第 2 章　数据类型、常量、变量与项目的使用

- 学习了解 Visual FoxPro 的基本数据类型、常量和变量的概念和定义方法。
- 掌握数据的查看方法。
- 熟练掌握常量、变量、运算符和表达式的使用方法。
- 理解数组的概念和基本使用。
- 学会 Visual FoxPro 数据库中提供的常用函数的使用方法。
- 掌握项目的建立和使用方法。

在计算机中，一切信息（数据）都是以二进制的形式表示和描述，即数据处理的方式都相同，因此处理的手段过于简单，也不易理解。我们知道，日常生活中有各种各样的信息（数据），有具有大小意义的数，如 12、-23、12.45 等；有具有日期时间性质的（信息）数据，如 2011 年 5 月 21 日、9:18:56 等。

为了方便计算机对这些（信息）数据存储和处理，Visual FoxPro 对各种数据进行了分类，这就是数据类型。

本章主要介绍构成 Visual FoxPro 应用程序的数据基本元素，包括：数据类型、数据的表现形式、数据的运算，具体就是指数据类型、常量与变量、运算符、表达式、内部函数等。

在本章的最后，我们还将向读者介绍项目的建立和使用方法。

2.1　数据类型

数据类型确定了数据在计算机内部的存储形式、值域及其所允许的运算。表 2-1 给出了 Visual FoxPro 的常用数据类型。在表 2-1 的第一列中，凡带下划线的字符，表示该类数据的类型缩写，在表的定义命令中将会用到。

表 2-1　Visual FoxPro 6.0 的常用数据类型

数据类型	说明	大小	范围
字符型（Character）	任意文本	每个字符用一个字节，最多可有 254 个字符	任意字符
货币型（Currency）	货币量	8 个字节	-922337203685477.5808~922337203685477.5807
日期型（Date）	含有年、月和日的数据	8 个字节	使用严格日期格式时，{^0001-01-01}（公元前 1 年 1 月 1 日）到{^9999-12-31}（公元 9999 年 12 月 31 日）

续表

数据类型	说明	大小	范围
日期时间型（DateTime）	含有年、月、日和时间的数据	8 个字节	使用严格日期格式时，{^0001-01-01}（公元前 1 年 1 月 1 日）到{^9999-12-31}（公元 9999 年 12 月 31 日），加上上午 00:00:00 到下午 11:59:59
逻辑型（Logical）	“真”或“假”的布尔值	1 个字节	真（.T.）或假（.F.）
数值型（Numeric）	整数或分数	在内存中占 8 个字节	在表中占 1～20 个字节 从.9999999999E+19～.9999999999E+20

除了上述数据类型外，表 2-2 还给出了在 Visual FoxPro 6.0 中仅可用于表中字段的有关的数据类型。

表 2-2 Visual FoxPro 6.0 仅用于字段的数据类型

字段类型	说明	大小	范围
双精度型（Double）	双精度浮点数	8 个字节	+/-4.94065645841247E-324～ +/-8.9884656743115E307
浮点型（Float）	与数值型一样	在内存中占 8 个字节；在表中占 1～20 个字节	-.9999999999E+19～.9999999999E+20
通用型（General）	OLE 对象引用	在表中占 4 个字节	受可用内存空间限制，通用型数据也是存放在与数据表同名、扩展名为.FPT 的备注文件中
整型（Integer）	整型值	4 个字节	从 2147483647～2147483646
备注型（Memo）	数据块引用	在表中占 4 个字节	受可用内存空间限制，存放在与数据表文件同名、扩展名为.FPT 的备注文件中
字符型（二进制）（VarChar Binary）	任意不经过代码页修改而维护的字符数据	每个字符用一个字节，最多可有 254 个字符	任意字符
备注型（二进制）（Memo Binary）	任意不经过代码页修改而维护的备注字段数据	在表中占 4 个字节	受可用内存空间限制

2.2 数据输出命令

下面将要介绍常量、变量和表达式的结果输出命令。最简单的数据输出命令是问号“?”和反斜杠“\”。问号命令分为单、双、三问号命令三种形式，反斜杠命令分为单、双反斜杠命令两种形式。

2.2.1 问号命令

1. 命令格式

? | ?? | Expression1 [FONT cFontName [, nFontSize] [STYLE cFontStyle]]
[, Expression2] ... [, Expression3]...

或

??? Expression

2. 功能

?：换行从下行首部开始显示表达式列表中各表达式的值。

??：不换行从光标的当前位置开始显示表达式列表中各表达式的值。

???：指定将字符型常量的内容直接发送到打印机上。

其中：

（1）Expression1：计算表达式 Expression1 的值，然后先输出一个回车和换行符，再输出计算结果。计算结果显示在 Visual FoxPro 主窗口或者活动的用户自定义窗口的下一行。

如果省略了表达式，则显示或打印一个空行。当包含多个表达式时，即有 Expression2、Expression3 等表达式时，显示的表达式结果之间将插入一个空格。

（2）FONT cFontName [, nFontSize]：指定用于“? | ??”输出的字体。cFontName 指定字体名称，nFontSize 指定字体大小。

【例 2-1】下列命令用 16 磅的“黑体”字体显示字符“中华人民共和国”，显示结果如图 2-1 所示。

? "中华人民共和国" FONT '黑体',16↙

注意：命令行中的符号“↙”，表示命令输入完毕后，并按下回车键，以下不再标出。

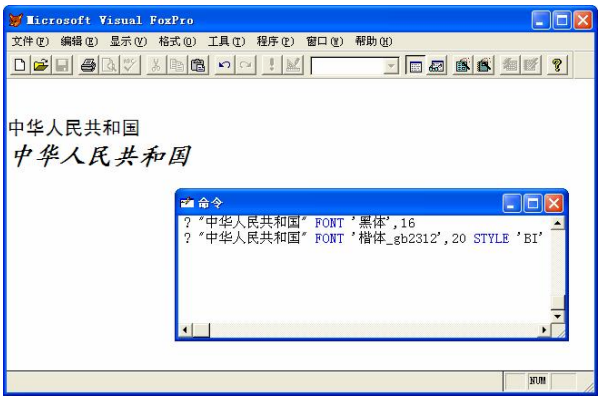

图 2-1　“?|??”数据输出命令的使用

如果给出 FONT 子句但是没有指定字体大小 nFontSize，此时字体大小为 10 磅。

如果省略 FONT 子句，并且“? | ??”的输出结果放在 Visual FoxPro 主窗口中，则输出的字体为 Visual FoxPro 主窗口字体（或用户自定义窗口字体）。

（3）STYLE cFontStyle：指定用于“? | ??”输出的字体样式。如果省略 STYLE 子句，则使用“正常”字体样式。STYLE 子句指定字体样式时，必须包含有 FONT 子句。cFontStyle 可以使用的字体样式，如表 2-3 所示。

可以使用多个字符的组合来指定字体样式。

【例 2-2】例如，下面的命令用 20 磅的“楷体”字体，加粗和斜体字样，显示字符“中华人民共和国”，显示结果如图 2-1 所示。

? "中华人民共和国" FONT '楷体_gb2312',20 STYLE 'BI'

“?|??”命令的后面还有许多短语可供选择，以修饰显示效果，读者可查阅帮助文件予以了解。

表 2-3　cFontStyle 指定的字体样式

字符	字体样式	字符	字体样式
B	粗体	S	阴影
I	斜体	-	删除线
N	正常	T	透明
O	轮廓	U	下划线
Q	不透明		

2.2.2　反斜杠命令

1. 命令格式

\|\\\TextLine

2. 功能

\：换行从下行首部开始显示文本行的内容。

\\：不换行从光标的当前位置开始显示文本行的内容。\和\\前面的空格不包含在输出行中，但是\和\\之后的空格包含在输出行中。

【例 2-3】在命令窗口中输入如下命令序列，在主窗口输出结果。

```
x=50
y=150
\x+y=
??x,"+",y
??"=",x+y
```

则在 Visual FoxPro 工作主窗口中显示：x+y= 50 + 150 = 200。

可以在文本行中嵌套一个表达式。如果表达式放在文本合并分隔符（默认情况下为<< >>）之内，并且 SET TEXTMERGE 设置为 ON，则计算表达式的值，并将结果以文本形式输出。

【例 2-4】在命令窗口中输入如下命令序列，在主窗口输出结果。

```
d=CTOD("10-1-2009")  &&此命令将字符串"10-1-2009"转化为一个日期
SET TEXTMERGE ON
\<<YEAR(d)>>年
\\<<MONTH(d)>>月
\\<<DAY(d)>>日
\是中华人民共和国成立 60 周年的喜庆日子。
```

命令后面的符号“&&”以及文字部分是注释说明，输入命令时可省略。命令序列显示的结果，如图 2-2 所示。

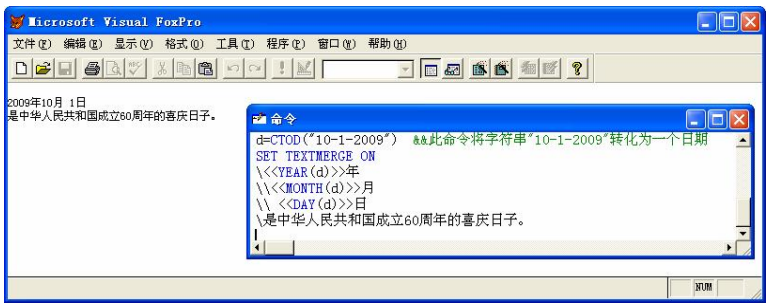

图 2-2　用“\|\\”文本输出命令的使用

2.3　常量

常量（Constant）是指在数据处理或在程序执行过程中其值不能发生变化的量，是在命令或程序中被直接引用的实际值。按是直接写出还是先定义成一个符号再引用，可将常量分为直写常量（字面常量）和符号常量。

最常用的直写常量在 Visual FoxPro 中有以下类型。

1．数值型常量

整数、小数和科学计数法表达的数是数值型常量，如 10，123.45，1.2e-3、2.5E+6 等。

2．字符型常量

字符型常量是使用一对界限符""，也可使用一对"或[]括起来的一个或几个字符，称为字符串。如'A'、"Abcd"、[123abc]。如果该字符串本身含有一种界限符，则需要用另一种界限符括起来，如[It's all right]等。字符串的最大长度为 254 个字符。

3．日期型常量

用花括号括起来的量，如{11/08/2000}，空白时间为{}或{/}。时间日期型为{11/08/2000 8:45 a}等，严格时间日期型可用{^11/08/2000 8:45 a}表示。

使用函数将一个字符串转换为一个日期，如 CTOD(″11/23/79″)等。

可以使用下面的命令对日期或日期时间常数进行明确检查。

SET STRICTDATE TO [0 | 1 | 2]

其中：0，默认值，关闭严格的日期格式检查；1，指定所有的日期和日期时间常数必须符合严格的日期格式；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。

下面再介绍几条影响日期格式的设置命令。

（1）SET MARK TO 命令

该命令用于设置显示日期型数据时使用的分隔符，命令格式如下：

SET MARK TO [cDelimiter]

其中，cDelimiter 设置显示日期型数据时使用的分隔符，如“-”、“.”等。如果 SET MARK TO 命令中没有指定任何分隔符，执行该命令时表示恢复系统默认的斜杠分隔符。

（2）SET DATE TO 命令

该命令用于指定日期表达式和日期时间表达式的显示格式，命令格式如下：

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

在该命令中，AMERICAN 等参数所定义的日期格式如表 2-4 所示。

表 2-4　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）SET CENTURY ON/OFF 命令

此命令用于设置显示日期型数据时是否显示世纪，命令使用格式如下：

```
SET CENTURY ON/OFF
```

【例 2-5】在命令窗口中输入下面的命令序列，其功能是设置年月日格式。

```
SET CENTURY ON          && 设置 4 位数字年份
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
?{^2011-08-12}
```

屏幕上显示以下结果：

```
2011/08/12
```

4. 逻辑型常量

逻辑型常量只有两个值，即.T.（真）和.F.（假）。.T.（真）也可写成.t.、.Y.或.y.；.F.（假）也可写成.f.、.N.或.n.。

5. 货币型常量

以$符号开头并四舍五入到小数 4 位的数值型数据，如$12、$23.45、$12.3457 等。

6. 符号常量

最常用的符号常量也有上述的 5 个类型，但它必须用预编译指令#DEFINE 进行预定义，方可使用。预定义的格式为：

```
#DEFINE ConstantName Expression
```

