第3章 Visual FoxPro 数据与数据运算

在进行数据处理时，除了需要表中的数据外，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，Visual FoxPro 有常量、变量、表达式和函数 4 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程序设计语言对数据进行运算和处理的能力与功能。本章将详细介绍这些程序设计基础要素。

3.1 常量与变量

每个数据都有自己所属的数据类型(分类)，数据类型决定了数据的存储方式和运算方式。每种类型的数据均有常量和变量之分。例如，在学生成绩处理系统中所处理的表 Student 如图 3-1 所示。该表中既有常量又有定义表结构的变量，它们所属的数据类型是不同的。

Student

学号	姓名	性别	年龄	籍贯	出生日期	专业	入学成绩
201001	王海	男	19	山东	02/03/90	网络工程	520.0
201002	李丽	女	20	山西	10/24/89	计算机技术	498.0
201003	刘立峰	男	20	河南	12/12/91	计算机应用	543.0
201004	张丽	女	19	山东	08/07/90	计算机技术	487.0
201005	赵毅	男	20	山东	01/12/89	计算机科学与技术	500.0
201006	赵子博	男	20	湖南	05/07/90	计算机科学与技术	512.0
201007	王伟真	男	20	新疆	06/12/89	网络工程	476.0

图 3-1 表 Student

表中的记录包括学号、姓名、性别、出生日期和入学成绩等字段。在向表中输入数据时，每个字段的数据类型是在表结构中定义的。Visual FoxPro 提供了许多数据类型，常用的数据类型有数值型(N)、字符型(C)、逻辑型(L)、货币型(Y)、日期型(D)和日期时间型(T)等十几种。

3.1.1 常量

常量是指在程序运行过程中其值不发生变化的量。常量用于表示一个具体的、不变的值，不同类型的常量采用不同的书写格式。Visual FoxPro 常量根据其值可分为数值型常量、字符型常量、逻辑型常量、货币型常量和日期(时间)型常量等。

1. 数值型常量

数值型常量也就是常数，用来表示数量的大小。数值型常量由数字 0～9、小数点和正负号构成，例如 19、520.3、－12 等。有些很大或很小的数值型常量也可以使用科学记数法形式书写。例如，用 8.234E12 表示 8.234×10^{12}，用 2.3E－12 表示 2.3×10^{-12}。

数值型数据在内存中占 8 个字节，取值范围是－0.9999999999E＋19～0.9999999999E＋20。

在 Visual FoxPro 中，具有数值特征的数据类型还有整型(Integer)、浮点型(Float)和双精度型(Double)，不过这 3 种数据类型只能用于字段变量。

2. 货币型常量

货币型常量用来表示货币值，其书写格式与数值型常量类似，但要在数字前加上货币符号($)。货币数据在存储和计算时采用 4 位小数，如果一个货币型常量多于 4 位小数，则系统会自动将多余的小数四舍五入。例如，常量 $369.875 678 9 存储为 $369.8757。

货币型数据用字母 Y 标识类型，货币型常量占据 8B 存储空间，取值范围是－922 337 203 685 477.5807～922 337 203 685 477.5807。货币型常量没有科学记数法形式。

3. 字符型常量

字符型(C)数据是不能进行算术运算的文字数据。字符型常量也称字符串，其表示方法是用半角单引号('')、双引号("")或方括号([])把字符(包括中文字符、英文字符、数字字符和其他 ASCII 字符)括起来，这里的单引号('')、双引号("")或方括号([])称为定界符，其作用是确定字符串的起始和终止界限，它本身不作为字符串的一部分。如"网络工程"。

不包含任何字符的字符串("")叫空串；空串与包含空格的字符串(" ")不同。一个英文字符或数字占 1B 内存，一个汉字占用 2B。字符型数据的长度范围是 0～254 个字符。

字符型常量的定界符必须成对匹配，不能一边用单引号而另一边用双引号。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该字符串定界，如["计算机"]。

例 3.1 显示几个字符型常量。

在命令窗口输入以下两条命令。

```
?"计算机",'等级',[考试],['三级' "数据库"]          && 显示几个字符型常量的值
??"工作",'字符串'," ",[实验方法]
```

其中，单问号(?)命令的功能是在下一行显示若干个表达式的值，表达式之间用逗号分开。双问号(??)命令的功能是在同一行显示表达式的值。分别按回车键执行以上命令后，在主窗口中的显示结果如下：

```
计算机 等级 考试 '三级' "数据库" 工作 字符串 实验方法
```

4. 日期型常量

日期型常量是用一对花括号({})将日期型数据括起来。花括号内包括年、月、日 3 部

分内容，各部分内容之间用分隔符分隔。分隔符可以是斜杠(/)、连字符(-)、句点(.)和空格。其中，斜杠(/)是系统在显示日期型数据时使用的默认分隔符，如图 3-1 中出生日期一栏所示。

1) 日期型常量的格式

日期型常量的格式有两种：

(1) 传统的日期格式。传统日期格式中的月、日各为 2 位数字，而年份可以是 2 位数字，也可以是 4 位数字。例如，{10/01/13}、{10-01-13}、{10 09 2013}等。

