返回总目录

GetLatestVersion 方法

获得源代码管理器中项目的一个文件的最新版本，并将一个只读版本复制到您本地的驱动器上。

语法

Object.GetLatestVersion（）

说明

GetLatestVersion 方法不签出该文件。

如果源代码管理器成功地获得了该文件，则返回“真”(.T.)。如果源代码管理器不能获得该文件，或者该项目不在源代码管理器中，则返回“假”(.F.)。

应用于

文件对象

请参阅

AddToSCC 方法, CheckIn 方法, CheckOut 方法, RemoveFromSCC 方法, UndoCheckOut 方法

GETNEXTMODIFIED（） 函数

返回一个记录号，对应缓冲表或临时表中下一个被修改的记录。

语法

GETNEXTMODIFIED(nRecordNumber [, cTableAlias | nWorkArea])

返值类型

数值型

参数描述

nRecordNumber

指定的记录编号，GETNEXTMODIFIED（） 函数在此记录后搜索下一个被修改的记录。*nRecordNumber* 可以指定为 0，从而确定表或临时表中第一个被修改的记录。

cTableAlias

指定表或临时表的别名，GETNEXTMODIFIED（） 函数返回该表中下一个被修改记录的记录号。

nWorkArea

指定表或临时表所在的工作区，GETNEXTMODIFIED（） 函数返回该工作区中下一个被修改记录的记录号。

如果没有指定别名或工作区，GETNEXTMODIFIED（）函数将返回当前选定表或临时表中下一个被修改记录的记录号。

说明

如果在所指定的记录后没有被修改的记录，GETNEXTMODIFIED（）函数返回 0。只要更改了记录中任一字段的内容（即使还原为字段的原始内容）或记录的删除状态，就认为该记录已被修改。

GETNEXTMODIFIED（）函数只能对启用表缓冲的表和临时表进行操作。可以利用 CURSORSETPROP（）函数启用表缓冲。

示例

下面的示例演示了如何利用 GETNEXTMODIFIED（）函数判定表中哪一个记录已作了更改。为满足表缓冲的要求，将 MULTILOCKS 设置为 ON。打开 testdate 数据库中的 customer 表，并调用 CURSORSETPROP（）函数，将缓冲方式设置为开放式表缓冲(5)。

发出 SKIP 命令，把记录指针移动到第二个记录上，并用 REPLACE 命令修改 cust_id 字段。从表的起始处开始，使用 GETNEXTMODIFIED(0) 显示从表中下一个被修改记录的记录号（2，第二个记录）。然后调用 TABLEREVERT（）函数使表返回到原始状态，并且再次使用 GETNEXTMODIFIED(0) 被显示下一个修改的记录号（0，表示没有修改过记录）。

```
CLOSE DATABASES
CLEAR

OPEN DATABASE SYS(HOME(2) + 'data\testdata')
SET MULTILOCKS ON &&必须为 ON 以请求表缓冲
USE Customer   &&打开 customer 表
```

```
=CURSORSETPROP("Buffering", 5, "customer")  && 启用表缓冲

SKIP  && 把记录指针移到第二个记录上

* 更改字段内容
REPLACE cust_id WITH "***"

* Call MESSAGEBOX function with results of GETNEXTMODIFIED
=MESSAGEBOX("Record " + ALLTRIM(STR(GETNEXTMODIFIED(0))) + ;
    " has changed.",0,"Results")

* Revert table and display results with MESSAGEBOX
=TABLEREVERT(.T.)  && 放弃对表的所有修改
nChange=GETNEXTMODIFIED(0)
IF nChange=0
   =MESSAGEBOX("Record(s) have been reverted.",0,"Results")
ELSE
   =MESSAGEBOX("Record " + ALLTRIM(STR(GETNEXTMODIFIED(0))) + ;
    " has changed.",0,"Results")
ENDIF
```

请参阅

CURSORSETPROP（）,CURVAL（）,GETFLDSTATE（）,OLDVAL（）

GETOBJECT（） 函数

激活 OLE 自动化对象，并创建此对象的引用。

语法

GETOBJECT(FileName [, ClassName])

返值类型

Object

参数描述

FileName

指明所要激活文件的完整路径与文件名。并不需要指定应用程序，因为 OLE 动态链接库将根据提供的文件名确定要启动的应用程序。

例如，下面的代码将启动 Microsoft Excel，打开名为 BUDGET.XLS 的文件，并通过名为 MBUDVAR 的对象变量创建引用：

```
MBUDVAR = GETOBJECT('C:\EXCEL\WORK\BUDGET.XLS')
```

ClassName

指定所要检索对象的类名。有的应用程序可以在同一文件中存储一个以上的对象类型，并允许使用类名指定要激活的对象。例如，如果某个文字处理应用程序在同一文件中存储了文档、宏定义与工具栏对象，则可用下列命令创

建对文档文件的引用：

```
MDOCFILE = GETOBJECT('C:\WRDPROC\MYDOC.DOC','WrdProc.Document')
    'WrdProc.Document')
```

对于某些服务程序，每次调用 GETOBJECT（）函数，都将启动该应用程序的一个额外实例，并占用额外内存。如果该应用程序已经运行，可以仿照下面的示例，忽略 *FileName* 而包含 *ClassName*，从而避免启动该应用程序的额外实例：

```
oleApp = GETOBJECT(, "Excel.Application")
```

说明

用 GETOBJECT（）函数从文件中激活某个 OLE 自动化对象，并通过变量或数组元素指定对该对象的引用。

如果指定了一个无效的文件或类名称，会显示 OLE 错误，并且 GETOBJECT（）函数返回一个空字符串。

请参阅

COMCLASSINFO（），CREATEOBJECT（），DEFINE CLASS，SET OLEOBJECT

GETPAD（） 函数

返回菜单栏上给定位置的菜单标题。

语法

GETPAD(cMenuBarName, nMenuBarPosition)

返值类型

Character

参数描述

cMenuBarName

指定菜单栏名称。

nMenuBarPosition

指定菜单栏上的某一位置。*nMenuPosition* 的取值范围从 1 开始（对应菜单栏上最左面的菜单标题），直到菜单中菜单标题的总数。

说明

可以添加、删除或重新调整标题栏上的菜单名。使用 DEFINE PAD 命令可以向标题栏中添加主菜单名，使用 RELEASE PAD 命令可以删除菜单标题。

示例

下面的示例用 GETPAD（） 来测试“编辑”菜单名称是否在 Visual FoxPro 系统菜单栏上。如果在，GETPAD（） 返回菜单名称（为了将“编辑”菜单栏还原到默认状

态，用 SET SYSMENU TO DEFAULT 命令）。

```
FOR gnCount = 1 TO CNTPAD('_msysmenu')    && pad 的数量
    IF PRMPAD('_msysmenu', GETPAD('_msysmenu', gnCount)) = 'Edit'
       RELEASE PAD (GETPAD('_msysmenu', gnCount)) OF _msysmenu
       EXIT
    ENDIF
ENDFOR
```

请参阅

DEFINE MENU, DEFINE PAD, RELEASE PAD

GETPEM（） 函数

返回事件或方法的属性值或程序代码。

语法

GETPEM(oObjectName | cClassName, cProperty | cEvent | cMethod)

返值类型

字符型、货币型、日期型、日期时间型、数值型或逻辑型

参数描述

oObjectName
指定对象，函数将返回它的属性值或其事件、方法的程序代码。*oObjectName* 可以是求值结果为对象的任意表达式，例如对象引用、对象变量或对象数组元素。

cClassName
指定类，函数将返回其属性值或其事件、方法的程序代码。

cProperty
指定要返回其值的属性。

cEvent
指定要返回其值的属性。

cMethod
指定要返回其值的属性。

说明

只有交互式的 Visual FoxPro session ，才支持 GETPEM（）函数。

请参阅

CREATE FORM, PEMSTATUS（）, SYS(1269), SYS(1270), SYS(1271), SYS(1272)

GETPICT（） 函数

显示“打开”对话框，并返回选定图片文件的名称。

语法

GETPICT([cFileExtensions] [, cFileNameCaption] [, cOpenButtonCaption])

返值类型

数值型

参数描述

cFileExtensions

没有选择“所有文件”菜单项时，在可滚动列表中指定显示图片文件的文件扩展名。

cFileExtensions 可以为下列形式：

- 如果 *cFileExtensions* 只包含单一扩展名（例如 .BMP），则只显示有这个扩展名的文件。
- cFileExtensions 也可以包含通配符（* 和 ?），此时将显示扩展名与通配符条件相匹配的所有文件。例如，如果 *cFileExtension* 为 ?X?，则显示扩展名为 .FXP、.EXE 和 .TXT 的所有文件。

- 如果 *cFileExtensions* 中包含空字符串 ('')，则显示扩展名为 .BMP 和 .DIB 的文件。

cFileNameCaption

指定在“文件名”文本框上方显示的标题，*cFileNameCaption* 将替换“文件名”。省略 *cFileNameCaption* 时，将显示“文件名”。

cOpenButtonCaption

为“确定”按钮指定标题。

说明

如果按 ESC 键、选择“取消”按钮、或单击“关闭”按钮，而退出“打开”对话框，GETPICT（） 函数将返回空字符串。

请参阅

GETFILE（）, GETEXPR, LOCFILE（）, PUTFILE（）

GETPRINTER（） 函数

显示 Windows 的“打印设置”对话框，并返回所选择的打印机名称。

语法

GETPRINTER（）

返值类型

字符型

说明

如果按 ESC **键、选择“取消”按钮、或从控制菜单上选择“关闭”而退出“打印设置”对话框，** GETPRINTER() **函数将返回空字符串。**

示例

```
CLEAR
cPrinter = GETPRINTER() && 显示 Windows 的“打印设置”对话框
*** 显示所选定的打印机的名字 ***
WAIT WINDOW IIF(EMPTY(cPrinter), '打印机未选择！ ', cPrinter)
```

请参阅

APRINTERS(), SET PRINTER

GO 或 GOTO 命令

将记录指针移动到指定位置。

语法

GO [RECORD] nRecordNumber [IN nWorkArea | IN cTableAlias]

–或者–

GO TOP | BOTTOM [IN nWorkArea | IN cTableAlias]

–或者–

GOTO [RECORD] nRecordNumber [IN nWorkArea | IN cTableAlias]

–或者–

GOTO TOP | BOTTOM [IN nWorkArea | IN cTableAlias]

参数描述

RECORD nRecordNumber

指定一个物理记录号，记录指针将移至该记录。您可以省略 GO 或 GOTO 命令而只指定记录号，但如果仅指定记录号，则只能在当前工作区中移动记录指针。

IN nWorkArea

指定表所在的工作区，记录指针在此表中移动。

IN cTableAlias

指定表的别名，记录指针在此表中移动。

TOP

将记录指针定位在表的第一个记录上。如果此表使用升序索引，则第一个记录是关键字值最小的记录；如果使用降序索引，则第一个记录是关键字值最大的记录。

BOTTOM

将记录指针定位在表的最后一个记录上。如果此表使用升序索引，则最后一个记录是关键字值最大的记录；如果使用降序索引，则最后一个记录是关键字值最小的记录。

说明

可以互换使用 GO 和 GOTO 命令。除非在 IN 子句中指定了另一个工作区，否则这两个命令都对当前工作区中的表进行操作。

示例

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE products  && 打开 Products 表
USE customer IN 0  &&打开 Customer 表
GO BOTTOM IN products
CLEAR
? RECNO('products')
GO TOP
? RECNO()     && 显示数值 1
GO 5
? RECNO()     && 显示数值 5
```

请参阅

RECNO(), SELECT, SKIP

GoBack 方法

在一个 Active Document 容器的历史列表中向后定位。

语法

HyperLink.GoBack（）

说明

只在包含在容器中的 Active Document 中支持 GoBack 方法。

当 Active Document 在容器之外运行时，会忽略 GoBack 方法。例如，当 Active Document 在 Visual FoxPro 运行期间或交互式的 Visual FoxPro 工作期中运行时，会被忽略。

应用于

超链接对象

请参阅

GoForward 方法, NavigateTo 方法

GoForward 方法

在一个 Active Document 容器的历史列表中向前定位。

语法

HyperLink.GoForward（）

说明

只在包含在容器中的 Active Document 中支持 GoForward 方法。

当 Active Document 在容器之外运行时，会忽略 GoForward 方法。例如，当 Active Document 在运行时刻 Visual FoxPro 中或一个交互的 Visual FoxPro 工作期中运行时，就会忽略 GoForward 方法。

应用于

超链接对象

请参阅

GoBack 方法, NavigateTo 方法

GOMONTH（） 函数

对于给定的日期表达式，返回指定数目的月份以前或以后的日期。

语法

GOMONTH(dExpression | tExpression, nNumberOfMonths)

返值类型

日期型

参数描述

dExpression

指定的日期表达式。相对于此日期，GOMONTH（） 函数返回日期。

tExpression

指定的日期时间表达式。相对于此日期时间，GOMONTH（） 函数返回日期。

nNumberOfMonths

指定从给定日期或日期时间开始的月数。如果 *nNumberOfMonths* 为正，GOMONTH（） 函数返回给定日期或日期时间 *nNumberOfMonths* 个月以后的日期；如果 *nNumberOfMonths* 为负，GOMONTH（） 函数返回给定日期或日期时间 *nNumberOfMonths* 个月以前的日期。

示例

```
SET CENTURY ON
STORE GOMONTH({^1998-02-16}, 5) TO gdDeadLine

CLEAR
? gdDeadLine  && 显示数值 07/16/1998
? GOMONTH({^1998-12-31}, 2)  && 显示数值 02/28/1999
? GOMONTH({^1998-12-31}, -2)  && 显示数值 10/31/1998
```

请参阅

CMONTH（）

GotFocus 事件

当通过用户操作或执行程序代码使对象接收到焦点时，此事件发生。

语法

```
PROCEDURE Object.GotFocus
[LPARAMETERS nIndex]
```

参数描述

nIndex

用以唯一标识控件数组中的某个控件。

说明

对象接收到焦点时，GotFocus 事件用来指定要发生的动作。例如，通过为表单中的每个控件附加 GotFocus 事件，可以显示简单说明或状态栏信息以指导用户；也可以通过激活、废止或显示依赖于拥有焦点控制的其他控件，提供可视化的提示。可根据用户的操作（例如单击鼠标）或在程序代码中调用 SetFocus 方法使控件接收焦点。当表单没有控件，或者它的所有控件已废止或不可见时，此表单才能接收焦点。

注意 只有当对象的 Enabled 属性和 Visible 属性均设置为“真”（.T.）时，此对象才能接收焦点。要为焦点的移动定制键盘操作方式，可以为表单上的控件设置 TAB 键次序或指定快捷键。在控件所在的容器 Activate 事件后，发生 GotFocus 事件。

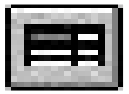
应用于

复选框，组合框，命令按钮，容器对象，控件对象，编辑框，表单，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，微调，文本框

请参阅

Activate 事件，ActiveControl 属性，Click 事件，Deactivate 事件，Enabled 属性，LostFocus 事件，SetFocus 方法，TabIndex 属性，TabStop 属性，Visible 属性

Grid 控件



创建表格。

语法

Grid

说明

表格是按行和列显示数据的容器对象，其外观与浏览窗口相似。表格是包含列对象的容器对象。列可以包含标头对象及控件。由于表格及其列、标头和控件都有各自的属性集，您可以完全控制表格中的每一个元素。可以使用表格生成器交互地创建表格。有关创建表格的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第十章“使用控件”。

属性

ActiveColumn	ActiveRow	AllowAddNew
AllowHeaderSizing	AllowRowSizing	Application
BackColor	BaseClass	ChildOrder

Class
Columns
DragIcon
FontBold
FontItalic
FontShadow
FontUnderline
GridLines
Height
HighLightRow
LinkMaster
Name
OLEDropEffects

Panel
ParentClass
RecordMark
RelationalExpr
RightToLeft
SplitBar
TabStop
Top

ClassLibrary
Comment
DragMode
FontCondense
FontName
FontSize
ForeColor
GridLineWidth
HelpContextID
Left
MouseIcon
OLEDragMode
OLEDropHasData
Property
PanelLink
Partition
RecordSource
RelativeColumn
RowHeight
StatusBarText
Tag
Value

ColumnCount
DeleteMark
Enabled
FontExtend
FontOutline
FontStrikeThru
GridLineColor
HeaderHeight
HighLight
LeftColumn
MousePointer
OLEDragPicture
OLEDropMode

Parent
ReadOnly
RecordSourceType
RelativeRow
ScrollBars
TabIndex
ToolTipText
View

Visible WhatsThisHelpID Width

事件

AfterRowColChange BeforeRowColChange Click

DblClick Deleted Destroy
DragDrop DragOver Error
Init MiddleClick Event MouseDown
MouseMove MouseUp MouseWheel
OLECompleteDrag OLEDragDrop OLEDragOver
OLEGiveFeedBack OLESetData OLEStartDrag
Resize RightClick Moved
Scrolled UIEnable Valid
When

方法

ActivateCell AddColumn AddObject
AddProperty CloneObject DeleteColumn
DoScroll Drag GridHitTest
Move OLEDrag ReadExpression
ReadMethod Refresh RemoveObject
ResetToDefault SaveAsClass SetAll

SetFocus　　WriteExpression　　WriteMethod　　ZOrder

示例

下列示例将一个表格控件放在表单上。打开 customer 表，并在表格中显示表中的内容。使用了 Caption 属性为 CUST_ID 字段指定不同的标头标题(Customer ID)。将一个命令按钮放在表单上，以关闭该表单。

使用了 SetAll 方法，以及 DynamicBackColor 属性，以指定记录的背景颜色。在表格中，如果一个记录的编号是偶数，则该记录的 DynamicBackColor 属性为白色，否则 DynamicBackColor 为绿色。

```
CLOSE ALL  && 关闭表和数据库
OPEN DATABASE (HOME(2) + 'data\testdata')

USE customer  IN 0  &&打开 Customer 表

frmMyForm = CREATEOBJECT('Form')  && 创建一个表单
frmMyForm.Closable = .F.  && 禁止窗口弹出菜单

frmMyForm.AddObject('cmdCommand1','cmdMyCmdBtn')  &&增加命令按钮
frmMyForm.AddObject('grdGrid1','Grid')  && 增加表格控件

frmMyForm.grdGrid1.Left = 25  && 调整表格位置

frmMyForm.grdGrid1.SetAll("DynamicBackColor", ;
    "IIF(MOD(RECNO（）, 2)=0, RGB(255,255,255) ;
    , RGB(0,255,0))", "Column")  && 将白与绿记录交换

frmMyForm.grdGrid1.Visible = .T.  && 显示表格控件
frmMyForm.cmdCommand1.Visible =.T.  &&显示 "Quit" 命令按钮
frmMyForm.grdGrid1.Column1.Header1.Caption = 'Customer ID'
```

```
frmMyForm.SHOW  && Display the form
READ EVENT  && Start event processing

DEFINE CLASS cmdMyCmdBtn AS CommandButton  &&创建命令按钮
    Caption = '\<Quit'  && 给命令按钮加标题
    Cancel = .T.  &&默认取消命令按钮(Esc键)
    Left = 125  && 命令按钮列
    Top = 210  && 命令按钮行
    Height = 25  && 命令按钮高

    PROCEDURE Click
       CLEAR EVENT  &&终止事件程序，关闭表单
       CLOSE ALL  && 关闭表和数据库
ENDDEFINE
```

请参阅

Column 对象, CREATE CLASS, CREATE FORM, DEFINE CLASS, Header 对象

GridHitTest 方法

作为输出参数返回一个表格控件在指定水平 (X) 和垂直 (Y) 坐标的组成部分。

语法

Grid.GridHitTest(nXCoord_In, nYCoord_In
[, nWhere_Out [, nRelRow_Out [, nRelCol_Out [, nView_Out]]]])

参数描述

nXCoord_In

指定包含表格的表单中的水平 (X) 位置。

nYCoord_In

指定包含表格的表单中的垂直 (Y) 位置。

nWhere_Out

一个输出参数，包含一个值，该值对应于 nXCoord_In **和** nYCoord_In ***位置*** **的表格成分。下表列出了** @nWhere_Out **的值，以及相应的表格成分。**

@nWhere_Out	表格成分
0	一个不能确定的表格成分。
1	列标头。
2	列标头之间。
3	单元格。
4	保留。
5	分隔栏。
6	记录删除标记。
7	保留。
8	保留。

9	保留。
10	保留。
11	左上角的框。
12	记录标记。
13	列标头大小调整区。
14	行大小调整区。
15	保留。
16	水平滚动栏
17	垂直滚动栏。

nRelRow_Out

一个输出参数，包含指定点的相关表格行。

nRelCol_Out

一个输出参数，包含指定点的相关表格列。

nView_Out

一个输出参数，包含一个值，该值对应于包含指定点的表格窗格。如果该表格被分成两个窗格，如果指定点位于左窗格，则该值为 0；如果指定点位于右窗格，则该值为 1。如果该表格没有被分成独立的窗格，则该参数为 1。

说明

如果指定点位于表格中，则 GridHitTest（）方法返回“真”(.T.)；否则返回“假”(.F.)。

可以在鼠标事件或 OLE 放落目标事件中使用 GridHitTest（） 方法，来判断鼠标指针位于表格的什么位置。nRelRow_Out和 nRelCol_Out参数可以传递给 ActivateCell（） 方法，以激活表格中指定的单元格。

应用于

表格控件

请参阅

ActivateCell 方法, ActiveColumn 属性, ActiveRow 属性, MCOL（）, MROW（）

GridLineColor 属性

指定在表格控件中分隔单元格的线的颜色。设计时可用；运行时可读写。

语法

Grid.GridLineColor[= nColor]

Grid.GridLineColor = RGB(nRedValue, nGreenValue, nBlueValue)

参数描述

nColor

指定单个值，该值代表颜色。在默认情况下，GridLineColor 设置为 0（黑）。

nRedValue, nGreenValue, nBlueValue

指定组成表格线颜色的三个独立的色度；必须与 RGB（） 函数一起使用，将三个颜色成分合成一个值，既 GridLineColor 属性。

注意

注意，在“属性”窗口中，您可以单击任何颜色属性，以显示“颜色”对话框。您可以在这个对话框中选择或定义颜色。当您关闭“颜色”对话框之后，对应于所选颜色的红、绿和蓝色度就成为这些属性的设置。

应用于

表格

请参阅

GridLines 属性，GridLineWidth 属性

GridLines 属性

确定在表格控件中是否显示水平和垂直线。设计时可用；运行时可读写。

语法

Grid.GridLines[= n 设置]

参数描述

n 设置

GridLines 属性的设置有：

设置	说明
0	无。没有表格线。.
1	水平。只显示水平表格线。
2	垂直。只显示垂直表格线。
3	（默认值）都显示。显示水平和垂直表格线。

应用于

表格

请参阅

GridLineColor 属性，GridLineWidth 属性

GridLineWidth 属性

指定表格控件中分隔单元格的线的宽度，以像素为单位。设计时可用；运行时可读写。

语法

Grid.GridLineWidth[= nWidth]

参数描述

nWidth

指定表格控件中分隔单元格的线的宽度。

应用于

表格

请参阅

GridLineColor **属性**, GridLines **属性**

HalfHeightCaption **属性**

指定表单标题高度是否为正常高度的一半。设计和运行时可用。

语法

Object.HalfHeightCaption[= lExpr]

参数描述

lExpr

下表列出了 HalfHeightCaption 属性的设置：

设置	说明
“真” (.T.)	表单标题高度为正常高度的一半。
“假” (.F.)	（默认值）表单标题高度为正常高度。

应用于

表单，_SCREEN

请参阅

Caption 属性

Header 对象

为表格控件的列创建标头。

语法

Header

说明

表格中的列包含有标头。标头在列的最上面显示列标题，并且可以响应事件。

有关创建表格的标头的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员

指南》的第十章“使用控件”。

属性

Alignment	Application	BackColor
BaseClass	Caption	Class
ClassLibrary	Comment	FontBold
FontCondense	FontExtend	FontItalic
FontName	FontOutline	FontShadow
FontSize	FontStrikeThru	FontUnderline
ForeColor	Name	Parent
ParentClass	Tag	

事件

Click	DblClick	MiddleClick Event
MouseDown	MouseMove	MouseUp
MouseWheel	RightClick	

方法

AddProperty	ReadExpression	ReadMethod
Refresh	ResetToDefault	SaveAsClass
WriteExpression	WriteMethod	

示例

下列示例使用 Header 对象以及 Caption 属性更改表格中第一个标头的标题。

将一个表格控件放在表单上。打开 customer 表，并在表格中显示表中的内容。使用了 Caption 属性为 CUST_ID 字段指定不同的标头标题(Customer ID)。将一个命令按钮放在表单上，以关闭该表单。

使用了 SetAll 方法，以及 DynamicBackColor 属性，以指定记录的背景颜色。在表格中，如果一个记录的编号是偶数，则该记录的 DynamicBackColor 属性为白色，否则 DynamicBackColor 为绿色。

```
CLOSE ALL  && 关闭表和数据库
OPEN DATABASE (HOME(2) + 'data\testdata')

USE customer  IN 0  &&打开 Customer 表

frmMyForm = CREATEOBJECT('form')  && 创建一个表单
frmMyForm.Closable = .f.  && 禁止 window 弹出菜单

frmMyForm.AddObject('cmdCommand1','cmdMyCmdBtn')  && 增加命令按钮
frmMyForm.AddObject('grdGrid1','Grid')  &&增加表格控件

frmMyForm.grdGrid1.Left = 25  && 调整表格位置

frmMyForm.grdGrid1.SetAll("DynamicBackColor", ;
    "IIF(MOD(RECNO（）, 2)=0, RGB(255,255,255) ;
    , RGB(0,255,0))", "Column")  &&将白和绿的记录调换

frmMyForm.grdGrid1.Visible = .T.  && 显示表格控件
frmMyForm.cmdCommand1.Visible =.T.  &&显示 "Quit" 命令按钮
frmMyForm.grdGrid1.Column1.Header1.Caption = 'Customer ID'

frmMyForm.SHOW  &&显示表单
```

```
READ EVENT  && 启动事件程序

DEFINE CLASS cmdMyCmdBtn AS CommandButton  && 创建命令按钮
    Caption = '\<Quit'  && 给命令按钮增加标题
    Cancel = .T.  && 默认取消命令按钮 (Esc 键)
    Left = 125  && 命令按钮列
    Top = 210  && 命令按钮行
    Height = 25  && 命令按钮高