其中：ConstantName 为符号常量名称，Expression 为要对符号常量预定义的值。

在程序中当不再使用符号常量时，可通过#UNDEF 语句予以释放。格式为：

```
#UNDEF ConstantName
```

【例 2-6】在命令执行的开始定义符号常量 PI，表示圆周率 3.1415926，设圆半径 R=10，求出圆的周长和面积。

在命令窗口中依次执行下面的命令序列：

```
#DEFINE PI 3.1415926
R=10
L=2*PI*R
S=PI*R*R
?"当圆半径 R=",R,"时："
?"圆周长 L=",L
?"圆面积 S=",S
#UNDEF PI
```

2.4 变量

在数据处理过程中，其值可以改变的量称为变量。一个变量必须有一个名字和相应的数据类型，通过名字可引用一个变量，而数据类型则决定了该变量的存储方式和在内存中占据存储单元的大小。变量名实际上是一个符号地址，在对程序编译连接时，由系统给变量分配一个内存地址，在该地址的存储单元中存放变量的值，如图 2-3 所示。程序从变量中取值，实际上是通过变量名找到相应的内存地址，从其存储单元中取得数据。

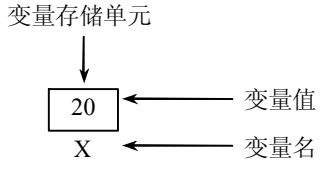

图 2-3 变量名与变量值

在 Visual FoxPro 中，变量根据是否与表的结构有关分为两大类，一类是与表的结构定义无关的变量，称为内存变量（又称为临时变量）；另一类则是与表的结构定义密切相关的变量，用来定义表的字段的数据类型，称为字段名变量。数组是一种特殊的内存变量。另外，Visual FoxPro 还提供了一系列系统内存变量，供用户使用，系统内存变量由系统规定。

内存变量是独立存在于内存中的变量，一般随程序结束或退出 Visual FoxPro 而释放，也可在程序代码中使用命令释放内存变量。内存变量常用于存储程序运行的中间结果或用于存储控制程序执行的各种参数。Visual FoxPro 定义 6 种内存变量：字符型、数值型、逻辑型、日期型、日期时间型和屏幕型内存变量。对于屏幕型内存变量，可用“SAVE SCREEN TO 变量名”命令存放当前屏幕上的信息，用“RESTORE SCREEN 变量名”命令从屏幕内存变量中恢复屏幕信息。Visual FoxPro 最多允许 65000 个内存变量，默认为 1024 个。

变量一般要先声明，再使用。

2.4.1　内存变量

1. 内存变量的命名

与常量不同，每个内存变量应有一个不能重复且固定的名字，以便与内存中为它们开辟的存储单元实现内外的对应联系，用户通过对变量名字的操作实现对存储单元的访问。

Visual FoxPro 规定，内存变量的名字不区分大小写，且必须是以英文字母或汉字或下划线开头，后跟字母、汉字、数字、下划线的一串字符，总长不得超过 128 个字符。

例如：xh、xm、A100、_scr、b_c 都是合法的内存变量名，而：12xy、*abc 则不是合法的变量名。

变量在命名时，应避免使用 Visual FoxPro 保留字。由于 Visual FoxPro 的系统内存变量均以下划线开头，为了不使用户自定义的内存变量与系统内存变量相混淆，建议用户在定义内存变量时，最好不要以下划线开头。

2. 内存变量的赋值

在 Visual FoxPro 中，内存变量的数据类型属于变体型，它的具体类型由赋给它的表达式的类型来确定。对内存变量赋值的方法有两种。

（1）使用“=”为内存变量赋值

“=”称为赋值命令，其使用格式如下：

```
MvarName = Expression
```

赋值命令的作用是将“=”右边表达式的值赋给它左边的内存变量。例如：

```
s='2008 年 8 月 8 日是一个骄傲的日子。'
?s      &&屏幕主窗口显示“2008 年 8 月 8 日是一个骄傲的日子。”
```

（2）使用 STORE 命令为内存变量赋值

STORE 命令的功能是将数据存入内存变量、数组或数组元素中，其语法格式如下：

```
STORE eExpression TO VarNameList | ArrayNameList
```

其中，VarNameList 表示内存变量列表。若内存变量列表中有若干个内存变量时，各变量之间必须用逗号“,”隔开（以后凡提到列表均如此）。

【例 2-7】为内存变量 d1、d2、d3 分别赋值一个日期{^2008-8-8}并显示。

在命令窗口中，输入如下的命令：

```
SET CENTURY ON              &&确定日期是否显示世纪部分
SET DATE TO LONG            &&确定日期显示格式和 Windows 系统日期格式一致
```

```
STORE {^2008-8-8} TO d1,d2,d3
?d1,d2,d3
```

在 Visual FoxPro 工作区将显示：

2008 年 8 月 8 日 2008 年 8 月 8 日 2008 年 8 月 8 日

3. 变量的作用范围

变量有一定的使用范围，在命令窗口中直接使用的变量是全局性的。其作用范围在命令窗口中以及所有的程序都可以使用。在 Visual FoxPro 程序中，还可以使用 Local、Private 和 Public 命令强制规定变量的作用范围。

（1）Local

用于定义本地变量，在命令窗口不能使用该命令。这类变量只能在创建它们的程序中使用和修改，不能被更高层或更低层的程序访问。

（2）Private

用于定义私有变量，在命令窗口该命令不起作用。它是仅用于当前程序的变量。这类变量可被程序本身或更低层的程序访问或修改。

（3）Public

用于定义全局变量，在本次 Visual FoxPro 运行期间，所有程序都可以使用这个全局变量。

2.4.2 数组变量

在 Visual FoxPro 数据处理过程中，常有很多变量，如 x0=301,x1=302,x2=303,x3=304,…。为方便对各个变量进行统一管理，可将各变量定义成一个集合，即数组，如图 2-4 所示。

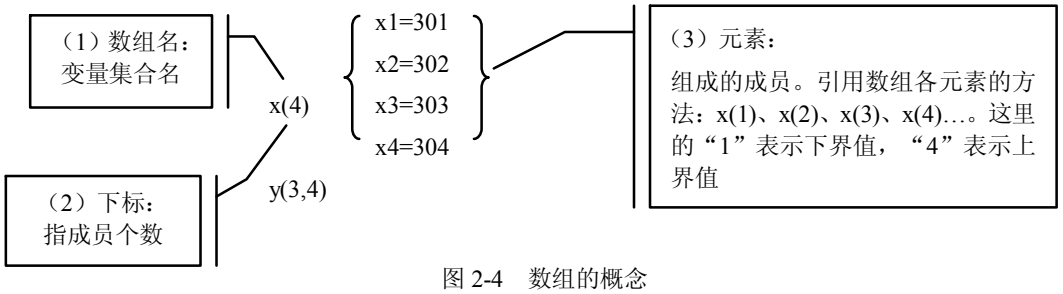

图 2-4 数组的概念

一个数组的基本要素如下：

（1）数组名（Array Name）：变量集合名，如 x、y、z 等。

（2）下标（Index）：指成员个数，如 x(4)中的数字 4。

（3）元素（Element）：组成数组的成员，使用时用 x(1)、x(2)、y(1,2)、x(3,4)等表示成员元素。

（4）维数（Dimension）：数组下标的个数，表示数组的复杂度。如数组 y(3,4)有两个下标，我们称其为二维数组。在 Visual FoxPro 中支持二维数组，但不支持更高维数组。

（5）下标的上界（Upper Boundary）和下界（Lower Boundary）。指定的上下标界，在 Visual FoxPro 中，各维的下界总是 1，上界大小由下标的数字决定。

一个数组可以存储多个值，通过数组名和下标对这些值进行存取。与变量相比，数组有以下优点：

（1）数组能够保存多个值。

（2）数组可与循环语句配合实现复杂算法。

（3）数组可用作通用过程的参数，以传递大量的值。

（4）数组常用来表示与一维、二维空间分布相关的数据，非常直观，即一维数组对应于二维表格中的一条记录，二维数组对应于一张表格。

与其他高级语言不同，Visual FoxPro 数组中的各个元素可以具有不同的数据类型。

1．数组的声明

数组原则上必须先声明然后使用，数组声明使用 DIMENSION 或 DECLARE 语句。数组声明的格式如下：

```
DIMENSION|DECLARE ArrayName1(nRows1 [, nColumns1])
                  [, ArrayName2(nRows2 [, nColumns2])] ...
```

功能是声明一维或二维数组，一个声明语句可以同时声明多个数组。其中：

- ArrayName：要声明的数组的名字。
- nRows：要声明的数组的行数。
- nColumns：要声明的数组的列数。

说明：

- 在 Visual FoxPro 中，数组的命名规则和内存变量相同。
- 在 Visual FoxPro 中，最多可声明 1024 个数组，每个数组最多 65000 个元素。
- 数组声明后各元素一律被初始化为.F.。
- 数组在声明和使用时，也可在数组名后使用方括号。
- 数组可重新声明从而改变其维数与大小，此时原元素值将保持不变，除非数组的容量变小使后面的元素丢失。
- 在计算机中数组按行存储，二维数组可以当成一维数组来使用，只要元素的序号相对应即可。设有一个二维数组：a(M,N)，一个一维数组：a(M*N)。则 a(i,j)和 a(k)的下标对应关系为：k=(i-1)*N+j（i=1,2,…,M，j=1,2…,N）。

【例 2-8】下面的命令行可声明一个有 10 个元素的一维数组 a，行数为 10，列数为 20 的数组 b。

```
DIMENSION  a(10),b(10,20)
```

2．数组的赋值

数组的赋值分为整体赋值和元素赋值。它们都可以像内存变量一样通过赋值语句或赋值符来实现。

（1）为数组整体赋值

数组的整体赋值只要将表达式的值赋给数组名即可。

【例 2-9】声明一个 2 行 3 列的数组 d，并将各元素的初值一律赋为{^2011-10-1 日}。

```
DIMENSION d(2,3)
d={^2011-10-1}   &&也可使用命令 STORE {^2011-10-1 日} TO d
SET CENTURY ON
?d(1,1),d(1,2),d(1,3)
?d(2,1),d(2,2),d(2,3)
```

显示结果为两行：

```
10/1/2011 10/1/2011 10/1/2011
10/1/2011 10/1/2011 10/1/2011
```

（2）为数组元素赋值

数组声明后，对数组元素的赋值，只要使用下标即可。

【例 2-10】对上例中的数组元素 d(1,3)、d(2,3)分别赋一个日期和一个字符串。

```
SET DATE LONG        &&设置日期的显示为长格式
d(1,3)= {^2011/10/1}
d(2,3)='中华人民共和国国庆日'
?d(1,3),d(2,3)
```

结果显示：

2011 年 10 月 1 日 中华人民共和国国庆日

2.4.3 字段名变量

字段名变量的声明与赋值和数据表密切相关，将在第 4 章“数据表的基本操作”中讲解。

由于内存变量存放在独立于数据库文件的临时存储单元中，所以，变量可以同名，但在这种情况下，变量具有更高的使用优先级。若访问内存变量，变量名前应加上“m.”或“m->”前缀来引用它。例如，一个内存变量和当前打开的表字段变量同名，都是 csrq，则在引用内存变量 csrq 时，要用形式 m.csrq 或 m->csrq，如果直接使用 csrq，则引用是的表字段变量 csrq。

2.4.4 内存变量的查看

声明或定义了内存变量后，用户可通过 DISPLAY MEMORY 或 LIST MEMORY 命令对所定义的内存变量进行查看。

DISPLAY MEMORY 命令的使用格式如下：

```
DISPLAY MEMORY [LIKE FileSkeleton] [NOCONSOLE]
[TO PRINTER [PROMPT] | TO FILE FileName]
```