系统默认的日期型数据为美国日期格式 mm/dd/yy(月/日/年)，这种格式的日期型常量要受命令 SET DATE TO 和 SET CENTURY TO 设置的影响。即在不同的设置状态下，计算机会对同一个日期型常量做出不同的解释。例如，{10/09/01}可以被解释为 2001 年 10 月 9 日、2001 年 9 月 10 日、2010 年 9 月 1 日。

(2) 严格的日期格式

{^yyyy-mm-dd}，用这种格式书写的日期常量能表达一个确切的日期，它不受 SET DATE 等语句的影响。这种格式的日期常量在书写时要注意：花括号内第一个字符必须是脱字符，年份必须用 4 位，年、月、日的次序不能颠倒、不能省略。日期型数据在内存中用 8 个字节表示，取值范围是{^0001-01-01}至{^9999-12-31}。

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。

2) 影响日期格式的设置命令

(1) 命令格式：

```
SET MARK TO [<日期分隔符>]
```

命令功能：用于设置显示日期型数据时使用的分隔符。如果执行 SET MARK TO 没有指定任何分隔符，则表示恢复系统默认的斜杠分隔符。

(2) 命令格式：

```
SET DATE [TO] AMERICAN|ANSI|BRITISH|FRENCH|GERMAN|ITALIAN| JAPAN|USA|MDY|DMY|YMD
```

命令功能：设置日期显示的格式。命令中各个短语所表示的日期格式如表 3-1 所示。

表 3-1　常用日期格式

短　语	格　式	短　语	格　式
AMERICAN	mm/dd/yy	ANSI	yy. mm. dd
BRITISH/FRENCH	dd/mm/yy	GERMAN	dd. mm. yy
ITALIAN	dd-mm-yy	JAPAN	yy/mm/dd
USA	mm-dd-yy	MDY	mm/dd/yy
DMY	dd/mm/yy	YMD	yy/mm/dd

(3) 命令格式：

```
SET CENTURY ON|OFF [世纪值] ROLLOVER [年份参照值]
```

命令功能：用于设置显示日期型数据时是否显示世纪。OFF 选项确定用两位数字表示年份所处的世纪，即如果该日期的两位数字年份大于或等于“年份参照值”，则它所处的世纪即为“世纪值”，否则为“世纪值＋1”。

(4) 命令格式：

```
SET STRICTDATE TO [0|1|2]
```

命令功能：用于设置是否对日期格式进行检查。

0 表示不进行严格的日期格式检查。

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，对 CTOD()和 CTOT()函数的格式也有效。

例 3.2 设置不同的日期格式。在命令窗口中输入如下 4 条命令，并分别执行。

```
SET CENTURY ON                && 设置 4 位数据年份
SET MARK TO                   && 恢复系统默认的斜杠日期分隔符
SET DATE TO YMD               && 设置年月日格式
?{^2013-09-05}
```

分别按回车键执行以上命令后，在主窗口中的显示结果为

```
2013/09/05
```

再在命令窗口中输入如下 4 条命令，并分别执行。

```
SET CENTURY OFF               && 设置 2 位数字年份
SET MARK TO "."               && 设置日期分隔符为西文句点
SET DATE TO MDY               && 设置年月日格式
?{^2013-09-05}
```

在主窗口中的显示结果为

```
09.05.13
```

接着在命令窗口中输入如下两条命令，并分别执行。

```
SET STRICTDATE TO 0
?{^2008-01-05},{08.01.05}
```

在主窗口中的显示结果为

```
01.05.08  01.05.08
```

5. 日期时间型常量

在保存日期、时间或二者兼有时，可使用日期时间数据类型。日期时间型常量包括日期和时间两部分：{＜日期＞,＜时间＞}。＜日期＞部分与日期型常量相似，也有传统而

严格的格式。

＜时间＞部分的格式为

```
[hh[:mm[:ss]AM|PM]]
```

其中，hh、mm 和 ss 分别代表时、分和秒，默认值分别为 12、0 和 0，AM(或 a)和 PM(或 p)分别代表上午和下午。系统默认的格式为 AM，如果指定的时间大于 12，就自然为下午时间。

日期时间型数据用 8B 存储，前 4B 保存日期，后 4B 保存时间。日期部分的取值范围与日期型数据相同，时间部分的取值范围是 00:00:00AM 至 11:59:59PM。

日期时间值可以包含完整的日期和时间，也可以只包含其中之一。如果省略日期值，则 Visual FoxPro 用系统默认值 1899 年 12 月 30 日填入；如果省略时间值，则 Visual FoxPro 用系统默认的午夜零点时间。

【试一试】 在命令窗口中输入如下命令，结果如何？

```
SET MARK TO
?{^2013-09-05,11:30 p},{^2012-01-01,},{^2010-10-1,3}
```

6. 逻辑型常量

逻辑型(logic)数据是描述客观事物真假的数据类型，表示逻辑判断的结果，用字母 L 表示。逻辑型数据只有逻辑真(.T.、.t.、.Y.和.y.)和逻辑假(.F.、.f.、.N.和.n.)两个值。逻辑值前后两个黑点作为逻辑型常量的定界符是必不可少的，否则会被误认为变量名。逻辑型数据在存储时只占用 1B 的内存空间。