    PROCEDURE Click
        CLEAR EVENT  && 终止 2 事件程序，关闭表单
        CLOSE ALL  && 关闭表和数据库
ENDDEFINE
```

请参阅

Column 对象, CREATE FORM, CREATE CLASS, DEFINE CLASS, Grid 控件

HEADER( ) 函数

返回表文件标题所占的字节数。

语法

HEADER([nWorkArea | cTableAlias])

返值类型

数值型

参数描述

nWorkArea | cTableAlias

返回在非当前工作区中打开的表的表头大小。参数 *nWorkArea* 指定工作区号，*cTableAlias* 指定表别名。如果省略工作区号和表别名，则 HEADER（）函数将返回当前工作区中打开的表的表头大小。

如果指定的工作区中没有打开的表，则 HEADER（）函数将返回 0。若指定的别名不存在，则 Visual FoxPro 将显示一条错误信息。

说明

表头中包含了该表本身的一些信息，其中包括该表的所有字段名、字段大小以及该表是否具有备注文件或结构索引文件等等。

请参阅

FSIZE（），RECSIZE（）

HeaderHeight 属性

指定表格控件中列标头的高度。设计时可用，运行时可读写。

语法

Header.HeaderHeight[= nHeight]

参数描述

nHeight

标头的高度，以像素为单位。

应用于

表格

请参阅

ScaleMode **属性**

Height **属性**

指定对象在屏幕上的高度。设计和运行时可用。

语法

Object.Height[= nHeight]

参数描述

nHeight

指定对象的高度，其单位由表单的 ScaleMode 属性决定。

说明

对于表单，高度不包括边框和标题栏。

对于控件，高度是从其边框的外侧算起的。

当用户或程序调整对象的大小时，Height 属性的值随之发生变化。

在计算对象的面积时，要用到 Height 和 Width 属性。

注意 当用于列对象包含的控件时，**Height** 属性只读。

示例

下面的示例演示了 Height 属性是如何指定表单中三个命令按钮的高度的。先使用 Addobject 方法向表单中添加了一个线条控件和三个命令按钮。再使用 Height 属性指定每个命令按钮的垂直高度。

```
frmMyForm = CREATEOBJECT('form') && 创建表单
frmMyForm.Closable = .F. && 使控件菜单框失效

frmMyForm.AddObject('shpLine','Line') && 向表单添加一个 Line 控件
frmMyForm.AddObject('cmdCmndBtn1','cmdMyCmndBtn1') && 向上命令按钮
frmMyForm.AddObject('cmdCmndBtn2','cmdMyCmndBtn2') && 向下命令按钮
frmMyForm.AddObject('cmdCmndBtn3','cmdMyCmndBtn3') && Quit 命令按钮

frmMyForm.shpLine.Visible = .T. && 显示线条控件
frmMyForm.shpLine.Top = 20 && 指定线条控件所在的行
frmMyForm.shpLine.Left = 125 && 指定线条控件所在的行

frmMyForm.cmdCmndBtn1.Visible =.T. && 向上命令按钮可见
frmMyForm.cmdCmndBtn2.Visible =.T. && 向下命令按钮可见
```

```
frmMyForm.cmdCmndBtn3.Visible =.T. && Quit 命令按钮可见

frmMyForm.SHOW && 显示表单
READ EVENT && 开始处理事件

DEFINE CLASS cmdMyCmndBtn1 AS COMMANDBUTTON && 创建命令按钮
    Caption = 'Slant \<Up' && 命令按钮的标题
    Left = 50 && 命令按钮所在的列
    Top = 100 && 命令按钮所在的行
    Height = 25 && 命令按钮的高度

    PROCEDURE Click
        ThisForm.shpLine.Visible = .F. && 隐藏 Line 控件
        ThisForm.shpLine.LineSlant ='/' && 向上倾斜
        ThisForm.shpLine.Visible = .T. && 显示线条控件
ENDDEFINE

DEFINE CLASS cmdMyCmndBtn2 AS CommandButton && 创建命令按钮
    Caption = 'Slant \<Down' && 命令按钮的标题
    Left = 200 && 命令按钮所在的列
    Top = 100 && 命令按钮所在的行
    Height = 25 && 命令按钮的高度

    PROCEDURE Click
        ThisForm.shpLine.Visible = .F. && 隐藏线条控件
        ThisForm.shpLine.LineSlant ='\' && 向下倾斜
        ThisForm.shpLine.Visible = .T. && 显示线条控件
ENDDEFINE

DEFINE CLASS cmdMyCmndBtn3 AS CommandButton && 创建命令按钮
```

```
    Caption = '\<Quit' && 命令按钮的标题
    Cancel = .T. && 默认为“取消”（ESC）命令按钮
    Left = 125 && 命令按钮所在的列
    Top = 150 &&命令按钮所在的行
    Height = 25 && 命令按钮的高度

    PROCEDURE Click
       CLEAR EVENT && 结束事件处理，关闭表单
ENDDEFINE
```

应用于

复选框，组合框，命令按钮，命令组，容器对象，控件对象，自定义，编辑框，表单，表格，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，选项组，页框，_SCREEN，形状，微调，文本框，计时器，工具栏

请参阅

Left 属性，Move 方法，ScaleMode 属性，Top 属性，Width 属性

HELP 命令

打开帮助窗口。

语法

HELP
[Topic | ID nContextID]
[IN [WINDOW] WindowName | IN [WINDOW] SCREEN | IN [WINDOW]
[NOWAIT]

参数描述

Topic
指定要显示的帮助主题。如果只给出主题词的部分拼写字母，Visual FoxPro会打开帮助窗口，并显示拼写最相近的主题。

ID nContextID
根据主题的上下文标识(ID)，指定要显示的帮助主题。

在使用DBF样式帮助时，*nContextID*是帮助表中Contextid字段的值。Contextid字段必须是该表的第一个字段。

当使用图形样式帮助时，*nContextID*是帮助项目文件中MAP部分的上下文编号。

IN [WINDOW] WindowName
在一个父窗口中打开帮助窗口，帮助窗口并不继承父窗口的任何特性。如果帮助窗口在一个父窗口内激活，则它不能移出父窗口。父窗口移动时，帮助窗口随之移动。在父窗口中打开帮助窗口之前，必须先用DEFINE WINDOW命令定义父窗口。

IN [WINDOW] SCREEN
把帮助窗口放置到 Visual FoxPro 主窗口中。

NOWAIT
在打开帮助窗口之后继续执行程序。程序不必等待帮助窗口的关闭，而是继续执行 HELP NOWAIT 之后的下一行程序。如果省略 NOWAIT，则在一个程序中执行 HELP 命令后，打开帮助窗口，并且在关闭帮助窗口之前暂停执行程序。

在图形样式帮助中，NOWAIT 参数无效，并且在发出 HELP 命令后程序继续执行。

说明

有关创建自己的帮助系统的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》的相应部分。

请参阅

SET HELP, SET HELPFILTER, SET TOPIC

Help 方法

打开“帮助”窗口。

语法

ApplicationObject.Help([cFileName] [, nContextID] [, cHelpTopic])

参数描述

cFileName

指定要打开的帮助文件的名称。如果帮助文件不在默认目录中，则在帮助文件名中应该包含路径。

nContextID

根据主题的上下文 ID，指定要显示的帮助主题。nContextID 是帮助项目文件 MAP 部分的上下文编号。

cHelpTopic

指定要显示的帮助主题。如果只包含了一个主题标题的部分拼写，Visual FoxPro 会打开“帮助”窗口，并且使用最匹配的标题显示主题。

说明

如果没有包含任何可选的参数，则显示主帮助主题。这个方法不支持 .dbf 样式的帮助。

应用于

Application 对象，VFP 系统变量

请参阅

HelpContextID 属性

为帮助文件的一个主题指定上下文标识，以便提供上下文相关帮助。设计和运行时可用。

语法

Object.HelpContextID[= nContextID]

参数描述

nContextID

指定主题在图形样式或 .DBF 样式的帮助文件中的上下文标识编码。context ID 数字的有效范围为 0 到 268,435,455。

说明

设计和运行时可用。若要在应用程序中为某个对象创建上下文相关帮助，必须在创建帮助文件时把相同的上下文编码同时赋给该对象和相关帮助主题。.

如果已经为应用程序创建了图形样式或 .DBF 样式帮助，Visual FoxPro 将调用帮助文件，并且请求当前上下文 ID 编码所标识的主题。可以使用 SET HELP TO 语句指定帮助文件，并指定一个键以便使用 ON KEY LABEL 语句激活这个帮助文件。创建图形样式帮助所需要的帮助编译器在专业版 Visual FoxPro 中提供。

当前的上下文标识编号是焦点对象的 HelpContextID 属性的设置。如果没有搜索到一个非零的当前上下文编号，则显示帮助文件的主目录。

对于一个服务程序对象，HelpContextID 属性指定了为该服务程序创建的类型库的上下文 ID。ServerHelpFile 属性指定了其中包含对应于这个上下文 ID 的帮助主题的帮助文件。

应用于

复选框，组合框，命令按钮，命令组，编辑框，表单，表格，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，选项组，页面，_SCREEN，Server 对象，形状，微调，文本框，工具栏

请参阅

ServerHelpFile 属性，SET HELP，SET TOPIC

HIDE MENU 命令

隐藏活动的用户自定义菜单栏。

语法

HIDE MENU MenuBarName1 [, MenuBarName2 ...] | ALL
[SAVE]

参数描述

MenuBarName1 [, MenuBarName2 ...]

指定要隐藏的菜单栏的名称或名称列表（以逗句隔开各个菜单栏名）。

ALL

隐藏所有的用户定义菜单栏。

SAVE

把菜单栏的图像放置到屏幕或某个窗口中。常用于程序的开发和调试。要从 Visual FoxPro 主窗口或用户自定义窗口中清除菜单栏的图像，可以使用 CLEAR 命令。

说明

HIDE MENU 命令可以从 Visual FoxPro 主窗口或用户自定义窗口中移去指定的某个菜单栏、一组菜单栏或所有菜单栏，但并不从内存中移去菜单的定义。只有那些用 DEFINE MENU 创建的菜单栏才能被隐藏。菜单栏的隐藏与废止是不同的。菜单栏被隐藏之后仍然驻留在内存中，并且可以使用 ACTIVATE MENU 或 SHOW MENU 命令重新显示。

请参阅

ACTIVATE MENU, DEFINE MENU, DEFINE PAD, SHOW MENU

Hide 方法

通过把 Visible 属性设置为“假”(.F.)，隐藏表单、表单集或工具栏。

语法

Object.Hide

说明

表单被隐藏后，用户不可访问它的控件，但是这些控件仍然可用，并且可以在代码中访问它们。虽然这些控件是不可见的，但这些保存在不可见表单中的控件仍然保留自己的 Visible 属性设置值。

当表单集的 Visible 属性为“假”(.F.)时，用户看不到它所包含的表单。表单集被隐藏后，用户不可访问其中的表单，但是这些表单仍然可用，并且可以在代码中访问它们。表单集的 Hide 方法并不设置子表单的 Visible 属性。因此，当一个表单包含在一个表单集中时，若要判断表单是否可见，必须同时检查表单和表单集的 Visible 属性设置。

隐藏表单集之后，Visual FoxPro 激活前一个活动对象。如在此之前没有活动对象，则激活 Visual FoxPro 主窗口。

为 _SCREEN 系统变量调用 Hide 方法无效。

应用于

表单，表单集，_SCREEN，工具栏

请参阅

Activate 事件，Deactivate 事件，Enabled 属性，GotFocus 事件，Show 方法，Visible 属性

HIDE POPUP 命令

隐藏由 HIDE POPUP 创建的一个或多个活动菜单。

语法

HIDE POPUP MenuName1 [, MenuName2 ...] | ALL
[SAVE]

参数描述

MenuName1 [, MenuName2 ...]

指定要隐藏的菜单的名称或名称列表（以逗号隔开各个菜单名）。

ALL

隐藏所有已定义的菜单。

SAVE

把菜单图像放到 Visual FoxPro 主窗口或用户自定义窗口上。常用于程序开发和调试。要从窗口中清除菜单的图像，可使用 CLEAR 命令。

说明

HIDE POPUP 命令可以从 Visual FoxPro 主窗口或用户自定义窗口中移去指定的一个、一组或全部菜单，但并未删除内存中的菜单定义。本命令只隐藏那些由 DEFINE POPUP 命令创建的菜单。菜单的隐藏与废止是不同的。菜单被隐藏之后仍然驻留在内存中，可用 ACTIVATE POPUP 或 SHOW POPUP 命令重新显示。

请参阅

ACTIVATE POPUP, DEFINE BAR, DEFINE POPUP, SHOW POPUP

HIDE WINDOW 命令

隐藏一个活动的用户自定义窗口或 Visual FoxPro 系统窗口。

语法

```
HIDE WINDOW WindowName1 [, WindowName2...] | ALL | SCREEN
  [ IN [WINDOW] WindowNameN | IN [WINDOW] SCREEN
  | IN [WINDOW]
  [BOTTOM | TOP | SAME]
```

参数描述

WindowName1 [, WindowName2 ...]

指定要隐藏的窗口的名称（名称列表以逗号隔开各个窗口名）。不包含任何参数的 HIDE WINDOW 命令将隐藏当前的活动窗口。在 Visual FoxPro 中，也可指定要隐藏的工具栏名。若要查阅 Visual FoxPro 的全部工具栏名称，请参阅 SHOW WINDOW。

ALL

隐藏所有窗口。

SCREEN

隐藏 Visual FoxPro 主窗口。若要再次显示 Visual FoxPro 主窗口，可使用

ACTIVATE WINDOW SCREEN 或 SHOW WINDOW SCREEN 命令。

IN [WINDOW] WindowNameN

在父窗口内隐藏一个窗口。

IN [WINDOW] SCREEN

隐藏 Visual FoxPro 主窗口中的一个窗口。

BOTTOM | TOP | SAME

指定要把窗口隐藏到什么位置（相对其他窗口而言）。BOTTOM 将窗口放到所有窗口的后面。TOP（默认值）则将窗口放到所有窗口的前面。SAME 在隐藏窗口时不影响原先的前后关系。在用 SHOW WINDOW ALL 命令显示所有隐藏窗口时，若要保持它们之间的相对位置，应使用 SAME 关键字。

说明

HIDE WINDOW 命令可从 Visual FoxPro 主窗口或用户自定义窗口中移去一个或多个窗口。HIDE WINDOW 命令也可以用来隐藏系统窗口，如命令窗口和查看窗口等。

隐藏窗口与关闭窗口是不同的。隐藏后的窗口仍驻留在内存中，并且仍然是活动窗口。可以向隐藏窗口输出数据，只是不能看见该窗口。

释放一个窗口可从内存中删除该窗口。要想重新显示已移去的窗口则必须重定义这个窗口。显示窗口可使用 ACTIVATE WINDOW 或 SHOW WINDOW 命令。

若要隐藏系统窗口和工具栏（在 Visual FoxPro 中），必须用引号把系统窗口或工具栏的全名括起来。例如，若要隐藏 Visual FoxPro 中“报表控制”工具栏，可执行下列命令：

```
HIDE WINDOW "Report Controls"
```

示例

下面的示例定义并激活了窗口 wOutput1 。程序等待您按一个键，然后隐藏此窗口。程序等待您再按一个键，然后显示窗口的数值。第三次按键会从屏幕和内存中删除此窗口。

```
DEFINE WINDOW wOutput1 FROM 6,1 TO 19,75 TITLE 'Output' ;
    CLOSE FLOAT GROW ZOOM
ACTIVATE WINDOW wOutput1

WAIT WINDOW 'Press a key to hide this window'
HIDE WINDOW wOutput1

WAIT WINDOW 'Press a key to see the window again'
SHOW WINDOW wOutput1

WAIT WINDOW 'Press a key to remove the window from memory'
DEACTIVATE WINDOW wOutput1
RELEASE WINDOW wOutput1
```

请参阅

ACTIVATE WINDOW, DEACTIVATE WINDOW, DEFINE WINDOW, RELEASE WINDOW, SHOW WINDOW

HideDoc 事件

当您从一个 Active Document 漫游时发生。

语法

PROCEDURE Object.HideDoc

说明

当您从一个 Active Document 漫游时，或者当关闭 Active Document 的容器时发生 HideDoc 事件。

当 Active Document 容器最小化，或者当容器恢复原来大小时，不发生 HideDoc 事件。

应用于

ActiveDoc 对象

请参阅

ShowDoc 事件

HideSelection 属性

指定当一个控件失去焦点时，已选中的文本是否显示为选中。设计和运行时可用。

语法

Control.HideSelection[= lExpr]

参数描述

lExpr

HideSelection 属性的设置有：

设置	说明
“真” (.T.)	（默认值）当控件失去焦点时，已选中的文本不显示为选中。
“假” (.F.)	当控件失去焦点时，已选中的文本显示为选中。

说明

可以使用这个属性表明当另一个表单或对话框获得焦点时所选中的文本。例如，在拼写检查例程中。

应用于

组合框，编辑框，微调，文本框

请参阅

DisabledBackColor, DisabledForeColor 属性, GotFocus 事件, LostFocus 事件

Highlight 属性

指定表格控件中具有焦点的单元格是否显示为选中。设计时可用；运行时可读写。

语法

Grid.Highlight[= lExpr]

参数描述

lExpr

Highlight **属性的设置有：**

设置	**说明**
“真” (.T.)	**（默认值）具有焦点的单元格显示为选中。**
“假” (.F.)	**具有焦点的单元格不显示为选中。**

说明

如果 HighLight **属性设置为“真”** (.T.)，**可以将它与一个列的** SelectOnEntry **属性一起使用，以确定整个单元格是否显示为选中。如果将** HighLight **属性设置为“假”** (.F.)，**则会忽略** SelectOnEntry **属性。**

应用于

表格

请参阅

HighlightRow 属性, SelectOnEntry 属性

HighlightRow 属性

指定一个表格控件中的当前行和单元格是否突出显示。设计时可用；运行时可读写。

语法

Grid.HighlightRow[= lExpr]

参数描述

lExpr

HighlightRow 属性的设置有：

设置	说明
“真” (.T.)	（默认值）当前行和单元格突出显示。
“假” (.F.)	当前行和单元格不突出显示。

应用于

表格

请参阅

Highlight 属性

HOME（） 函数

返回启动 Visual FoxPro **和** Visual Studio **的目录名。**

语法

HOME([nLocation])

返值类型

字符型

参数描述

nLocation

指定 HOME（） **返回特定的** Visual FoxPro **或** Visual Studio **目录的名称。注意，如果省略** nLocation **则等同于** HOME(0)。

下表列出了 nLocation **的值，以及** HOME（） **返回的目录。假设您在安装** Visual FoxPro **或** Visual Studio **时选择了默认的安装目录。**

nLocation	目录
0 **或省略**	**启动** Visual FoxPro **的目录。**
1	Visual FoxPro **的安装根目录。**

续表

2	包含 Visual FoxPro 示例的目录。这个目录等同于 _SAMPLES 系统变量中的示例目录。
3	Visual Studio Common 目录。
4	Visual Studio Common\Graphics 目录。
5	Visual Studio 和 MSDN 的示例目录。
6	Visual Studio Common\Tools 目录。

说明

HOME() 函数与 SYS(2004) 命令等价，包含它是为了提供与 dBASE IV 的兼容性。

示例

```
CLEAR
? 'Visual FoxPro   Starup directory: ',   HOME()
? 'Visual FoxPro installation directory:   ', HOME(1)
? 'Visual FoxPro samples directory:        ',HOME(2)
? 'Visual Studio common directory:         ',HOME(3)
? 'Visual Studio graphics directory:    ', HOME(4)
? 'Visual Studio samples directory:     ', HOME(5)
? 'Visual Studio tools directory:       ',   HOME(6)
```

请参阅

_SAMPLES 系统变量，SYS() 函数概述，SYS(2004)

HomeDir 属性

指定项目的主目录。

语法

Object.HomeDir[= cDirectory]

参数描述

cDirectory

指定项目的主目录。默认值是 Visual FoxPro 当前目录，可以用 CURDIR（）函数确定这个目录。

说明

项目的主目录是项目中所有文件的相对路径。HomeDir 属性对应于“项目信息”对话框“项目”选项卡中“主目录”文本框中的目录。

应用于

项目对象

请参阅

CURDIR（）,项目信息对话框

HostName 属性

返回或设置您的 Visual FoxPro 应用程序的用户可读的宿主名。设计时可用；运行时可读写。

语法

Control.HostName [= cName]

参数描述

cName

返回或设置一个字符串表达式，该表达式指定了宿主名。

说明

当编辑一个对象时，HostName 属性的设置会显示为该对象的窗口主题。但是，有些提供对象的应用程序不显示这个 HostName 属性的值。

应用于

OLE 绑定型控件，OLE 容器控件

请参阅

DocumentFile 属性

HOUR（） 函数

返回日期时间表达式的小时部分。

语法

HOUR(tExpression)

参数描述

tExpression

指定的日期时间表达式，HOUR（） 函数将返回它的小时部分。

返值类型

数值型

说明

HOUR（） 函数返回的数值是以 24 小时制为基础的，并且不受 SET HOURS 当前设置的影响。例如，不论 SET HOURS 设置为 12 还是 24，下列命令都将返回 13：

```
? HOUR({^1998-02-16 1:00p})
```

示例

下面的示例将显示当前时间的小时部分和一个指定时间的小时部分。

```
CLEAR
```

```
? HOUR(DATETIME())
? HOUR({^1998-02-16 10:42a}) && 显示数值 10
```

请参阅

CTOT()，DATE()，DATETIME()，DTOT()，MINUTE()，SEC()，SECONDS()，SETSECONDS，TIME()

Hours 属性

指定在一个日期时间值的小时部分是显示为 12 或 24 小时时间格式。设计运行时可用。

语法

Object.Hours[= nValue]

参数描述

nValue

取下列设置之一：

设置	说明
0	（默认值）由 SET HOURS 的设置确定在一个日期时间值的小时部分是显示为 12 还是 24 小时时间格式。如果 SET HOURS 为 12，则这个日期时间值的小时部分显示为 12 小时时间格式。如果 SET HOURS 为 24，则这个日期时间值的小时部分显示为 24 小时时间格式。

续表

12	日期时间值的小时部分显示为 12 小时时间格式。
24	日期时间值的小时部分显示为 24 小时时间格式。

说明

如果 DateFormat 属性的设置为 Short 或 Long，则忽略 Hours 属性的设置。

应用于

文本框

请参阅

Century 属性, DateFormat 属性, DateMark 属性, Seconds 属性, SETHOURS, StrictDateEntry 属性

HScrollSmallChange 属性

指定当单击一个表单的水平滚动箭头时表单的水平滚动量。设计和运行时可用。

语法

Object.HScrollSmallChange[= nExpression]

参数描述

nExpression

指定水平滚动的增量。度量单位由 Scalemode 属性的设置决定。当 Scalemode 属性设置为 3 – 像素时，默认的增量为 10 个像素。

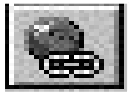
应用于

表单。

请参阅

ScaleMode 属性, ScrollBars 属性, VScrollSmallChange 属性

Hyperlink 对象



创建一个超链接对象。

语法

Hyperlink

说明

超链接对象为 Visual FoxPro 应用程序和包含在容器（例如 Microsoft Internet Explorer）中的 Active Documents 提供了漫游能力。使用超链接对象，Visual FoxPro 应

用程序可以要求一个 Active Document 容器跳转到一个给定的 URL (Uniform Resource Locator)。

超链接对象没有用户界面。超链接对象的 NavigateTo 方法允许您跳转到一个给定的 URL，GoBack 和 GoForward 方法允许您在容器的历史列表中前后移动。

附注只在 Microsoft Internet Explorer 中支持超链接对象。使用 Visual FoxPro 的 Hyperlink 按钮、Hyperlink 图像或组件管理库中的 Hyperlink Label 基类可以获得独立于浏览器的漫游能力。

属性

Application	BaseClass	Class
ClassLibrary	Comment	Name
Parent	ParentClass	Tag

事件

Destroy	Error	Init

方法

AddProperty	GoBack Method	GoFoward Method
NavigateTo Method	ReadExpression	ReadMethod
ResetToDefault	SaveAsClass	WriteExpression

请参阅

Icon 属性

指定最小化表单时显示的图标。对于表单，指定在运行时刻一个表单最小化时显示的图标。设计和运行时可用。

对于项目对象，指定为一个发布的 .exe 应用程序显示的图标。

语法

Object.Icon[= cFileName]

参数描述

cFileName

指定表单最小化时所显示图标的文件名和路径。对于表单，指定表单最小化时显示的图标的文件名和路径。对于一个发布的 .exe 应用程序，为该应用程序显示的图标的文件名和路径。

说明

对于表单，Icon 属性可以为任何表单指定自定义图标，运行时，供用户最小化表单时使用。例如，可以指定一个特殊图标来说明表单的功能。设计时，从属性窗口中输入图标的文件名，文件名必须以 .ICO 为扩展名，并且具有图标的格式。如果不指定图标，系统使用 Visual FoxPro 的默认表单图标。

可以使用 GRAPHICS\ICONS 子目录中的 Visual FoxPro 图标库。

注意 如果在设计时设置了 Icon 属性，而指定的文件不存在，Visual FoxPro 会显示错误信息，但该属性仍指向指定文件。如果在运行时 Icon 属性指向一个不存在的文件，Visual FoxPro 将忽略这个 Icon 属性。

对于项目对象，Icon 属性指定为一个发布的 .exe 应用程序显示的图标。不能为一个发布的 Visual FoxPro .app 应用程序指定图标。

应用于

表单，项目对象，_SCREEN

请参阅

Caption 属性，Picture 属性

IDXCOLLATE( ) 函数

返回索引或索引标识的排序序列。

语法

IDXCOLLATE([cCDXFileName,] nIndexNumber [, nWorkArea | cTableAlias])

返值类型

字符型

参数描述

cCDXFileName

指定复合索引文件名。该复合索引文件可以是随表自动打开的结构复合索引文件，也可以是独立的复合索引文件。

nIndexNumber

指定索引或索引标识，IDXCOLLATE（）返回其排序序列。当 *nIndexNumber* 从 1 递增到打开的索引文件和索引标识总数时，IDXCOLLATE（）按如下次序返回索引或索引标识的排序序列：

1. 首先返回单项索引 .IDX 文件（如果有此类文件打开）的排序序列，返回的排序序列取决于 USE 或 SET INDEX 中包含的单项索引文件的序列。
2. 然后返回结构复合索引（如果存在）中标识的排序序列，标识的排序序列按照标识在结构复合索引中创建的次序返回。
3. 最后返回打开的独立复合索引中标识的排序序列，标识的排序序列按照标识在独立的复合索引中创建的顺序返回。

如果 *nIndexNumber* 大于打开的单项索引文件、结构复合索引标识和独立的复合索引标识的总数目，则返回空字符串。

nWorkArea

指定表的工作区，IDXCOLLATE（）函数返回该表的索引文件和索引标识

的排序序列。

如果指定的工作区中没有打开的表，IDXCOLLATE（） 函数将返回空字符串。

cTableAlias

指定表的别名，IDXCOLLATE（） 返回该表的索引文件和索引标识的排序序列。

如果指定的表别名不存在，Visual FoxPro 产生错误信息。

说明

IDXCOLLATE（） 函数可以返回多项复合索引文件中每一标识的排序序列，这样只需使用一系列的 SET COLLATE 和 INDEX 命令，就可以完全删除一个索引文件，然后再正确地重建这个索引文件。

附注 REINDEX 的某些功能并不需要 IDXCOLLATE（），因为其排序序列信息已经存放在索引和索引标识中。

有关 Visual FoxPro 国际化支持的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第十八章“开发国际化应用程序”。

示例

下面示例先打开 testdata 数据库的 customer 表，然后使用 FOR...ENDFOR 创建一个循环，在该循环中应用 IDXCOLLATE（） 函数来显示 customer 结构索引中每个索引标识的排序序列，每个结构索引标识的名称和它的排序序列一起显示。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE Customer   && 打开 customer 表
```

```
CLEAR

FOR nCount = 1 TO 254
   IF !EMPTY(TAG(nCount)) && 检查索引标识
   ? TAG(nCount) + ' ' && 显示标识名
   ?? IDXCOLLATE(nCount) && 显示排序序列
   ELSE
      EXIT && 在找到所有的标识之后退出循环
   ENDIF
ENDFOR
```