其中：

- LIKE FileSkeleton：可以通过包含 LIKE 子句有选择地显示符合匹配条件要求的内存变量和数组。匹配条件中可以包含问号“?”和星号“*”通配符，例如，要显示所有以字母 A 开头的内存变量，可执行命令：DISPLAY MEMORY LIKE A*
- NOCONSOLE：不在 Visual FoxPro 主窗口或活动的用户自定义窗口输出。
- TO PRINTER [PROMPT]：将 DISPLAY MEMORY 的结果定向输出到打印机。包含 PROMPT 子句时，在打印开始前显示“打印”对话框。
- TO FILE FileName：将查看的结果定向输出到 FileName 指定的文件（*.txt）中。如果文件已经存在，且 SET SAFETY 设为 ON，Visual FoxPro 会提示是否要改写此文件。
- DISPLAY MEMORY 也显示有关系统内存变量、菜单、菜单栏、菜单标题和窗口的信息。

【例 2-11】内存变量显示示例。

在命令窗口中依次输入并执行如下命令：

```
a=10
ab="Visual FoxPro"
abc=.Y.
b=30
bc="Hello China!"
DIMENSION c(3)
DISPLAY MEMORY LIKE a*
DISPLAY MEMORY TO nc
```

其中：

（1）命令 DISPLAY MEMORY LIKE a*显示内容如下

```
A               Pub         N       10                  (           10.00000000)
AB              Pub         C       "Visual FoxPro"
ABC             Pub         L       .T.
```

（2）命令 DISPLAY MEMORY TO nc 执行的功能是：在主窗口既显示内存变量，同时也显示系统变量，并将这些变量保存到文件 nc.txt 中。屏幕显示和文件 nc.txt 的部分内容如下：

```
A               Pub         N       10                  (           10.00000000)
AB              Pub         C       "Visual FoxPro"
B               Pub         N       30                  (           30.00000000)
BC              Pub         C       "Hello China!"
C               Pub         A
    (    1)                 L       .F.
    (    2)                 L       .F.
    (    3)                 L       .F.
ABC             Pub         L       .T.
已定义       6 个变量，占用了 39 个字节
1018 个变量可用
打印系统内存变量
_ALIGNMENT      Pub         C       "LEFT   "
_ASCIICOLS      Pub         N       80                  (           80.00000000)
……
_WRAP           Pub         L       .F.
   74 个系统变量已定义
菜单和主菜单定义
    0 个菜单已定义
弹出式菜单定义
    0 个弹出式菜单已定义
窗口定义
    0 个窗口已定义
```

在上面的显示结果中，其中：

- 第 1 列为变量名。
- 第 2 列为变量特性，若为全局变量，则显示 Pub；若为私有变量，则显示 Priv。
- 第 3 列为变量类型。字符型显示 C，数值型显示 N，日期型显示 D，逻辑型显示 L，数组显示 A 等。
- 第 4 列为变量的值。对于字符型变量，显示带双引号的字符串；对于数值型变量，显示数值，其后面的是机内表示法；对于日期型变量，显示格式为“月月/日日年年”（可用 SET DATE TO 命令改变其显示格式）；对于逻辑型变量，显示.T.或.F.。
- 对于内存变量，占用了多少个字节数的计算方法是字符型变量的长度（包括字符型数组元素）+7 后的总和。

此外，要查看内存变量的使用情况，也可使用 LIST MEMORY 命令，该命令和 DISPLAY MEMORY 的区别是滚屏显示，而不是分屏显示。LIST MEMORY 命令的语法格式如下：

```
LIST MEMORY [LIKE FileSkeleton] [NOCONSOLE]
            [TO PRINTER [PROMPT] | TO FILE FileName]
```

2.4.5　内存变量的保存、恢复和清除

当退出 Visual FoxPro 系统后，用户所建立的内存变量将不会存在，如果希望以后再使用

这些变量，这时可将这些内存变量保存到一个指定的内存变量文件中，同时也可将指定的内存变量文件中的变量恢复到 Visual FoxPro 系统中。如果内存变量不再使用，也可清除。

1. 内存变量的保存

用户所建立的内存变量可用下面的命令将它们保存到内存变量文件中。

```
SAVE TO FileName | MEMO MemoFieldName
        [ALL LIKE Skeleton | ALL EXCEPT Skeleton]
```

其中：

- FileName：指定保存内存变量和数组的内存变量文件。内存变量文件的默认扩展名是.MEM。
- MEMO MemoFieldName：指定保存内存变量和数组的备注字段。
- ALL LIKE Skeleton：指定保存所有符合匹配条件要求的内存变量和数组。匹配条件中可以包含问号“?”和星号“*”通配符。
- ALL EXCEPT Skeleton：指定保存所有不符合匹配条件要求的内存变量和数组。

【例 2-12】对例 2-11 中所定义的内存变量进行保存。

```
SAVE TO NC        &&保存所有内存变量
SAVE TO NC1 ALL LIKE A*   &&保存所有以 A 字符开始的内存变量
SAVE TO NC1 ALL EXCEPT A*   &&保存所有除 A 字符开始的内存变量
```

2. 内存变量的恢复

如果要重新使用已保存在内存变量文件中的内存变量，可用命令 RESTORE FROM 将内存变量调入内存，该命令的使用语法格式如下：

```
RESTORE FROM FileName | MEMO MemoFieldName [ADDITIVE]
```

其中：

- FileName：指定保存内存变量和内存变量数组的内存变量文件。内存变量文件指定扩展名是.MEM。
- MEMO MemoFieldName：指定保存内存变量和内存变量数组的备注字段。
- ADDITIVE：防止删除当前内存中已有的内存变量或内存变量数组。如果使用 ADDITIVE 时，要添加的内存变量或内存变量数组的数目加上已有内存变量的数目超过了系统对内存变量数目的限制，Visual FoxPro 将从内存变量文件或备注字段中恢复尽可能多的内存变量和内存变量数组。

恢复内存变量或内存变量数组时，如果内存变量或内存变量数组与已有内存变量或内存变量数组有相同的名称，则用恢复的内存变量或内存变量数组的值改写原有内存变量或内存变量数组中的值。

3. 内存变量的清除

为节省存储空间，不再使用的内存变量应使用清除命令来释放其所占的内存空间。可利用如下几条命令清除内存变量。其语法格式如下：

（1）CLEAR ALL

该命令的功能是从内存中释放所有的内存变量和数组以及所有用户自定义菜单栏、菜单和窗口的定义。

（2）CLEAR MEMORY

该命令的作用是从内存中释放所有公共或私有内存变量和数组，但不释放系统内存变量。

（3）RELEASE <内存变量名表>

该命令的作用是从内存中释放指定的内存变量和数组，各内存变量和数组名称可用逗号分隔。

（4）RELEASE ALL [EXTENDED] [LIKE Skeleton | EXCEPT Skeleton]

其中：

- ALL：从内存中释放所有的内存变量和数组。
- EXTENDED：在程序中，指定释放所有的公共变量。当在程序中执行 RELEASE ALL、RELEASE ALL LIKE 或 RELEASE ALL EXCEPT 时，并不释放公共变量。
- LIKE Skeleton | EXCEPT Skeleton：从内存中释放所有符合匹配条件的内存变量和数组，Skeleton 可以包含通配符“?”和“*”。

例如，清除所有以 A 开头的内存命令，可输入下面的命令：

```
RELEASE ALL LIKE a*
```

2.5　运算符与表达式

在数据加工时，Visual FoxPro 提供了一组运算指令，这些指令以某些特殊符号表示，称为运算算符，简称运算符（Operator）。

用运算符按一定的规则连接常量、变量和函数所组成的式子称为表达式（Expression），如，m+3，s+Cos(x*3.14/180)等都是表达式，单个变量或常数也可以看成是表达式。每一个表达式都有一个值，即计算结果，称为表达式的值（Value）。

运算符可分为数值（算术）运算符、字符串运算符、日期运算符、关系（比较）运算符、逻辑运算符和类与对象运算符等六类，这里介绍前五类。

2.5.1　数值运算符

数值运算符要求参与运算的运算对象是数值型数据，运算的结果也是数值型，除取负号运算符“-”是单目运算符（要求一个运算对象）之外，其余都是双目运算符（要求两个运算对象）。数值运算符及优先级如表 2-5 所示。

表 2-5　数值运算符及优先级

运算	运算符	运算优先级	例子	结果
括号	()	1		
取负	-	2	-2+3	1
幂	^或**	3	-2^4	16
乘法	*	4	2*3	6
浮点除法	/		3/2	1.5
取模（余）	%	5	5 % 3	2
加法	+	6	2+3	5
减法	-		2-3	-1

说明：

（1）由数值运算符按一定的规则连接常量、变量和函数所组成的式子称为数值表达式。

（2）在 Visual FoxPro 中，不支持数学的花括号（一对“{}”）和中括号（一对“[]”）的用法，如果在表达式中有多重括号，则可使用多个括号“()”进行表示，如下面的表达式：

(x-y)/(x+y)+(1+sin(x)^2+cos(30))*ln(x+y)

（3）取模（余）运算符%。取余运算就是对两数进行除法运算后取商的余数部分，如果第一个对象是一个整数，则余数一定为整数，即被除数中的小数位数决定了计算结果中的小数位数。

求余的结果的正负号始终与第二个运算对象的符号相同。例如：

5 % 3 结果值为 2。

5 % -3 结果值为-1。

-5 % -3 结果值为-2。

5 % 2.2 结果值为 1。

5.0 % 2.2 结果值为 0.6。

当除数和被除数的符号相同时，结果的绝对值为：|被除数|%|除数|；当除数和被除数的符号不同时，结果的绝对值为：(|被除数|-|除数|)%|除数|-|除数|。

思考题：执行命令?13%4,-13%-4,8.00%2.6,- 8%-2.5 后，结果是什么？

2.5.2 字符串运算符

字符串运算符有“+”、“-”和“$”，用字符串运算符按一定的规则连接常量、变量和函数所组成的式子称为字符串表达式。其中：

（1）运算符“+”和“-”：作用是将两个字符串连接在一起，系同级别运算符。例如：

"Hello␣" + "China!"　'结果为：Hello China!，其中“␣”表示空格

"Hello␣" - "China!"　结果为：HelloChina!␣

说明：

- “+”：将两个字符串、一个字符串和一个字段或一个字符串和一个内存变量连接起来。
- “-”：如果第一个字符串有后缀空格，则将第二个字符串追加到第一个字符串之后并将原第一个字符串的后缀空格移动到结果字符串之后。

（2）运算符“$”：用于比较，在一个字符串表达式中寻找另一个字符串表达式。由于该运算符的优先级比“+”和“-”低，但比关系运算符高，因此介绍关系运算符时再行讲解。

2.5.3 日期时间运算符

日期型、日期时间型运算符有“+”和“-”两个，它们是同一优先级的，对应的表达式分别为日期型和日期时间型表达式。

表 2-6 给出了日期型和日期时间型运算符、表达式的功能。

表 2-6　日期、日期时间型运算符及表达式

运算符	结果	结果单位	实例	结果
+	得到新日期：d	天（day）	{^2011/08/01}+7	08/08/11
	得到新日期时间：t	秒（Sec）	{^2011/08/08 09:00:00}+3600	08/08/11 10:00:00 AM
-	相差的天数：n	天（day）	{^2000/08/08}-{^1999/08/08}	366

续表

运算符	结果	结果单位	实例	结果
-	相差的秒数：n	秒（Sec）	{^2011/08/08 20:00}-{^2011/08/08 1:00}	68400
	得到新日期：d	天（day）	{^2011/08/10}-2	08/08/11
	得到新日期时间：t	秒（Sec）	{^2011/08/08 20:30}-5400	08/08/11 07:00:00 PM

2.5.4　关系运算符

关系运算符用于进行两个类型相容的数据之间的大小比较，比较的结果是逻辑值.T.或.F.，即比较成立时，返回逻辑值.T.；反之，返回逻辑值.F.。

关系运算符分为关系运算符和字符关系运算符“$”。关系运算符有 7 个，分别为“<”、“>”、“=”、“<>”（或“#”，或“!=”）、“<=”、“>=”、“==”，它们均系同级别运算符，字符关系运算符是“$”，但该运算符比关系运算符的优先级要高。

各个关系运算符及表达式的功能说明，如表 2-7 所示。

表 2-7　关系运算符和关系表达式

运算符	功能	运算优先级	实例	结果
$	包含于	1	'人民'$[中国人民]	.T.
<	小于	2	2<3	.T.
<=、=<	小于等于		{^2008/08/08}<={^2007/08/08}	.F.
=	等于		"abc"=[abd]	.F.
>=、=>	大于等于		[北京奥运]>=[奥运北京]	.T.
>	大于		.T.>.F.	.T.
<>、#、!=	不等于		5!=6	.T.
==	精确等于		'北京奥运'=='北京奥运　'	.F.