【知识扩展】 用户可根据编程的需要声明符号常量，定义常量的语句的一般格式为

```
#DEFINE 常量名 表达式
```

例如：

```
#DEFINE PI 3.1415926
```

其中，“常量名”为要定义的常量，一般为了与变量名相区别，常量名一般用大写字母表示。“表达式”是常量所代表的数据内容，即在程序运行时实际使用的值。在此定义后，程序中的符号常量 PI 都用 3.141 592 6 来表示。注意：此语句只能在程序方式下运行和使用。

3.1.2 变量

变量是指在程序运行过程中其值可以发生变化的量。每个变量都有一个变量名，变量名的命名规则是：变量名以字母、汉字或下划线开头，后面可以包含字母、汉字、数字或下划线，例如，姓名、gz、x、y、z 都是合法变量名。

注意：变量名不能是 Visual FoxPro 保留字，变量名中字母不区分大小写，变量名总长度不超过 128 个字符。

Visual FoxPro 变量可以分为内存变量、数组变量和系统变量等，下面展开讨论。

1. 内存变量

内存变量是内存中的一些临时工作单元，用于存放程序或命令执行过程中的输入数据、输出数据和中间数据。内存变量的类型取决于存放在其中的数据的类型，一旦退出系统或关机，内存变量就会消失。给内存变量赋值常用以下两种格式。

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＞

例如：

```
date={^2012/12/12}              && 给变量 date 赋日期型数据{^2012/12/12}
STORE 12 TO a,b                 && 给变量 a 和 b 赋数值型数据 12
STORE "北京" TO ch              && 给变量 ch 赋字符型数据"北京"
```

关于给内存变量赋值的说明如下：

(1) 等号一次只能给一个内存变量赋值。STORE 命令可以同时给多个变量赋予相同的值，此时各个内存变量名之间应用逗号分隔开。

(2) 一个内存变量在使用之前并不需要特别声明。当用 STORE 命令给内存变量赋值时，如果该变量不存在，那么系统会自动建立它。

(3) 如果要改变内存变量的内容和类型，可以通过对内存变量重新赋值来完成。

在 Visual FoxPro 中，可以把不同类型的数据赋给同一个内存变量，内存变量的类型取决于所赋值的类型。

当内存变量与当前表中的字段变量同名时，使用时须在内存变量前面加“M.”或者“M->”，即

```
M.内存变量名   或   M->内存变量名
```

否则系统将访问同名的字段变量。

2. 数组变量

数组变量简称数组，是按一定顺序排列的一组内存变量的集合。它由一系列元素组成，每个数组元素可以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内存变量，各个数组元素可以分别赋值。在 Visual FoxPro 中，一个数组中各个元素的数据类型可以相同也可以不同。

与简单内存变量不同，数组在使用之前一般要用 DIMENSION 或 DECLARE 命令显式创建，并且规定数组是一维数组还是二维数组，以及数组名和数组大小。数组大小由下标值的上、下限决定，下限规定为 1。

创建数组的命令格式如下：

```
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值.F.。

例 3.3 DIMENSION a(3),b(2,3)命令一次定义了两个数组。

一维数组 a 含 3 个元素：a(1)、a(2)、a(3)。

二维数组 b 含 6 个元素：b(1,1)、b(1,2)、b(1,3)、b(2,1)、b(2,2)、b(2,3)。

例 3.4 数组的定义及赋值。

```
DIMENSION a(3),b(4,3)                    && 定义两个数组
a=10                                     && 将 a 数组所有元素赋值为 10
b(3,1)=a(1)                              && 引用 a 数组的元素给 b 数组的元素赋值
STORE .T. TO b(3,1)                      && 给 b 数组的一个元素重复赋逻辑值
b(2,2)='上海大学'                        && 给 b 数组的一个元素赋字符值"上海大学"
b(4,1)={^2013/01/16}                     && 给 b 数组的一个元素赋日期值
b(2,3)={^2013-01-16 10:00:00 a}          && 给 b 数组的一个元素赋严格的日期时间值
```

在使用数组和数组元素时，应注意如下问题。

(1) 在一切可以使用简单内存变量的地方，均可以使用数组元素。

(2) 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。

(3) 在同一个运行环境下，数组名不能与简单变量名重复。

(4) 在赋值语句中的表达式位置不能出现数组名，可以出现具体的数组元素名。

(5) 可以用一维数组的形式访问二维数组。例如，例 3.3 中的数组 b 中的各元素可以用一维数组的形式依次表示为 b(1)、b(2)、b(3)、b(4)、b(5)、b(6)，其中 b(5)与 b(2,2)是同一变量。

3. 系统变量

系统变量是 Visual FoxPro 自动创建并维护的内存变量，用于控制 Visual FoxPro 的输出和显示的格式。为了和普通内存变量加以区别，系统变量名以下划线“_”开头。常用的系统变量如表 3-2 所示。

表 3-2 常用的系统变量的类型及应用说明

系统变量	变 量 说 明
_SCREEN	指定 Visual FoxPro 主窗口的属性
_VFP	指向当前运行的 Visual FoxPro 应用程序对象
_PAGETOTAL	在报表中设置“共 Y 页第 X 页”页数
_TEXT	可以把文本合并命令的结果输出到低级文件中
_DIARYDATE	默认存储当前日期
_SHELL	Visual FoxPro 在执行程序时阻止访问命令窗口

3.1.3 内存变量的常用操作命令

1. 表达式值的显示

格式 1：?[<表达式表>][AT <数值表达式>]

格式 2：??[<表达式表>][AT <数值表达式>]

功能：先计算表达式表中各表达式的值，再把结果显示在屏幕上；若无表达式表，则

输出一个空行。[AT<数值表达式>]项表示输出的起始列号。

例如：