请参阅

SET COLLATE

IF...ENDIF 命令

根据逻辑表达式值，有选择地执行一组命令。

语法

```
IF lExpression [THEN]
  Commands
[ELSE
  Commands]
```

ENDIF

参数描述

lExpression

指定要计算的逻辑表达式。如果 *lExpression* **的计算结果为“真”（.T.），则执行** IF **语句之后、**ELSE **或** ENDIF **语句（以先出现的语句为准）之前的所有命令。**

- **如果** *lExpression* **为“假”（.F.）而且包含** ELSE **语句，则执行** ELSE **语句之后、**ENDIF **语句之前的所有命令。**
- **如果** *lExpression* **为“假”（.F.）但不包含** ELSE **语句，则忽略** IF **语句和** ENDIF **之间的所有命令。在这种情况下，程序从** ENDIF **语句后面的第一条命令开始，继续往下执行。**

说明

一个 IF...ENDIF **语句块之中可以嵌套另一个** IF...ENDIF **语句块。**

注释可以放在 IF、ELSE **和** ENDIF **所在行的后面。在编译和执行时，程序将忽略这些注释。**

示例

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE Customer    && 打开 customer 表
```

```
GETEXPR 'Enter condition to locate ' TO gcTemp;
    TYPE 'L' DEFAULT 'COMPANY = ""'
LOCATE FOR &gcTemp  && 输入 LOCATE 表达式
IF FOUND()  && 是否找到?
    DISPLAY  && 是，显示记录
ELSE  && If not found
    ? 'Condition ' + gcTemp + ' was not found '  && 显示相应信息
ENDIF
USE
```

请参阅

DO CASE...ENDCASE, DO WHILE...ENDDO, FOR...ENDFOR, IIF(), SCAN...ENDSCAN

IIF() 函数

根据逻辑表达式的值，返回两个值中的某一个。

语法

IIF(lExpression, eExpression1, eExpression2)

返值类型

字符型、数字型、货币型、日期型或日期时间型

参数描述

lExpression

指定要计算的逻辑表达式。

eExpression1, eExpression2

如果 *lExpression* **计算结果为“真”**(.T.)，**返回** *eExpression1*；**如果** *lExpression* **为“假”**(.F.)，**则返回** *eExpression2*。

说明

该函数也称作 Immediate IF。**它计算一个逻辑表达式的值，然后根据计算结果，返回两个表达式中的一个。如果逻辑表达式的值为“真”**(.T.)，**则** IIF（） **返回第一个表达式；如果逻辑表达式的值为“假”**(.F.)，**则** IIF（） **返回第二个表达式。**

提示 **对于简单的条件表达式，该函数可以代替** IF...ENDIF **语句。在按条件指定报表和标签表达式中的字段内容时** IIF（） **特别有用。** IIF（） **函数比等价语句** IF...ENDIF **执行速度快得多。**

示例

下面的示例用 IIF（） **来检查** employee **表的备注字段是否为空。如果为空，显示“没有说明”；否则显示备注字段的内容。**

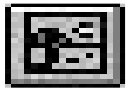
```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE employee  && 打开 Employee 表
CLEAR

SCAN
```

```
   ?IIF(EMPTY(Notes), 'No Notes', notes)   && 备注字段为空否?
ENDSCAN
```

请参阅

IF...ENDIF

Image 控件



创建一个可以显示 .BMP 图片的图像控件。

语法

Image

说明

图像控件是一种图形控件，可以显示 .BMP 图片，但不能直接修改图片。然而，同其他控件一样，图像控件具有一整套属性、事件和方法，因而可以响应事件，并且可以在运行时改变自己。

有关创建图像的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第十章“使用控件”。

属 性

Application	BackStyle	BaseClass
BorderColor	BorderStyle	Class
ClassLibrary	ColorSource	Comment
DragIcon	DragMode	Enabled
Height	HelpContextID	Left
MousePointer	Name	OLEDragMode
OLEDragPicture	OLEDropEffects	OLEDropHasData
OLEDropMode	Parent	ParentClass
Picture	Stretch	Tag
ToolTipText	Top	Visible
WhatsThisHelpID	Width	

事 件

Click	DblClick	Destroy
DragDrop	DragOver	Error
Init	MiddleClick Event	MouseDown
MouseMove	MouseUp	MouseWheel
OLECompleteDrag	OLEDragDrop	OLEDragOver

续表

OLEGiveFeedBack	OLESetData	OLEStartDrag
RightClick	UIEnable	

方法

AddProperty	CloneObject	Drag
Move	OLEDrag	ReadExpression
ReadMethod	ResetToDefault	SaveAsClass
WriteExpression	WriteMethod	Zorder

请参阅

CREATE CLASS, CREATE FORM, DEFINE CLASS

IMEMode 属性

为一个独立的控件指定 Input Method Editor (IME) **窗口设置。设计和运行时可用。**

语法

Object.IMEMode[= nExpression]

参数描述

nExpression

取下列设置之一：

nExpression	IME 窗口动作
0	（默认值）不控制。由操作系统确定当控件获得焦点时是否打开 IME 窗口。如果当控件获得焦点时，IME 窗口是关闭的，可以通过按激活 IME 窗口的组合键打开 IME 窗口。
1	打开 IME。当控件获得焦点时打开 IME 窗口。
2	关闭 IME。当控件获得焦点时关闭 IME 窗口。可以通过按激活 IME 窗口的组合键打开 IME 窗口。

说明

除非您运行的是远东版 Microsoft Windows 95 或 Windows NT，否则会忽略这个属性。

应用于

组合框，编辑框，文本框

请参阅

IMESTATUS（），SET BROWSEIME

IMESTATUS( ) 函数

打开或关闭 IME（输入法编辑器）窗口，或者返回当前的 IME 状态。

语法

IMESTATUS([nExpression])

返值类型

数值型

参数描述

nExpression

打开或关闭 IME（输入法编辑器）窗口。下表列出了 *nExpression* 的值和相应的 IME 窗口的状态。

nExpression	IME 窗口动作
0	关闭 IME 窗口
1	打开 IME 窗口

如果省略 *nExpression*，IMESTATUS( ) 返回当前的 IME 状态。下表列出了 IME 状态的返回值，使用 VERSION(3) 决定当前环境。

下面的表列出了为日语地区的 IME 状态返回的值。

返回值	IME 状态
0	没有安装 IME
1	打开 IME
2	关闭 IME
3	使 IME 不可用
4	Hiragana 模式 (双精度)
5	Katakana 模式 (双精度)
6	Katakana 模式 (单精度)
7	Alpha 数值型模式 (双精度)
8	Alpha 数值型模式 (单精度)

下面的表列出了为朝鲜语地区的 IME 状态返回的值。

返回值	IME 状态
0	没有安装 IME
1	Hangul 模式 (单精度)
2	English 模式 (单精度)
11	English 模式 (双精度)
15	Hangul 模式 (双精度)
23	Hanja 转换模式 (Hangul + 单精度模式)
31	Hanja 转换模式 (Hangul + 双精度模式)

说明

详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第十八章“开发国际化应用程序”中的“输入国际化字符”。

操作对于如 Hiragana 和 Katakana 语言的双精度字符设置时，该函数非常有用。

请参阅

SET BROWSEIME, IMEMode 属性, ISLEADBYTE（）, STRCONV（）

IMPORT 命令

从外部文件导入数据，创建一个 Visual FoxPro 新表。

语法

```
IMPORT FROM FileName
  [DATABASE DatabaseName [NAME LongTableName]]
  [TYPE] FW2 | MOD | PDOX | RPD | WK1
  | WK3 | WKS | WR1 | WRK | XLS
  | XL5 [SHEET cSheetName]
  | XL8 [SHEET cSheetName]
  [AS nCodePage]
```

参数描述

FileName

指定导入数据的文件名。如果不包含文件扩展名，则使用默认扩展名。

DATABASE DatabaseName

创建的新表将加入到本处指定的数据库中。

NAME LongTableName

为新表指定一个长表名。长表名最多可包含 128 个字符。若不指定长表名，则使用 *FileName* 作为表名。

TYPE

关键字 TYPE 是可选的，但 IMPORT 命令中必须包含以下某种文件类型。

文件类型	说明
FW2	包含 FW2 表示要导入由 Framework II 创建的 FW2 型文件。
MOD	包含 MOD 表示要导入由 Microsoft Multiplan 4.0 创建的 MOD 型文件。
PDOX	包含 PDOX 可导入 Paradox 文件，要导入 Borland 公司的 Paradox 3.5 和 4.0 中的数据库文件，可以包含 PDOX。
7RPD	包含 RPD 可导入由 RapidFile 创建的 RPD 型文件。
WK1 | WK3 | WKS	包含 WK1 可以从 Lotus 1-2-3 的电子表格中导入数据。电子表格中的列变为表中的字段，行变为表中的记录。Lotus 1-2-3 2.X 创建的电子表格以 WK1 为扩展名；Lotus 1-2-3 3.X 创建的电子表格以 WK3 为扩展名；Lotus 1-2-3 1-A 创建的电子表格以 WKS 为扩展名。

续表

WR1 | WRK	包含 WR1 可由 Lotus Symphony 电子表格导入数据。电子表格中的列变为表中的字段，行变为表中的记录。Symphony 1.10 产生的电子表格以 .WR1 为扩展名，Symphony 1.1 产生的电子表格以 .WRK 为扩展名。
XLS	包含 XLS 可从 Microsoft Excel 2.0、3.0 和 4.0 的工作表中导入数据。工作表中的列变为表中的字段，行变为表中的记录。Microsoft Excel 创建的工作表文件以 .XLS 为扩展名。
XL5 [SHEET cSheetName]	包含 XL5 可以从 Microsoft Excel 5.0 中导入数据。工作表中的列变为表中的字段，行变为表中的记录。Microsoft Excel 创建的工作表文件以 .XLS 为扩展名。 如果省略了 SHEET 子句，则 Visual FoxPro 从 SHEET1 中导入数据。若要从某一指定工作表中导入数据，就应包含关键字 SHEET，并用参数 *cSheetName* 来指定工作表的名称。
XL8 [SHEET *cSheetName*]	包含 XL8 可以从 Microsoft Excel 97 中导入数据。工作表中的列变为表中的字段，行变为表中的记录。Microsoft Excel 创建的工作表文件以 .XLS 为扩展名。 如果省略了 SHEET 子句，则 Visual FoxPro 从 SHEET1 中导入数据。若要从某一指定工作表中导入数据，就应包含关键字 SHEET，并用参数 *cSheetName* 来指定工作表的名称。

AS nCodePage

指定导入文件的代码页。Visual FoxPro 复制导入文件的内容，并且在复制同

时，把数据自动转换成当前 Visual FoxPro 代码页。

如果 Visual FoxPro 不支持指定的 *nCodePage* 的值，会产生错误信息。也可以使用获取 *nCodePage* 的 GETCP（）函数来显示“代码页”对话框，并在对话框中指定一个代码页。

如果省略了 AS *nCodePage*，而 Visual FoxPro 不能决定导入文件的代码页，Visual FoxPro 会复制导入文件的内容，而且在复制的同时把数据自动转换成当前的 Visual FoxPro 代码页；如果省略了 AS *nCodePage* 子句而 Visual FoxPro 可以决定导入文件的代码页，则 Visual FoxPro 会自动将导入文件中的数据从数据代码页转换成当前的 Visual FoxPro 代码页。使用 CPCURRENT（）可以确定当前的 Visual FoxPro 代码页。

如果 *nCodePage* 的值为 0，Visual FoxPro 假定导入文件的代码页即为当前的 Visual FoxPro 代码页，并且不进行代码页转换。

说明

大部分软件包以自己可以使用的文件格式存储数据，Visual FoxPro 不能直接打开它们。IMPORT 命令使用存储于 Visual FoxPro 不能直接读的文件格式中的数据创建新的 Visual FoxPro 表。

新创建的表与导入数据的文件名同名，但以 .DBF 为扩展名。

请参阅

APPEND FROM, COPY TO, EXPORT, GETCP（）

_INCLUDE 系统变量

指定一个默认的头文件，其中包含用户定义的类、表单或表单集。

语法

_INCLUDE = HeaderFileName

参数描述

HeaderFileName

指定预定义编译常数的默认头文件，其中自动包含用户定义的类、表单或表单集。如果头文件没有位于当前默认目录中，则在头文件名中应该包含路径。

说明

可以通过选项对话框中的默认文件项目确定默认的头文件。

请参阅

CREATE CLASS 命令, CREATE FORM 命令, File Locations Tab, 选项对话框, #INCLUDE

Increment 属性

单击上箭头或下箭头时，微调控件中数值增加或减小的量。设计和运行时可用。

语法

Spinner.Increment[= nIncrement]

参数描述

nIncrement

指定单击上箭头时微调中增加的数值和单击下箭头时微调减少的数值。默认值为 1.00。

应用于

微调

请参阅

RangeLow 事件

IncrementalSearch 属性

指定控件是否支持对键盘操作的递增搜索。设计和运行时可用。

语法

Control.IncrementalSearch[= lExpr]

参数描述

lExpr

下表列出了 IncrementalSearch 属性的设置：

设置	说明
“真” (.T.)	（默认设置）支持递增搜索。
“假” (.F.)	不支持递增搜索。

说明

设计和运行时可用。下面举例说明什么是递增搜索。比如要搜索“ELASTIC”这个单词，可以键入 E-L-A，…等等。在键入字母的时候，Visual FoxPro 会逐步搜索所键入字母的组合，逐字匹配要找的单词。在非递增搜索的情况下，它会找首字母为 E 的第一个单词，然后再找首字母为 L 的第一个单词，…以此类推。

附注 DBLCLICK 系统变量的设置确定了对于下一个键入的字母等待多长时间。可以调整 _DBLCLICK 的值，使得递增搜索正常工作。

应用于

组合框，列表框

请参阅

组合框，_DBLCLICK，列表框

INDBC（） 函数

如果指定的数据库对象在当前数据库中，则返回“真”（T）；否则返回“假”（F）。

语法

INDBC(cDatabaseObjectName, cType)

返值类型

逻辑值

参数描述

cDatabaseObjectName

指定一个命名连接、字段、索引、表或者 SQL 视图的名称，INDBC（） 函数将据此返回一个逻辑值，表明该对象是否存在于当前数据库中。

cType

指定 *cDatabaseObjectName* 的数据库对象类型。下表列出了 *cType* 可能的取值和相应的数据库对象类型：

cType	数据库对象类型
CONNECTION	命名连接
FIELD	字段
INDEX	索引
TABLE	表
VIEW	SQL 视图

CONNECTION、FIELD、INDEX、TABLE 和 VIEW 不能简写。

说明

执行 INDBC（） 函数时，必须打开一个数据库并把它设置为当前数据库。否则，Visual FoxPro 会产生错误信息。

示例

下面示例中，创建一个名为 mydbc 的临时数据库，并把名为 mytable 的临时表添加到这个数据库中，然后用 INDBC（） 函数来检查这个新创建的表是否在该数据库中，最后关闭所创建的数据库和表，并删除它们。

```
CLOSE DATABASES
CREATE DATABASE mydbc  && 创建一个新数据库
CREATE TABLE mytable (field1 C(10)) && 自动添加到数据库中

? 'MyTable in the database? '
?? INDBC('mytable', 'TABLE')  && 返值类型 .T.
```

CLOSE DATABASES
DELETE DATABASE mydbc DELETETABLES

请参阅

ADBOBJECTS（），CREATE DATABASE, DELETE DATABASE, OPEN DATABASE, SET DATABASE

_INDENT 系统变量

使每段的第一行产生缩进。包含此变量是为了提供向后兼容性。可以用报表设计器代替。

INDEX 命令

创建一个索引文件，利用该文件可以按某种逻辑顺序显示和访问表记录。

语法

INDEX ON eExpression TO IDXFileName | TAG TagName [OF CDXFileName]
[FOR lExpression]
[COMPACT]
[ASCENDING | DESCENDING]
[UNIQUE | CANDIDATE]
[ADDITIVE]

参数描述

eExpression

指定一个索引表达式，该表达式中可以包含当前表中的字段名。在索引文件中，按索引表达式给每个表记录都创建一个索引关键字，Visual FoxPro 使用这些关键字来显示和访问表中的记录。

注意 **尽管不提倡，事实上 *eExpession* 也可以是一个变量、数组元素或者其他工作区中表的字段或字段表达式。备注字段不能单独用于索引文件表达式中，它们必须与其他的字符表达式结合起来。如果索引中包含的变量或字段不存在或不能定位，Visual FoxPro 会产生错误信息。**

如果索引表达式中的字段以表别名或工作区字母开头，Visual FoxPro 会产生错误信息。虽然在包含别名字段的情况下，可以用 Rushmore 技术优化 FOR 子句，但在创建索引时最好避免使用别名字段。在有些情况下（USE...AGAIN、SQL 查询等等），Visual FoxPro 会自动给表指定一个不同的别名，这时可能不能正确更新或使用索引。

有关 Rushmore 技术的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“理解 Rushmore 技术”。

如果要建立一个具有可变长度关键字的索引，关键字会被空格填充。Visual FoxPro 不支持可变长度的索引关键字。

索引关键字的长度可以为 0。例如，当索引表达式是空备注字段的一个子字符串时，所创建的索引关键字长度就为 0。当 Visual FoxPro 创建索引时，它检验表中第一个记录的字段，如有一个字段为空，可能需要向第一个记录中这样的字段内填入一些临时数据，以免产生长度为 0 的索引关键字。

.idx 索引的索引关键字长度必须在 1 到 100 个字符间。

.cdx 索引的索引关键字长度必须在 1 到 240 个字符间。

TO IDXFileName

创建 .IDX 索引文件。索引文件的默认扩展名为 .IDX，可以使用一个与之不同的扩展名，也可以在 Visual FoxPro 配置文件内改变索引文件的这个默认扩展名。创建索引文件时，必须遵循标准 Windows 的文件命名规则。

TAG TagName [OF CDXFileName]

创建一个复合索引文件。复合索引文件是一种可包含任意数量的独立标识（索引项）的单个索引文件，每一个标识都由其唯一标识名确定。标识名必须以字母或下划线开头，最多可由 10 个字母、数字或下划线组成。

复合索引文件中的标识数目仅受可用内存和磁盘空间的限制。

多项复合索引文件一般是压缩的。创建复合索引文件时不必包含 COMPACT。复合索引文件的扩展名为 .CDX。

可创建的复合索引文件有两种类型：结构复合索引文件和非结构复合索引文件。

在 TAG *TagName* 参数中不包含可选的 OF *CDXFileName* 子句，便可以创建结构复合索引文件。结构复合索引文件的基本名（不含扩展名的文件名）总是与表的基本名相同，并且自动与表同时打开。

如果一个表的复合索引文件不能被定位、被删除或者已被重命名，则在打开该表时会显示一个对话框。这时如果在对话框中选择默认的“取消”按钮，则不打开表；如选择“忽略”按钮，则打开该表，并且删除表头中的标记。表头可表明该表与结构复合索引文件相关联。

提示　如果一个结构复合索引已经与它的表脱离关系，则用下面的命令可以使它重新和表相关联：

```
USE TableName INDEX CDXFileName
```

在 TAG *TagName* 参数之后包含 OF *CDXFileName*，便可以创建非结构复合索引文件。与结构复合索引不同的是，必须明确使用 SET INDEX 命令或 USE 命令中的 INDEX 子句打开非结构复合索引文件。

创建并打开一个复合索引文件之后，执行带有 TAG *TagName* 参数的 INDEX 命令可以在该复合索引文件中添加一个标识。

其中，*CDXFileName* 是与表脱离关系的结构复合索引名。如果结构复合索引与表脱离关系以后对表进行了修改，则需要重索引该表。

FOR lExpression

指定一个条件，只显示或访问满足这个条件表达式 *lExpression* 的记录，索引文件只为那些满足条件表达式的记录创建索引关键字。

如果 *lExpression* 是一个可优化表达式，Rushmore 将优化 INDEX...FOR *lExpression* 命

令。要获取最佳性能，请在 FOR 子句中使用可优化表达式。

详细内容，请参阅稍后的 SET OPTIMIZE 命令与《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”中的“掌握 Rushmore 技术”。

COMPACT

使用 COMPACT 可以创建一个压缩的 .IDX 文件。

ASCENDING

指定 .CDX 文件为升序。在默认情况下，按升序创建 .CDX 标识（包含 ASCENDING 参数可以将索引文件的排序方式明确指示出来）。类似地，包含 DESCENDING 可按降序索引一个表。

DESCENDING

指定 .CDX 文件为降序。在创建 .IDX 文件时不能包含 DESCENDING 参数，但可以用 SET INDEX 或 SET ORDER 命令将 .IDX 索引文件指定为降序。

UNIQUE

对于一个索引关键字值，只有第一个满足该值的记录包含在 .IDX 文件或 .CDX 标识中。利用 UNIQUE 子句可以避免显示或访问记录的重复值。所有添加到表中的记录，如果与表中原有记录有重复的索引关键字值，则不包含在索引文件之内。使用 INDEX 命令的 UNIQUE 选项，与在执行 INDEX 或 REINDEX 命令之前执行 SET UNIQUE ON 命令完全等效。

在 UNIQUE 索引或索引标识处于激活状态时，如果更改了一个有重复索引关键字值的记录，则同时需要更新索引或索引标识。但在重新用 REINDEX 命令重索引该文件之前，仍然不能访问下一个有相同索引关键字值的记录。

CANDIDATE

创建候选结构索引标识。只有在创建结构索引标识时才能包含关键字 CANDIDATE；否则，Visual FoxPro 会产生错误信息。

使用候选索引标识可以避免索引表达式 *eExpression* 指定的字段或字段组合有重复值。Candidate（候选）一词是指索引类型；因为候选索引中不同的记录没有重复值，所以它们可以作为主索引的“候选”索引。

如果一个字段或字段组合已包含重复值，那么为它创建候选索引标识时，Visual FoxPro 会产生错误信息。

有关候选与主索引标识的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》第七章“处理表”中的“设置主索引或候选索引”。

ADDITIVE

所有先前已打开的索引文件保持打开状态。如省略 ADDITIVE 子句，则在用 INDEX 命令创建索引文件或表时，关闭所有先前已打开的索引文件（结构复合索引文件除外）。

说明

Index 可用来在具有索引文件的表中，按索引表达式所规定的顺序显示或访问记录，但并不会根据索引文件更改表的物理存储顺序。

如果 SET TALK 设置成 ON，则在索引过程中，Visual FoxPro 会报告已建立索引的记录数目。索引过程中显示记录的计数器间隔可由 SET ODOMETER 指定。

使用 DISPLAY STATUS 命令可以显示打开的索引文件的许多内容，其中包括所有打开的索引文件的文件名、类型（结构、非结构、.CDX、.IDX）、索引表达式、排序序列以及主控索引文件名或主控标识名。

可以打开的索引文件（.CDX 或 .IDX）数目只受内存或系统资源的限制。在 Visual FoxPro、FoxPro for Windows 和 FoxPro for MS-DOS 中，能打开的文件总数目由 MS-DOS 的配置文件 CONFIG.SYS 中的 FILES 设置决定。有关 FILES 设置的详细内容，请参阅 MS-DOS 手册。

索引类型 Visual FoxPro 允许创建两种类型的索引文件：

- 包含多个索引标识项（索引名）的 .CDX 复合索引文件
- 包含一个索引项的 .IDX 单项索引文件

您也可以创建一个结构复合索引文件，该文件同表一起自动打开。

提示 也可以创建结构复合索引文件，这种索引文件在打开表时自动打开。因为结构复合索引文件随表的打开而自动打开，所以这种类型比较常用。

包含 COMPACT 文件可创建压缩的 .IDX 索引文件，而复合索引文件总是压缩的。在 Visual FoxPro 与 FoxBASE+ 中共享文件时，应该使用非压缩的 .IDX 索引文件。否则，在创建 .IDX 时，应包含 COMPACT 子句以利用 Visual FoxPro 的 Rushmore 技术。

索引的顺序与更新 表的显示或访问顺序只由一个索引文件（主控索引文件）或标识（主控标识）控制。有一些命令（如 SEEK 命令）使用主控索引文件或标识搜索记录，但是在修改表时，所有已打开的 .IDX 和 .CDX 索引文件都将被更新。使用 USE 命令的 INDEX 子句或 SET INDEX 和 SET ORDER 命令可以指定主控索引文件或标识。

User-Defined Functions 索引表达式中虽然可以包含用户自定义函数，但最好不要这

样做，因为索引表达式中使用用户自定义函数会增加创建或更新索引所需的时间。另外，如果索引表达式中使用了用户自定义函数，有可能不更新这个索引。

如果索引表达式中使用了用户自定义函数，则应该保证 Visual FoxPro 能够找到这个函数。当 Visual FoxPro 创建索引时，索引表达式存储在索引文件中，但用户自定义函数并不存储在索引文件中，索引文件中只保存指向用户自定义函数的引用。

示例

示例 1 打开 customer 表，创建一个名为 complist 的索引文件，显示数值，并且按 company 字段的字母顺序处理记录。

示例 2 再次打开 customer 表，并且根据 city 字段前五个字符和 company 字段前六个字符的子字符串创建一个名为 citycomp 的索引文件。当使用这个索引文件时，表中的记录主要根据 city 字段排序，其次根据 company 字段排序。

示例 3 创建了索引标识。第一个标识是 address 字段的结构复合索引标识。第二个标识是在名为 custcdx 的非结构索引文件中创建的。