说明：

（1）对于数值型数据比较大小很简单，数值大者为大，数值小者为小。逻辑型数据比较时逻辑真值.T.大于逻辑假值.F.。日期型或日期时间型数据比较大小以日历和时钟为准，按年、月、日、时、分、秒的次序逐项比较，朝后者为大，朝前者为小。例如：

```
?{^201108/08}>{^2010/08/08},{^2011/08/08 21:30}<{^2011/08/08 20:00}
```

结果显示：.T.　.F.

（2）字符串比较总的原则是：从第一个字符开始逐字符比较，当碰到某个对应字符不同时，该字符的排列次序大者所在的字符串大。

字符串在进行比较时，可按拼音（PinYin）、机内码（Machine）和笔画（Stroke）进行比较，选用的排列方法不同会有不同的结果。因此，在进行字符大小比较时，必须清楚当前所用的字符排序次序属于哪一种，如果不是希望的排序次序，就要重新设置。

- 设置字符排序次序

字符的排序次序既可通过 Visual FoxPro 主菜单“选项”对话框中的“数据”选项卡来设置，也可用命令设置。命令格式如下：

SET COLLATE TO cOption

其中：参数 cOption 指定要选用的字符排序次序的方式，它必须以字符串形式给出，可以是："Pinyin"、"Machine"、"Stroke"，分别表示字符串按拼音、机器码、笔画进行排序。"Pinyin"是系统的缺省设置。

- 确定操作符“=”如何对不同长度字符串进行比较

使用命令 SET ANSI，可以确定在 Visual FoxPro 命令中如何用操作符“=”对不同长度字符串进行比较。该命令的语法格式如下：

SET ANSI ON | OFF

其中：

字符串顺序：在 Visual FoxPro 命令中，两个字符串在比较中的左右顺序是有关系的。把字符串从操作符“=”或“==”的一边移到另一边将影响比较的结果。

ON：将“=”右侧较短的字符串后面加入空格以使它与较长的字符串具有相同的长度。然后，对两个字符串的每个字符进行一个字符与一个字符的比较。比如下面的比较：

'Tommy' = 'Tom'

如果 SET ANSI 设置为 ON，则上述比较结果为“假”（.F.）。这是因为在'Tom'中加入空格后成了'Tom '，而字符串'Tom '与字符'Tommy'并不是每个对应的字符都相等。

OFF：指定“=”右侧较短的字符串不要填充空格。两个字符串比较到较短的字符串结束就可以了，系统默认为 OFF。比如下面的比较：

'Tommy' = 'Tom'

如果 SET ANSI 设置为 OFF，则上述比较结果为“真”（.T.）。这是因为比较完字符串'Tom'后便不再继续比较了。

SET ANSI 对操作符“==”没有任何影响。当使用操作符“==”时，较短的字符串总是在其后加上空格后再进行比较的。

（3）字符串比较分为精确比较和非精确比较

- 精确比较

精确比较的运算符为“==”，其比较方法是：当且仅当两个字符串完全相同——长度相等（对应字符相同时），表达式的结果为真，其余情况均为假。它与命令 SET EXACT OFF|ON 状态的设置无关。

- 非精确比较

非精确比较所用的比较运算符是“=”，它与 SET EXACT OFF|ON 状态的设置有密切的关系。设有两个字符串 S1 和 S2，有表达式 S1=S2。

在 SET EXACT OFF 状态（默认设置）下，S1=S2 的比较方法是：当 S2 为 S1 从首字符开始的一个子字符串时结果即为真，否则为假。与精确比较不同，在非精确比较的状态下，如果将 S1、S2 位置颠倒，则可能产生不同结果。

在 SET EXACT ON 状态下，S1=S2 的比较方法分两步进行。首先，在较短的字符串后面添加空格到和较长的串等长，然后再逐字符比较大小。

【例 2-13】比较"北京奥运会"和"奥运会"的大小。

```
SET COLLATE TO "machine"
SET EXACT OFF
?"北京奥运会"="北京","北京"="北京奥运会"        &&结果为  .T.      .F.
SET EXACT ON
?"北京奥运会"="北京","北京"="北京奥运会"        &&结果为  .F.      .F.
```

（4）包含运算符$

包含运算符的运算定义为：设有两个字符串 S1 和 S2，有表达式：S1$S2。当 S1 属于 S2 的一个子字符串时表达式结果为真，否则为假。例如：

?"奥运会"$"北京奥运会"

结果显示：

.T.

2.5.5　逻辑运算符

逻辑运算符对运算对象进行逻辑运算，共 3 个。按其优先级从高到低的排列为 NOT（逻辑非）、AND（逻辑与）、OR（逻辑或），运行的结果为逻辑型数据。当逻辑关系成立时，运算结果为.T.；当逻辑关系不成立时，运算结果为.F.。除运算符 NOT 是一个单目运算符外，其余都是双目运算符。表 2-8 列出了逻辑运算符，表 2-9 列出了逻辑运算的真值表。

表 2-8　逻辑运算符

运算符	说明	运算结果的说明	优先级	例子	结果
NOT 或.NOT. 或!	逻辑非	取反，即：假取真， 真取假	1	Not(3>2)	.F.
AND 或.AND.	逻辑与	操作数均为真时， 结果为真，其余为假	2	(3>2) And (5>=5)	.T.
OR 或.OR.	逻辑或	操作数均为假时， 结果为假，其余为真	3	(3<2) Or (5>=5)	.T.

表 2-9　逻辑运算真值表（用 T 表示真，用 F 表示假）

操作数 1	操作数 2	逻辑非	逻辑与	逻辑或
A	B	not A	A and B	A or B
F	F	T	F	F
F	T	T	F	T
T	F	F	F	T
T	T	F	T	T

说明：

①逻辑运算中的各运算符的优先级各不相同，NOT（逻辑非）最高，但它低于关系运算符。

②逻辑运算符可用于对多个关系表达式进行逻辑判断。如，要判断 X 是否为[3,9)中的一个数，就要使用逻辑运算符 AND，正确的写法是：

3<=X AND X<9　或　X>=3 AND X<9

又如：A+B=C AND X=Y，也是逻辑表达式。其含义是，当 A+B 等于 C 并且 X 等于 Y 时，该表达式的结果为真。

又如，判断闰年的条件是：①能被 4 整除，但不能被 100 整除；②能被 400 整除。假设用 IntYear 表示输入的年份，则该年是否为闰年的条件如下。

IntYear % 4 = 0 AND IntYear % 100 <> 0 OR IntYear % 400 = 0

而下面一段程序的输出结果是：False。

```
a = 10
b = 20
?a = b
```

2.5.6 名称表达式和宏替换表达式

1. 名称表达式

在 Visual FoxPro 中，许多命令和函数需要提供一个名，如表（.dbf）的文件名、字段名、索引文件名、内存变量和数组名、菜单名、表单名、属性名等。

名不是变量或字段，但是可以定义一个表达式，以代替同名的变量或字段的值。将名称保存到变量或数组元素中，然后再由括号括起该变量或数组，称为名称表达式。然后就可用这个名称表达式来替换字符型变量或数组元素的值。名称表达式的使用语法格式如下：

```
(VarName)
```

其中，VarName 为字符型的内存变量名，名称并不是表达式、变量、数组元素或字段，名称不应以引号括起。

【例 2-14】执行下面的命令，查看运行结果。

```
nvar=10
var_name="nvar"    &&将名称 nvar 保存到变量 var_name 中
STORE 1.23 TO (var_name)    &&该命令的作用等于 store 1.23 to nvar
?(var_name)
?(var_name),nvar       &&  显示结果为 nvar    1.23
```

2. 宏替换表达式

宏替换表达式的使用语法格式如下：

```
&VarName[.cExpression]
```

其中，VarName 为字符型变量，其作用是将存储在字符型内存变量中的字符串替换，即将此变量值作为名称使用。宏替换符号与内存变量名之间不能有空格。宏替换的范围是从“&”起，直到遇到一个点号（.）或空白为止。

同样是完成替换，宏替换的速度比名称表达式慢，但使用或作用更为广泛。

【例 2-15】例 2-14 中，用宏替换表达式替换名称表达式。

```
var_name="nvar"     &&将名称 nvar 保存到变量 var_name 中
STORE 1.23 TO &var_name
?&var_name,nvar       &&  显示结果为 nvar    1.23
```

【例 2-16】宏替换可以用于类型转换，也可以构成表达式，执行下面的命令，查看运行结果。

```
m='25'
?15+&m    &&显示 40
```

【例 2-17】宏替换函数可以嵌套调用。嵌套是指宏替换函数取代的字符串本身可包含宏替换符&。执行下面的命令，查看运行结果。

```
楚留香="老师"
XM='楚留香'
职称='教授'
ZC='职称'
?"&XM."+&xm+"的&zc.是"+&ZC
```

屏幕上显示的结果是：

```
楚留香老师的职称是教授
```

思考题：

①有下面的命令序列，运行后结果是什么？

```
C1= '1'
C2= '2'
?C&C1
?C&C2+C&C1
```

②运行下面的命令序列，查看显示结果。

```
N1="15"
?2*&N1.0
?2*10&N1
?2*10.&N1
```

在使用宏替换表达式时还应注意以下几点，以免在应用中出现错误。

（1）宏替换表达式的自变量必须是字符型的内存变量，不能是数值型或其他任何类型的数据，也不能是字段变量（参见第 3 章）。

（2）符号“&”与自变量之间不得有空格。

（3）当表达式自变量与其尾随字符容易混淆时，必须以“.”号分隔。

2.5.7　表达式的运算顺序

一个表达式由常量、变量、函数、运算符以及圆括号“()”，按照一定的规则组成。该表达式的运算结果与类型是由参与运算的数据和运算符决定的。

表达式的书写规则：

①表达式中的字符不区别大小写，从左到右在同一基准并排书写，不能出现上下标。

②在表达式中可以多次使用圆括号“()”，但圆括号必须成对出现；Visual FoxPro 表达式中的乘号“*”不能省略。

③能用内部函数的地方尽量使用内部函数。

例如，有数学表达式 $\frac{-b+\sqrt{b^2-4ac}}{2a}$，则在 Visual FoxPro 中可写成：

```
(-b + Sqr(b ^ 2 - 4 * a * c)) / (2 * a)
```

④可以用括号改变优先顺序，强令表达式的某些部分优先运行。括号内的运算总是优先于括号外的运算。对于多重括号，总是由内到外。

当表达式中有多个运算符时，表达式要按运算符的优先级来进行运算。

- 同类型运算符的表达式，按本类型内各种运算符的优先级确定运算次序。
- 不同类型运算符的混合表达式，优先级从高到低依次为数值、字符、日期和日期时间运算符、关系运算符和逻辑运算符。
- 同优先级时，表达式按自左向右的次序执行。

例如：500/16-6 与 500/(16-6)的结果分别是 25.25 和 50。

【例 2-18】在 SET EXACT OFF 的状态下，对下面的表达式，判断结果。

```
.NOT.10+8>=2^4 .AND. 5>8 .AND.(.t. .OR. .f.).OR. .NOT.'计算机学院'='计算机'
```

对表达式判断结果的步骤如下：

（1）解括号

```
.NOT. 10+8>=2^4 .AND. 5>8 .AND. .t. .OR. .NOT. '计算机学院'='计算机'
```

（2）做数值、字符运算

.NOT. 18>=16 .AND. 5>8 .AND. .t. .OR. .NOT. '计算机学院'='计算机'

（3）做关系运算

.NOT. .T. .AND. .F. .AND. .t. .OR. .NOT. .T.

（4）做逻辑运算，先做逻辑非

.F. AND .F. AND .T. OR .F.

（5）做逻辑与

.F. OR .F.