```
x=23
y=26
c="计算机程序设计课"
?x,y,c
?"x+y=",x+y
```

输出结果为

```
23,26,计算机程序设计课
x+y=49
```

说明：? 是换行输出，不管有没有指定表达式的情况下都会输出一个回车符。?? 命令中各表达式值紧接在当前行的光标所在处直接输出，不会输出一个回车符。

2. 显示内存变量

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>]

功能：显示内存变量的当前信息，包括变量名、作用域、类型和取值。

- 选用 LIKE 短语只显示与通配符相匹配的内存变量，通配符包括“*”和“?”。“*”表示任意多个字符，“?”表示任意一个字符。
- 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。DISPLAY 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，按任意键后就可以继续显示下一屏。
- 可选子句 TO PRINTER 或 TO FILE <文件名>用于在显示的同时送往打印机，或者存入给定文件名的文本文件中，文件的扩展名为.txt。

3. 清除内存变量

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 清除所有的内存变量，在人机会话状态其作用与格式 1 相同。如果出现在程序中，则应加上短语 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。

例 3.5 释放内存变量。

```
RELEASE y1,y2              && 释放内存变量 y1 和 y2
RELEASE ALL LIKE B*        && 释放所有以字母 B 开头的内存变量
```

4. 表中数据与数组数据之间的交换

数组是把一大批数据组织在一起的数据处理方法，而表文件的数据内容是以记录的方式存储和使用的。为了使它们之间方便地进行数据交换，Visual FoxPro 提供了表和数组相互之间数据传递的功能，可以方便地完成表记录与内存变量之间的数据交换。

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]

功能：格式 1 的功能是将表的当前记录从指定的字段名表中的第一个字段的内容开始，依次复制到数组名中的从第一个数组元素开始的内存变量中。如果不使用 FIELDS 短语指定字段，则复制除备注型(M)和通用型(G)之外的全部字段。

如果事先没有创建数组，系统将自动创建；如果已创建的数组元素个数少于字段数，系统自动建立其余数组元素；如果已创建的数组元素个数多于字段数，其余数组元素的值保持不变。

如果选用 MEMO 短语，则同时复制备注型字段。如果选用 BLANK 短语，则产生一个空数组，各数组元素的类型和大小与表中当前记录的对应字段相同。

格式 2 的功能是用通配符指定包括或排除的字段。FIELDS LIKE <通配符>和 FIELDS EXCEPT <通配符>可以同时使用。

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> | FIELDS EXCEPT <通配符>] [MEMO]

功能：格式 1 的功能是将数组中的数据作为一个记录复制到表的当前记录中。从第一个数组元素开始，依次向字段名表指定的字段填写数据。如果省略 FIELDS 选项，则依次向各个字段复制；如果数组元素个数多于记录中字段的个数，则多余部分被忽略。

如果选用 MEMO 短语，则在复制时包括备注型字段，否则备注型字段不予考虑。

格式 2 的功能是用通配符指定包括或排除的字段。FIELDS LIKE <通配符>和 FIELDS EXCEPT <通配符>可以同时使用。

3.2 表 达 式

在 Visual FoxPro 中，表达式是由常量、变量和函数通过特定的运算符连接起来的式子。其形式包括：单一的运算对象(如常量、变量和函数)和由运算符将运算对象连接起

来形成的式子。

根据表达式值的类型，表达式可以分为数值表达式、字符表达式、日期时间表达式、关系表达式和逻辑表达式。

表达式中的变量、对象和函数参数等都需要名称。Visual FoxPro 中的命名规则如下：

(1) 只能以字母、汉字或下划线开头。

(2) 由字母、数字、汉字和下划线组成。

(3) 名字长度可以取 1～128 个字符，但自由表的字段名和索引名最多只能是 10 个字符。

命名时要避免使用 Visual FoxPro 中的保留字，尽量做到见名知意。

3.2.1 数值、字符型与日期时间表达式

1. 数值表达式

数值表达式又叫做算术表达式。数值表达式是由算术运算符将数值型数据连接起来形成的，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、变量或函数。

1) 算术运算符

算术运算符有＋、－、＊、/、＊＊(或^)、％以及()。算术运算符及其含义和优先级如表 3-3 所示。

表 3-3 算术运算符及优先级

优先级	运 算 符	说 明	优先级	运 算 符	说 明
1	()	形成表达式内的子表达式	3	＊、/、％	乘、除、求余运算
2	＊＊或^	乘方运算	4	＋、－	加、减运算

例 3.6 算术运算符的使用。