```
* 示例 1
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE Customer    && 打开 customer 表
INDEX ON company TO complist
CLEAR
DISPLAY STATUS

* 示例 2
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE Customer    && 打开 customer 表
INDEX ON SUBSTR(city,1,5) + SUBSTR(company,1,6) TO citycomp
```

```
CLEAR
DISPLAY STATUS
```

* **示例** 3

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE Customer    && 打开 customer 表
INDEX ON address TAG address
INDEX ON company TAG company OF custcdx
CLEAR
DISPLAY STATUS
```

请参阅

ALTER TABLE, CDX（）, COPY INDEXES, COPY TAG, DELETE TAG, FOR（）, INDEXSEEK（）, KEY（）, MDX（）, NDX（）, ORDER（）, REINDEX, SET COLLATE, SETINDEX , SET ODOMETER, SET ORDER, SET TALK, SET UNIQUE, SORT, SYS(14), SYS(21), SYS(22), SYS(2021), TAG（）, TAGCOUNT（）, USE

返回总目录

ISDIGIT（）函数
ISEXCLUSIVE（）函数
ISFLOCKED（）函数
ISHOSTED（）函数
ISLEADBYTE（）函数
ISLOWER（）函数
ISMOUSE（）函数
ISNULL（）函数
ISREADONLY（）函数
ISRLOCKED（）函数
ISUPPER（）函数
Item 方法
ItemBackColor, ItemForeColor 属性
ItemData 属性
ItemIDData 属性
ItemIDToIndex 方法
ItemTips 属性
JOIN 命令
JUSTDRIVE（）函数
JUSTEXT（）函数
JUSTFNAME（）函数
JUSTPATH（）函数

INDEXSEEK（）函数

在一个索引表中搜索第一次出现的某个记录，该记录的索引关键字与指定的表达式匹配，可以不移动记录指针。

语法

INDEXSEEK(eExpression [, lMovePointer [, nWorkArea | cTableAlias [, nIndexNumber | cIDXIndexFileName | cTagName]]])

返值类型

逻辑值

参数描述

eExpression

指定索引关键字表达式，您需要 INDEXSEEK（）搜索这个表达式。

lMovePointer

指定是否将记录指针移动到匹配记录。如果 lMovePointer 为“真” (.T.)，并且匹配记录存在，则将记录指针移动到该匹配记录。如果 lMovePointer 为“真” (.T.)，而匹配记录不存在，则不移动记录指针。如果 lMovePointer 为“假” (.F.) 或省略，则即使匹配记录存在，也不移动记录指针。

nWorkArea

指定表的工作区编号，在该表中搜索索引关键字。

cTableAlias

指定该表的别名。如果省略 nWorkArea 和 cTableAlias，则搜索当前选中工作区中的表。

nIndexNumber

指定索引文件或标识的编号，用于搜索索引关键字。nIndexNumber 引用 USE 或 SET INDEX 命令中列出的索引文件。在 USE 或 SET INDEX 命令中打开的 .IDX 文件列为第一个。在结构 .cdx 文件（如果存在）中的标识是按创建顺序编号的。最后，任何打开的独立 .cdx 文件中的标识是按创建顺序编号的。有关索引编号的详细信息，请参阅 SET ORDER。

cIDXIndexFileName

指定用于搜索索引关键字的一个 .idx 文件。

cTagName

指定一个 .cdx 文件的标识，该文件用于搜索索引关键字。标识名可以来自一个结构 .cdx 文件，或任何打开的独立 .cdx 文件。

注意 如果 .cdx 文件和标识名重名，则优先使用 .cdx 文件。

说明

如果匹配记录找到了，则 INDEXSEEK（）返回“真” (.T.)；否则返回“假” (.F.)。只能将 INDEXSEEK（）用于索引排序设置的表，而且只能搜索一个索引关键字。除非 SET EXACT 设置为 OFF，否则匹配必须严格。

INDEXSEEK（）可以在不移动记录指针的情况下快速搜索一个记录。由于不移动记录指针，就不执行规则和触发器。如果 INDEXSEEK（）返回“真” (.T.)，则表明找到了一个匹配记录，可以再次使用 INDEXSEEK()，并且将第二个参数

lMovePointer 设置为“真” (.T.)，以便将记录指针移动到匹配记录。

请参阅

INDEX, KEYMATCH(), LOCATE, SEEK, SEEK()

IndexToItemID 方法

返回一个指定项的 ID 号。

语法

[nItemID =] Control.IndexToItemID(nIndex)

参数描述

nItemID

指定唯一标识号。

nIndex

指定编号，该编号代表在控件中显示对象的位置。

说明

每一个添加到组合框或列表框中的项都有两个标识号：

- *nItemID*，唯一的标识编号。
- *nIndex*，一个整数，对应于控件中显示项的排列位置：列表中的第一个项对

应于 *nIndex* = 1。

在对象刚添加到控件中时，这两个编号是相同的。但随着对象的排序、删除和添加，它们就不一定再相等了。

如果知道控件中一个特定项的 *nIndex* 号时，可使用 IndexToItemID 方法返回它的 *nItemID* 号。

应用于

组合框，列表框

请参阅

AddItem 方法，AddListItem 方法，ItemIDToIndex 方法

Init 事件

在创建对象时发生。

语法

PROCEDURE Object.Init
[LPARAMETERS Param1, Param2,...]

参数描述

Param1, Param2...

参数是可选的。但是如果传递参数，就必须用 LPARAMETERS 或

PARAMETERS 语句列出每一个参数，否则 Visual FoxPro 将产生错误信息。

说明

对于表单集和其他容器对象来说，容器中对象的 Init 事件在容器的 Init 事件之前触发，因此容器的 Init 事件可以访问容器中的对象。容器中对象的 Init 事件的发生顺序与它们添加到容器中的顺序相同。

如果不创建控件或 Active Document，可在 Init 事件中返回“假”（.F.），这时不触发 Destroy 事件。例如，下面的代码在 Invoice 表不存在时返回“假”（.F.）：

```
PROCEDURE INIT
  IF NOT FILE("INVOICE.DBF")
  ERROR 'Initialization Failed: File not found'
  RETURN .F.
  ELSE
  USE INVOICE IN 0 AGAIN
  THIS.WorkArea = SELECT()
  ENDIF
ENDPROC
```

应用于

ActiveDoc 对象，复选框，组合框，命令按钮，命令组，容器对象，控件对象，临时表，自定义控件，数据环境，编辑框，表单，表单集，表格，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，选项组，页面，页框，ProjectHook 对象，关系，形状，微调，文本框，计时器，工具栏

请参阅

AddObject 方法，CREATEOBJECT( )，Load 事件

InitialSelectedAlias 属性

在加载数据环境时，指定一个与临时表对象相关联的别名作为当前别名。

与 SELECT 语句的执行方式相似。

语法

DataEnvironment.InitialSelectedAlias[= cText]

设置

cText

指定一个与临时表对象相关联的别名。

说明

与 SELECT 语句的执行方式相似。

应用于

数据环境

请参阅

Alias 属性, SELECT

INKEY（）函数

返回一个编号，该编号对应于键盘缓冲区中第一个鼠标单击或按键操作。

语法

INKEY([nSeconds] [, cHideCursor])

返值类型

数值型

参数描述

nSeconds

以秒为单位，指定 INKEY（）函数对键击的等待时间。如果不包含 *nSeconds*，INKEY（）函数立即返回一次键击的值；如果 *nSeconds* 为 0，INKEY（）函数一直等待到有键击为止。

cHideCursor

显示或隐藏光标，或者检查鼠标单击。若要显示光标，请在 *cHideCursor* 中包含 S；若要隐藏光标，请在 *cHideCursor* 中包含 H；如果既包含 S 又包含 H，则使用后一个字符的设置。

默认时，INKEY（）函数不检查鼠标单击。如果要检查鼠标单击，可在 *cHideCursor* 中包含 M。若在 *cHideCursor* 中包含了 M，则 INKEY（）函数返回 151 表示一次鼠标单击。

如果既要检查鼠标单击又要显示光标，可在 *cHideCursor* 中包含 M 和 S；若要检查鼠标单击并且隐藏光标，可包含 M 和 H。

当为一个键或组合键指定了键盘宏时，在 *cHideCursor* 中包含 E 可以扩展键盘宏。包含 E 时，INKEY（）函数将返回指定给键盘宏的第一个键击所对应的值，重复执行包含 E 的 INKEY（）函数，可返回后续键击所对应的值；如省略 E，INKEY（）函数将返回触发键盘宏的键或组合键本身的对应值。在 *cHideCursor* 中，除了 H、M、S 和 E 之外的其他字符都被忽略.

下表列出了单键以及单键与 SHIFT、CTRL 和 ALT 键组合时 INKEY（）函数的返回值。破折号(—)表示组合键没有返回值。

键名	单键	SHIFT	CTRL	ALT
F1	28	84	94	104
F2	–1	85	95	105
F3	–2	86	96	106
F4	–3	87	97	107
F5	–4	88	98	108
F6	–5	89	99	109
F7	–6	90	100	110
F8	–7	91	101	111
F9	–8	92	102	112

续表

F10	–9	93	103	113
F11	133	135	137	139
F12	134	136	138	140
1	49	33	–	120
2	50	64	–	121
3	51	35	–	122
4	52	36	–	123
5	53	37	–	124
6	54	94	–	125
7	55	38	–	126
8	56	42	–	127
9	57	40	–	128
0	48	41	–	19
a	97	65	1	30
b	98	66	2	48
c	99	67	3	46
d	100	68	4	32
e	101	69	5	18
f	102	70	6	33
g	103	71	7	34
h	104	72	127	35

续 表

I	105	73	9	23
j	106	74	10	36
k	107	75	11	37
l	108	76	12	38
m	109	77	13	50
n	110	78	14	49
o	111	79	15	24
p	112	80	16	25
q	113	81	17	16
r	114	82	18	19
s	115	83	19	31
t	116	84	20	20
u	117	85	21	22
v	118	86	22	47
w	119	87	23	17
x	120	88	24	45
y	121	89	25	21
z	122	90	26	44
INS	22	22	146	162
HOME	1	55	29	151
DEL	7	7	147	163

续表

END	6	49	23	159
PAGE UP	18	57	31	153
PAGE DOWN	3	51	30	161
上箭头	5	56	141	152
下箭头	24	50	145	160
右箭头	4	54	2	157
左箭头	19	52	26	155
ESC	27	–/27	–*/27	–*/1
ENTER	13	13	10	–/166
BACKSPACE	127	127	127	14
空格键	9	15	148/*	*
SPACEBAR	32	32	32/–	57

* Windows保留的键。

说明

如果没有按下键，则INKEY()函数返回0；如果键盘缓冲区中有多个键，INKEY()函数只返回第一个输入到缓冲区的键的值。

请参阅

_DBLCLICK, KEYBOARD, KeyPress事件, LASTKEY(), ON KEY, READKEY(), SET TYPEAHEAD

INLIST（）函数

判断一个表达式是否与一组表达式中的某个匹配。

语法

INLIST(eExpression1, eExpression2 [, eExpression3 ...])

返值类型

逻辑型或 null 值

参数描述

eExpression1

指定 INLIST（）函数要在表达式组中搜索的表达式。

eExpression2 [, eExpression3 ...]

指定要搜索的表达式组。表达式组中必须至少包含一个表达式（*eExpression2*），最多可包含 24 个。

表达式组中的所有表达式必须具有相同的数据类型。

说明

如 INLIST（）函数在表达式组中找到了要搜索的表达式，就返回“真”（.T.）；否则，返回“假”（.F.）。如果 *eExpression1* 为 null 值，则 INLIST（）函数返回 null 值；如果 *eExpression1* 与表达式组中的任何表达式都不匹配，或者表达式组中有一个表达式为 null 值，INLIST（）函数也返回 null 值。

示例

本示例中，INLIST（）判断当前月份在一年中属于哪个季度。当前月存储在变量 gcMonth 中。每个 EACH 子句用 INLIST（）来判断 gcMonth 的内容是否能在月份名称的列表中找到。返回的季度名称存储在变量 gcReporTitle 中。

```
SET TALK ON
STORE CMONTH(DATE( )) TO gcMonth
DO CASE
    CASE INLIST(gcMonth,'January','February','March')
        STORE 'First Quarter' TO gcReporTitle
    CASE INLIST(gcMonth,'April','May','June')
        STORE 'Second Quarter' TO gcReporTitle
    CASE INLIST(gcMonth,'July','August','September')
        STORE 'Third Quarter' TO gcReporTitle
    OTHERWISE
        STORE 'Fourth Quarter' TO gcReporTitle
ENDCASE
WAIT WINDOW gcReporTitle
```

请参阅

BETWEEN( )

INPUT 命令

包含此项是为了提供向后兼容性。请使用文本框控件。

InputMask 属性

指定控件中数据的输入格式和显示方式。设计和运行时有效。

语法

Control.InputMask[= cMask]

设置

cMask

下表列出了对 InputMask 属性的设置：

设置	说明
X	可输入任何字符。
9	可输入数字和正负符号，如负号 (–)。
#	可输入数字、空格和正负符号。

续表

$	在某一固定位置显示（由 SET CURRENCY 命令指定的）当前货币符号。
$$	在微调控件或文本框中，货币符号显示时不与数字分开。
*	在值的左侧显示星号。
.	句点分隔符指定小数点的位置。
,	逗号可以用来分隔小数点左边的整数部分。

说明

设计和运行时可用。该属性与 Format 恰恰相反，Format 指定整个输入字段的输入方式。可以同时用几种 Format 代码，它们都影响整个输入字段。

应用于

列,组合框， 微调， 文本框

请参阅

DynamicInputMask 属性, Format 属性

INSERT 命令

在当前表中插入新记录。包含此命令是为了提供向后兼容性。可以使用 APPEND 或

INSERT - SQL 命令代替。

INSERT – SQL 命令

在表尾追加一个包含指定字段值的记录。

语法

```
INSERT INTO dbf_name [(fname1 [, fname2, ...])]
  VALUES (eExpression1 [, eExpression2, ...])
```

–或者 –

```
INSERT INTO dbf_name FROM ARRAY ArrayName | FROM MEMVAR
```

参数描述

INSERT INTO dbf_name

指定要追加记录的表名。*dbf_name* 中可以包含路径，也可以是一个名称表达式。

如果指定的表没有打开，则 Visual FoxPro 先在一个新工作区中以独占方式打开该表，然后再把新记录追加到表中。此时并未选定这个新工作区，选定的仍然是当前工作区。

如果所指定的表是打开的，INSERT 命令就把新记录追加到这个表中；如果表不是在当前工作区打开的，则追加记录后表所在的工作区仍然不是选定工作区，选定的仍然是

当前工作区。

[(fname1 [, fname2 [, ...]])]

指定新记录的字段名，INSERT - SQL 命令将向这些字段中插入字段值。

VALUES (eExpression1 [, eExpression2 [, ...]])

新插入记录的字段值。如果省略了字段名，那么必须按照表结构定义字段的顺序来指定字段值。如果 SET NULL 的值为 ON，INSERT – SQL 会试图将 null 值插入在 VALUES 子句中没有指定的任意字段中。

FROM ARRAY ArrayName

指定一个数组，数组中的数据将被插入到新记录中。从第一个数组元素开始，数组中的每个元素的内容依次插入到记录的对应字段中。第一个数组元素的内容插入到新记录的第一个字段，第二个元素的内容插入到第二个字段，… 依此类推。

当包含 FROM ARRAY 子句时，会忽略字段的任何默认值。

FROM MEMVAR

把变量的内容插入到与它同名的字段中。如果某一字段不存在同名的变量，则该字段为空。

说明

新记录中包含了 VALUE 子句列出的值，或包含指定的数组或变量中的值。插入新记录后，记录指针指向新记录。

示例

下面的示例打开了 employee 表并添加一个记录。

```
USE employee
```

```
INSERT INTO employee (emp_no, fname, lname, officeno) ;
    VALUES (3022, "John", "Smith", 2101)
```

下面的示例打开了数据库 testdata 中的 customer 表。当前记录的内容分散到变量中，并且表的结构被复制到新表 cust2 中。可用 INSERT – SQL 在表 cust2 中插入新记录，也可以发出 BROWSE 来显示新记录。

```
CLOSE DATABASES
CLEAR

OPEN DATABASE (HOME(2) + 'Data\testdata')
USE Customer    && 打开 customer 表
*把当前记录存入内存变量中
SCATTER MEMVAR

*把当前表的结构复制到 EXAMPLE 表中
COPY STRUCTURE TO cust2

*从内存变量中插入记录
INSERT INTO cust2 FROM MEMVAR

SELECT CUST2
BROWSE

*关闭并删除 EXAMPLE 表
USE
DELETE FILE cust2.dbf
```

请参阅

CREATE QUERY, CREATE TABLE – SQL, MODIFY QUERY, SELECT –

SQL

INSMODE（）函数

返回当前的插入方式，或者把插入方式设置成 ON 或 OFF。

语法

INSMODE([lExpression])

返值类型

逻辑值

参数描述

lExpression

把插入方式设置成 ON 或 OFF。INSMODE(.T.)将插入方式设置成 ON，INSMODE(.F.)将插入方式设置成 OFF。返回的逻辑值表示在执行 INSMODE(.T.)或 INSMODE(.F.)之前的插入方式。

说明

省略可选参数时，如果插入方式处于 ON 状态（在光标前面插入字符），则 INSMODE（）函数返回“真”(.T.)；如果插入方式处于 OFF 状态（改写光标所在位置的字符），则 INSMODE（）函数返回“假”(.F.)。

示例

下面的示例利用 INSMODE( ) 函数把插入方式设置成 ON，然后再把插入方式切换成相反的状态。

```
SET TALK ON
=INSMODE(.T.) && 把插入方式置为 ON
? INSMODE( )
= INSMODE(!INSMODE( )) && 把插入方式切换到相反的状态
? INSMODE( )
```

请参阅

CAPSLOCK( ), NUMLOCK( )

Instancing 属性

指定项目中的一个服务程序如何被实例化。设计和运行时可用。

语法

Object.Instancing[= nExpression]

设置

nExpression

指定如何实例化一个服务程序。下表列出了 nExpression 的值，以及每个值的说明。

设置	FoxPro.h 常数	说明
1	SERVERINSTANCE_SINGLEUSE	（默认值）允许您在 Visual FoxPro 内外使用 OLE 自动服务创建该类的一个实例。项目之外的一个自动服务客户程序对该类一个实例的每一个请求都会启动自动服务程序的一个独立副本。
2	SERVERINSTANCE_NOTCREATABLE	只允许您在 Visual FoxPro 内创建该类的一个实例。
3	SERVERINSTANCE_MULTIUSE	允许您在 Visual FoxPro 内外使用 OLE 自动服务创建该类的一个实例。 项目之外的一个自动服务客户程序对该类一个实例的每一个请求会使自动服务程序的一个已经运行的副本成为新实例的源。

应用于

服务程序对象

请参阅

INT（） 函数

计算一个数值表达式的值，并返回其整数部分。

语法

INT(nExpression)

返值类型

数值型

参数描述

nExpression

指定 INT（） 计算的数值表达式。

示例

```
CLEAR
? INT(12.5) && 显示数值 12
? INT(6.25 * 2) && 显示数值 12
? INT(-12.5) && 显示数值 -12
STORE -12.5 TO gnNumber
? INT(gnNumber) && 显示数值 -12
```

请参阅

CEILING( ), FLOOR( ) , ROUND( )

IntegralHeight 属性

指定是否自动调整编辑框或列表框控件的高度，这样可以正确显示控件中的最后一项。指定是否自动调整文本框控件的高度，以显示一行文本。设计时可用；运行时只读。

语法

Object.IntegralHeight[= lExpr]

参数描述

lExpr

取下列一个值：

lExpr	说明
“真” (.T.)	自动调整编辑框或列表框控件的高度，这样可以正确显示控件中的最后一项。自动调整文本框控件的高度，以显示一行文本。
“假” (.F.)	默认值。编辑框或列表框的高度不可调，不可以正确显示控件的最后一项。不可以自动调整文本框控件的高度，以显示一行文本。

说明

如果编辑框或列表框控件的高度不合适，则控件中的最后一行文字会只显示一部分。将 IntergralHeight 设置为“真” (.T.)，可以自动调整控件的高度，这样可以正确显示控件中的最后一项。

对于一个文本框，将 IntergralHeight 设置为“真” (.T.) 可以保证当控件的 FontSize 属性改变时，该控件能自动调整自己的高度。

当 IntegralHeight 属性设置为“真” (.T.) 时，Height 属性的值可能与控件的真实高度不符。

应用于

编辑框，列表框，文本框

请参阅

Height 属性

InteractiveChange 事件

在使用键盘或鼠标更改控件的值时，该事件发生。

语法

PROCEDURE Control.InteractiveChange

[LPARAMETERS nIndex]

参数描述

nIndex

对于控件数组中的控件，指定唯一标识号。

说明

在每次交互地更改对象时，都要发生该事件。例如，当用户在文本框中键入字符时，每一次击键都会触发 InteractiveChange 事件。

应用于

复选框，组合框，命令组，编辑框，列表框，选项组，微调，文本框

请参阅

Click 事件

Interval 属性

指定计时器控件的 Timer 事件之间的时间间隔毫秒数。设计和运行时可用。

语法

Timer.Interval[= nTime]

参数描述

nTime

指定 Timer 事件之间的间隔毫秒数。默认为 0，不触发 Timer 事件。

说明

有关 interval 属性的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第十章“使用控件”中的“计时器控件的初始化”。

应用于

计时器

请参阅

Reset 方法, Timer 事件

ISALPHA( ) 函数

判断字符表达式的最左边一个字符是否为字母。

语法

ISALPHA(cExpression)

返值类型

逻辑值

参数描述

cExpression

ISALPHA() 函数所要判断的字符表达式。*cExpression* 中第一个字符之后的所有字符都将被忽略。

说明

如果字符表达式的第一个字符是字母，ISALPHA() 函数将返回“真”(.T.)；否则，返回“假”(.F.)。

示例

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE Customer    && 打开 customer 表
CLEAR

DISPLAY contact
? ISALPHA(contact)  && 显示 .T.
DISPLAY maxordamt
? ISALPHA(cust_id)  && 显示 .F.
```

请参阅

ISLOWER(), ISUPPER(), LOWER(), UPPER()

ISBLANK（）函数

判断表达式是否为空值。

语法

ISBLANK(eExpression)

返值类型

逻辑值

参数描述

eExpression

ISBLANK（）要判断的表达式。*eExpression* 可以是表中的一个字段、一个变量或数组元素，也可以是一个表达式。

对一个字段来说，如果该字段包含下述值，则 ISBLANK（）函数将返回“真”(.T.)。

类型	内容
字符型	空字符串、空格或无值（新追加的空记录或用 BLANK 命令清除后的记录）
数值型	无值（新追加的空记录或用 BLANK 命令清除后的记录）
浮点型	无值（新追加的空记录或用 BLANK 命令清除后的记录）
日期型	空日期时间（{ / / : : }）或无值（新追加的空记录或用 BLANK 命令清除后的记录）

续表

日期时间型	空日期时间（{ / / : : }）或无值（新追加的空记录或用 BLANK 命令清除后的记录）
逻辑型	无值（新追加的空记录或用 BLANK 命令清除后的记录）
备注型	空（无备注内容）
通用型	空（无 OLE 对象）
图片	空（无图片）

说明

如果表达式 *eExpression* 为空值，则 ISBLANK（）函数返回“真”(.T.)；否则，返回“假”(.F.)。

APPEND BLANK 和 BLANK 命令可用来创建空记录，BLANK 命令还可以用来清除一个记录内某些字段中的数据。用 ISBLANK（）函数则可以判断一个字段是否为空值。

附注货币型、整型、双精度型表达式永远不可能为空值，因而对这些表达式 ISBLANK（）函数总是返回“假”(.F.)。

ISBLANK（）函数与 EMPTY（）和 ISNULL（）函数不同。例如，当字符表达式是 null 值、空格、Tab 字符、回车、换行符的某种组合时，EMPTY（）函数将返回“真”(.T.)；但只有字符表达式中仅包含空字符串或空格时，ISBLANK（）函数才返回“真”(.T.)。

示例

下面的示例创建了表 mytable 并追加了一个空白记录。ISBLANK（）返值类型 true

(.T.)，因为 myfield 为空。在 myfield 中放置值，并且 ISBLANK() 返值类型 false

(.F.)，因为 myfield 不再为空。

