DATE TO YMD      &&设置日期显示格式为 yy/mm/dd 形式，即年/月/日的形式
SET MARK TO "-"      &&设置日期型数据用"-"分隔
? {^2011/03/28}
```

显示结果为：2011-03-28

（2）设置日期格式的菜单方式

选择“工具”菜单中的“选项”命令，在弹出的“选项”对话框中选择“常规”选项卡，其中的“2000 年兼容性”栏可以设置“严格的日期级别”；在“区域”选项卡中的“日期和时间”区域中可以设置日期格式、日期分隔符以及年份的 4 位或 2 位显示。

6. 日期时间型常量

日期时间型常量包括日期和时间两部分内容，日期部分与日期型常量相似，也分为传统的和严格的两种格式。严格的日期时间型常量的格式为：{^yyyy/mm/dd[,][hh[:mm[:ss]][a|p]]}，其中[hh[:mm[:ss]][a|p]]为时间部分的格式，hh、mm 和 ss 分别代表时、分和秒，分隔符为冒号（:）；a 和 p 分别代表上午和下午，如果指定的小时大于等于 12，则自然是指下午的时间了。

### 1.2.2 变量

变量是在命令操作和程序运行过程中其值可变的量。其构成要素有 3 个：变量名、数据类型和变量值。每一个变量都可以通过变量名来访问变量值。Visual FoxPro 的变量

分为字段变量和内存变量两大类，其中内存变量又可细分为简单内存变量、数组变量和系统变量 3 类。

1. 简单内存变量

简单内存变量是一种独立于数据库存在的变量，通常简称为内存变量或变量。在维护和操作数据库时，经常需要一些内存单元来存放一些临时的数据或一些计算结果，这些临时开辟的内存单元就是内存变量。其数据类型有数值型、字符型、货币型、日期型、日期时间型、逻辑型。

（1）内存变量的命名

内存变量的命名规则如下。

1）由字母、汉字、数字或下划线组成，不允许有空格。

2）以字母、汉字或下划线开始。

3）长度为 1～254 个字符，每个汉字占两个字符。

4）避免使用 Visual FoxPro 6.0 的保留字作为内存变量名。

（2）内存变量的赋值

内存变量的赋值命令除了具有赋值功能之外，还同时具有定义内存变量和确定内存变量的数据类型的功能。也就是说，内存变量赋值之前不必定义，所赋值的数据类型就是内存变量的数据类型。Visual FoxPro 6.0 可以用如下两种命令格式来创建内存变量并为其赋值。

**【格式 1】**　<内存变量名>=<表达式>

**【格式 2】**　STORE <表达式> TO <内存变量名表>

**【说明】**　格式 1 一次只能给一个内存变量赋值；格式 2 一次可以将一个值同时赋给多个内存变量，各内存变量名之间必须用逗号隔开；可以通过对内存变量重新赋值来改变其内容和数据类型。

**【例 1.4】**　赋值示例。

```
X=.F.                  && 内存变量 X 的值为.F.，数据类型是逻辑型
STORE 2*3 TO Y,Z       && 内存变量 Y、Z 的值都是 6，数据类型是数值型
X=Y+Z                  && 内存变量 X 的值变为 12，数据类型是数值型
X="数据库"             && 内存变量 X 的值为数据库，数据类型是字符型
```

（3）内存变量的显示

**【格式】**　LIST|DISPLAY MEMORY [LIKE<通配符>]

**【功能】**　显示当前内存变量的信息，包括变量名、作用域、数据类型和当前值。

**【说明】**

1）LIST MEMORY 为滚屏显示所有内存中的变量信息，一直显示到结束；而 DISPLAY MEMORY 为分屏显示所有内存中的变量信息，显示满一屏后暂停，按键盘上任意键继续显示下一屏，以此类推，直至显示结束。

2）LIKE 子句为可选项，用于显示变量名与通配符相匹配的内存变量。通配符包括*和？，其中*表示任意多个字符，？表示任意一个字符。例如，LIST MEMORY LIKE a*，显示的就是以字母 a 开头的变量。

2. 数组变量

数组是内存中连续的一片存储区域，是由一组名字相同但下标不同的有序的内存变量组成的集合。其中每一个内存变量都是这个数组的一个元素，在内存中独占一个内存单元，可以给每个数组元素分别赋值。使用时通过数组名和下标来区分各个数组元素。一个数组中所含元素的个数是由数组的下标值决定的，而下标的个数决定了数组的维数，只有一个下标的数组叫做一维数组，有两个下标的数组叫做二维数组。数组在使用前要先定义数组的名字，并说明其维数。

（1）数组的定义

**【格式】** DIMENSION|DECLARE <数组名 1>(<下标 1>[,<下标 2>])[,<数组名 2>(<下标 1>[,<下标 2>])]...

**【功能】** 定义一个或多个一维或二维数组。

**【说明】**

1）DIMENSION 和 DECLARE 在功能上完全相同。

2）命名时除数组名最多不超过 10 个字符外，其他规则与内存变量完全相同。

3）数组的最小下标是 1，<下标 1>、<下标 2>用来指定数组的最大下标，缺省<下标 2>时定义的是一维数组，否则为二维数组。

4）数组中各元素的初始值默认为逻辑值.F.，同一个数组元素在不同时刻可以存放不同类型的数据。

5）可以用一维数组的形式访问二维数组。这是由于在内存中，二维数组元素是按行排列的。

6）在同一个运行环境下，数组名不能和简单内存变量名重名。

7）在赋值和输入语句中使用数组名时，表示将同一个值同时赋给该数组的全部数组元素。

（2）数组变量的赋值

对数组变量的赋值，可同样使用对内存变量赋值的两种命令格式。

**【例 1.5】** 数组应用示例。