```
?2+3                              && 结果为 5
?(2*3+SQRT(36)/3)/2               && 结果为 4
```

2) 求余运算

求余运算％和取余函数 MOD()的作用相同。余数的正负号与除数一致。如果被除数与除数同号，那么函数即为两数相除的余数；如果被除数与除数异号，则余数为两数相除的余数加上除数的值。当表达式中出现乘、除和求余运算时，它们具有相同的优先级。

例 3.7 求余运算。

```
?15%4  15%-4                      && 结果为 3 -1
```

2. 字符型表达式

字符型表达式由字符串连接运算符将字符型常量、变量或者函数连接起来形成，其运算结果仍然是一个字符型数据。字符串连接运算符有以下两个，它们的优先级相同。

＋：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。

例 3.8 字符型表达式运算。

```
? "计算机 "+ "实验室"          && 结果为"计算机 实验室"
? "计算机 "- "实验室"          && 结果为"计算机实验室 "
```

3. 日期时间表达式

在 Visual FoxPro 中，日期时间表达式中可以使用的运算符也有＋和－两个。日期时间表达式的格式有一定的限制，不能任意组合。合法的日期时间表达式格式见表 3-4，其中＜天数＞和＜秒数＞都是数值表达式。

表 3-4 日期时间表达式的格式

格 式	结果及类型
＜日期＞＋＜天数＞	日期型。指定若干天后的日期
＜天数＞＋＜日期＞	日期型。指定若干天后的日期
＜日期＞－＜天数＞	日期型。指定若干天前的日期
＜日期＞－＜日期＞	数值型。两个指定日期相差的天数
＜日期时间＞＋＜秒数＞	日期时间型。指定日期时间若干秒后的日期时间
＜秒数＞＋＜日期时间＞	日期时间型。指定日期时间若干秒后的日期时间
＜日期时间＞－＜秒数＞	日期时间型。指定日期时间若干秒前的日期时间
＜日期时间＞－＜日期时间＞	数值型。两个指定日期时间相差的秒数

例 3.9 日期时间运算。

```
?{08/10/06}+1                  && 结果为{08/11/06}
?{08/15/06}-{08/02/06}         && 结果为 13
```

符号＋和－既可以作为日期时间运算符，也可以作为算术运算符和字符串连接运算符。到底作为哪种运算符使用，要根据它们所连接的运算对象的数据类型而定。

3.2.2 关系表达式

1. 关系表达式

关系表达式又称为简单逻辑表达式，它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>
```

关系运算符的作用是比较两个表达式的大小或进行子串包含测试。其运算结果是逻辑型数据。Visual FoxPro 共有 8 种关系运算符。关系运算符及其含义如表 3-5 所示，它们的优先级相同。

表 3-5　关系运算符及其优先级

运 算 符	说　明	运 算 符	说　明
<	小于	<=	小于或等于
>	大于	>=	大于或等于
=	等于	==	字符串精确比较
<>、#或！=	不等于	$	子串包含测试

说明：运算符==和$仅适用于字符型数据，其他运算符适用于任何类型的数据，但前后两个运算对象的数据类型要一致。运算符==用于两个字符串的精确比较，包括空格字符。当使用运算符==时，SET EXACT 命令将被忽略。

1）数值型数据和货币型数据比较

这两种数据按数值的大小进行比较，包括负号。例如：3>-3，$14<$24。

2）日期型数据和日期时间型数据比较

越早的日期或时间越小，越晚的日期或时间越大。例如：{^2008-08-22}>{^2007-01-12}。

3）逻辑型数据比较

.T.大于.F.。

4）子串包含

关系表达式<字符型表达式 1>$<字符型表达式 2>为子串包含测试，如果前者是后者的一个子字符串，则结果为逻辑真，否则为逻辑假。

例 3.10　子串包含测试。

```
STORE "计算机" TO x1
STORE "多媒体计算机" TO x2
?x1$ x2,x2$ x1                    && 结果为.T.和.F.
```

2. 设置字符的排序次序

比较两个字符串时，系统会对两个字符串的字符从左向右逐个进行比较，当发现两个对应字符不同时，就根据这两个字符的排序顺序决定两个字符串的大小。

设置字符比较次序的命令是

```
SET COLLATE TO "<排序次序名>"
```

其中，排序次序名必须放在引号当中。次序名可以是 Machine、PinYin 或者 Stroke。

Machine 为机器次序，即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，大写字母序列 A～Z 排在小写字母序列 a～z 的前面。因此，大写字母小于小写字母。汉字的机内码与汉字国标码一致。对常用的一级汉字而言，根据它们的拼音顺序决定大小。

PinYin 为拼音次序，即按照拼音次序排序。对于英文字符而言，空格在最前面，小写字母序列 a～z 在前，大写字母序列 A～Z 在后。

Stroke 为笔画次序，即无论中文还是英文，都按照书写笔画的多少排序。

3. 字符串精确比较与 EXACT 设置

当用单等号运算符=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。该命令可以在命令窗口或在程序中执行，也可以通过“数据”选项卡设置。