```
CREATE TABLE mytable FREE (myfield C(20))
APPEND BLANK && 添加新的空记录
CLEAR

? ISBLANK(myfield) && 显示.T.
REPLACE myfield WITH 'John Smith' && 在字段中插入一个值
? ISBLANK(myfield) && 显示.F.
```

请参阅

APPEND, BLANK, EMPTY( ), ISNULL( ) , LEN( )

ISCOLOR( ) 函数

判断当前计算机能否显示彩色。

语法

ISCOLOR( )

返值类型

逻辑值

说明

如果当前计算机具有彩色显示能力（不管当前所用的是否为彩色监视器），ISCOLOR（）函数将返回“真”(.T.)；如果当前计算机不支持彩色显示，ISCOLOR（）函数将返回“假”(.F.)。

请参阅

颜色概览，SET DISPLAY，SYS(2006)

ISDIGIT（）函数

判断字符表达式的最左边一个字符是否为数字（0到9）。

语法

ISDIGIT(cExpression)

返值类型

逻辑值

参数描述

cExpression

指定ISDIGIT（）函数要判断的字符表达式。*cExpression*中第一个字符之后的所有字符都被忽略。

说明

如果字符表达式的最左边一个字符是数字（0 到 9），则 ISDIGIT（）函数返回“真”(.T.)；否则，ISDIGIT（）函数返回“假”(.F.)。

示例

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE orders  && 打开 Orders 表
CLEAR

DISPLAY cust_id
? ISDIGIT(cust_id)  && 显示.F.
DISPLAY order_dsc
? ISDIGIT(ALLTRIM(STR(order_dsc)))  && 显示.T.
```

请参阅

ISALPHA()

ISEXCLUSIVE（）函数

如果一个表或数据库是以独占方式打开的，则返回“真”(.T.)；否则，返回“假”(.F.)。

语法

ISEXCLUSIVE([cTableAlias | nWorkArea | cDatabaseName [, nType]])

返值类型

逻辑值

参数描述

cTableAlias

指定的表别名，ISEXCLUSIVE（）函数将返回它的独占使用状态。如指定的表别名不存在，Visual FoxPro会显示错误信息。

nWorkArea

指定的工作区，ISEXCLUSIVE（）函数将返回指定工作区中表的独占使用状态；如果所指定的工作区中没有打开的表，则ISEXCLUSIVE（）函数返回“假”(.F.)。

cDatabaseName

指定的数据库名，ISEXCLUSIVE（）函数返回该数据库的独占使用状态。

nType

指定所返回的是表的独占使用状态还是数据库的独占使用状态。下表列出了*nType*的值和相应的返回状态。

nType	要返回独占使用状态的对象
1	表
2	数据库

若要判断数据库的独占使用状态，必须把*nType*设置为2。

说明

如果省略了 *cTableAlias*、*nWorkArea* 和 *cDatabaseName* 三个可选参数，ISEXCLUSIVE（）函数返回当前选定工作区中表的独占使用状态。

要以独占方式打开一个表，可以在 USE 命令中包含关键字 EXCLUSIVE 子句，也可以在打开表之前把 SET EXCLUSIVE 设置成 ON。

要以独占方式打开数据库，可在 OPEN DATABASE 命令中包含关键字 EXCLUSIVE。

示例

在下例中，使用 ISEXCLUSIVE（）函数检验表是否以独占方式打开。由于当前工作区中的表不是以独占方式打开，所以表没有被重索引。

```
cExclusive = SET('EXCLUSIVE')
SET EXCLUSIVE OFF
SET PATH TO (HOME(2) + 'data\')
OPEN DATA testdata  &&打开一个数据库
USE customer    &&不以独占方式打开 customer 表
USE employee IN 0 EXCLUSIVE   &&在另一个工作区中以独占方式打开
IF ISEXCLUSIVE( )
   REINDEX  &&只有当表以独占方式打开时，才重建索引
ELSE
   WAIT WINDOW 'The table has to be exclusively opened'
ENDIF
SET EXCLUSIVE &cExclusive
```

请参阅

OPEN DATABASE, SET EXCLUSIVE, USE

ISFLOCKED（）函数

返回表的锁定状态。

语法

ISFLOCKED([nWorkArea | cTableAlias])

返值类型

逻辑值

参数描述

nWorkArea

指定数据区，需要返回其中锁定状态的表就在该数据区中。如果您既不指定 *nWorkArea*，也不指定 *nWorkArea*，则当前工作区中的表的锁定状态被返回。

cTableAlias

指定返回锁定状态的表的别名。如果指定了别名的表没有打开，也将出现“没有找到别名”的错误信息。

说明

如果指定的表被锁定，ISFLOCKED（）返回真 (.T.)；否则返回假 (.F.)。ISFLOCKED（）与 SYS(2011) 类似。但 ISFLOCKED（）返回逻辑值；SYS(2011) 返回字符值。

请参阅

中运行。

请参阅

GETHOST()

ISLEADBYTE( ) 函数

如果字符表达式第一个字符的首字节是前导字节，则返回“真”(.T.)。

语法

ISLEADBYTE(cExpression)

返值类型

逻辑值

参数描述

cExpression

指定 ISLEADBYTE( ) 求值的字符表达式，忽略 *cExpression* 中第一个字符的第一个字节之后的任何字节。

说明

如果字符表达式第一个字符的首字节是前导字节，ISLEADBYTE( ) 返回“真”(.T.)，否则返回“假”(.F.)。

该函数可用于管理双字节字符集。

请参阅

ISALPHA( ), ISDIGIT( ) , ISLOWER( ) , ISUPPER( )

ISLOWER（）函数

判断字符表达式最左边的字符是否为小写字母。

语法

ISLOWER(cExpression)

返值类型

逻辑值

参数描述

cExpression

指定 ISLOWER（）函数要判断的字符表达式。*cExpression* 中第一个字符之后的所有字符都被 ISLOWER（）函数忽略。

说明

如果指定字符表达式的第一个字符是小写字母，ISLOWER（）函数返回“真”(.T.)；否则返回“假”(.F.)。

示例

```
CLEAR
?ISLOWER('redmond') &&显示数值 .T.
?ISLOWER('Redmond') &&显示数值 .F.
```

请参阅

ISALPHA( ), ISUPPER( ) , LOWER( ) , UPPER( )

ISMOUSE（） 函数

如果所用计算机装有鼠标，则返回“真”(.T.)。

语法

ISMOUSE( )

说明

若有鼠标，ISMOUSE（）**函数返回“真”(.T.)；否则返回“假”(.F.)。**

示例

下面的示例在有鼠标时显示“真”(.T.)；否则显示“假”(.F.)。

```
CLEAR
? 'Mouse hardware present?'
?? ISMOUSE( )
```

请参阅

MCOL( ), MDOWN( ) , MROW( ) , SYSMETRIC( )

ISNULL() 函数

如果一个表达式的计算结果为 null 值，则返回“真”(.T.)；否则返回“假”(.F.)。

语法

ISNULL(eExpression)

返值类型

逻辑值

参数描述

eExpression

指定要计算的表达式。

说明

ISNULL() 函数可用于判断字段、变量或数组元素是否为 null 值，也可以判断表达式的计算结果是否为 null 值。

示例

下面的示例中，ISNULL() 用于检查 null 值。

```
STORE .NULL. TO mNullvalue &&将 null 值存入内存变量

CLEAR
? mNullvalue &&显示内存变量的值
? ISNULL(mNullvalue) && 返回.T.,表示为 null 值
```

```
? TYPE('mNullvalue')    && 返回 L, 表示为逻辑值
? (mNullvalue = .NULL.) && 返回 .NULL., null 值测试有误
```

请参阅

NVL(), SET NULL

ISREADONLY（）函数

判断是否以只读方式打开表。

语法

ISREADONLY([nWorkArea | cTableAlias])

返值类型

逻辑值

参数描述

nWorkArea | cTableAlias

返回其他工作区中打开表的只读状态。 *nWorkArea* **指定工作区号，** *cTableAlias* **指定表或工作区的别名。若在指定的工作区中没有打开的表，** ISREADONLY（）**函数返回“假”** (.F.)。

如果没有指定工作区号、表别名或工作区别名，则返回当前工作区中打开的表的“只读”状态。

说明

若表是以只读方式打开，ISREADONLY（）返回“真”(.T.)；否则，返回“假”(.F.)。

要以只读方式打开表，可以使用包含 NOUPDATE 选项的 USE 命令，也可以在“打开”对话框打开该表时，在对话框中选定“只读”复选框，还可以把表文件的 MS-DOS 文件属性设置为只读。

用 SELECT - SQL 命令创建的临时表总是只读的。

示例

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer  NOUPDATE && 以只读方式打开 customer 表

CLEAR
? ISREADONLY('customer') && return .T.
```

请参阅

USE

ISRLOCKED（）函数

返回记录的锁定状态。

语法

ISRLOCKED([nRecordNumber, [nWorkArea | cTableAlias]])

返值类型

逻辑值

参数描述

nRecordNumber

指定一个记录号。该记录的锁定状态被返回。如果您省略记录号，当前记录的锁定状态被返回。

nWorkArea

指定记录所在的表所在的工作区。如果您既不指定 *nWorkArea*，也不指定 *nWorkArea*，则默认返回当前工作区中的表中的记录的锁定状态。

cTableAlias

指定表的别名。

说明

如果指定记录处于锁定状态，ISRLOCKED（）返回真 (.T.)；否则返回假 (.F.)。

请参阅

FLOCK(), ISFLOCKED() , LOCK(), RLOCK() , SYS(2011)

ISUPPER（）函数

判断字符表达式的首字符是否为大写字母 (A～Z)。

语法

ISUPPER(cExpression)

返值类型

逻辑值

参数描述

cExpression

指定要判断的字符表达式。*cExpression* 中第一个字符以后的所有字符都将被忽略。

说明

若字符表达式的第一个字符为大写字母 (A～Z)，ISUPPER（）函数返回“真” (.T.)；否则，返回“假” (.F.)。

示例

```
? ISUPPER('Redmond') && 显示数值 .T.
? ISUPPER('redmond') && 显示数值 .F.
```

请参阅

ISALPHA( ), ISLOWER( ) , LOWER( ) , UPPER( )

Item 方法

返回对文件集合、项目集合或服务程序集合中一个指定文件、项目或服务程序的对象引用。

语法

Object.Item(nIndex)

nIndex

一个指定文件集合、项目集合或服务程序集合中一个文件、项目或服务程序的值。如果 nIndex 比集合中的文件、项目或服务程序数目多，则会造成一个“无效引用”错误。

说明

使用 Name 属性和 Item 属性，可以确定一个文件、项目或服务程序的名称，以及它们的目录。

应用于

文件集合，项目集合，服务程序集合

请参阅

Name 属性

ItemBackColor, ItemForeColor 属性

在组合框或列表框控件中，指定显示数据项文本的背景色或前景色。设计和运行时可用。

语法

Control.ItemBackColor[= nColor]

– 或 –

Control.ItemBackColor = RGB(nRedValue, nGreenValue, nBlueValue)

Control.ItemForeColor[= nColor]

– 或 –

Control.ItemForeColor = RGB(nRedValue, nGreenValue, nBlueValue)

参数描述

nColor

指定表示文本颜色的整数。

注意 在属性窗口中双击颜色属性，系统显示“颜色”对话框，可以在此对话框中选择或定义颜色。关闭“颜色”对话框后，与所选颜色相对应的红、绿、蓝三色的深度就变成这些属性的设置。

有关详细内容，请参阅 BackColor, ForeColor 属性中的颜色表主题。

应用于

组合框，列表框

请参阅

BackColor, ForeColor 属性, SelectedItemBackColor, SelectedItemForeColor 属性

ItemData 属性

使用索引引用一维数组，该数组包含的数据项数目与组合框或列表框控件中 List 属性的设置相等。设计时不可用，运行时可读写。

语法

Control.ItemData(nIndex)[= nData]

参数描述

nIndex

指定要保存或检索的数据项的索引，*nIndex* 对应于列表中数据项的显示顺序。

nData

ItemData 列表中，要保存或检索的编号。

说明

使用 ItemData 属性，可使组合框或列表框中的每个数据项都与一个指定编号相联系，然后可以在程序中使用这些编号来标识列表中的各个数据项。例如，在列表框中，可用编号来标识每个雇员的姓名。在填充列表框的同时，也把雇员的编号填充到 ItemData 数组的对应元素中。

注意 当使用 AddItem 方法向列表中插入数据项时，ItemData 数组中也将自动插入一个数据项，但并未对该数据项的值进行初始化，即这个数据项所在位置的值并没有发生变化。因而，如果使用 ItemData 属性，当向列表中添加新数据项时，一定要给每个数组元素赋值。

另请注意 ItemData 属性的值只有在组合框或列表框的项少于 60 时才正确。

应用于

组合框，列表框

请参阅

AddItem 方法，ItemIDToIndex 方法，List 属性，ListItemID 属性，NewItemID 属性 RemoveItem 方法，Selected 属性，TopItemID 属性

ItemIDData 属性

使用唯一的标识编号来引用一维数组，该数组中包含的数据项数目与组合框或列表框

控件中 List 属性的设置相等。设计时不可用，运行时可读写。

语法

Control.ItemIDData(nItemID)[= nData]

参数描述

nItemID

指定数据项的唯一标识号。

nData

ItemIDData 列表中要保存或检索的编号。

说明

使用 ItemIDData 属性，可使组合框或列表框中的每个数据项都与一个指定的编号相联系，然后可以在程序中使用这些编号来标识列表中的各个数据项。例如，对于每一个雇员的列表可以有一个对应的编号，当在列表中添加新数据项时，ItemIDData array 数组中对应编号的相应项也做相应改动。

注意 当使用 AddItem 方法向列表中插入数据项时，ItemIDData 数组中也将自动插入一个数据项，但并未对该数据项的值进行初始化，即这个数据项所在位置的值并没有发生变化。因而在使用 ItemIDData 属性向列表中添加新数据项时，一定要给每个数组元素赋值，该数组与 ItemData 属性所访问的数组相同。

应用于

组合框，列表框

请参阅

IndexToItemID 方法, List 属性, ListItemID 属性, NewItemID 属性, RemoveItem 方法, Selected 属性, TopItemID 属性

ItemIDToIndex 方法

返回 *nIndex* 值，这个值指示数据项在控件列表中的位置。

语法

[nIndex =]Control.ItemIDToIndex(nItemID)

参数描述

nItemID

与数据项相联系的唯一标识号。

说明

添加到组合框或列表框中的每个数据项都具有两个编号：

- *nItemID*，唯一的标识编号。
- *nIndex*，一个整数，它对应于控件中数据项显示顺序。列表中的第一个数据项对应于 *nIndex* = 1。

在数据项刚添加到控件中时，这两个编号是相同的。但随着数据项的排序、删除和添加，这两个编号可能不相同。

当知道控件列表中某一指定项的 *nItemID* 编号时，可使用 ItemIDToIndex 方法获得它的 *nIndex* 编号。

应用于

组合框，列表框

请参阅

AddItem 方法，AddListItem 方法，IndexToItemID 方法，ListItemID 属性，List 属性，ListCount 属性

ItemTips 属性

指定组合框或列表框中的项是否显示条目提示。设计时可用；在运行时刻可读写。

语法

Control.ItemTips[= lExpression]

参数描述

lExpression

ItemTip 属性的设置有：

设置	说明
“真” (.T.)	组合框或列表框中的项显示条目提示。
“假” (.F.)	（默认值）组合框或列表框中的项不显示条目提示。

说明

条目提示是一个小窗口，当鼠标指针位于组合框或列表框的一项上时，条目提示显示整个项。当组合框或列表框中的项比控件长时，可将这个属性设置为“真”任何时候组合框或列表框控件中的项都比控件的宽度大。

应用于

组合框，列表框

请参阅

ItemData 属性

JOIN 命令

包含此命令是为了提供向后兼容性。可使用 SELECT - SQL 命令代替。

JUSTDRIVE（）函数

从完整路径中返回驱动器的字母。

语法

JUSTDRIVE(*cPath*)

返值类型

字符型

参数描述

cPath

指定完整路径名，您只需要驱动器的字母。

请参阅

ADDBS( ), DEFAULTEXT( ), FILE( ), FORCEEXT( ), FORCEPATH( ), JUSTEXT( ), JUSTFNAME( ), JUSTPATH( ), JUSTSTEM( )

JUSTEXT( ) 函数

从完整路径中返回三字母的扩展名。

语法

JUSTEXT(*cPath*)

返值类型

字符型

参数描述

cPath

指定文件的名称，其中包含完整路径，您只需其中的扩展名。

请参阅

ADDBS( ), DEFAULTEXT( ), FILE( ), FORCEEXT( ), FORCEPATH( ), JUSTDRIVE( ), JUSTFNAME( ), JUSTPATH( ), JUSTSTEM( )

JUSTFNAME( ) 函数

返回完整路径和文件名中的文件名部分。

语法

JUSTFNAME(*cFileName*)

返值类型

字符型

参数描述

cFileName

指定文件的名称，其中包含完整路径，您只需其中的文件名。

请参阅

ADDBS( ), DEFAULTEXT( ), FILE( ), FORCEEXT( ), FORCEPATH( ), JUSTDRIVE( ), JUSTEXT( ), JUSTPATH( ), JUSTSTEM( )

JUSTPATH（）函数

返回完整路径中的路径名。

语法

JUSTPATH(*cFileName*)

返值类型

字符型

参数描述

cFileName

指定文件的名称，其中包含完整路径，您只需其中的路径。

请参阅

ADDBS( ), DEFAULTEXT( ), FILE( ), FORCEEXT( ), FORCEPATH( ), JUSTDRIVE( ), JUSTEXT( ), JUSTFNAME( ), JUSTSTEM( )

JUSTSTEM（）函数

返回完整路径和文件名中的根名（扩展名前的文件名）。

语法

JUSTSTEM(*cFileName*)

返值类型

字符型

参数描述

cFileName

指定文件的名称，其中包含完整路径，您只需其中的根名。

请参阅

ADDBS( ), DEFAULTEXT( ), FILE( ), FORCEEXT( ), FORCEPATH( ), JUSTDRIVE( ), JUSTEXT( ), JUSTFNAME( ), JUSTPATH( )

KEY（）函数

返回索引标识或索引文件的索引关键字表达式。

语法

KEY([CDXFileName,] nIndexNumber [, nWorkArea | cTableAlias])

返值类型

字符型

参数描述

CDXFileName

指定复合索引文件的文件名。KEY（）函数将返回 .CDX 文件中索引标识的索引关键字表达式。所指定的复合索引文件可以是和表一起自动打开的结构复合索引文件，也可以是独立的复合索引文件。

nIndexNumber

指定要返回哪一个索引关键字表达式。

USE 和 SET INDEX 命令都支持一个索引文件列表，从中可以为一个表同时打开多个索引文件。索引文件列表中可以包含 .IDX 单项索引文件、结构复合索引文件或独立的复合索引文件等三种索引文件的任意组合。

数值表达式 *nIndexNumber* 指定从打开的索引文件中返回哪一个索引表达式。随着 *nIndexNumber* 从 1 递增到所打开的 .IDX 单项索引文件、结构复合索引文件和独立复合

索引文件三者的总数，KEY（）函数将按下列顺序依次从打开的各索引文件中返回索引表达式：

1. 首先返回 .IDX 单项索引文件（如果有此类文件打开）中的索引表达式。各个单项索引文件在 USE 或 SET INDEX 命令中的排列顺序决定索引表达式的返回顺序。
2. 其次返回结构复合索引（如果有）中的每个标识的索引表达式。多个标识的索引表达式的返回顺序取决于各标识在结构复合索引中的创建顺序。
3. 最后返回打开的独立复合索引中的每个标识的索引表达式。多个标识的索引表达式的返回顺序取决于标识在独立复合索引中创建的顺序。

若 *nIndexNumber* 大于打开的索引文件总数（包括 .IDX 单项索引文件、结构复合索引文件和独立的复合索引文件），则返回空字符串。

nWorkArea

指定表所在的工作区号，KEY（）函数将返回该表的索引关键字表达式。

如果指定的工作区中没有表打开，则 KEY（）函数返回空字符串。

cTableAlias

指定表别名，KEY（）函数将返回该表的索引关键字表达式。

若指定的表别名不存在，Visual FoxPro 将产生错误信息。

若省略 *nWorkArea* 和 *cTableAlias* 参数，将返回当前工作区中打开表的索引关键字表达式。

说明

用 INDEX 命令创建索引标识或索引文件时可指定其索引关键字表达式。当该索引标识或索引文件作为主控索引标识或主控索引文件打开时，其索引关键字表达式将决定

表的显示或访问方式。

有关创建索引标识、索引文件以及索引关键字表达式的详细内容，请参阅 INDEX。

示例

下面的示例先打开 testdata 数据库中的 customer 表，然后在 FOR ... ENDFOR 创建的循环中使用 KEY（）函数来显示 customer 结构索引文件中每个索引标识的索引表达式。每个结构索引标识名和它的索引表达式同时显示。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE Customer    && 打开 customer 表
CLEAR

FOR nCount = 1 TO 254
   IF !EMPTY(TAG(nCount)) &&  检查索引文件中的标识
   ? TAG(nCount) + ' ' && 显示标识的名称
   ?? KEY(nCount) && 显示索引表达式
   ELSE
      EXIT && 所有标识都搜索完以后退出循环
   ENDIF
ENDFOR
```

请参阅

INDEX, REINDEX, SET INDEX, SYS(14)

KEYBOARD 命令

把指定的字符表达式放置到键盘缓冲区中。

语法

KEYBOARD cKeyboardValue
[PLAIN] [CLEAR]

参数描述

cKeyboardValue

指定要放入键盘缓冲区的字符表达式。字符表达式可以是字符串、一个键标记、一组键标记，也可以是一个可返回字符表达式的用户自定义函数。

若 *cKeyboardValue* 是一个键标记，则必须把它用大括号和引号括起来。例如：

```
KEYBOARD '{CTRL+LEFTARROW}'
```

有关键标记的列表，请参阅 ON KEY LABEL。键盘缓冲区中最多可容纳 128 个字符。当键盘缓冲区完全充满以后，多余的字符将被忽略。

PLAIN

如果您定义了键盘宏或激活了 ON KEY LABEL 命令，包含 PLAIN 参数可忽略这些键盘指定。包含 PLAIN 参数的 KEYBOARD 命令向键盘缓冲区中放置的是原义字符，而非键盘指定值。

例如，若您已用 ON KEY LABEL 命令给“A”键指定了一条命令而且 *cKeyboardValue* 中又包含有“A”，使用 PLAIN 参数可把字母“A”放到键盘缓冲区中，而没有执行指定给“A”的 ON KEY LABEL 命令。

CLEAR

在把 *cKeyboardValue* 放置到键盘缓冲区以前，先清除该缓冲区内的所有内容。

说明

使用 KEYBOARD 命令可把字符放置到键盘缓冲区中。在 Visual FoxPro 查找键盘输入之前，这些字符将一直保存在键盘缓冲区中。在查找键盘输入时，这些字符将被读出并执行，如同我们直接从键盘输入一样。

利用 KEYBOARD 命令可创建自动运行的演示系统，来演示您的应用程序。

请参阅

CHRSAW(), ON KEY LABEL, PLAY MACRO, SET FUNCTION

KeyboardHighValue, KeyboardLowValue 属性

指定可用键盘输入到微调控件文本框中的最大、最小值。设计和运行时可用。

语法

Spinner.KeyboardHighValue[= nHigh]

Spinner.KeyboardLowValue[= nLow]

参数描述

nHigh

指定可输入的最大值。 *nHigh* **的默认值为** 2,147,483,647。

nLow

指定可输入的最小值。 *nLow* **的默认值为** 2,147,483,647。

应用于

微调

请参阅

SpinnerHighValue, SpinnerLowValue **属性**

KEYMATCH( ) 函数

在索引标识或索引文件中搜索一个索引关键字。

语法

KEYMATCH(eIndexKey [, nIndexNumber [, nWorkArea | cTableAlias]])

返值类型

逻辑值

参数描述

eIndexKey

指定要搜索的索引关键字。索引文件或索引标识中的索引关键字是由索引表达式决定的。用 INDEX 命令创建索引文件或索引标识时可以指定其索引表达式。KEY（）函数和 SYS(14) 命令都可返回索引文件或索引标识的索引表达式。有关创建索引文件、索引表达式以及索引关键字的详细内容，请参阅 INDEX。

若省略所有可选参数，KEYMATCH（）函数将在主控索引文件或主控索引标识中搜索指定的索引关键字。如果不存在有效的主控索引文件或主控索引标识（例如，已用不包含任何参数的 SET ORDER TO 命令把表设置成了物理记录顺序），Visual FoxPro 将产生错误信息。

nIndexNumber

指定在哪一个索引文件或索引标识中搜索。通常情况下，若要搜索其他的索引标识，*nIndexNumber* 一般是一个从 1 开始以 1 递增的整数。

如 *nIndexNumber* 为 1，则搜索主控单项索引 (.IDX) 文件或主控索引标识（如果有）。

随着 *nIndexNumber* 的增大，依次搜索结构复合索引文件（如果有）中各个后续标识。各个标识的搜索次序取决于它们在结构复合索引中的创建顺序。

随着 *nIndexNumber* 的继续增大，在搜索结构复合索引文件中的所有标识后，将搜索打开的独立复合索引文件中的标识。各标识的搜索次序取决于它们在独立复合索引中的创建顺序。