```
DIMENSION X(7),Y(2,3)          && 定义一维数组 X，二维数组 Y
? Y(1,2)                       && 显示数组 Y 的第 2 个元素的初值
X(1)={^2011/03/15}             && 给数组 X 的第 1 个元素赋值
? X(1)                         && 显示数组 X 的第 1 个元素的值
? X                            && 数组变量不带下标时只显示第一个元素
X=9                            && 给数组 X 的全部元素赋值，此时可不带下标
? X(3),X(6)                    && 显示的结果都是 9
Store 10 to Y                  && 给数组 Y 的全部元素赋值，此时可不带下标
? Y(4)                         && 用一维数组的形式访问二维数组
```

3. 字段变量

字段变量是表中的数据项，是用户在创建表时建立的，而表是以文件的形式存储的，因此字段变量是永久性变量。字段变量简称字段。每个字段都有 3 个要素构成：字段名、

【功能】 返回<数值表达式>值的整数部分。例如：

```
? INT(12.8),INT(5-12.5)
```

显示结果为：12 -7

此外，函数 CEILING(<数值表达式>)的功能是返回大于或等于<数值表达式>值的最小整数，函数 FLOOR(<数值表达式>)的功能是返回小于或等于<数值表达式>值的最大整数。例如：

```
? CEILING(9.1),FLOOR(5.7)
```

显示结果为：10 5

2. 求余函数

【格式】 MOD(<数值表达式 1>,<数值表达式 2>)

【功能】 返回<数值表达式 1>的值除以<数值表达式 2>的值的余数。

【说明】 函数值的正负号与除数的正负号相同。函数的值分两种情况：①被除数与除数是同号，函数值即为两数相除的余数，即 MOD(X1,X2)=X1-INT(X1/X2)*X2；②被除数与除数是异号，则函数的值为两数相除的余数再加上除数的值，即 MOD(X1,X2)=X1-INT(X1/X2)*X2+X2。另外，本函数与%运算符等价。例如：

```
? MOD(9,4),MOD(-9,-4),MOD(-9,4),MOD(9,-4),9%4,9%(-4)
```

显示结果为：1 -1 3 -3 1 -3

3. 四舍五入函数

【格式】 ROUND(<数值表达式 1>,<数值表达式 2>)

【功能】 返回指定表达式在指定位置四舍五入后的结果。

【说明】 <数值表达式 1>用来指定被进行四舍五入处理的数据，<数值表达式 2>用来指定进行四舍五入的位置。如果<数值表达式 2>的值为非负数，表示按此值保留的小数位数；如果<数值表达式 2>的值为负数，表示按此值的绝对值在整数部分进行四舍五入运算的位数，也就是小数点左端包含零的个数；如果<数值表达式 2>的值为非整数，则先对<数值表达式 2>的值取整，之后按此值对<数值表达式 1>的值进行四舍五入运算。例如：