当处于 ON 状态时，字符串的比较运算将进行到两个字符串全部结束为止，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。

当处于 OFF 状态时，字符串的比较以右边的字符串为目标，右字符串结束即终止比较。只要右边的字符串与左边字符串的前面部分相匹配，即可以得到逻辑真的结果。即字符串比较因右面的字符串结束而终止。

当用双等号运算符＝＝比较两个字符串时，只有当两个字符串完全相同时，运算结果才会是逻辑真(.T.)，否则为逻辑假(.F.)。运算符＝＝对两个字符串的运算结果与 SET EXACT 设置无关。

SET EXACT 设置对字符串的影响，如表 3-6 所示。表中 TRIM()函数的功能是去掉字符串尾部空格。

表 3-6　SET EXACT 对字符串比较的影响

比　较	=(EXACT OFF)	=(EXACT ON)	==(EXACT ON 或 OFF)
"abc"="abc"	.T.	.T.	.T.
"ab"="abc"	.F.	.F.	.F.
"abc"="ab"	.T.	.F.	.F.
"abc"="ab "	.F.	.F.	.F.
"ab"="ab "	.F.	.T.	.F.
"ab"="ab"	.T.	.T.	.F.
" "="ab"	.F.	.F.	.F.
"ab"=" "	.T.	.F.	.F.
TRIM("ab ")="ab"	.T.	.T.	.T.
"ab"=TRIM("ab ")	.T.	.T.	.T.

3.2.3　逻辑表达式

逻辑表达式是由逻辑运算符将逻辑型数据连接而形成的，其运算结果仍然是逻辑型数据。逻辑型运算符有 3 个：.NOT.或!(逻辑非)、.AND.(逻辑与)和.OR.(逻辑或)。其优先级顺序依次为.NOT.、.AND.、.OR.。

逻辑运算符的运算规则如表 3-7 所示，＜LE1＞和＜ LE2＞分别代表两个逻辑型数据。

表 3-7　逻辑运算规则

＜LE1＞	＜LE2＞	.NOT.＜LE1＞	＜LE1＞.AND. ＜LE2＞	＜LE1＞.OR. ＜LE2＞
.T.	.T.	.F.	.T.	.T.
.T.	.F.	.F.	.F.	.T.
.F.	.T.	.T.	.F.	.T.
.F.	.F.	.T.	.F.	.F.

3.2.4 运算符优先级

当一个表达式包含多种运算符时，其运算的优先级由高到低排列为

算术运算符→字符运算符→日期运算符→关系运算符→逻辑运算符

圆括号作为运算符，其优先级最高，圆括号可以嵌套。圆括号可以改变表达式的运算顺序。有时在表达式的适当地方插入圆括号，可以提高表达式的可读性。

例 3.11 运算符优先级。

```
?8>3.AND."男">"男生"OR.T.<.F.                    && 结果为.F.
?(15%3=0)AND(24%5=4)OR"学习"!="工作"             && 结果为.T.
```

3.3 常 用 函 数

函数是用程序来实现的一种特定的功能。每个函数一般需要若干个运算对象，即自变量，但只有一个函数返回值。Visual FoxPro 函数根据使用特点可分为系统函数和自定义函数两大类。

系统函数是系统内部“编制”好的一段程序。Visual FoxPro 6.0 提供了 400 多个系统函数（也称为标准函数或内部函数），用户可以直接使用，并不需要知道函数内部究竟是如何工作的，只需要正确地按照“函数名（实参）”的形式调用即可。本节重点介绍常用的几类标准函数，需要程序员先定义才能使用的自定义函数将在第 7 章讨论。

3.3.1 数值函数

数值函数是指函数值为数值的一类函数，其自变量和返回值往往都是数值型数据。

1. 绝对值函数

格式：ABS(＜数值表达式＞)

功能：计算并返回自变量的绝对值。

例 3.12

```
STORE 30 TO x
?ABS(20-x),ABS(x-20)                             && 结果为 10   10
? ABS(64-2*50)                                   && 结果为 36
```

2. 符号函数

格式：SIGN(＜数值表达式＞)

功能：返回指定数值表达式的符号。当表达式的运算结果为正、负和 0 时，函数值分别为 1、－1 和 0。

例 3.13

```
? SIGN(1),SIGN(-3),SIGN(2-2)                     && 结果为 1   -1   0
```

3. 求平方根函数

格式：SQRT（<数值表达式>）

功能：返回数值表达式的平方根。其中数值表达式的值必须是非负数。

例 3.14

```
? SQRT(100)                    && 结果为 10.00
```

【即学即用】 将 $\sqrt{|ab-c^3|}$ 写为等价的 Visual FoxPro 算术表达式。（答案：SQRT(ABS(a * b－c^3))）

4. 圆周率函数

格式：PI()

功能：返回数值常量 π 的近似值。

5. 求整数函数

格式：INT(<数值表达式>)

CEILING(<数值表达式>)

FLOOR(<数值表达式>)