若 *nIndexNumber* 大于打开的索引文件总数（包括 .IDX 单项索引文件、结构复合索引标识和独立的复合索引标识），Visual FoxPro 将产生错误信息。

nWorkArea | cTableAlias

用于搜索非当前工作区中的索引文件或标识。*nWorkArea* 指定工作区号，*cTableAlias* 指定表别名。若省略这两个参数，KEYMATCH（）函数将搜索由当前工作区中的表打开的索引文件或标识。

如果指定的表别名不存在，Visual FoxPro 将产生错误信息。

说明

KEYMATCH（）函数在索引标识或索引文件中搜索指定的索引关键字，若找到，则返回“真”(.T.)；否则，返回“假”(.F.)。该函数可以用来防止重复出现的索引关键字。

KEYMATCH（）函数运行完之后，仍把记录指针放置到运行该函数之前记录指针所指的记录上。

请参阅

INDEX, INDEXSEEK(), KEY() , SET INDEX, SYS(14), USE

KeyPress 事件

当用户按下并释放某个键时发生此事件。

语法

PROCEDURE Object.KeyPress
LPARAMETERS [nIndex,] nKeyCode, nShiftAltCtrl

–或者–

LPARAMETERS nKeyCode, nShiftAltCtrl

参数描述

在该事件的处理程序中，必须包含 LPARAMETERS 或 PARAMETERS 语句，并为每个参数指定名称。

nIndex

唯一地标识控件数组中的一个控件。

nKeyCode

包含一个数值，用该数值标识被按下的键。有关特殊键和组合键的编码，请参阅 INKEY( )。

nShiftAltCtrl

如果按下在 nKeyCode 中标识的键时，也按下修改键，则设置一个位。

有效的修改键是 SHIFT、CTRL 和 ALT 键。

下表列出了为单独的修改键在 nShiftAltCtrl 中返回的值。

对 nShiftAltCtrl 的修改键值

键	值
SHIFT	1
CTRL	2
ALT	4

参数是位的总数，重要性最小的位对应于 SHIFT 键 (bit 0)、CTRL 键 (bit 1) 与 ALT 键

(bit 2)。

位分别对应于值 1、2 和 4。该参数指出了键的状态。可以设置一些位、所有位或不能设置位，表示按下了一些键、所有键或没有按键。例如，如果同时按下 CTRL 与 ALT 键，nShiftAltCtrl 的值为 6。

说明

具有焦点的对象接收该事件。

在两种情况下，表单可接收 KeyPress 事件：

- 表单中不包含控件，或表单的控件都不可见或未激活。
- 表单的 KeyPreview 属性设置为“真”(.T.)。表单首先接收 KeyPress 事件，然后具有焦点的控件才接收此事件。
- 表单的 KeyPreview 属性设置为“真”(.T.)。表单首先接收 KeyPress 事件，然后具有焦点的控件才接收此事件。

KeyPress 事件常用于截取输入到控件中的键击。它使您可以立即检验键击的有效性或对键入的字符进行格式编排。使用 KeyPreview 属性可以创建全局键盘处理程序。

对任何与 ALT 键的组合键，不发生 KeyPress 事件。

应用于

复选框，组合框，命令按钮，编辑框，表单，列表框，选项按钮，微调，文本框

请参阅

I ，KeyPreview 属性

KeyPreview 属性

指定表单的 KeyPress 事件是否优先于控件的 KeyPress 事件。设计和运行时可用。

语法

Object.KeyPreview[= lExpr]

参数描述

lExpr

下表列出了 KeyPreview 属性的设置：

设置	说明
“真” (.T.)	表单首先接收 KeyPress 事件，然后活动控件才接收。
“假” (.F.)	（默认值）由活动控件接收 KeyPress 事件，表单不接收。

说明

使用 KeyPreview 属性可使表单先对 KeyPress 事件进行处理，然后再由当前活动控件对这些事件进行处理。

可以使用这个属性为表单创建一个全局键盘处理程序。例如，在应用程序中用到功能键时，可以在表单层次上对这些键击做出处理，而不用为每一个接收到键击事件的控

件分别编写代码。

若表单没有可见的活动控件，它将自动接收所有键盘事件。

应用于

表单，页面，_SCREEN，工具栏

请参阅

Enabled 属性, KeyPress 事件, Visible 属性

LABEL 命令

根据表文件和标签定义文件打印标签。

语法

```
LABEL [FORM FileName1 | FORM ?],
  [ENVIRONMENT]
  [Scope]
  [FOR lExpression1]
  [WHILE lExpression2]
  [NOCONSOLE]
  [NOOPTIMIZE]
  [PDSETUP]
```

[PREVIEW [NOWAIT]]
[NAME ObjectName]
[TO PRINTER [PROMPT] | TO FILE FileName2]

参数描述

FORM FileName1
指定要打印的标签所对应的标签定义文件名。标签定义文件的默认扩展名为 .LBX**。若标签定义文件不在当前驱动器或当前目录中，则必须指定驱动器和目录。**

FORM ?
显示“打开”对话框，从中可以选择一个已有的标签文件。

ENVIRONMENT
包含此参数是为了提供与 FoxPro 2.x **标签的向后兼容性。若要恢复与** Visual FoxPro **标签相联系的数据环境，可把该数据环境的** AutoOpenTables **属性置为默认值“真”**(.T.)**。为了确保在标签打印完毕后关闭数据环境，可把该数据环境的** AutoClose **属性设置为默认值“真”** (.T.)**。**

当创建或修改标签时，可把当前的 Visual FoxPro **数据环境同标签定义文件一起保存起来。保存** Visual FoxPro **数据环境，可把所有打开的表和索引文件、索引顺序以及表之间的关系等其他信息作为附加记录放置在标签定义表中。**

Scope
指定记录的范围。系统只打印该范围以内的记录。 Scope **子句有：** ALL**，**

NEXT *nRecords*，RECORD *nRecordNumber* 和 REST 等。包含 *Scope* 参数的命令只对活动工作区中的表进行操作。

LABEL 命令默认的范围是所有 (ALL) 记录。

FOR lExpression1

指定一个条件，只打印满足逻辑条件 *lExpression1* 的记录。这样可以把不想打印的记录筛选掉。

若 *lExpression1* 是一个可优化的表达式，则 Rushmore 将对 LABEL…FOR 命令所创建的查询进行优化。为了达到最好的运行效率，应在 FOR 子句中使用可优化的表达式。

详细内容，请参阅稍后的 SET OPTIMIZE 命令与《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”中的“掌握 Rushmore 技术”。

WHILE lExpression2

指定一个条件，只要逻辑表达式 *lExpression2* 计算为“真” (.T.)，就连续打印记录。

NOCONSOLE

在打印标签或向文件中传送标签时，不在 Visual FoxPro 主窗口或用户自定义窗口显示标签的输出结果。

NOOPTIMIZE

关闭 Rushmore 对 LABEL 命令进行优化。

详细内容，请参阅稍后的 SET OPTIMIZE 命令与《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”中的“掌握 Rushmore 技

术”。

PDSETUP
加载打印机驱动程序设置。

在 Visual FoxPro 中，包含 PDSETUP 关键字，可以使用打印机驱动设置来打印在 FoxPro for MS-DOS 中创建的基于字符的标签。打印 Visual FoxPro 中建立的基于图形的标签时，忽略 PDSETUP。

PREVIEW [NOWAIT]
打印之前先在“预览”窗口中显示标签，而不打印。若要打印标签，必须再发出一条不含 PREVIEW 子句的 LABEL 命令。

在 Visual FoxPro 中，若包含可选的 NOWAIT 子句，则在运行时 Visual FoxPro 将不等待关闭“页面预览”窗口，而是在该窗口打开的情况下继续往下执行。

NAME ObjectName
为标签的数据环境指定一个对象变量名。数据环境和其中的对象具有属性和方法（例如 AddObject），运行时需要设置或调用它们。利用对象变量可对这些属性和方法进行访问。若不指定对象变量名，Visual FoxPro 将把标签文件的文件名作为对象变量名的默认值，供事件引用。

TO PRINTER [PROMPT]
向打印机发送标签。在 Visual FoxPro 中，若包含可选项 PROMPT，则在开始打印以前提示“打印”对话框。

TO FILE FileName2
把 FoxPro for MS-DOS 中创建的基于字符的标签发送到文本文件 *FileName2*

中。包含 TO FILE 子句将创建该文件，默认扩展名为 .TXT。

说明

用 MODIFY LABEL 或 CREATE LABEL 命令可以创建标签定义文件。

若不带任何参数发出 LABEL 命令，“打开”对话框出现，显示已有的标签文件以供选择。

请参阅

CREATE LABEL, MODIFY LABEL

Label 控件

创建用以显示文本的标签。

语法

Label

说明

标签控件是用以显示文本的图形控件，其中的文本不能直接更改。但是，由于标签控

件具有与其他控件相同的一套属性、事件和方法，所以运行时它也可以对事件做出反应，或者动态地被更换。
若要给标签指定一个访问键，可在标题中作为访问键的字符前面加上反斜杠和小于号(\<)。显示这一标签时，该字符带有下划线。按下标签的访问键时，将激活 Tab 键次序所指定的下一个控件。使用 TabIndex 属性可给标签指定一个 Tab 键次序。
不同的对象，所显示的标题也不同。
标签控件的 Caption 属性允许包含的最大字符数目为 256 个。
有关创建标签控件的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第十章“使用控件”。

属性

Alignment	Application	AutoSize
BackColor	BackStyle	BaseClass
BorderStyle	Caption	Class
ClassLibrary	ColorScheme	ColorSource
Comment	DisabledBackColor	DisabledForeColor
DragIcon	DragMode	Enabled
FontBold	FontCondense	FontExtend
FontItalic	FontName	FontOutline
FontShadow	FontSize	FontStrikeThru
FontUnderline	ForeColor	Height
HelpContextID	Left	MouseIcon
MousePointer	Name	OLEDragMode

续 表

OLEDragPicture	OLEDropEffects	OLEDropHasData
OLEDropMode	Parent	ParentClass
RightToLeft	TabIndex	Tag
ToolTipText	Top	Visible
WhatsThisHelpID	Width	WordWrap

事 件

Click	DblClick	Destroy
DragDrop	DragOver	Error
Init	MiddleClick	MouseDown
MouseMove	MouseUp	MouseWheel
OLECompleteDrag	OLEDragDrop	OLEDragOver
OLEGiveFeedBack	OLESetData	OLEStartDrag
RightClick	UIEnable	

方 法

AddProperty	CloneObject	Drag
Move	OLEDrag	ReadExpression
ReadMethod	ResetToDefault	SaveAsClass
UIEnable	WriteExpression	WriteMethod
ZOrder		

请 参 阅

LASTKEY（）函数

返回最近一次按键所对应的整数。

语法

LASTKEY( )

返值类型

数值型

说明

LASTKEY（）函数的返回值与 INKEY（）函数的返回值是相等的。当在各控制间移动时，LASTKEY（）函数的值被更新。.

请参阅

CHRSAW( ), INKEY( ), READKEY( )

LastModified 属性

包含项目中一个文件最后更改的日期和时间。设计和运行时只读。

语法

Object.LastModified

说明

LastModified 属性包含一个日期时间值，表明文件最后更改的时间。

应用于

文件对象

请参阅

FDATE( ), FTIME( )

Left 属性

对于表单对象，确定表单的左边界与 Visual FoxPro 主窗口左边界之间的距离。设计和运行时可用。

主窗口中包含了一个表单，零位置立即到主窗口的左边界的右侧。如果工具栏停放在主窗口的左侧，零位置立即到工具栏的右侧。在 Macintosh 的 Visual FoxPro 中，如果表单在桌面上（不包含 Visual FoxPro 主窗口中），零位置在屏幕的左边。

Left，Top，Height 和 Width 属性多用于与对象的外部尺寸有关的操作（如移动或调整对象大小）。可以用 ScaleMode 属性更改度量单位。

注意 用于列对象所包含的控件时，Left 属性只读。

示例

下面的示例演示了如何使用 Left 属性确定控件在表单中的位置。使用 AddObject 方法向表单中添加一个线条控件和三个命令按钮，Left 属性指定了表单上每个控件的水平位置。

```
frmMyForm = CREATEOBJECT('Form') && 创建一个表单
frmMyForm.Closable = .F. && 禁止控件菜单框

frmMyForm.AddObject('shpLine','Line') && 向表单中添加线条控件
frmMyForm.AddObject('cmdCmndBtn1','cmdMyCmndBtn1') && “向上”命令按钮
```

```
frmMyForm.AddObject('cmdCmndBtn2','cmdMyCmndBtn2') && “向下”命令按钮
frmMyForm.AddObject('cmdCmndBtn3','cmdMyCmndBtn3') && “退出”命令按钮

frmMyForm.shpLine.Visible = .T. &&使线条控件可见
frmMyForm.shpLine.Top = 20 &&指定线条控件行
frmMyForm.shpLine.Left = 125 &&指定线条控件列

frmMyForm.cmdCmndBtn1.Visible =.T. && “向上”命令按钮可见
frmMyForm.cmdCmndBtn2.Visible =.T. && “向下”命令按钮可见
frmMyForm.cmdCmndBtn3.Visible =.T. && “退出”命令按钮可见

frmMyForm.SHOW &&显示表单
READ EVENTS &&开始事件处理

DEFINE CLASS cmdMyCmndBtn1 AS COMMANDBUTTON &&创建命令按钮
    Caption = 'Slant \<Up' &&命令按钮标题
    Left = 50 &&命令按钮行
    Top = 100 &&命令按钮列
    Height = 25 &&命令按钮高度

    PROCEDURE Click
        ThisForm.shpLine.Visible = .F. &&隐藏线条控件
        ThisForm.shpLine.LineSlant ='/' &&斜向上
        ThisForm.shpLine.Visible = .T. &&显示线条控件
ENDDEFINE

DEFINE CLASS cmdMyCmndBtn2 AS CommandButton &&创建命令按钮
    Caption = 'Slant \<Down' &&命令按钮标题
    Left = 200 &&命令按钮行
    Top = 100 &&命令按钮列
    Height = 25 &&命令按钮高度
```

```
    PROCEDURE Click
        ThisForm.shpLine.Visible = .F. && 隐藏线条控件
        ThisForm.shpLine.LineSlant ='\' && 斜向下
        ThisForm.shpLine.Visible = .T. && 显示线条控件
ENDDEFINE

DEFINE CLASS cmdMyCmndBtn3 AS CommandButton && 创建命令按钮
    Caption = '\<Quit' &&  命令按钮标题
    Cancel = .T. &&  默认取消命令按钮(Esc)
    Left = 125 && 命令按钮行
    Top = 150 && 命令按钮列
    Height = 25 && 命令按钮高度

    PROCEDURE Click
        CLEAR EVENTS && 结束事件处理，关闭表单
```

应用于

复选框，组合框，命令按钮，命令组，容器对象，控件对象，自定义，编辑框，表单，表格，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，选项组，页框，_SCREEN，形状，微调，文本框，计时器，工具栏

请参阅

Height 属性, Move 方法, ScaleMode 属性, Top 属性, Width 属性

LEFT（）函数

从字符表达式最左边字符开始，返回指定数目的字符。

语法

LEFT(cExpression, nExpression)

返值类型

字符型

参数描述

cExpression

指定的字符表达式，LEFT（）函数从中返回字符。

nExpression

指定从字符表达式中返回的字符个数。若 *nExpression* 的值大于 *cExpression* 的长度，则返回字符表达式的全部字符。否则，返回空字符串。

LEFT（）函数与起始位置为 1 的 SUBSTR（）函数是等价的。

示例

```
CLEAR
? LEFT('Redmond, WA', 4) && 显示数值 Redm
```

请参阅

A T ( ), LTRIM( ), RIGHT( ), RTRIM( ), SUBSTR( )

LEFTC（）函数

返回字符表达式中指定数目的字符，从最左边的字符开始计数。

语法

LEFTC(cExpression, nExpression)

返值类型

字符型

参数描述

cExpression

指定要从中返回字符的字符表达式。

nExpression

指定从字符表达式中返回的字符数目。如果 *nExpression* 大于 *cExpression* 长度，则返回 *nExpression* 的所有字符。否则返回空字符串。

说明

LEFTC( ) 函数是为包含双字节字符的表达式设计的。如果该表达式只包括单字节字符，LEFTC( ) 函数等同于 LEFT( ) 。

LEFTC（）从包含单字节和双字节的任意组合的字符表达式中返回指定数目的字符。

LEFTC（）与起始位置为 1 的 SUBSTRC（）等价。
该函数适用于管理 Hiragana 或 Katakana 等语言的双字节字符集。
请参阅

AT_C( ), LEFT( ), RIGHTC( ), SUBSTRC( )

LeftColumn 属性

指定表格控件显示的最左列的编号。设计和运行时只读。
语法

Grid.LeftColumn
说明

用 LeftColumn 可以指定哪些列对用户是不可见的。例如，若 LeftColumn = 3，则第 1 列和第 2 列就是不可见的。
应用于

表格
请参阅

ActiveColumn 属性, ActiveRow 属性, ColumnOrder 属性, RelativeColumn 属性, RelativeRow 属性

LEN（）函数

返回字符表达式中字符的数目。

语法

LEN(cExpression)

返值类型

数值型

参数描述

cExpression

指定的字符表达式，LEN（）函数返回其字符数目。

说明

LEN（）函数可用于确定字符表达式的长度。

示例

下面的示例打开了数据库 testdata 中的 customer 表。LEN（）用于显示 cust_id 与 contact 字段的宽度。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE Customer    && 打开 customer 表

CLEAR
```

```
? 'Width of contact field: '
?? LEN(contact)
? 'Width of cust_id field: '
?? LEN(cust_id)
```

请参阅

FSIZE( ), TXTWIDTH( )

LENC（）函数

返回字符表达式或备注字段中字符的数目。

语法

LENC(cExpression)

返值类型

数值型

参数描述

cExpression

指定由 LENC（）**返回其字符数的字符表达式。**

说明

LENC（）**用于包含双字节字符的表达式。如果表达式只包含单字节字符，** LENC（）

等同于 LEN( ) 。

LENC( ) 返回包含单字节和双字节任意组合的字符表达式或备注字段的字符数。该函数适用于管理双字节字符集。

请参阅

LEN( )

LIKE( ) 函数

确定字符表达式是否与另一个字符表达式相匹配。

语法

LIKE(cExpression1, cExpression2)

返值类型

逻辑值

参数描述

cExpression1

指定要与 *cExpression2* 相比较的字符表达式。*cExpression1* 中可以包含通配符 * 和 ?。问号 (?) 可与 *cExpression2* 中的任何单个字符相匹配，星号 (*) 可与任意数目的字符相匹配。在 *cExpression1* 中可以把任何数目的通配符进行任意组合。

cExpression2

指定要与 *cExprssion1* 相比较的字符表达式。只有在 *cExpression1* 与 *cExpression2* 中的字逐个匹配的情况下，LIKE() 函数才返回“真”(.T.)。

说明

如果 *cExpression1* 与 *cExpression2* 相匹配，则 LIKE() 函数返回“真”(.T.)；否则，返回“假”(.F.)。

SET COMPATIBLE 决定 LIKE() 函数如何比较 *cExpression1* 和 *cExpression2* 中的空格。若 SET COMPATIBLE 设置为 ON 或 DB4，则在比较之前删除 *cExpression1* 和 *cExpression2* 中的后缀空格；若 SET COMPATIBLE 设置为 OFF 或 FOXPLUS，则两字符表达式中的后缀空格也参加比较。

示例

在以下示例中，显示 products 表中前两个字母为“Ch”的所有产品名。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE products  && 打开 Products 表

CLEAR
? 'All product names with first two letters Ch:'
?
SCAN FOR LIKE('Ch*', prod_name)
    ? prod_name
ENDSCAN
USE
```

请参阅

LIKEC( ) 函数

确定字符表达式是否与另一个字符表达式相匹配。

语法

LIKEC(cExpression1, cExpression2)

返值类型

逻辑值

参数描述

cExpression1

指定要与 *cExpression2* **相比较的字符表达式。** *cExpression1* **中可以包含通配符 * 和 ?。问号 (?) 可与** *cExpression2* **中的任何单个字符相匹配，星号 (*) 可与任意数目的字符相匹配。**

cExpression2

指定要与 *cExprssion1* **相比较的字符表达式。只有在** *cExpression1* **与** *cExpression2* **中的字符逐个匹配的情况下，** LIKEC( ) **函数才返回“真”** (.T.)。

说明

LIKEC（）用于包含双字节字符的表达式。如果表达式只包含单字节字符，L等同于 LIKE()。

LIKEC（）决定一个字符表达式是否与另一个字符表达式相匹配。字符表达式可以包含单字节和双字节的任意组合。如果 *cExpression1* 与 *cExpression2* 相匹配，则 LIKEC（）函数返回“真”(.T.)；否则，返回“假”(.F.)。

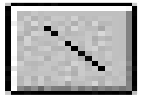
SET COMPATIBLE 决定 LIKEC（）函数如何比较 *cExpression1* 和 *cExpression2* 中的空格。若 SET COMPATIBLE 设置为 ON 或 DB4，则在比较之前删除 *cExpression1* 和 *cExpression2* 中的所有后缀空格。若 SET COMPATIBLE 设置为 OFF 或 FOXPLUS，则两字符表达式中的后缀空格也参加比较。

该函数适用于管理双字节字符集。

请参阅

A T _ C ( ), ATCC( ), LIKEC( ), RATC( )

Line 控件



创建一个显示水平、竖直或对角线条的控件。

语法

Line

说明

线条控件是一种用来显示水平、竖直或对角线条的图形控件，所显示的线条不能直接更改。但是，由于线条控件与其他控件一样具有一整套属性、事件和方法，所以运行时它可以对事件做出反应或者动态地被更改。

有关创建线条的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第十章“使用控件”。

属性

Application
BaseClass
BorderColor
BorderStyle
BorderWidth
Class
ClassLibrary
ColorSource
Comment
DragIcon
DragMode
DrawMode
Enabled
Height
HelpContextID
Left
LineSlant
MousePointer
Name
OLEDragMode
OLEDragPicture
OLEDropEffects
OLEDropHasData
OLEDropMode
Parent
ParentClass
Tag
Top
Visible
WhatsThisHelpID
Width

事件

Click	DblClick	Destroy
DragDrop	DragOver	Error
Init	MiddleClick	MouseDown
MouseMove	MouseUp	MouseWheel
OLECompleteDrag	OLEDragDrop	OLEDragOver
OLEGiveFeedBack	OLESetData	OLEStartDrag
RightClick	UIEnable	

方法

AddProperty	CloneObject	Drag
Move	OLEDrag	ReadExpression
ReadMethod	ResetToDefault	SaveAsClass
WriteExpression	WriteMethod	ZOrder

示例

下面的示例演示了如何添加线条控件到表单中，以及如何用 LineSlant 属性来指定线条倾斜的方向。

可用 AddObject 方法添加线条控件到表单中。添加了三个命令按钮，允许您改变线条倾斜的方向或关闭表单。

```
frmMyForm = CREATEOBJECT('Form') && 创建一个表单
frmMyForm.Closable = .F. && 取消控件目录箱的作用

frmMyForm.AddObject('shpLine','Line') && 向表单增加线条控件
```

```
frmMyForm.AddObject('cmdCmndBtn1','cmdMyCmndBtn1') && 上移 Cmnd 按扭
frmMyForm.AddObject('cmdCmndBtn2','cmdMyCmndBtn2') && 下移 Cmnd 按扭
frmMyForm.AddObject('cmdCmndBtn3','cmdMyCmndBtn3') && 退出 Cmnd 按扭

frmMyForm.shpLine.Visible = .T. &&让线条控件可见
frmMyForm.shpLine.Top = 20 &&指定线条控件所在行
frmMyForm.shpLine.Left = 125 &&指定线条控件所在列

frmMyForm.cmdCmndBtn1.Visible =.T. &&让向上命令按扭可见
frmMyForm.cmdCmndBtn2.Visible =.T. &&让向下命令按扭可见
frmMyForm.cmdCmndBtn3.Visible =.T. &&让退出命令按扭可见

frmMyForm.SHOW &&显示表单
READ EVENT &&启动事件程序

DEFINE CLASS cmdMyCmndBtn1 AS COMMANDBUTTON &&创建命令按扭
    Caption = 'Slant \<Up' &&给命令按扭增加说明
    Left = 50 &&命令按扭列
    Top = 100 &&命令按扭行
    Height = 25 &&命令按扭高度

    PROCEDURE Click
        ThisForm.shpLine.Visible = .F. &&隐藏线条控件
        ThisForm.shpLine.LineSlant ='/' &&启动
        ThisForm.shpLine.Visible = .T. &&显现线条控件
ENDDEFINE

DEFINE CLASS cmdMyCmndBtn2 AS CommandButton &&创建命令按扭
    Caption = 'Slant \<Down' &&给命令按扭增加说明
    Left = 200 &&命令按扭列
    Top = 100 &&命令按扭行
    Height = 25 &&命令按扭高
```

```
    PROCEDURE Click
        ThisForm.shpLine.Visible = .F.  && 隐藏线条控件
        ThisForm.shpLine.LineSlant ='\'  && 向下倾斜
        ThisForm.shpLine.Visible = .T.  && 显示线条控件
ENDDEFINE
DEFINE CLASS cmdMyCmndBtn3 AS CommandButton  && 创建命令按扭
    Caption = '\<Quit'  && 给命令按扭增加说明
    Cancel = .T.  && 默认取消命令按扭(Esc 键)
    Left = 125  && 命令按扭列
    Top = 150  && 命令按扭行
    Height = 25  && 命令按扭高

    PROCEDURE Click
        CLEAR EVENTS  &&终止事件程序，退出表单
ENDDEFINE
```