（6）最后做逻辑或

最终结果为.F.。

思考题：求表达式('a'+'bc'<'abc'.or.3+2-5>=0).and..not..f.的运算结果。

2.6 内部函数

在 Visual FoxPro 中，为了方便程序开发者的使用，系统将常用的一些数学、统计等公式，以程序的方式编写出来，并将这些程序命名为函数，供其他程序调用。Visual FoxPro 提供了 500 余种内部函数。

Visual FoxPro 内部函数分为数值运算函数、字符串运算函数、日期时间函数、转换函数、测试函数和用户自定义函数（将在第 6 章中给予介绍）等六类。

一般函数的调用方法如下：

函数名（参数列表） '有参函数
函数名（） '无参函数

说明：

①使用函数要注意参数的个数及参数的数据类型。函数中即使没有一个参数，函数调用时圆括号也不得省略。圆括号内的参数称为实参。

②Visual FoxPro 函数的调用可单独出现或出现在表达式中，目的是使用函数取得一值。

③要注意函数的定义域（自变量或参数的取值范围）和值域。如：sqr(x)，要求：x>=0；而函数 exp(23773)的值就超出实数在计算机中的表示范围。

2.6.1 数值运算函数

数值运算函数是指自变量和结果一般均为数值型数据的函数。表 2-10 给出了常用的数值函数。其中的参数 n 代表数值型表达式，en 代表任意表达式。

表 2-10 Visual FoxPro 常用数值运算函数

函数名称	功能描述	举例	结果
ABS(n)	返回由 n 指定数值表达式的绝对值	?ABS(-5)	5
COS(n)	余弦函数，n 为弧度值	COS(PI()/3)	0.50
EXP(n)	返回 e^x 的值，其中 x 是某个给定的数值型表达式 n；e 为自然对数的底，大小约等于 2.71828	EXP(2)	7.39
FLOOR(n)	对于给定的数值型表达式值 n，返回小于或等于它的最大整数	FLOOR(-10.5) FLOOR(10.5)	-11 10

续表

函数名称	功能描述	举例	结果
INT(n)	计算一个数值表达式 n 的值，并返回其整数部分	?INT(12.5) ?INT(6.25 * 2) ?INT(-12.5)	12 12 -12
LOG(n)	计算数值表达式 n 的自然对数	LOG(10)	2.30
LOG10(n)	计算数值表达式 n 的常用对数	LOG10(10)	1.00
MAX(en1,en2[,en3...])	对几个表达式求值，并返回具有最大值的表达式。所有表达式必须为同一数据类型	?MAX(12,34.5,-23)	34.5
MIN(en1,en2[,en3 ...])	对几个表达式求值，并返回具有最小值的表达式	?MIN(12,34.5,-23)	-23
MOD(n1,n2)	同于 n1%n2 运算	MOD(5,-3)	-1
PI()	圆周率函数	PI()	3.14
RAND([n])	根据种子数 n，返回一个 0～1 之间的随机纯小数	?INT(1+100*RAND( ))	91
ROUND(n1,n2)	四舍五入到指定小数位数的数值	SET DECIMALS TO 4 SET FIXED ON ?ROUND(1234.1962,3) ?ROUND(1234.1962,2) ?ROUND(1234.1962,0) ?ROUND(1234.1962,-1) ?ROUND(1234.1962,-2) SET FIXED OFF SET DECIMALS TO 3 ?ROUND(1234.1962, 2)	 1234.196 1234.2000 1234.0000 1230.0000 1200.0000 1234.20
SIGN(n)	当表达式 n 的值为正、负或 0 时，分别返回 1、-1 或 0	?SIGN(10) ?SIGN(0) ?SIGN(-10)	1 0 -1
SIN(n)	返回角度 n 的正弦值	?SIN(PI()/6)	0.500
SQRT(n)	开平方根，n≥0	SQRT(4)	2.00
TAN(n)	求正切函数，n 为弧度值	TAN(PI()/4)	1.00

说明：

（1）对于函数的返回值，系统缺省为 2 位小数。若要改变小数位数，可通过如下命令进行设置：

SET DECIMALS TO n

其中，参数 n 是要设置的小数位数。

（2）参数可以是常数、变量或表达式，还可以是函数（称函数的嵌套）。

（3）表中的三角函数的参数单位为弧度。例如，n 为 30°，则求正弦 Sin()值，需将 n 转

化成弧度，即 Sin(n*3.1415/180)或 Sin(n*PI()/180)，才能得到正确的结果。

（4）SIGN(n)函数，当 N<0 时，返回-1；当 N=0 时，返回 0；当 N>0 时，返回 1。

（5）RAND(n)函数产生一个 0～1 之间的随机小数，包括 0，但不包括 1。RAND()函数中的参数 n，称为种子（Seed）数。其中：

①在第一次发出 RAND()函数时用种子数 n，然后再使用不带 n 参数的 RAND()函数，将得到一个随机数序列。如果第二次发出 RAND()函数时使用同样的种子数 n，那么 RAND()返回同样的随机数序列。

②如果第一次发出 RAND()函数时使用的 n 参数是负数，那么将使用来自系统时钟的种子值。若要获得随机程度最大的数字序列，可以最初用一个负的参数发出 RAND()函数，然后再不带参数发出 RAND()函数。

③如果省略了 n 参数，RAND()函数将使用默认的种子数值 100001。

④为了生成某个范围内的随机整数，可使用以下公式：

Int((上限值 – 下限值 + 1) * RAND + 下限值)

例如，随机产生 100～200（不包括 100 和 200）之间的整数的表达式：Int(99*RAND+101)；如果包括 100 而不包括 200，则表达式为 Int(100*RAND+100)。

（6）当函数的参数是另一个函数时，称为函数的嵌套调用，Visual FoxPro 中允许多次嵌套调用。如：

```
?SQRT(ABS(-36))      &&结果显示 6
```

由于嵌套在内层的函数是外层函数的参数，因此要注意内层函数值的数据类型与外层函数的参数类型是否一致或兼容，值的范围也要符合要求。如：

```
?Log(Cos(3.1415926))
```

由于值的范围不符合要求，因此会产生运行时错误提示，如图 2-5 所示。

图 2-5　错误提示信息

（7）四舍五入函数 ROUND(n1,n2)，其中：

参数 n1 是进行四舍五入的数值型表达式。n2 是要精确到的小数点的位数，可以大于 0、等于 0、小于 0。各种情况下函数的意义如下：

- n2>0，精确到小数点后的 n2 位。
- n2=0，精确到整数个位。
- n2<0，精确到小数点前的|n2|+1 位，小数点前的|n2|位数值均以 0 给出。

值得注意的是，ROUND()函数不理会 SET DECIMALS TO 命令设置的小数位数。若要使 SET DECIMALS TO 命令对 ROUND()函数有效，则应将 SET FIXED 设置为 ON。

2.6.2　常用字符处理函数

字符及字符串处理函数是指函数的处理对象均为字符型数据，但其返回值类型各异，即：

字符型处理函数是指自变量一般为字符型数据的函数。字符型函数的返回值的类型是多样的，有的函数结果是字符型，也有的函数结果却是数值型或逻辑型等。表 2-11 给出了常用的字符型函数。表中的参数 s 表示字符型表达式，en 表示任意数据类型的表达式。

表 2-11　常用的字符型函数

函数名称	功能描述	举例	结果
AT(s1,s2 [,n]) 或 ATC(s1,s2 [,n])	数值型。查找字符表达式 s1 在另一个字符表达式 s2 中第 n 次出现的位置，从最左边开始计数。n 缺省，指第 1 次。 ATC()与 AT()功能类似，但不区分串中的大小写字符	AT([ab],[abcdabefAbdeaabg],3) ATC([ab],[abcdabefAbdeaabg],3)	14 9
CHRTRAN(s1,s2,s3)	字符型。在字符表达式 s1 中，把与 s2 字符表达式字符相匹配的字符替换为 s3 表达式中相应字符	?CHRTRAN('abcdef', 'ace', 'xyz') ?CHRTRAN('abcd', 'abc', 'yz') ?CHRTRAN('abcdef','ace','xyzqrst')	xbydzf yzd xbydzf
LEFT(s,n)	字符型。从 s 的左边取 n 个字符	LEFT("2008 北京奥运会",8)	2008 北京
LEN(s)	数值型。求字符串 s 的长度。	LEN('2008 北京奥运会')	14
LIKE(s1,s2)	逻辑型。s1 字符表达式是否与 s2 字符表达式相匹配，当两串对应字符都匹配时结果为.T.；否则为.F.；s1 中可用通配符：*、?	LIKE([ab*],[abcd])	.T.
LOWER(s)	字符型。将串 s 中的大写字母改为小写	LOWER('FoxPro')	foxpro
OCCURS(s1,s2)	数值型。子串 s1 在串 s2 中出现的次数，若 s1 不是 s2 的子串，则返回 0	OCCURS([ab],[abcdabefabdeadbg])	3
REPLICATE(s,n)	字符型。生成 n 个串 s 组成的新串	REPLICATE('Fox',2)	FoxFox
RIGHT(s,n)	字符型。从 s 的右边取 n 个字符	RIGHT("Visual FoxPro",6)	FoxPro
SPACE(n)	字符型。生成 n 个空格组成的一个字符串	LEN(SPACE(18)-SPACE(5))	23
STUFF(s1,n1,n2[,s2])	字符型。用 s2 替换 s1 中从 n1 开始的 n2 个字符。如果 s2 是空字符串，则从 s1 中删除用 n2 指定的字符数目。如果 n2 是 0，则替换字符串 s2 插入到 s1 中	STUFF([北京奥林匹克运动会],7,10,'运')	北京奥运会
SUBSTR(s,n1[,n2])	字符型。从 s 的第 n1 个字符开始，取 n2 个字符，组成新串，n2 缺省指从 n1 一直取到最后	SUBSTR("北京奥运会",5,6)	奥运会

续表

函数名称	功能描述	举例	结果
TRANSFORM(en,[cF])	字符型。以指定的格式 cF 输出字符表达式或数值表达式 en。详细使用情况，请参见第 7 章@...SAY 的使用	?TRANSFORM(12.34, '$$$$.99') ?TRANSFORM("abc","@!") ?TRANSFORM("abc","!!X") 其中带有“@”表示整体效果	$12.34 ABC ABc
UPPER(s)	字符型。将串 s 中的小写字母改为大写字母	UPPER('Olympiad')	OLYMPIAD
LTRIM(s)	字符型。将字符串 s 中的先导空格删除	LTRIM([␣␣喜迎奥运])	喜迎奥运
TRIM(s)、RTRIM(s)	字符型。将字符串 s 中的后缀空格删除	LEN(TRIM('北京奥运␣␣'))	8
ALLTRIM(s)	字符型。将字符串 s 中的前后空格删除	LEN(ALLTRIM('␣␣喜迎奥运␣'))	8

2.6.3 常用日期和时间类函数

日期和时间类函数提供时间和日期的信息。表 2-12 给出了常用的日期和时间函数。表中的自变量 d 代表日期型表达式、t 代表日期时间型表达式、dt 代表日期或日期时间型表达式。

表 2-12 常用日期时间型函数

函数名称	功能描述	举例	结果
DATE()	日期型。返回系统当前日期	?DATE()	08/08/11
DATETIME()	日期时间型。返回系统当前日期时间（12 小时制）	?DATETIME()	08/08/11 06:26:45 PM
DAY(dt)	数值型。返回表达式中是月内的第几天	?DAY(DATETIME())	8
HOUR(t)	数值型。返回表达式中的小时数（24 小时制）	HOUR(DATETIME())	18
MIMUTE(t)	数值型。返回表达式中的分钟数	MINUTE(DATETIME())	26
MONTH(dt)	数值型。返回表达式中的月份	?MONTH(DATE())	8
SEC(t)	数值型。返回表达式中的秒数部分	SEC(DATETIME())	45
TIME()	字符型。返回系统当前时间（24 小时制）	?TIME()	18:26:45
YEAR(dt)	数值型。返回表达式中的年份	?YEAR(DATE())	2011

对于日期和日期时间型数据，缺省的显示格式是按传统的“mm/dd/yy hh:mm:ss”格式进行。用户可根据需要自行设置其显示格式，设置不同，对于同一日期会得到不同的显示结果。