功能：INT()函数计算数值表达式的值，然后返回它的整数部分。

CEILING()函数返回大于或等于数值表达式的最小整数。

FLOOR()函数返回小于或等于数值表达式的最大整数。

例 3.15

```
STORE 2.3 TO x
?INT(x),INT(-x)              && 结果为 2    -2
STORE 2.3 TO x
?CEILING(x),CEILING(-x)      && 结果为 3    -2
STORE 2.3 TO x+++
?FLOOR(x),FLOOR(-x)          && 结果为 2    -3
```

【思考】 如何求实数 x 的小数部分？如何判断 n 是否偶数？试写出相应的 Visual FoxPro 表达式。

6. 四舍五入函数

格式：ROUND(<数值表达式 1>,<数值表达式 2>)

功能：该函数返回数据表达式在指定位置四舍五入后的结果。<数值表达式 2>表示四舍五入的位置。如果<数值表达式 2>大于或等于 0，那么它表示的是要保留的小数位数；如果<数值表达式 2>小于 0，那么它表示的是整数部分的舍入位数。

例 3.16

```
X=123.456
? ROUND(X,2),ROUND(X,1),ROUND(X,0),ROUND(X,-1)
```

显示结果为

```
123.46  123.5  123  123
```

说明：为了叙述的方便，将数值表达式 2 用 n 来表示。如果 n 的值非负，表示需保留的是十进制数的小数位数；如果 n 的值为负数，则 ROUND()返回的结果在小数点左端包含 n 个零；如果 n 的值为非整数，先对其取整再四舍五入。

7. 求余数函数

格式：MOD(＜数值表达式 1＞,＜数值表达式 2＞)

功能：该函数返回＜数值表达式 1＞和＜数值表达式 2＞两个数相除后的余数。余数的正负号与除数相同。如果被除数与除数同号，那么函数值就为两数相除的余数；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 3.17

```
? MOD(13,5)                    && 结果为 3
? MOD(13,-5)                   && 结果为-2
? MOD(-13,5)                   && 结果为 2
? MOD(-13,-5)                  && 结果为-3
```

【思考】 如何求出 168 的十位数字？试写出相应的表达式。

8. 求最大值和最小值函数

格式：MAX(＜数值表达式 1＞ [,＜数值表达式 2＞[,...]])

MIN(＜数值表达式 1＞ [,＜数值表达式 2＞[,...]])

功能：MAX()函数返回数值表达式中的最大值。MIN()函数返回数值表达式中的最小值。

例 3.18

```
?MIN(6,INT(4.2))               && 结果为 4
?MAX ("8","64")                && 结果为 64
```

说明：自变量表达式的类型可以是数值型、字符型、日期型、日期时间型、货币型、双精度型和浮点型，但是在一个具体的函数中所有表达式的类型必须相同。

3.3.2 字符函数

字符函数是指自变量一般是字符型数据的函数。

1. 求字符串长度函数

格式：LEN(＜字符表达式＞)

功能：返回字符表达式值的长度，函数值为数值型。

例 3.19

```
? LEN("计算机实验室")          && 结果 12
? LEN("welcome to vfphome")    && 结果为 18
```

2. 大小写转换函数

格式：LOWER(<字符表达式>)

UPPER(<字符表达式>)

功能：LOWER()函数将字符表达式值中的大写字母转换成小写字母，其他字符不变。

UPPER()函数将字符表达式值中的小写字母转换成大写字母，其他字符不变。

例 3.20

```
? LOWER("Xyz")                    && 结果为 xyz
? UPPER ("xYz")                   && 结果为 XYZ
```

3. 空格字符串生成函数

格式：SPACE(<数值表达式>)

功能：该函数生成由数值表达式指定空格数的空字符串。

例 3.21

```
?"欢迎"+SPACE(2)+"计算机实验室"        && 结果为"欢迎  计算机实验室"
```

4. 删除前后空格函数

格式：TRIM(<字符表达式>)

功能：删除字符表达式值前面的空格字符。

格式：LTRIM(<字符表达式>)

功能：删除字符表达式值尾部的空格字符。

格式：ALLTRIM(<字符表达式>)

功能：删除字符表达式值前面和尾部空格字符。

例 3.22

```
? "欢迎  "+"实验室"                  && 结果为"欢迎  实验室"
? TRIM("欢迎  ") +"实验室"           && 结果为"欢迎实验室"
? "欢迎 "+"  实验室"                 && 结果为"欢迎   实验室"
? "欢迎 "+LTRIM("  实验室")          && 结果为"欢迎实验室"
? SPACE(3)                          && 结果为"   "
? ALLTRIM("   数据库 ")              && 结果为"数据库"
```

5. 取子串函数

格式：LEFT(<字符表达式>,<长度>)

功能：从字符表达式值的左端取一个指定长度的子串作为函数值。

格式：RIGHT(<字符表达式>,<长度>)

功能：从字符表达式值的右端取一个指定长度的子串作为函数值。

格式：SUBSTR(<字符表达式>,<起始位置>,<长度>)

功能：从字符表达式值的起始位置取一个指定长度的子串作为函数值。

例 3.23

```
STORE "WELCOME" TO x
? LEFT(x,3)                    && 结果为 WEL
? LEFT(x,-1)                   && 结果为无显示
? RIGHT (x,4)                  && 结果为 COME
? RIGHT(x,-1)                  && 结果为无显示
? SUBSTR(x,3,3)                && 结果为 LCO
? SUBSTR(x,4)                  && 结果为 COME
? SUBSTR(x,8)                  && 结果为无显示
```