请参阅

CREATE CLASS, CREATE FORM, DEFINE CLASS

Line 方法

在表单对象中画一条线。

语法

Object.Line(nXCoord2, nYCoord2)

–或者–

Object.Line(nXCoord1, nYCoord1, nXCoord2, nYCoord2)

参数描述

nXCoord1, nYCoord1

指定线条起始点的坐标。度量单位由表单的 ScaleMode 属性指定。

nXCoord2, nYCoord2

指定线条终点的坐标。

说明

所画线条的宽度取决于 DrawWidth 属性的设置。在背景上画线条的方式取决于 DrawMode 和 DrawStyle 属性的设置。运行 Line 方法后，CurrentX 和 CurrentY 属性分别被设置为 *nXCoord2* 和 *nYCoord2*。

应用于

表单，_SCREEN

请参阅

CurrentX, CurrentY 属性，DrawMode 属性，DrawStyle 属性，DrawWidth 属性，ScaleMode 属性

返回总目录

MaxButton 属性
MaxHeight 属性
MaxLeft 属性
MaxLength 属性
MaxTop 属性
MaxWidth 属性
MCOL() 函数
MD 或 MKDIR 命令
MDIForm 属性
MDOWN() 函数
MDX() 函数
MDY() 函数
MEMLINES() 函数
MEMORY() 函数
MemoWindow 属性
MENU 命令
MENU() 函数
MENU TO 命令
Message 事件
MESSAGE() 函数
MESSAGEBOX() 函数
MiddleClick 事件

LINENO() 函数

返回程序中正在执行的那一行的行号。行号是相对于主程序第一行。

语法

LINENO([1])

返值类型

数值型

参数描述

1

返回相对于当前程序或过程第一行的行号。若省略参数 1，则返回相对于主程序第一行的行号。

说明

程序行从程序头开始计数，包括注释行、继续行和空行。若程序在执行中被挂起，则 LINENO() 返回程序挂起处的程序行号。若取消程序，则 LINENO() 返回 0。

默认情况下，返回相对于主程序开始的行号。如果调用一个过程，行号从调用程序头部开始计数。

LINENO() 对于调试程序很有用。要设置断点并在指定行号处终止程序执行，可在调试窗口中执行下列命令：

LINENO( ) = nExpression

当 LINENO( ) 的值等于 *nExpression* 时挂起程序。

示例

下面的示例是一个简单的错误处理例程的一部分。

```
ON ERROR DO bug_proc WITH LINENO( )
BRWS  && 出现错误
ON ERROR

*** 错误处理 ***

PROCEDURE bug_proc
PARAMETERS gnBadLine
WAIT WINDOW 'Error occurred at line: ' + ALLTRIM(STR(gnBadLine))
RETURN
```

请参阅

ERROR( ), MESSAGE( ), PROGRAM( ), SYS(16)

LineSlant 属性

指定线条倾斜方向，是从左上到右下还是从左下到右上。设计和运行时可用。

语法

Line.LineSlant[= cSlant]

参数描述

cSlant

LineSlant 属性的设置是：

设置	说明
\	（默认值）线条从左上到右下倾斜。
/	线条从左下到右上倾斜。

示例

下面示例演示了如何使用 LineSlant 属性指定线条控件的倾向。

先向表单添加线条控件，再向表单添加三个命令按钮来更改线条倾向或关闭表单。

```
frmMyForm = CREATEOBJECT('Form') && 创建一个表单
frmMyForm.Closable = .F. && 使控件菜单框无效

frmMyForm.AddObject('shpLine','Line') && 向表单添加线条控件
frmMyForm.AddObject('cmdCmndBtn1','cmdMyCmndBtn1') && “向上”命令按钮
frmMyForm.AddObject('cmdCmndBtn2','cmdMyCmndBtn2') && “向下”命令按钮
frmMyForm.AddObject('cmdCmndBtn3','cmdMyCmndBtn3') && “退出”命令按钮

frmMyForm.shpLine.Visible = .T. && 使线条控件可见
frmMyForm.shpLine.Top = 20 && 指定线条控件行
frmMyForm.shpLine.Left = 125 && 指定线条控件列

frmMyForm.cmdCmndBtn1.Visible =.T. && “向上”命令按钮可见
frmMyForm.cmdCmndBtn2.Visible =.T. && “向下”命令按钮可见
```

```
frmMyForm.cmdCmndBtn3.Visible =.T. && “退出”命令按钮可见

frmMyForm.SHOW &&显示表单
READ EVENTS &&开始事件处理

DEFINE CLASS cmdMyCmndBtn1 AS COMMANDBUTTON &&创建命令按钮
    Caption = 'Slant \<Up' && 命令按钮标题
    Left = 50 &&命令按钮列
    Top = 100 &&命令按钮行
    Height = 25 &&命令按钮高度

    PROCEDURE Click
        ThisForm.shpLine.Visible = .F. &&隐藏线条控件
        ThisForm.shpLine.LineSlant ='/' &&上斜
        ThisForm.shpLine.Visible = .T. &&显示线条控件
ENDDEFINE

DEFINE CLASS cmdMyCmndBtn2 AS CommandButton &&显示线条控件
    Caption = 'Slant \<Down' &&命令按钮标题
    Left = 200 &&命令按钮列
    Top = 100 &&命令按钮行
    Height = 25 &&命令按钮高度

    PROCEDURE Click
        ThisForm.shpLine.Visible = .F. &&隐藏线条控件
        ThisForm.shpLine.LineSlant ='\' &&下斜
        ThisForm.shpLine.Visible = .T. &&显示线条控件
ENDDEFINE

DEFINE CLASS cmdMyCmndBtn3 AS CommandButton &&创建命令按钮
```

```
    Caption = '\<Quit' && 命令按钮标题
    Cancel = .T. && 默认“取消”命令按钮(ESC)
    Left = 125 && 命令按钮列
    Top = 150 && 命令按钮行
    Height = 25 && 命令按钮高度

    PROCEDURE Click
       CLEAR EVETNTS && 终止事件处理，关闭表单
ENDDEFINE
```

应用于

线条

请参阅

DrawMode 属性, DrawStyle 属性

LinkMaster 属性

指定表格控件中的子表所链接的父表。设计时可用，运行时可读写。

语法

Grid.LinkMaster[= cName]

参数描述

cName

指定用来在表格控件中驱动子表显示的父表别名。

说明

使用 LinkMaster **属性可以在表单的父表（或主表）与表格** RecordSource **属性所引用的表之间设置一对多关系。**

应用于

表格

请参阅

ChildOrder **属性**, RelationalExpr **属性**

LIST **命令**

连续显示表的信息。

语法

LIST
[FIELDS FieldList]
[Scope] [FOR lExpression1] [WHILE lExpression2]

```
  [OFF]
  [NOCONSOLE]
  [NOOPTIMIZE]
  [TO PRINTER [PROMPT] | TO FILE FileName]
```
– 或 –
```
LIST FILES
  [ON Drive]
  [LIKE FileSkeleton]
  [TO PRINTER [PROMPT] | TO FILE FileName]
```
– 或 –
```
LIST MEMORY
  [LIKE FileSkeleton]
  [NOCONSOLE]
  [TO PRINTER [PROMPT] | TO FILE FileName]
```
– 或 –
```
LIST STATUS
  [NOCONSOLE]
  [TO PRINTER [PROMPT] | TO FILE FileName]
```
– 或 –
```
LIST STRUCTURE
  [IN nWorkArea | cTableAlias]
  [NOCONSOLE]
```

[TO PRINTER [PROMPT] | TO FILE FileName]

说明

除了下列差别，LIST 命令与 DISPLAY 命令相同：

- LIST 的默认范围是所有记录。
- 信息充满 Visual FoxPro 主窗口或用户自定义窗口以后，LIST 不给提示，继续显示。
- 当 SET DELETED 为 ON 时，LIST 不显示带删除标记的记录。

有关 LIST 命令的详细内容，请参阅 DISPLAY 中的相应命令。

请参阅

DISPLAY, DISPLAY FILES, DISPLAY MEMORY, DISPLAY STATUS, DISPLAY STRUCTURE, SET DELETED

LIST CONNECTIONS 命令

连续显示有关当前数据库中命名连接的信息。

语法

LIST CONNECTIONS

[TO PRINTER [PROMPT] | TO FILE FileName]
[NOCONSOLE]

参数描述

TO PRINTER [PROMPT]

将 LIST CONNECTIONS 的输出结果送到打印机。

在 Visual FoxPro 中，包含 PROMPT 子句可在打印之前显示“打印”对话框。PROMPT 应直接放置在 TO PRINTER 后面。

TO FILE FileName

将 LIST CONNECTIONS 的输出结果送到 *FileName* 指定的文件中。若文件已存在并且 SET SAFETY 为 ON，则 Visual FoxPro 询问是否改写文件。

NOCONSOLE

不向 Visual FoxPro 主窗口或活动的用户自定义窗口输出。

说明

显示信息包括当前数据库中命名连接、数据源和连接字符串的名称。使用 DBGETPROP( ) 可以返回有关当前数据库中连接的其他内容。

示例

下列示例假设一个名为 MyFoxSQLNT 的 ODBC 数据源可用，并且数据源的用户标识是“sa”。示例打开 testdata 数据库，创建一个名为 Myconn 的连接。然后用 LIST CONNECTIONS 来列出数据库中的命名连接。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')

CREATE CONNECTION Myconn DATASOURCE "MyFoxSQLNT" USERID "sa"
```

CLEAR

LIST CONNECTIONS && 列出数据库中的命名连接。

请参阅

CREATE CONNECTION, DELETE CONNECTION, DBGETPROP(), DISPLAY CONNECTIONS, RENAME CONNECTION

LIST DATABASE 命令

连续显示有关当前数据库的信息。

语法

LIST DATABASE
[TO PRINTER [PROMPT] | TO FILE FileName]
[NOCONSOLE]

参数描述

TO PRINTER [PROMPT]

将 LIST DATABASE 的输出结果送到打印机。

在 Visual FoxPro 中，包含 PROMPT 子句可在打印启动之前显示打印对话框。PROMPT

应紧接着放置在 TO PRINTER 的后面。

TO FILE FileName

将 LIST DATABASE 的输出结果送到 *FileName* 指定的文件中。若该文件已存在并且 SET SAFETY 为 ON，则 Visual FoxPro 询问是否改写文件。

NOCONSOLE

不向 Visual FoxPro 主窗口或活动的用户自定义窗口输出。

说明

使用 DBGETPROP( ) 可以返回有关当前数据库的其他内容。

示例

下例创建了一个命名为 people 的数据库。表 friends 被创建并自动添加到数据库中。DISPLAY TABLES 命令用来显示数据库中的表，而 LIST DATABASES 命令列出数据库中表的相关信息。

```
CREATE DATABASE people
CREATE TABLE friends (FirstName C(20), LastName C(20))
CLEAR
DISPLAY TABLES  && 显示数据库中的表

LIST DATABASE  && 列出表的信息
```

请参阅

DISPLAY DATABASE

LIST DLLS 命令

连续显示有关信息。DLL 函数可用 DECLARE - DLL 命令在 Visual FoxPro 中注册。

语法

LIST DLLS
[TO PRINTER [PROMPT] | TO FILE FileName]
[NOCONSOLE]

参数描述

TO PRINTER [PROMPT]

将 LIST DLLS 的输出结果送到打印机。

在 Visual FoxPro 中，包含 PROMPT 子句可在打印之前显示“打印”对话框。PROMPT 应直接放置在 TO PRINTER 后面。

TO FILE FileName

将 LIST DLLS 的输出结果送到 *FileName* 指定的文件中。若该文件已存在并且 SET SAFETY 为 ON，则 Visual FoxPro 询问是否改写文件。

NOCONSOLE

不向 Visual FoxPro 主窗口或活动的用户自定义窗口输出。

请参阅

LIST OBJECTS 命令

连续显示有关一个对象或一组对象的信息。

语法

```
LIST OBJECTS
  [LIKE cObjectSkeleton]
  [TO PRINTER [PROMPT] | TO FILE FileName]
  [NOCONSOLE]
```

参数描述

LIKE cObjectSkeleton

显示一组对象的信息。*cObjectSkeleton* 是对象梗概，支持通配符（* 和 ?）。例如，要连续显示所有以 A 开头的对象，可使用下列命令：

```
LIST OBJECTS LIKE A*
```

TO PRINTER [PROMPT]

将 LIST OBJECTS 的输出结果送到打印机。

包含 PROMPT 子句可在打印之前显示“打印”对话框。PROMPT 关键字应直接放置在

TO PRINTER 后面。

TO FILE FileName

将 LIST OBJECTS 的输出结果送到 *FileName* 指定的磁盘文件中。若该文件已存在并且 SET SAFETY 为 ON，则 Visual FoxPro 询问是否改写文件。

NOCONSOLE

不向 Visual FoxPro 主窗口或活动的用户自定义窗口输出。

说明

LIST OBJECTS 显示全部现有对象的下列信息：

- 属性和属性值
- 方法
- 成员对象及其所基于的类或子类
- 对象所基于的类或子类
- 对象的类等级

LIST OBJECTS 连续不停顿地填满整个 Visual FoxPro 主窗口。

示例

下面的示例用 DEFINE CLASS、CREATEOBJECT() 从 Visual FoxPro 的 Form 基类中创建了两个自定义类，FormChild 与 FormGrandChild。LIST OBJECTS 列出了对象和它们的属性信息。

```
CLEAR
frmMyForm = CREATEOBJECT("FormGrandChild")
```

```
LIST OBJECTS LIKE frm*
RELEASE frmMyForm

DEFINE CLASS FormChild AS FORM
ENDDEFINE

DEFINE CLASS FormGrandChild AS FormChild
ENDDEFINE
```

请参阅

DISPLAY OBJECTS

LIST PROCEDURES 命令

连续显示当前数据库内部存储过程的名称。

语法

```
LIST PROCEDURES
  [TO PRINTER [PROMPT] | TO FILE FileName]
  [NOCONSOLE]
```

参数描述

TO PRINTER [PROMPT]

将 LIST PROCEDURES 的返回信息输出到打印机。

包含 PROMPT 可在打印之前显示“打印”对话框。PROMPT 关键字应直接放置在 TO PRINTER 后面。

TO FILE FileName

将 LIST PROCEDURES 的输出信息送到 *FileName* 指定的磁盘文件中。若该文件已存在并且 SET SAFETY 为 ON，则 Visual FoxPro 询问是否改写文件。

NOCONSOLE

不向 Visual FoxPro 主窗口或活动的用户自定义窗口输出。

说明

用 MODIFY PROCEDURES 可创建内部存储过程。

示例

下面示例打开 testdata 数据库，并且使用 LIST PROCEDURES 列出数据库中的内部存储过程（如果存在）。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')

CLEAR

LIST PROCEDURES && 列出数据库中的内部存储过程
```

请参阅

APPEND PROCEDURES, COPY PROCEDURES, CREATE DATABASE, DISPLAY DATABASE, DISPLAY PROCEDURES, MODIFY PROCEDURE

List 属性

用来访问组合框或列表框控件中各数据项的字符串数组。设计时不可用，运行时可读写。

语法

Control.List(nRow [, nCol])[= cChar]

参数描述

nRow

根据显示顺序指定需要检索的数据项所在的行。例如，*nRow* = 3 指定在列表中显示第三行。

nCol

根据显示顺序指定需要检索的数据项所在的列。例如，*nCol* = 2 指定在列表中显示第二列。若未指定 *nCol*，则 List 属性默认检索第一列。只能对多于一列的组合框和列表框指定 *nCol*。

说明

可将 List 属性与 ListCount 属性联合使用，从 1 到 ListCount 逐一返回列表中的所有

项。

不能同时使用 List 属性和数组函数。然而，如果 RowSourceType 属性设置为 5（数组）并且 RowSource 属性设置为列表中值所在的数组，则可对 RowSource 属性指定的数组使用数组函数。

注意 当 RowSourceType 设置为 0 或 1 时，可以用 AddItem 方法向组合框或列表框中添加数据项。要删除数据项，可使用 RemoveItem 方法。要按字母顺序放置数据项，可在向列表添加数据项之前将控件的 Sorted 属性设置为“真”(.T.)。

示例

在下面示例中，使用 ListCount 遍历所有数据项。这些数据项由组合框或列表框的 List 属性指定。

下面示例创建了一个列表框，列表框数据项中的数据项源是一个数组。该数组用 RowSourceType 和 RowSource 属性指定。

列表框的 MultiSelect 属性设置为“真”(.T.)，允许从列表框中做多项选择。在 FOR ... ENDFOR 循环中使用 ListCount 属性来显示列表框中选定的数据项。Selected 属性用来确定所选择的数据项。List 属性用来返回数据项。

```
CLEAR

DIMENSION gaMyListArray(10)
FOR gnCount = 1 to 10  && 用字母填充数组
    STORE REPLICATE(CHR(gnCount+64),6) TO gaMyListArray(gnCount)
ENDFOR
```

```
frmMyForm = CREATEOBJECT('Form') && 创建一个表单
frmMyForm.Closable = .F. && 使控件数据框无效

frmMyForm.Move(150,10) && 移动表单

* 添加“取消”命令按钮
frmMyForm.AddObject('cmbCommand1','cmdMyCmdBtn')
frmMyForm.AddObject('lstListBox1','lstMyListBox')

* && 指定一个数组
frmMyForm.lstListBox1.RowSourceType = 5
frmMyForm.lstListBox1.RowSource = 'gaMyListArray'

frmMyForm.cmbCommand1.Visible =.T. && “取消”命令按钮可见
frmMyForm.lstListBox1.Visible =.T. && 列表框可见

frmMyForm.SHOW && 显示表单
READ EVENTS && 开始事件处理

DEFINE CLASS cmdMyCmdBtn AS CommandButton && 创建命令按钮
    Caption = '\<Quit' &&命令按钮标题
    Cancel = .T. && 默认取消命令按钮(Esc)
    Left = 125 && 命令按钮列
    Top = 210 && 命令按钮行
    Height = 25 && 命令按钮高度

    PROCEDURE Click
        CLEAR EVENTS && 终止事件处理，关闭表单
        CLEAR && 清除 Visual FoxPro 主窗口
ENDDEFINE

DEFINE CLASS lstMyListBox AS ListBox && 创建列表框控件
```

```
    Left = 10  && 列表框列
    Top = 10  && 列表框行
    MultiSelect = .T.  && 允许选择多于 1 个数据项

PROCEDURE Click
    ACTIVATE SCREEN
    CLEAR
    ? "Selected items:"
    ? "--------------"
    FOR nCnt = 1 TO ThisForm.lstListBox1.ListCount
        IF ThisForm.lstListBox1.Selected(nCnt)  && 是否选择了数据项?
            ? SPACE(5) + ThisForm.lstListBox1.List(nCnt)  && 显示数据项
        ENDIF
    ENDFOR

ENDDEFINE
```

应用于

组合框，列表框

请参阅

AddItem 方法, IndexToItemID 方法, ItemIDToIndex 方法, ListCount 属性, ListItem 属性, ListItemID 属性, RowSource, RowSourceType,Sorted 属性

LIST TABLES 命令

连续显示包含在当前数据库中的所有表和表的信息。

语法

LIST TABLES
[TO PRINTER [PROMPT] | TO FILE FileName]
[NOCONSOLE]

参数描述

TO PRINTER [PROMPT]

将 LIST TABLES 的返回内容输出到打印机。

包含 PROMPT 可在打印之前显示“打印”对话框。PROMPT 关键字应直接放置在 TO PRINTER 之后。

TO FILE FileName

将 LIST TABLES 的输出结果送到 *FileName* 指定的磁盘文件中。若文件已存在并且 SET SAFETY 为 ON，则 Visual FoxPro 询问是否改写文件。.

NOCONSOLE

不向 Visual FoxPro 主窗口或活动的用户自定义窗口输出。

说明

此命令返回的信息中包括表名和路径，是 LIST STATUS 显示内容的子集。LIST

TABLES 显示的信息仅包含与表有关的信息，而不管表是否打开。

示例

下面示例打开 testdata 数据库中的 customer 表，用 LIST TABLES 命令列出数据库中关于表的信息。

```
CLOSE DATABASES
SET PATH TO (HOME(2) + 'data\') && 设置到数据库路径
OPEN DATABASE testdata && 打开 testdata 数据库
CLEAR

LIST TABLES && 列出表单信息
```

请参阅

ADD TABLE, CLOSE DATABASES, CREATE DATABASE, DISPLAY TABLES,
OPEN DATABASE, REMOVE TABLE

LIST VIEWS 命令

连续显示当前数据库中有关 SQL 视图的信息。

语法

LIST VIEWS
[TO PRINTER [PROMPT] | TO FILE FileName]
[NOCONSOLE]

参数描述

TO PRINTER [PROMPT]

将 LIST VIEWS 返回的内容输出到打印机。

包含 PROMPT 可在打印开始之前显示“打印”对话框。PROMPT 关键字应直接放置在 TO PRINTER 后面。

TO FILE FileName

将 LIST VIEWS 输出的结果送到 *FileName* 指定的磁盘文件中。若文件已存在并且 SET SAFETY 为 ON，则 Visual FoxPro 询问是否改写文件。

NOCONSOLE

不向 Visual FoxPro 主窗口或活动的用户自定义窗口输出。

说明

LIST VIEWS 显示当前数据库中 SQL 视图的名称并且说明 SQL 视图是基于本地表还是远程表。使用 DBGETPROP( ) 可以返回当前数据库中关于 SQL 视图的其他内容。可用 CREATE SQL VIEW 创建 SQL 视图。

示例

下面的示例打开数据库 testdata 。用 CREATE SQL VIEW 创建一个本地的 SQL 视图 myview。显示“视图设计器”以允许您指定 SQL 视图的表和条件。保存了所创建的

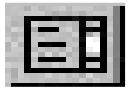
SQL 视图后，数据库中有关 SQL 视图的信息会被列出。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
CREATE SQL VIEW myview

CLEAR

LIST VIEWS
```

请参阅

CREATE DATABASE 命令, CREATE SQL VIEW 命令, DBGETPROP( ) , DISPLAY DATABASE 命令, DISPLAY VIEWS 命令

ListBox 控件



创建列表框。

语法

ListBox

说明

列表框显示一系列数据项，从中可选择一项或多项。列表框与组合框相似，不同的是，组合框初始时只显示一个数据项。

属性

Application
BaseClass
BorderColor
BoundColumn
BoundTo
Class
ClassLibrary
ColorScheme
ColorSource
ColumnCount
ColumnLines
ColumnWidths
Comment
ControlSource
DisabledBackColor
DisabledForeColor
DisabledItemBackColor
DisabledItemForeColo
DisplayValue
DragIcon
DragMode
Enabled
FirstElement
FontBold
FontCondense
FontExtend
FontItalic
FontName
FontOutline
FontShadow
FontSize
FontStrikeThru
FontUnderLine
Height
HelpContextID
IncrementalSearch
IntegralHeight
ItemBackColor
ItemData
ItemForeColor
ItemIDData
ItemTips
Left
List
ListCount
ListIndex
ListItem
ListItemID
MouseIcon
MousePointer
MoverBars

续 表

MultiSelect	Name	NewIndex
NewItemID	NullDisplay	NumberOfElements
OLEDragMode	OLEDragPicture	OLEDropEffects
OLEDropHasData	OLEDropMode	Parent
ParentClass	Picture	RightToLeft
RowSource	RowSourceType	Selected
SelectedID	SelectedItemBackColor	SelectedItemForeColor
Sorted	SpecialEffect	StatusBarText
TabIndex	TabStop	Tag
TerminateRead	ToolTipText	Top
TopIndex	TopItemID	Value
Visible	WhatsThisHelpID	Width

事 件

Click	DblClick	Destroy
DragDrop	DragOver	Error
ErrorMessage	GotFocus	Init
InteractiveChange	KeyPress	LostFocus
Message	MiddleClick Event	MouseDown
MouseMove	MouseUp	MouseWheel

续表

OLECompleteDrag	OLEDragDrop	OLEDragOver
OLEGiveFeedBack	OLESetData	OLEStartDrag
ProgrammaticChange	RangeHigh	RangeLow
RightClick	UIEnable	Valid
When		

方法

AddItem	AddListItem	AddProperty
Clear	CloneObject	Drag
IndexToItemID	ItemIDToIndex	Move
OLEDrag	ReadExpression	ReadMethod
Refresh	RemoveItem	RemoveListItem
Requery	ResetToDefault	SaveAsClass
SetFocus	WriteExpression	WriteMethod
ZOrder		

示例

下面的示例创建了一个列表框控件。列表框中的项的源是由 RowSourceType **和** RowSource **属性指定的数组。**

设置列表框的 MultiSelect **属性为真**(.T.)**，这样您就可以在列表框中有多个选择。可以用** ListCount、Selected **和** List **属性（决定在列表和所选的项中项的数目）显示项或所选的项。**

```
CLEAR

DIMENSION gaMyListArray(10)
FOR gnCount = 1 to 10  && 在数组中填充字母
    STORE REPLICATE(CHR(gnCount+64),6) TO gaMyListArray(gnCount)
NEXT

frmMyForm = CREATEOBJECT('Form')  && 创建表单
frmMyForm.Closable = .f.  && 取消控件目录箱作用

frmMyForm.Move(150,10)  && 移动表单

frmMyForm.AddObject('cmbCommand1','cmdMyCmdBtn')  && 增加“退出”命令按扭
frmMyForm.AddObject('lstListBox1','lstMyListBox')  && 增加 ListBox 控件

frmMyForm.lstListBox1.RowSourceType = 5  && 指定一个数组
frmMyForm.lstListBox1.RowSource = 'gaMyListArray' && 数组包括 listbox 项目

frmMyForm.cmbCommand1.Visible =.T.  &&使“退出”命令按扭可见
frmMyForm.lstListBox1.Visible =.T.  && 使 List Box 可见

frmMyForm.SHOW  && 显示表单
READ EVENTS  && 启动事件程序

DEFINE CLASS cmdMyCmdBtn AS CommandButton  && 创建命令按扭
    Caption = '\<Quit'  && 给命令按扭增加说明
    Cancel = .T.  &&默认取消命令按扭(Esc 键)
    Left = 125  && 命令按扭列
    Top = 210  && 命令按扭行
    Height = 25  && 命令按扭高
```