表 2-11 给出的结果是按传统格式显示的，并假设此时的系统日期时间如下：

{^2011-8-8 18:26:45}

2.6.4 常用类型转换类函数

类型转换函数的作用是将自变量的数据类型转换成另一类型。常用的数据类型转换函数如表 2-13 所示。表中，假设已使用了世纪显示设置命令：SET CENTURY ON，并假设此时的系统日期时间为：{^2011-8-8 18:26:45}。

表 2-13　常用类型转换函数

函数名称	功能描述	举例	结果
ASC(s)	数值型。返回表达式 s 中第一个字符的机内码值，忽略其他字符	?ASC(“FoxPro”)	70
CHR(n)	字符型。n 是一个大于 0 的正整数，CHR()返回与之对应的 ASCII 字符或汉字	?CHR(65)	A
Val(s)	数值型。将数字组成的字符表达式转换成数字值	?Val('123.456abc') ?Val("1.23e5") ?Val("-1.23e5") ?Val("abc-1.23e5")	123.46 123000.00 -123000.00 0.00
STR(n1,[n2[,n3]])	字符型。将数值 n1 转换成总长度为 n2、小数为 n3 位的字符串	?STR(123.456,6,2)	123.46
CTOD(s)	日期型。将形如日期型数据的字符串转换成对应的日期型数据	?CTOD('08/08/2011') ?CTOD('^2011-8-8')	08/08/2011 08/08/2011
CTOT(s)	日期时间型。将形如日期时间型数据的字符串转换为对应的日期时间型数据	?CTOT([08/08/2011 18:00])	08/08/2011 06:00 PM
DTOC(d [,1])	字符型。将日期型数据转换成对应的字符串	?DTOC({^2011/08/08}) ?DTOC({^2008/8/8},1)	08/08/2011 20110808
DTOR(n)	数值型。将数值表达式 n（度）转换为弧度。如果是一个用“度：分：秒”格式表示的度数应先转换为实数形式	?sin(dtor(30))	0.50
DTOS(dt)	字符型。将指定日期或日期时间表达式 dt 转换成八位字符串。此函数不受命令 SET DATE |CENTURY 的影响	Dt={^2011-8-8 18:36:45} ?dtos(dt)	20110808
TTOC(t [,1|2])	字符型。将日期时间型数据转换成对应的字符串	SET HOUR TO 24 ?TTOC({^2011/08/8 18:26:35}) ?TTOC({^2011/08/8 18:26:35},1)	08/08/2011 18:26:35 20110808182635
RTOD(n)	数值型。将表示弧度的数值表达式 n 转换成度	? ?RTOD(PI()/3)	60.00
DTOT(d)	日期时间型。将日期型表达式 d 转换成日期时间型值	?DTOT(DATE())	08/08/2011 18:26:35
TTOD(t)	日期型。从日期时间表达式 t 中返回一个日期值	SET CENTURY ON ?TTOD(CTOT([08/08/2011 18:00]))	08/08/2011

转换函数的几点说明。

（1）在 DTOC(d [,1])函数中，无参数 1 时结果为 MM/DD/YY[YY]格式，有参数 1 时结果为 YYYYMMDD 格式。

（2）在 TTOC(t [,1|2])函数中，如果 t 中只包含时间，则系统把日期“12/30/1899”添加到 t 中；如果 t 中只包含日期，则系统将默认的午夜时间“12:00:00A.M.”添加到 t 中。

参数 1：表示返回一个有 14 位的字符串，格式为“yyyy:mm:dd:hh:mm:ss”，且格式不受 SET CENTURY 或 SET SECONDS 的影响。

参数 2：TTOC()返回的字符串只包含日期时间表达式的时间部分。字符串中是否包含秒由 SET SECONDS 和 SET DATE 决定。如果 SET DATE 设为 LONG 或 SHORT，则字符串的格式由“控制面板”中的相应设置决定。

（3）在 DTOT(d)中，函数返回的日期时间型值的格式由 SET DATE 和 SET MARK 的当前设置值决定。若未提供世纪值，则假定为二十世纪。

DTOT()向日期中加上午夜(12:00:00 A.M.)的默认时间来生成有效的日期时间值。

在 TTOD(t)中，如果表达式 t 只包含时间，则系统将默认的日期“12/30/1899”添加到 t 中，并返回这个默认的日期。

（4）函数 VAL(s)说明如下：

- 如果 s 以非数字字符开头，转换结果为 0。
- 如果 s 以数字字符、“.”、“+”、“-”开头，转换结果为：截至到第 1 个非数字字符出现的位置为止。如 VAL(".123A")的结果为.123，且受 SET DECIMALS TO 的影响。
- 如果字符串的前面部分如同科学记数常量的形式，则将其转为对应的数值型数据。
- 字符型表达式 s 最多由 16 位数字组成，若超过 16 位，则对其四舍五入。

（5）函数 STR(n1[, n2 [, n3]])说明如下：

- n1：要转换成字符串的数值型数据。
- n2：要转换成的字符串总长度。
- n3：要转换成的字符串的小数位数，n3>0，表示精确到小数点后的位数；n3=0（缺省），表示不保留小数位；n3<0，则出错。系统转换时，对 n3 位后面的紧跟一位数值作四舍五入处理。
- 当 n2、n3 均缺省时，表示将 n1 转换为长度为 10 个字符的仅由 n1 的整数部分组成的字符串。
- 小数点也占 1 个字符位置。
- 转换时，首先保证整数部分能精确转换，然后剩余的长度才考虑小数部分。如果 n2 连整数部分都无法容纳，则结果将以 n2 个“*”给出。例如，执行如下命令：

```
a=123456.378
?STR(a,8,5),STR(a,5)
```

屏幕上显示的结果是：

```
123456.4        *****
```

（6）CHR(n)函数中的正整数的物理意义是字符的机器码值，它的取值范围分别是：

- n≤127：标准 ASCII 字符的机器码值。
- 128≤n≤255：扩展 ASCII 字符的机器码值。
- 41377≤n≤43518：标准图形符的机器码值。
- 45217≤n≤55289：一级汉字的机器码值。
- 55457≤n≤63485：二级汉字的机器码值。

ASC()与 CHR()函数功能相反。

【例 2-19】验证汉字“成”的机内码是否在 45217≤n≤55289 之中。

```
?ASC("成")
```

命令执行后，屏幕显示的结果为 46025，说明在机内码 45217≤n≤55289 之中。

而执行下面的命令，则显示的结果是：VFP 数据库。

```
?CHR(86)+CHR(70)+CHR(80)+CHR(51965)+CHR(48861)+CHR(49122)
```

2.6.5　测试函数

测试函数用来测试操作对象的状态，表 2-14 给出常用测试函数。函数中的自变量 n 表示工作区号，s 表示表的别名。同时假设有一个表 xscj.dbf，其中有 100 条记录，并已经打开。

表 2-14　常用测试函数

函数名称	功能描述	举例	结果
ALIAS(n | s)	字符型。返回当前表或指定工作区表的别名。 如果省略参数 n 或 s，函数将返回在当前工作区中打开的表的别名。如果当前或指定工作区没有打开的表，则函数返回空字符串	USE xscj ?alias( ) select 0 use xscj again alias cj ? alias( )	xscj cj
BETWEEN(vt,vl,vh)	逻辑型或 Null 值。测试 vt 是否在 vl～vh 之间，在其中返回.T.，否则返回.F.。vl 一定小于或等于 vh；如果 vl 或 vh 为 Null 值，则返回 Null 值	?BETWEEN(5,2,10) ?BETWEEN('C','a','z')	.T. .F.
BOF([n|s])	逻辑型。测试记录指针是否指向表文件的开始标记，指向则返回.T.，否则返回.F.；空表时 BOF()为.T.	GO TOP SKIP -1 ?BOF()	.T.
DBF([s | n])	字符型。返回指定工作区中打开的表名，或根据表别名返回表名（包含的路径）。 如果省略 s 和 n，DBF() 返回当前工作区中打开的表名；如果指定的工作区中没有打开的表，DBF() 返回一个空字符串；如果表没有 s 别名，则 Visual FoxPro 产生错误	select 0 use xscj alias cj ?dbf("cj")	D:\jxgl\data\xscj.dbf
DBC( )	字符型。返回当前数据库的名称和路径。如果没有当前数据库，则 DBC()返回空字符串	Open DataBase jxgl ?dbc()	d:\jxgl\data\jxgl.dbc
DELETED([n|s])	逻辑型。删除标记测试函数：测试当前记录是否加有逻辑删除标记，加有则返回.T.，否则返回.F.	DELETE ?DELETED()	.T.
EMPTY(Exp)	逻辑型。当表达式为该数据类型规定的“空”值时，返回.T.，否则返回.F.	?EMPTY(0)	.T.
EOF([n|s])	逻辑型。测试记录指针是否指向表文件的结束标记，指向则返回.T.，否则返回.F.；空表时 EOF()为.T.	List ?EOF()	.T.

续表

函数名称	功能描述	举例	结果
FCOUNT([n | s])	数值型。返回表中的字段数目	?fcount()	9
FIELD(n1 [, n | s])	字符型。根据编号 n1 返回表中的字段名。如果 n1 等于 1，则返回表中的第一个字段名；如果 n1 等于 2，则返回第二个字段名，依此类推	?field(2)	姓名
FILE(s)	逻辑型。如果在磁盘上找到指定的文件，则返回.T.	?FILE("xsda.dbf")	.T.
FOUND([n | s])	逻辑型。如果最近 CONTINUE、FIND、LOCATE 或 SEEK 命令执行成功，则函数的返回值为“真”(.T.)。可以使用这个函数来判定子表是否有记录和父表的记录相匹配	USE xsda INDEX 姓名 TAG xm FIND 阿童木 ?FOUND()	.T.
FSIZE(cfn[,n| s] | FN)	数值型。以字节为单位，返回指定字段或文件的大小。其中，参数 cfn 和 FN 分别表示字段名或文件名	?fsize("姓名")	8
IIF(lExp,Exp1,Exp2)	条件测试函数；当 lExp 为真时，返回 Exp1 的值，否则返回 Exp2 的值。该函数的数据类型由 Exp1 或 Exp2 的数据类型决定	?IIF(性别='男','樱木花道','花仙子')	花仙子
ISNULL(Exp)	逻辑型。当 Exp 的值为 Null 时，返回.T.，否则返回.F.	x=.null. ?ISNULL(x)	.T.
RECCOUNT([n|s])	数值型。返回表的物理记录条数，注意空表时，该函数为 0	?RECCOUNT()	100
RECNO([n|s])	数值型。返回当前记录的物理记录号，注意当 BOF()为.T.时，RECNO()为 1；EOF()为.T.时，RECNO()为记录条数+1	GO 10 ?RECNO()	10
RECSIZE([n | s])	数值型，测试表中记录的大小。该函数显示的记录的大小与 DISPLAY STRUCTRUE 命令显示的结果相同	?recsize()	44
SEEK(eExp [, n | s [,n1| cfn| ctn]])	逻辑型。在一个已建立索引的表中搜索一个记录第一次出现的位置，该记录的索引关键字与指定表达式相匹配。seek()函数返回一个逻辑值，指示搜索是否成功。 如果省略了 n 或 s，则在当前工作区中搜索表。其中：①n1 参数，指定用来搜索关键字的索引文件或索引标识编号。②cfn 参数：指定用来搜索索引关键字的.IDX 文件。③ctn 参数：指定用来搜索索引关键字的.CDX 文件的标识。标识名称可以来自结构文件.CDX，也可以来自任何打开的独立.CDX 文件。 说明：如果存在相同的.IDX 文件和标识名称，优先使用.IDX 文件	use xsda order xm ?seek('阿童木')	.T.