```
? ROUND(123.456,2),ROUND(123.456,-2),ROUND(123.456,1.8)
```

显示结果为：123.46 100 123.5

4. 求绝对值函数

【格式】 ABS(<数值型表达式>)

【功能】 返回<数值型表达式>值的绝对值。

例如：

```
? ABS(-1)
```

显示结果为：1

5. 求符号函数

【格式】 SIGN(<数值型表达式>)

来恢复默认状态。若在第一次使用此函数时，选择一个负值的<数值表达式>，以后将使用来自系统时钟的种子值。例如：

```
? rand()
```

显示结果为：0.85

此值为初次使用此函数返回的值，每次均为随机数。

### 1.3.2 字符函数

字符函数一般是指自变量是字符的函数。

1. 求字符串长度函数

【格式】 LEN(<字符型表达式>)

【功能】 返回字符串的长度，即所包含的字符个数。

【说明】 函数值为数值型。例如：

```
CLENG="Visual FoxPro 程序设计教程"
? LEN(CLENG)
```

显示结果为：25

2. 取子串函数

【格式】 SUBSTR(<字符串表达式>,<数值表达式 1>,[<数值表达式 2>])

【功能】 函数从<字符串表达式>的值中取出一子串。

【说明】 <数值表达式 1>规定了子串的在<字符表达式>值中的开始位置，<数值表达式 2>规定了子串的长度。当<数值表达式 1>的值大于<字符串表达式>值的长度时，得一个空串；当<数值表达式 2>的值大于<字符串表达式>值的长度或者省略此项时，则取到字符串结束为止。例如：

```
S="Visual FoxPro 程序设计教程"
? SUBSTR(S,8,6),SUBSTR(S,23)
```

显示结果为：FoxPro 教程

此外，函数 RIGHT(<字符串表达式>,<数值表达式>)的功能是从<字符串表达式>值的尾部（右侧）取一个子串，子串长度由<数值表达式>决定。函数 LEFT(<字符串表达式>,<数值表达式>)的功能是从<字符串表达式>值的前部（左侧）取一个子串，子串长度由<数值表达式>决定。

3. 求子串位置函数

【格式】 AT(<字符串表达式 1>,<字符串表达式 2>)

【功能】 返回<字符串表达式 1>在<字符串表达式 2>中第一次出现的起始位置值。

【说明】 函数值为整数。若不存在，函数值为零。ATC()与 AT()的用法基本相同，唯一的不同是：ATC()搜索时不区分大小写，而 AT()是严格区分大小写的。例如：

```
? AT("Do","Easy does it."),ATC("Do","Easy does it."),AT("医学院","全科医学")
```

显示结果为：0 6 0

4. 生成空格串函数

【格式】　SPACE(<数值表达式>)

【功能】　返回一个完全由空格组成的字符串。

【说明】　空格个数由<数值表达式>确定。例如：

```
STORE SPACE(15) TO X
? LEN(X)
```

显示结果为：15

5. 删除字符串前后空格函数

【格式】　LTRIM(<字符串表达式>)
RTRIM|TRIM(<字符串表达式>)
ALLTRIM(<字符串表达式>)

【功能】　LTRIM()的功能是删除字符串的前导空格；RTRIM()和 TRIM()的功能是删除字符串的尾部空格；ALLTRIM()的功能是删除字符串的前导和尾部空格。

【说明】　RTRIM()与 TRIM()相同，ALLTRIM()兼具前两者的功能。例如：

```
c=space(4)+"see"+space(4)
?"I"+ltrim(c)+"!"
?"I"+trim(c)+"!"
?"I"+alltrim(c)+"!"
```

显示结果为：

```
Isee    !
I   see!
Isee!
```

6. 大小写字母转换函数

【格式】　UPPER(<字符表达式>)
LOWER(<字符表达式>)

【功能】　UPPER()的功能是将<字符表达式>值中的小写字母全部转换成大写字母；LOWER()的功能是将<字符表达式>值中的大写字母全部转换成小写字母。

【说明】　只对字母进行大小写转换，其他字符保持不变。例如：

```
? LOWER("Visual FoxPro"),UPPER("Visual FoxPro")
```

显示结果为：visual foxpro　　VISUAL FOXPRO

7. 字符复制函数

【格式】　REPLICATE(<字符表达式>,<数值表达式>)

【功能】　复制<字符表达式>的值若干次。

【说明】　复制的次数由<数值表达式>的值确定。例如：

```
? REPLICATE("*",3)
```

显示结果为：***