说明：对 LEFT()和 RIGHT()函数，设长度为 n，由于 1 个汉字占 2B，汉字字符串取子串时，如果 n 的值为奇数，可能会出现乱码。对于 SUBSTR()函数，假设起始位置为 m，长度为 n。若省略 n，则从 m 开始截取以后的所有字符串；若 n 大于从 m 开始的字符串长度，则从 m 开始截取以后的所有字符串；若 m 大于字符表达式值的长度，则截取的字符串为空白字符串。SUBSTR()函数若省略第三个自变量＜长度＞，则该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

格式：OCCURS(＜字符表达式 1＞,＜字符表达式 2＞)

功能：返回第一个字符串在第二个字符串中出现的次数，如果第一个字符串不是第二个字符串的子串，则函数值为 0。

7. 求子串位置函数

格式：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：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，则函数返回 0。

ATC()与 AT()功能类似，但在子串比较时不区分字母大小写。

＜数值表达式＞用于表明要在＜字符表达式 2＞值中搜索＜字符表达式 1＞值是第几次出现，其默认值为 1。

例 3.24

```
STORE "WELCOME" TO x
? AT("come",x)                 && 结果为 0
? ATC("come",x)                && 结果为 4
? AT("E",x)                    && 结果为 1
```

8. 子串替换函数

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

功能：该函数用＜字符表达式 2＞值替换＜字符表达式 1＞值中由＜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。

例 3.25

```
STORE "WELCOME" TO x
STORE "LIKE" TO y
? STUFF(x,3,5,y)                    && 结果为 WELIKE
? STUFF(x,3,4,"    ")               && 结果为 WE    E
```

9. 字符替换函数

格式：CHRTRAN(<字符表达式 1>,<字符表达式 2>,<字符表达式 3>)

功能：函数的自变量是 3 个字符表达式。当第 1 个字符串中的一个或多个相同字符与第 2 个字符串中的某个字符相匹配时，就用第 3 个字符串中的对应字符替换这些字符。如果第 3 个字符串包含的字符个数少于第 2 个字符串包含的字符个数，那么第 1 个字符串中相匹配的各字符将被删除。如果第 3 个字符串包含的字符个数多于第 2 个字符串包含的字符个数，多余字符被忽略。将 ASCII 码转化为相应的字符，函数返回值为字符型。

例 3.26

```
? CHRTRAN("你们好","你们","您")      && 结果为"您好"
```

10. 字符串匹配函数

格式：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，如果所有对应字符都相匹配，函数返回逻辑值真(.T.)，否则返回逻辑值假(.F.)。

例 3.27

```
STORE "xyz" TO x
? LIKE("ab*",x)                     && 结果为.T.
? LIKE("abC",x)                     && 结果为.F.
```

说明：<字符表达式 1>中可以包含通配符“*”和“?”。“*”可与任何数目的字符相匹配，“?”可以与任何单个字符相匹配。

3.3.3 日期和时间函数

日期和时间函数的自变量一般是日期型数据或日期时间型数据。

1. 求系统日期和时间函数

格式：DATE()

TIME()

DATETIME()

功能：DATE()函数返回当前系统日期，函数值为日期型。

TIME()函数以 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()函数返回当前系统日期时间，函数值为日期时间型。

例 3.28

```
? DATE(),TIME(),DATETIME()          && 显示当前系统的日期、时间
```

说明：如果选择了 nExp，则不管为何值，返回的系统时间都包括秒的小数部分，精确至小数点后两位，函数返回值为字符型。

2. 求年份、月份、天数函数

格式：YEAR（<日期表达式>|<日期时间表达式>）

MONTH（<日期表达式>|<日期时间表达式>）

DAY（<日期表达式>|<日期时间表达式>）

功能：YEAR()函数从指定的日期表达式或日期时间表达式中返回年份。

MONTH()函数从指定的日期表达式或日期时间表达式中返回月份。

DAY()函数从指定的日期表达式或日期时间表达式中返回月份中的天数。

例 3.29

```
STORE{^2008-01-05} TO x
? YEAR(x),MONTH(x),DAY(x)           && 结果为 2008   1   5
```

3. 求时、分和秒函数

格式：HOUR(<日期时间表达式>)

MINUTE(<日期时间表达式>)

SEC(<日期时间表达式>)

功能：HOUR()函数从日期时间表达式中返回小时部分(24 小时制)。

MINUTE()函数从日期时间表达式中返回分钟部分。

SEC()函数从日期时间表达式中返回秒数部分。

这 3 个函数的返回值都为数值型。

例 3.30

```
STORE ({^2013-09-05 01:34:22 PM }) TO t
? HOUR(t ),MINUTE(t),SEC(t)         && 结果为 13   34   22
```

3.3.4 数据类型转换函数

数据类型转换函数的功能是将某一种类型的数据转换成另一种类型的数据。

1. 数值转换成字符串

格式：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 L，则字符串加前导空格以满足规定的<长度>要求；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