续表

函数名称	功能描述	举例	结果
SELECT([0 | 1 | s])	数值型。返回当前工作区编号或未使用工作区的最大编号。其中：0（或省略），函数返回当前工作区的编号；1，函数返回未使用工作区的最大编号；s，指定表别名，函数返回其所在工作区编号	select 10 ?select() ?select(0)	10 10
TYPE(cExp)	字符型。返回字符表达式其内容的数据类型	?TYPE('(12 * 3) + 4') ?TYPE('date( )') ?TYPE('.f. or .t.') ?TYPE('answer=42') ?TYPE('$19.99')	N D L U Y
USED([n|c])	逻辑型。如果在指定的工作区 n 中打开了一个表 c，函数就返回“真”（.T.）；否则，返回“假”（.F.）。	UES 学生 ?USED()	.T.
VARTYPE(Exp)	字符型。与 TYPE()函数功能相同，但是 VARTYPE()更快，而且表达式外面不需要引号	?VARTYPE(date( ))	D

下面给出关于测试函数的一些说明。

（1）函数 EMPTY()在自变量为“空”值时，将返回一个逻辑真。表 2-15 给出了各类数据的“空”值定义。

表 2-15　不同数据类型的“空”值定义

数据类型		“空”值定义	数据类型	“空”值定义
数值类型数据	数值型（N）	0	字符型（C）	空串、空格、制表符、回车符、换行符及它们的组合
	货币型（Y）	0	日期型（D）	空，如 CTOD("")
	浮点型（F）	0	日期时间型（T）	空，如 CTOT("")
	双精度型（B）	0	逻辑型（L）	.F.
	整型（I）	0	备注型（M）	空（无内容）
二进制型		空（0h）或仅含 0 位	通用型（G）	空（无 OLE 对象）
变体型		空（0h）或仅含 0 位		

（2）TYPE(cExp)函数返回的数据，如表 2-16 所示。

表 2-16　TYPE()函数所返回的字符值及其对应的数据类型

数据类型	返回的字符	数据类型	返回的字符
字符型	C	备注型	M
数值型	N	对象型	O
货币型	Y	通用型	G
日期型	D	Screen（用 SAVE SCREEN 命令建立）	S
日期时间型	T	未定义的表达式类型	U
逻辑型	L		

（3）函数 VARTYPE(Exp[,lExp])的返回值，如表 2-17 所示。

表 2-17　VARTYPE()函数返回的数据类型

返回的字符	数据类型	返回的字符	数据类型
N	数值型、整型、浮点型或双精度型	C	字符型或备注型
Y	货币型	D	日期型
L	逻辑型	T	日期时间型
O	对象	X	Null
G	通用型	U	未知

注意：当表达式 Exp 的值为.NULL.时，将根据 lExp 逻辑表达式的值决定返回的值，如果 lExp 的值为.T.，则返回 Exp 的原数据值；如果 lExp 的值为.F.或缺省，则返回“X”以表明 Exp 的运算结果为.NULL.。

【例 2-20】分析下面命令序列的执行结果。

```
x="中国人民"
y=121
y=.NULL.
z=123.4567
?VARTYPE(x),VARTYPE(y),VARTYPE(y,.T.),VARTYPE(z),VARTYPE(u)
```

结果显示：

```
C   X   N   N   U
```

2.6.6　其他函数

表 2-18 给出了其他几个常用的功能函数。

表 2-18　其他函数

函数名	功能描述	操作	显示结果
MESSAGEBOX(cExp1 [,n [, cExp2]])	数值型。显示一个用户自定义对话框	?MessageBox("真的要退出吗?",4+32+0, "提示信息")	
CURDIR(cExp)	字符型。给出当前磁盘的当前目录	SET DEFAULT d:\xsda ?CURDIR()	\xsda\
SYS(5)	字符型。返回当前 Visual FoxPro 的默认驱动器	?SYS(5)	D
SYS(2004) 或 HOME()	字符型。返回启动 Visual FoxPro 的目录或文件夹名称	?HOME()	显示 Visual FoxPro 安装启动目录名称
DISKSPACE([cExp])	数值型。返回默认磁盘驱动器上可用的字节数。无参数时，默认的驱动器或卷由 SET DEFAULT 命令指定	SET DEFAULT TO D ?DISKSPACE()	5211013120
RGB(nR, nG, nB）	根据红 nR、绿 nG、蓝 nB 颜色成份合成一个颜色值	_SCREEN.BACKCOLOR= RGB(255,0,0)	屏幕显示红色

续表

函数名	功能描述	操作	显示结果
COL()和 ROW()	判断光标列（行）位置函数	CLEAR @5,5 SAY '' &&定位于 5,5 处 @ROW()+6,COL() SAY '胜利' @ROW()+1,$+4 SAY '万岁!'	胜利 万岁!
INKEY([<n>][, cH]))	数值型。检测用户所击键对应的 ASCII 码函数，数值表达式以秒为单位等待击键的时间	K=INKEY([<n>]) 执行该命令并按下 A 键后 ?k	65

其他函数的几点说明：

（1）INKEY()函数。INKEY()函数的参数 n，以秒为单位，指定 INKEY()函数对击键的等待时间。如果不包含 n，INKEY()函数立即返回一次击键的值；如果 n 为 0，INKEY()函数一直等待到有击键为止。

INKEY()函数的参数 cH，决定是否显示或隐藏光标，或者检查鼠标单击。若要显示光标，cH 为 S；若要隐藏光标，cH 为 H；如果 cH 既包含 S 又包含 H，则使用后一个字符的设置。

（2）有关 MessageBox 函数的详细介绍和使用方法，请参见第 9 章有关内容。

2.7　Visual FoxPro 的可视化设计工具

为了减少用户工作量，高效开发出高质量的 Visual FoxPro 应用程序，Visual FoxPro 提供了一整套的可视化设计工具供用户使用。这些工具可分为向导、设计器、生成器三大类。各个类型的设计工具，使用方法均大体雷同，故对每个类型的设计工具只需讲解一个，即可逐类旁通。

2.7.1　向导（Wizard）

向导提供了用户完成某些工作所需要的详细操作步骤，在这些步骤的引导下，用户可以一步一步方便地完成任务，不用编程就可以创建良好的应用程序界面，并完成许多与数据库有关的操作。Visual FoxPro 提供了 25 种向导工具。常用的向导有：表向导、报表向导、表单向导、查询向导等。

向导的启动有两种方法，现以建立表单文件为例讲解向导的使用步骤。

1. 通过“文件”菜单

操作步骤如下：

（1）单击“文件”菜单的“新建”命令，打开“新建”对话框，如图 2-6 所示。

（2）选择有关的文件类型，例如选“表单”。

（3）单击“向导”按钮，即可启动“向导”，用户在该向导的指引下，可逐步完成该类文件的创建。

2. 通过“工具”菜单

操作步骤如下：

（1）单击“工具”菜单中的“向导”命令，在展开的“向导”二级子菜单中，选择“表

单”命令，启动创建“表单”向导，如图 2-7 所示。

图 2-6 “新建”对话框

图 2-7 使用“工具”菜单启动向导

（2）在“向导选择”对话框中，单击需要的向导图标。

Visual FoxPro 提供了多个向导工具，常用向导工具的功能如表 2-19 所示。

表 2-19 向导功能表

向导名称	功能
表向导	在表结构基础上创建一个新表
报表向导	利用单独的表来快速创建报表
一对多报表向导	从相关的数据表中快速创建报表
标签向导	快速创建一个标签
分组/统计报表向导	快速创建分组统计报表
表单向导	快速创建一个表单
一对多表单向导	从相关的数据表中快速创建表单
查询向导	快速创建查询
交叉表向导	创建交叉表查询
本地视图向导	利用本地数据创建视图
远程视图向导	创建远程视图
导入向导	导入或添加数据
文档向导	从项目文件和程序文件的代码中产生格式化的文本文件
图表向导	快速创建图表
应用程序向导	快速创建 Visual FoxPro 的应用程序
SQL 升迁向导	引导用户利用 Visual FoxPro 数据库功能创建 SQL Server 数据库
数据透视表向导	快速创建数据透视表
安装向导	从文件中创建一整套安装磁盘
邮件合并向导	创建一个邮件合并文件

2.7.2　设计器（Designer）

Visual FoxPro 系统的设计器为用户提供了一个友好的图形界面。用户可以通过它创建并定制数据表结构、数据库结构、报表格式和应用程序组件等。常用的设计器有：表设计器、查询设计器、视图设计器、报表设计器、数据库设计器、菜单设计器等。和向导不同，设计器是一个不分步骤的集成设计环境。

以打开数据库设计器为例，其操作步骤如下：

（1）打开“文件”菜单，单击“新建”命令，打开“新建”对话框。

（2）在“新建”对话框中，选中“数据库”单选按钮，单击“新建文件”按钮，打开“创建”对话框。

（3）输入数据库文件名，选择文件保存路径，单击“保存”按钮返回，这时就会打开“数据库设计器”窗口，如图 2-8 所示。

图 2-8　数据库设计器

Visual FoxPro 有多种设计器，其功能如表 2-20 所示。

表 2-20　设计器功能表

名称	功能
表设计器	创建表并建立索引
查询设计器	用于创建本地表的查询
视图设计器	用于创建远程数据源的查询并可更新查询
表单设计器	创建表单，用以查看并编辑表的数据
报表设计器	创建报表，以便显示及打印数据
标签设计器	创建标签布局以便打印标签
数据库设计器	建立数据库，查看并创建表之间的关系
连接设计器	为远程视图创建连接
菜单设计器	创建菜单或快捷菜单

2.7.3　生成器（Builder）

生成器一般附属于设计器，其作用是协助设计器通过交互式操作，自动生成表达式、程

序过程等，从而简化操作、提高效率。在生成器中，用户只要告诉它做什么，至于如何做则是生成器自己的事情了。

常用的生成器有：组合框生成器、命令组生成器、表达式生成器、表单生成器、列表框生成器等。

以打开表单生成器为例，其操作步骤如下：

（1）打开“文件”菜单，单击“新建”命令，打开“新建”对话框。

（2）选中“表单”单选按钮，单击“新建文件”按钮，打开“表单设计器”窗口。

（3）在“表单设计器”窗口中，在表单上单击鼠标右键，然后在弹出的快捷菜单中单击“生成器”命令，打开“表单生成器”对话框，如图 2-9 所示。

图 2-9 “表单生成器”对话框

Visual FoxPro 具有多种类生成器，其功能如表 2-21 所示。

表 2-21 生成器功能表

名称	功能	名称	功能
自动格式生成器	生成格式化的一组控件	列表框生成器	生成列表框
组合框生成器	生成组合框	选项生成器	生成选项按钮
命令组生成器	生成命令组按钮框	文本框生成器	生成文本框
编辑框生成器	生成编辑框	表达式生成器	生成并编辑表达式
表单生成器	生成表单	参照完整性生成器	生成参照完整性规则
表格生成器	生成表格		

2.8 项目管理器

项目管理器也是 Visual FoxPro 提供的一种辅助设计工具。一个有一定规模的数据库应用系统，不仅包含了各种类型的文件，而且每一类文件的数目也不止一个。Visual FoxPro 的项目管理器把每一类文件的组成作为一类模块，如表模块、表单模块、报表模块等，通过创建一个项目文件把应用系统的所有组成模块统一管理起来。用户可利用项目管理器简便地、可视化地创建、修改、调试和运行项目中各类文件，还能把应用项目集成为一个在 Visual FoxPro 环境下运行的应用程序，或者编译（连编）成脱离 Visual FoxPro 环境而运行的可执行文件。可以认为项目管理器是 Visual FoxPro 的管理和控制中心。

2.8.1　创建项目

1. 创建方法

通常可以使用两种方法创建一个新的项目文件，一种是使用 Visual FoxPro 的菜单命令，另一种是在命令窗口输入命令。具体操作如下所述。

（1）菜单操作：打开“文件”菜单，单击“新建”命令，选择文件类型为“项目”，单击“新建文件”按钮，为文件取名，单击“保存”按钮。

（2）命令窗口：CREATE PROJECT <项目文件名>。

在使用以上两种方法后，都可以创建一个新的项目文件，项目文件的扩展名是.PJX。在 Visual FoxPro 系统的窗口中会出现一个项目管理器来表示项目文件，同时在系统的菜单栏中还会出现“项目”菜单，提供对项目文件操作的相关命令。项目管理器的界面如图 2-10 所示。

2. 项目的关闭与保存

在“项目管理器”对话框中，单击对话框右上角的“关闭”按钮（或按下 Esc 键，或按下 Ctrl+Q 组合键），也可按下 Ctrl+W 组合键（保存），系统都会弹出“系统信息提示”对话框，如图 2-11 所示。

图 2-10　Visual FoxPro 的项目管理器

图 2-11　“系统信息提示”对话框

如果项目中没有任何文件，可将其删除或保存，在图 2-11 中，单击“删除”按钮，可删除该项目，单击“保持”按钮，可保存该项目。

说明：

- 如果单击图 2-6 中的“向导”按钮，系统将弹出“应用程序向导”对话框，利用它可创建项目并同时生成一个应用程序框架。
- 对于已存在的项目，双击该项目文件名同样会激活项目管理器，并将该项目所在的路径（文件夹）作为当前操作（如保存文件）时的默认路径（文件夹）。

3. 项目管理器界面组成

从项目管理器界面可以看出，项目管理器由以下几部分组成：

（1）标题栏。项目管理器标题栏显示的标题就是项目文件的主文件名，在创建项目文件时，默认项目文件名为“项目 1、项目 2、……”，用户可以删除，输入自己选择的项目文件名。

（2）选项卡。标题栏下方是选项卡，共有 6 个。选择不同的选项卡，则在下面的工作区显示所管理的相应文件的类型。现对各选项卡的意义做如下说明。

- 全部——可显示和管理应用项目中使用的所有类型的文件，“全部”选项卡包含了它

右边的五个选项卡的全部内容。

- 数据——管理应用项目中各种类型的数据文件，数据文件有数据库、自由表、视图、查询文件等。
- 文档——显示和管理应用项目中使用的文档类文件，文档类文件有表单文件、报表文件、标签文件等。
- 类——显示和管理应用项目中使用的类库文件，包括 Visual FoxPro 系统提供的类库和用户自己设计的类库。
- 代码——管理项目中使用的各种程序代码文件，如程序文件（.PRG）、API 库和用项目管理器生成的应用程序（.APP）。
- 其他——显示和管理应用项目中使用的、但在以上选项卡中没有的文件，如菜单文件、文本文件等。

（3）工作区。从图 2-10 可以看出，项目管理器的工作区采用分层结构的方式来组织和管理项目中的文件。左边的最高一层用明确的标题标识了文件的分类，单击“+”号可展开该类文件的下属组织层次，“+”号也变成了“-”号。单击“-”号可把展开的层次折叠起来，“-”号变成了“+”号。用鼠标逐层单击某类文件的“+”号，展开到最后是没有“+”或“-”号的文件名，选中某个文件后，就可以用项目管理器的命令按钮来修改和运行这个文件。

（4）命令按钮。项目管理器右边的命令按钮为工作区窗口的文件提供各种操作命令。

2.8.2 项目管理器的使用

在开发一个数据库应用系统时，可以有两种方法使用项目管理器，一种方法是先创建一个项目管理文件，再使用项目管理器的界面来创建应用系统所需的各类文件；另一种方法是先独立地建立应用系统的各类文件，再把它们一一添加到一个新建的项目管理文件中。究竟使用哪种方法，完全看开发者的个人习惯，项目管理器中的“新建”和“添加”命令按钮给开发者提供了选择的自由。

1. 命令按钮的功能

在刚创建和打开一个项目文件后，可看到以下命令按钮，它们的功能如下：

- 新建(N)... ——在工作区窗口选中某类文件后，单击“新建”按钮，新建的文件就添加到该项目管理器中。
- 添加(A)... ——可把用 Visual FoxPro“文件”菜单下的“新建”命令和“工具”菜单下的“向导”命令创建的各类独立的文件添加到该项目管理器中，把它们统一地组织管理起来。
- 修改(M) ——使用设计器界面来修改项目中已存在的各类文件。
- 打开(O) ——可打开项目中已存在的如数据库文件等。
- 浏览(B) ——可浏览表中记录。
- 关闭(C) ——关闭已打开的文件，如数据库文件等。
- 运行(U) ——在工作区窗口选中某个具体文件后，可运行该文件。
- 移去(V)... ——把选中的文件从该项目中移去。
- 连编(D)... ——把项目中相关的文件连编成应用程序和可执行文件。

上述命令按钮并不是一成不变的，若在工作区打开一个数据库文件，“运行”按钮会变成“关闭”按钮；打开一个自由表文件，“运行”按钮会变成“浏览”按钮，单击该按钮，系统

提供浏览方式显示表的记录。

此外上述命令按钮有时是可用的，有时是不可用的。它们的可用和不可用状态与工作区的文件选择状态相对应，如在“全部”选项卡的工作区中，各种文件类型都是“+”号没有展开，也就是没有选中要操作的具体文件，此时像“新建”、“运行”等按钮呈现灰色，表示是不可用的。如果在工作区展开某类文件，如单击“文档”类文件，选中了“表单”类文件，这些按钮就变成了黑色，表示是可用的，现在就可修改和运行选中的表单文件了。

2. 项目管理器中命令的操作

在项目管理器中管理文件，可进行新建、添加、运行、重命名等各种操作。在工作区窗口用鼠标单击展开各类文件和选择要操作的文件，可用以下几种方式进行。

（1）使用命令按钮：即用前面介绍的项目管理器界面右边的命令按钮，如单击按钮“新建”、“添加”、“运行”等。

（2）使用“项目”菜单：启动了项目管理器之后，会在 Visual FoxPro 的菜单栏自动添加“项目”菜单。“项目”菜单下的命令除了包括项目管理器的按钮命令外，还有不同的内容，如图 2-12 所示。

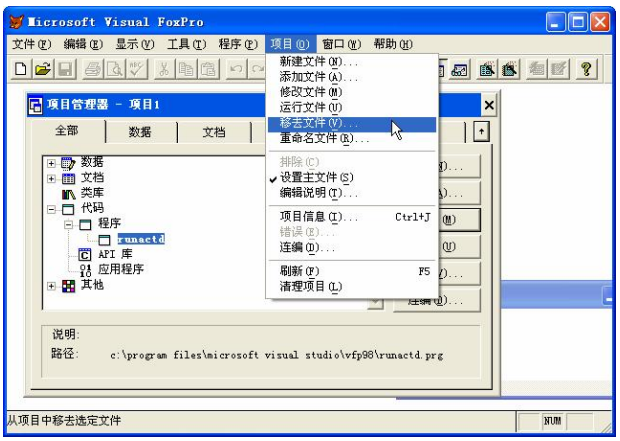

图 2-12　“项目”菜单下的命令

可以用“项目”菜单下的命令对项目管理器管理的文件进行“重命名”和“设置主文件”等操作，这些操作是项目管理器的命令按钮没有提供的。

（3）使用快捷菜单：在项目管理器的工作区选择了某类文件后，单击鼠标右键，可弹出一个快捷菜单，如图 2-13 所示，快捷菜单的命令和命令按钮以及“项目”菜单下的命令也有所不同。选择其中的“生成器”命令，可使用一个名为“应用程序生成器”的辅助工具来把项目中设计的大部分文件生成一个应用程序。

3. 项目管理器对应用程序的开发运用

在开发一个数据库应用系统时，创建一个项目管理器后，首先通过项目管理器的“新建”和“添加”命令按钮把应用系统所有的各类文件组织到项目中，然后再对项目中的各类文件进行调试和修改，就可选择应用系统中程序运行的起点文件——主文件，这个文件可以是调用其他程序的主程序，或者是调用其他表单的主表单。图 2-13 中的例子就是把一个实际应用项目中名称为“main”的程序文件选作主文件。在选择了“主文件”后，就可以使用项目管理器右下方的“连编”命令按钮，通过弹出的“连编选项”对话框（见图 2-14），把项目文件生成为

可在 Visual FoxPro 环境下运行的应用程序或脱离 Visual FoxPro 运行的可执行文件。

图 2-13　“项目管理器”中的快捷菜单

图 2-14　“连编选项”对话框

2.8.3　定制项目管理器

从图 2-8 可看出，项目管理器是一个 Windows 类型的对话框，它有 2 个特殊按钮，一个关闭按钮和一个缩放按钮，它们的意义如下。

（1）关闭按钮

关闭项目管理器，常用于退出 Visual FoxPro 时。但如果在退出 Visual FoxPro 时并没有关闭项目管理器，则下次启动 Visual FoxPro 时将自动打开前一次正在使用的项目管理器。

（2）缩放按钮

屏幕的显示范围毕竟是有限的，为了避免项目管理器占用太多的屏幕，可按下此按钮，此时项目管理器将折叠（收缩）成一个工具栏窗口，原本向上的收缩箭头按钮则变为向下的展开箭头按钮“”，如图 2-15 所示。再次单击缩放按钮，项目管理器又将展开恢复原样。

图 2-15　被缩成工具栏的项目管理器

1. 改变大小和位置

（1）改变项目管理器的位置：鼠标拖动项目管理器的标题栏可改变其在屏幕上的位置。

（2）改变项目管理器的大小：鼠标拖动项目管理器的四边可改变它的长或者宽，拖动它的四角可同时改变它的长和宽。

2. 分离项目管理器中的选项卡

在项目管理器折叠之后，可把其中的一个选项卡分离出来，以方便单独使用。方法是用鼠标向下拖动其中一个选项卡，就可把它分离出来了。在图 2-16 的下面部分是分离出来的“代码”选项卡，这时在“代码”标签的右边有个图钉图标“”，单击这个图标可设置为该选项卡在最前面显示，再单击这个图标可取消设置。单击图钉图标右边的关闭按钮，可把分离出来的选项卡还原到原来的位置。

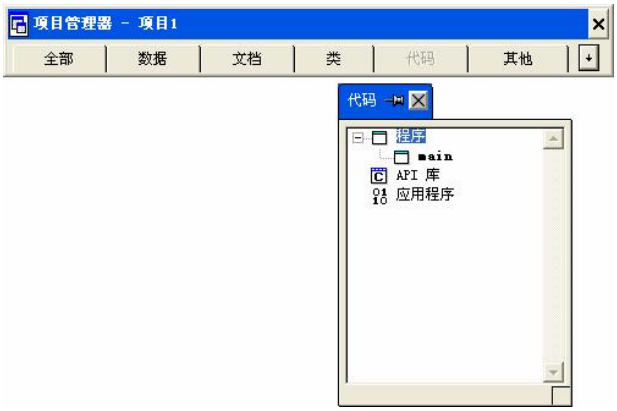

图 2-16　分离选项卡

3. 停放项目管理器

可用鼠标拖动项目管理器的标题栏到 Visual FoxPro 主窗口的菜单栏和工具栏附近，项目管理器就变成了系统工具栏的一个工具条，这时它没有工作区窗口，因此不能使用。但可用分离选项卡的方法，把其中某个选项卡分离出来单独使用。

2.9　在 Visual FoxPro 环境下使用操作系统命令创建用户文件夹

要开发一个应用程序，用户最好先为自己创建一个独立的文件夹。这既可以在 Windows 环境下创建，又可以在 Visual FoxPro 命令窗口下使用操作系统命令创建。在 Visual FoxPro 环境中创建用户文件夹，可使用下面的命令。

RUN|! MD [Driver:][Path]FolderName

其中：

- RUN|!：指定在 Visual FoxPro 命令窗口执行操作系统的有关命令或运行非 Visual FoxPro 程序。
- MD：操作系统中创建目录的命令。
- Driver：指定要创建的用户文件夹所在的磁盘驱动器符，缺省时为当前盘。
- Path：指定要创建的用户文件夹的路径，缺省时为当前目录。
- FolderName：指定要创建的用户的文件夹名。

【例 2-21】在 Visual FoxPro 环境中创建一个用户子目录：d:\VISUAL FOXPRO\myFolder，并把它设置为缺省目录。

```
!MD d:\VISUAL FOXPRO\myFolder
SET DEFAULT TO d:\VISUAL FOXPRO\myFolder
```

其中，指定使用默认的驱动器、目录或文件夹命令 SET DEFAULT TO，也可使用如下命令进行对话式的设定：

```
CD | CHDIR cPath | ?
```

参数 cPath 表示完整的路径，如 d:\Jxgl；参数“?”表示在设置默认路径时，打开一个对话框，供用户进行路径的选择。

在 Visual FoxPro 中，同一个文件夹只需创建一次，以后只要未被删除就可打开直接使用。