```
                                    主窗口
ENDDEFINE

DEFINE CLASS lstMyListBox AS ListBox  && 创建 ListBox 控件 l
    Left = 10  && List Box 列
    Top = 10  && List Box 行
    MultiSelect = .T.  && 允许多项选择

PROCEDURE Click
    ACTIVATE SCREEN
    CLEAR
    ? "Selected items:"
    ? "--------------"
    FOR nCnt = 1 TO ThisForm.lstListBox1.ListCount
        IF ThisForm.lstListBox1.Selected(nCnt)  &&该项目被选否?
            ? SPACE(5) + ThisForm.lstListBox1.List(nCnt) && 显示项目
        ENDIF
    ENDFOR

ENDDEFINE
```

请参阅

CREATE CLASS, CREATE FORM, DEFINE CLASS

ListCount 属性

存放组合框或列表框控件的列表中的项数。设计时不可用。运行时只读。

语法

Control.ListCount

示例

下面的示例创建了一个列表框。列表框中的项的源是由 RowSourceType 和 RowSource 属性指定的数组。

ListCount 用于对组合框或列表框的 List 属性指定的所有项进行循环遍历。

设置列表框的 MultiSelect 属性为真(.T.)，这样您就可以在列表框中有多个选择。在 FOR...ENDFOR 循环中用 ListCount 属性可显示项或在列表框中所选的项。可用 Selected 和 List 属性来决定选择的项。

```
CLEAR

DIMENSION gaMyListArray(10)
FOR gnCount = 1 to 10  && 用字母填充数组
    STORE REPLICATE(CHR(gnCount+64),6) TO gaMyListArray(gnCount)
ENDFOR

frmMyForm = CREATEOBJECT('Form')  &&创建一个表单
```

```
ENDDEFINE

DEFINE CLASS lstMyListBox AS ListBox  && 创建 ListBox 控件 |
    Left = 10  && List Box 列
    Top = 10  && List Box 行
    MultiSelect = .T.  && 允许选择多项
```

```
PROCEDURE Click
    ACTIVATE SCREEN
    CLEAR
    ? "Selected items:"
    ? "--------------"
    FOR nCnt = 1 TO ThisForm.lstListBox1.ListCount
        IF ThisForm.lstListBox1.Selected(nCnt)  && 该项是否被选
            ? SPACE(5) + ThisForm.lstListBox1.List(nCnt) && 显示项
        ENDIF
    ENDFOR

ENDDEFINE
```

应用于

组合框，列表框

请参阅

AddItem **方法**，List **属性**，ListItemID **属性**，RemoveItem **方法**

ListIndex **属性**

指定组合框或列表框中选定数据项的索引号。设计时不可用，运行时可读写。

语法

Control.ListIndex[= nIndex]

设置

nIndex

ListIndex **属性的设置是：**

设置	说明
0	（默认值）指示没有选定数据项。对组合框来说，这说明用户输入了列表中没有的值。
1 ... ListCount	选定数据项的索引。

说明

下面显示了选定数据项的字符串。

```
? List(MyList.ListIndex)
```

使用控件的 Value **属性可返回相同的值。**

应用于

组合框，列表框

请参阅

AddItem **方法**, AddListItem **方法**, IndexToItemID **方法**, ItemIDToIndex **方法**, List **属性**, ListItem **属性**, RemoveItem **方法**, Value **属性**

ListItem 属性

该属性是一个字符串数组，用于通过 ID 值访问组合框或列表框控件中的数据项。设计时不可用。运行时可读写。

语法

Control.ListItem(nItemID)[= cChar]

设置

nItemID

使用数据项的唯一标识 (ID) 指定数据项。

说明

ListItem 属性与 ListCount 属性联合使用，从 1 到 ListCount 逐一返回列表中的所有项。使用 List 属性可按显示顺序检索数据项；使用 ListItem 属性可按项的标识检索项。

注意 要向组合框或列表框中添加数据项，可使用 AddItem 或 AddListItem 方法。要删除项，可使用 RemoveItem 或 RemoveListItem 方法。要按字母顺序放置项，在向列表中添加项之前设置控件的 Sorted 属性为“真” (.T.)。

应用于

组合框，列表框

请参阅

AddItem 方法，AddListItem 方法，IndexToItemID 方法，ItemIDToIndex 方法，ListCount 属性，List 属性，ListItemID 属性

ListItemID 属性

指定组合框或列表框控件中选定项的唯一标识号。设计时不可用，运行时可读写。

语法

Control.ListItemID[= nItemID]

设置

nItemID

ListItemID 属性的设置和说明是：

设置	说明
–1	指示没有选定项。对于组合框，这说明用户输入了列表中没有的值。

1(或大于 1 的任何数)	选定项的项标识。

说明

设计时不可用，运行时可读写。若用 RemoveListItem 方法从列表中删除一项，则所有剩余项都保持各自的标识号。当用 AddItem 方法向列表中添加项并且 Sorted 属性设置为“假”(.F.)时，指定 *nItemID* 为可用的最低值。当用 AddListItem 方法向列表中添加项并且 Sorted 属性设置为“假”(.F.)时，可指定 *nItemID* 为任何数。

应用于

组合框，列表框

请参阅

AddItem 方法，AddListItem 方法，IndexToItemID 方法，ItemIDToIndex 方法，ListCount 属性，List 属性，ListItem 属性，RemoveItem 方法，RemoveListItem 方法

_LMARGIN 系统变量

包含此变量是为了提供向后兼容性。可用报表设计器代替。

LOAD 命令

包含此变量是为了提供向后兼容性。可用 SET LIBRARY 代替。

Load 事件

在创建对象前发生。

语法

PROCEDURE Object.Load

[LPARAMETERS nIndex]

参数描述

nIndex

唯一地标识控件数组中的某个控件。

说明

LOAD 事件先为表单集发生，然后再为其包含的表单发生。 Load 事件先于 Activate 和 GotFocus 事件发生。

为了避免创建表单，Load 事件返回 false (.F.)；不执行 Destroy 事件。

应用于

表单，表单集

请参阅

Activate 事件, GotFocus 事件, Unload 事件

LOADPICTURE() 函数

为位图文件，图标文件或 Windows 图元文件 (Windows meta file) 创建一个对象。

语法

LOADPICTURE([cFileName])

返值类型

对象

参数描述

cFileName

指定位图文件 (.BMP)，图标文件 (.ICO)或 Windows 图元文件 (.WMF)。如果省略 *cFileName*，则返回一个空图对象（不是 .NULL.）。您可以用 GETPICT ( ) 做为 *cFileName*，这样，函数运行时将开启“打开”对话框，您可以从中选取您需要的位图文件 (.BMP)。

说明

许多 ActiveX 控件要求使用对象作为其设置。例如，ActiveX Outline 控件的 PictureOpen 属性就需要一个对象作为其设置。

请参阅

GETPICT( ), SAVEPICTURE( )

LOCAL 命令

创建局部变量和变量数组。

语法

LOCAL VarList

–或者–

LOCAL [ARRAY] ArrayName1(nRows1 [, nColumns1])
[, ArrayName2(nRows2 [, nColumns2])] ...

参数描述

VarList

指定要创建的一个或多个局部变量。

[ARRAY] ArrayName1 (nRows1 [, nColumns1])
[, ArrayName2 (nRows2 [, nColumns2])] ...

指定要创建的一个或多个局部数组。有关每个参数的说明请参阅 DIMENSION。

说明

局部变量和变量数组只能在创建它们的过程或函数中使用和更改，不能被高层或低层程序访问。一旦包含局部变量和数组的过程或函数执行完毕，则这些局部变量和数组将被释放。

用 LOCAL 创建的变量和数组都初始化为“假”(.F.)。必须在赋值之前把变量或数组声明为局部。若在用 LOCAL 声明一个变量或数组为局部变量或数组之前，对该变量或数组进行赋值，则 Visual FoxPro 产生错误信息。

局部变量可以由引用方式传递。

不能缩写 LOCAL，因为 LOCAL 和 LOCATE 的前四个字母相同。

请参阅

DIMENSION, FUNCTION, LPARAMETERS, PARAMETERS, PARAMETERS(), PRIVATE, PUBLIC, RELEASE

LOCATE 命令

按顺序搜索表，从而找到满足指定逻辑表达式的第一个记录。

语法

LOCATE FOR lExpression1
[Scope]
[WHILE]
[NOOPTIMIZE]

参数描述

FOR lExpression1

LOCATE **按顺序搜索当前表以找到满足逻辑表达式** *lExpression1* **的第一个记录。**

若 *lExpression1* **是可优化表达式，则** Rushmore **优化由** LOCATE FOR **创建的查询。为了获得最佳执行效果，可在** FOR **子句中使用可优化表达式。**

详细内容，请参阅稍后的 SET OPTIMIZE **命令与《**Microsoft Visual FoxPro 6.0 **中文版**

程序员指南》第十五章“优化应用程序”中的“掌握 Rushmore 技术”。

Scope

指定要定位的记录范围。只有范围内的记录才被定位。Scope 子句有：ALL、NEXT *nRecords*、RECORD *nRecordNumber* 和 REST。包含 *Scope* 的命令只能在活动工作区中的表上操作。

LOCATE 的默认范围是所有 (ALL) 记录。

WHILE lExpression2

指定一个条件，只要逻辑表达式 *lExpression2* 计算值为“真” (.T.)，就继续查找记录。

NOOPTIMIZE

关闭 LOCATE 的 Rushmore 优化。

详细内容，请参阅稍后的 SET OPTIMIZE 命令与《Microsoft Visual FoxPro 6.0 中文版程序员指南》第十五章“优化应用程序”中的“掌握 Rushmore 技术”。

说明

被搜索的表不必已有索引。

若 LOCATE 发现一个满足条件的记录，可使用 RECNO( ) 返回该记录号。若发现满足条件的记录，则 FOUND( ) 返回“真” (.T.)，EOF( ) 返回“假” (.F.)。若 SET TALK 是 ON，则显示满足条件的记录号。

LOCATE 发现一个满足条件的记录之后，可执行 CONTINUE，从而在表的剩余部分寻找其他满足条件的记录。当执行 CONTINUE 时，搜索操作从满足条件的记录的下

一条记录开始继续执行。可重复执行 CONTINUE，直到到达范围边界或表尾。

若找不到满足条件的记录，则 RECNO( ) 返回表中的记录数加 1，FOUND( ) 返回“假”(.F.)，EOF( ) 返回“真”(.T.)。

LOCATE 和 CONTINUE 只能用于当前工作区。若选择了另一工作区，则当重选原来的工作区时可继续原来的搜索过程。

示例

下面的示例定位了来自德国的顾客记录。并且显示总数。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE customer  && 打开 Customer 表
SET TALK OFF

STORE 0 TO gnCount

LOCATE FOR ALLTRIM(UPPER(customer.country)) = 'GERMANY'
DO WHILE FOUND( )
    gnCount = gnCount + 1
    ? company
    CONTINUE
ENDDO

? 'Total companies Germany: '+ LTRIM(STR(gnCount))
```

请参阅

CONTINUE, EOF( ), FIND, FOUND( ) , INDEXSEEK( ) , RECNO( ) , SEEK, SEEK( ) , SET OPTIMIZE

LOCFILE() 函数

在磁盘上定位文件，并返回带有路径的文件名。

语法

LOCFILE(cFileName [, cFileExtensions] [, cFileNameCaption])

返值类型

字符型

参数描述

cFileName

指定要定位的文件名称。如果 *cFileName* 只包含文件名，LOCFILE( ) 函数将首先搜索 Visual FoxPro 的默认目录或文件夹。如果在默认目录或文件夹中找不到该文件，则搜索 Visual FoxPro 路径。可以用 SET PATH 命令指定 Visual FoxPro 路径。

如果 *cFileName* 包含路径及文件名，则在指定的位置搜索。如果在指定的位置上没有找到该文件，LOCFILE( ) 函数将先搜索 Visual FoxPro 默认目录或文件夹，然后再搜索 Visual FoxPro 路径。

如果已定位了该文件，LOCFILE( ) 返回文件名与路径。

cFileExtensions

指定要定位文件的扩展名。如果用 *cFileName* 指定的文件名不含扩展名，Visual FoxPro 将把 *cFileExtensions* 中列出的各个文件扩展名作为该文件的扩展名，并再作搜索。

cFileExtensions 还可以指定当不能定位指定的文件时，在“打开”对话框中显示的文件的扩展名。

cFileExtensions 可以有不同的形式：

- 如果 *cFileExtensions* 包含单个扩展名（例如，PRG），则只显示具有此扩展名的文件。
- *cFileExtensions* 还可以包含通配符（* 与 ?），此时显示扩展名满足通配符条件的所有文件。例如，如果 *cFileExtensions* 为 ?X?，则所有具有扩展名 .FXP、.EXE 或 .TXT 的文件都将显示出来。
- 在 Visual FoxPro for Windows 中，cFileExtensions 可以包含一个文件说明，后面跟一个文件扩展名，或者跟一组逗号分隔的文件扩展名。该文件说明出现在“文件类型”列表框中。文件说明和扩展名之间用冒号 (:) 分隔。多个文件说明和扩展名之间用分号 (;) 分隔。

  例如，如果 cFileExtensions 为 “文本文件:TXT”，说明“文本文件”会出现在“文件类型”列表框中，并且还会显示所有以 .txt 为后缀的文件。

  如果 cFileExtensions 为 “表:DBF; 文件:TXT,BAK”，说明“表”和“文件”会出现在“文件类型”列表框中。从“文件类型”列表框中选择“表”时，会显示所有以 .dbf 为后缀的文件。从“文件类型”列表框中选择“文件”时，会显示所有以 .txt 和 .bak 为后缀的文件。

cFileNameCaption

指定用来提示用户的文本。该文本出现在输入文件名的文本框的左侧。如果省略本参数，则显示“文件名”。

关于 Visual FoxPro 的文件扩展名与相应的创建程序类型，请参阅帮助中的“文件扩展名与文件类型”。

说明

如果在默认目录或文件夹下、在 Visual FoxPro 路径上以及指定位置上都无法找到该文件，将显示“打开”对话框。该对话框可用于定位文件。当从对话框中选定某个文件时，返回带有文件路径的文件名。

如果选择“取消”、按 ESC 键或从控制菜单上选择“关闭”而退出“打开”对话框，Visual FoxPro 将产生错误信息，LOCFILE( ) 也不会返回值。

请参阅

FILE( ), GETFILE( ), GETPICT( ), PUTFILE( ), SET PATH

LOCK() 函数

尝试锁定表中一个或更多的记录。

语法

LOCK([nWorkArea | cTableAlias]
| [cRecordNumberList, nWorkArea | cTableAlias])

返值类型

逻辑值

参数描述

nWorkArea | cTableAlias

尝试锁定指定工作区中已打开表的当前记录。*nWorkArea* 指定工作区号，*cTableAlias* 指定表别名。如果不指定工作区和表别名，LOCK( ) 函数尝试锁定当前工作区中的表的当前记录。

cRecordNumberList

尝试锁定多个记录时，指定必须包含的一个或更多记录编号的列表。SET MULTILOCKS 必须为 ON，并且必须要包含给多个记录加锁的工作区或表的别名。

LOCK( ) 函数尝试锁定您所指定的所有记录。*cRecordNumberList* 指定的记录编号之间用逗号分隔。例如尝试给某表的前四个记录加锁时，*cRecordNumberList* 应为 1,2,3,4。

若要锁定多个记录，可以对其中的每一个记录重复下列步骤：移动记录指针指向要锁定的记录，再发出 LOCK( ) 或 RLOCK( ) 命令。

在 Visual FoxPro 中，可以将 0 指定为记录编号。指定 0 时试图锁定表头。

警告 锁定表头的时间应尽可能地短，因为表头锁定时其他用户不能向表中添加记录。

可以用 UNLOCK RECORD 0，UNLOCK 或 UNLOCK ALL 命令解除对表头的锁定。

如果在 *cRecordNumbers* 中指定的所有记录均锁定成功，LOCK( ) 函数返回“真”

(.T.)。由 *cRecordNumbers* 指定的记录中即使有一个记录不能锁定，LOCK( ) 函数也将返回“假”(.F.) 并且不锁定任何记录。不过，原来已有的记录锁仍保留原样。多个记录的锁定是一个叠加过程。设置额外的记录锁并不释放对其他记录的锁定。

在一个工作区中可以锁定的最大记录数约为 8000，锁定整个表的速度总是比只锁定一部分记录要快。

说明

LOCK( ) 函数与 RLOCK( ) 函数等价。

如果成功地设置了锁定和解锁，则 LOCK() 返回“真”(.T.)。设置锁定的用户对锁定的记录有读写权限，而网络上的其他用户有只读权限。

执行 LOCK( ) 函数并不保证记录锁设置成功。不能对已由另一用户锁定的记录或位于其他用户锁定的表中的记录设置记录锁。如果由于某种原因不能设置记录锁，LOCK( ) 函数将返回“假”(.F.)。

在默认情况下，LOCK( ) 函数尝试锁定记录一次。使用 SET REPROCESS 命令可以在第一次尝试失败后自动地重设记录锁。SET REPROCESS 确定初始锁定失败时锁定操作的尝试次数或锁定尝试应当持续的时间。详细内容，请参阅稍后的 SET REPROCESS 命令。

SET MULTILOCKS 决定能否锁定表中的多个记录。如果 SET MULTILOCKS 为 OFF（默认值），则只能锁定表中的单个记录。而 SET MULTILOCKS 为 ON 时，可以锁定表中的多个记录。详细内容，请参阅稍后的 SET MULTILOCKS 命令。

对记录解锁　表记录只能由设置锁定的用户进行解锁。发出 UNLOCK 命令、关闭表或退出 Visual FoxPro 都可以释放记录锁定。

UNLOCK 可用于释放当前工作区、指定的工作区或所有工作区中的记录锁定。详细内容，请参阅稍后的 UNLOCK 命令。

将 SET MULTILOCKS 从 ON 转换到 OFF 或从 OFF 转换到 ON 即隐含地执行了 UNLOCK ALL — 释放所有工作区中的记录锁定。

可以用 USE、CLEAR ALL 或 CLOSE DATABASES 命令来关闭表。

有关网络上的记录、文件锁定和共享表的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第十七章“共享访问程序设计”。

示例

下面的示例对 customer 和 employee 表中的前四个记录设置锁定和解锁。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
SET REPROCESS TO 3 AUTOMATIC
STORE '1,2,3,4' TO gcRecList
gcOldExc = SET('EXCLUSIVE')
SET EXCLUSIVE OFF
SELECT 0
USE employee  && 打开 Employee 表
SELECT 0
USE customer  && 打开 Customer 表
? LOCK('1,2,3,4', 'customer')  && 锁住 customer 表的前四个记录
? RLOCK(gcRecList, 'employee')  && 锁住 employee 表的前四个记录
UNLOCK IN customer
UNLOCK IN employee
SET EXCLUSIVE &gcOldExc
```

请参阅

CLEAR, CLOSE, FLOCK( ), RLOCK( ), SET MULTILOCKS, SET

LockScreen 属性

确定表单是否以批处理方式执行对表单及所含对象属性设置的更改。设计和运行时可用。

语法

Object.LockScreen[= lExpr]

参数描述

lExpr

LockScreen 属性的设置有：

设置	说明
“真” (.T.)	表单及其包含的对象以批处理方式反映对属性设置的更改，或者说在同一时刻反映，而不是在更改后立即反映。
“假” (.F.)	（默认值）表单及其包含的对象在对属性设置更改后立即反映出更改。

说明

需要在运行时对 BackColor、FontName 等显示属性进行更改，应设置 LockScreen 为“真”(.T.)以减少令人厌烦的屏幕刷新动作。

LockScreen 属性不能防止对表单的更改立即反应出来。例如，即使 LockScreen 属性设置为“真”，在调用 Move 方法时表单会被移动。

注意 如果设置 LockScreen 为“假”(.F.)，将重画表单及其所有的控件。

应用于

表单，_SCREEN，工具栏

请参阅

Paint 事件

LOG() 函数

返回给定数值表达式的自然对数（底数为 *e*）。

语法

LOG(nExpression)

返值类型

数值型

参数描述

nExpression

指定的数值表达式，LOG( )函数返回方程 $e^x = nExpression$ 中的 x 值。*nExpression* 必须大于 0。

说明

自然对数以常数 *e* 为底。在返回的结果中，小数的位数由 SET DECIMALS 命令指定。

示例

```
CLEAR
? LOG(1) && 显示 0.00
STORE EXP(2) TO gneSquare
? LOG(gneSquare) &&显示 2.00
```

请参阅

EXP( ), LOG10( ), SET DECIMALS

LOG10( ) 函数

返回给定数值表达式的常用对数（以 10 为底）。

语法

LOG10(nExpression)

返值类型

数值型

参数描述

nExpression

指定的数值表达式，LOG10( )**函数返回方程** $10^x = nExpression$ **中的** x **值。***nExpression* **必须大于** 0。

说明

常用对数以 10 **为底。在返回的结果中，小数的位数由** SET DECIMALS **命令指定。**

示例

```
CLEAR
? LOG10(10) && 显示 1.00
STORE 100 TO gnBaseTen
? LOG10(gnBaseTen) && 显示 2.00
? LOG10(gnBaseTen^2) && 显示 4.00
```

请参阅

EXP( ), LOG( ), SET DECIMALS

LOOKUP() 函数

在表中搜索字段值与指定表达式匹配的第一个记录。

语法

LOOKUP(ReturnField, eSearchExpression, SearchedField [, cTagName])

返值类型

字符型、数值型、货币型、浮点型、整型、双精度型、日期型、日期时间型或逻辑型。

参数描述

ReturnField

指定的一个字段，搜索成功时，LOOKUP()函数返回此字段中的内容。如果搜索不成功，LOOKUP()函数返回与 *ReturnField* 长度和数据类型都相同的空字符串。

eSearchExpression

指定搜索表达式。搜索表达式常常是表中某个字段的内容，或者与活动索引或复合索引标识的索引表达式相对应。

SearchedField

指定要搜索的字段。如果该表没有活动索引，LOOKUP()函数对由

SearchedField 指定的字段执行顺序搜索。

如果打开的某个索引文件或索引标识的索引关键字表达式正是您指定的搜索字段，LOOKUP( )函数将利用索引文件或索引标识执行快速搜索。

cTagName

指定 LOOKUP( )函数在搜索时使用的复合索引标识名。复合索引搜索是 LOOKUP( ) 函数所能执行的最快的搜索方式。

说明

如果搜索成功，LOOKUP( )函数移动记录指针指向匹配记录，并返回该记录中指定字段的内容。

如果没有找到搜索表达式，LOOKUP( )函数返回与 *ReturnField* 长度和数据类型相同的空字符串，并将记录指针定位在文件尾。

如果使用 LOOKUP( )在父表中搜索，则把所有相关子表的记录指针移动到相关记录上。

本函数不能用 Rushmore 优化。

示例

在下面的示例中，LOOKUP( )函数使用索引标识 company 搜索字符串 "Ernst Handel" 的第一次出现。如果搜索成功，LOOKUP( ) 函数返回字段 contact 的内容，并由 @ ... SAT 命令显示这个返回值。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer ORDER company  && 打开 Customer 表
CLEAR
@ 2,2 SAY LOOKUP(contact, 'Ernst Handel', company, 'company')
```

请参阅

FIND, INDEX, LOCATE, SEEK, SEEK()

LostFocus 事件

当某个对象失去焦点时发生。

语法

PROCEDURE Object.LostFocus
[LPARAMETERS nIndex]

参数描述

nIndex

唯一地标识控件数组中的一个控件。

说明

这一事件发生的时间取决于对象的类型：

- **控件由于用户的操作而失去焦点，这类操作包括选中另一个控件或在另一个控件上单击，或在代码中用** SetFocus **方法更改焦点。表格由于用户在** Microsoft **窗口下击** CTRL+TAB **键或在** Macintosh **机上击** CONTROL+TAB **键**

退出时失去焦点。

- 只有当表单不包含任何控件，或者所包含的所有控件的 Enabled 和 Visible 属性的设置均为“假”(.F.)时，表单失去控件，或者另一个表单获得焦点。

对于表单，LostFocus 事件在 Deactivate 事件之前发生。

应用于

复选框，组合框，命令按钮，容器对象，控件对象，编辑框，表单，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，微调，文本框

请参阅

Deactivate 事件，Enabled 属性，GotFocus 事件，Visible 属性

LOWER() 函数

以小写字母形式返回指定的字符表达式。

语法

LOWER(cExpression)

返值类型

字符型

参数描述

cExpression

指定要由 LOWER( ) 函数转换的字符表达式。

说明

LOWER( ) 函数将字符表达式中所有的大写字母 (A–Z) 转换为小写字母 (a–z)。字符表达式中的其他字符保持不变。

示例

```
STORE 'FOX' TO gcName
CLEAR
? LOWER(gcName) && 显示 fox
```

请参阅

ISALPHA( ), ISLOWER( ), ISUPPER( ), PROPER( ), UPPER( )

LPARAMETERS 命令

将调用程序传入的数据赋给局部变量或数组。

语法

LPARAMETERS ParameterList

参数描述

ParameterList

为接收数据的局部变量或数组指定名称。

在 *ParameterList* 中，参数之间用逗号分隔。在 LPARAMETERS 语句中，至少应有与 DO ... WITH 语句中同样多的参数。如果在 LPARAMETERS 语句中列出的变量或数组的数目多于 DO ... WITH 语句传递的参数数目，则多出的变量初始化为“假”(.F.)。最多可以传递 27 个参数。

可以使用 PARAMETERS( ) 函数确定向最近一次执行的程序、过程或用户自定义函数传递的参数数目。

说明

LPARAMETERS 命令在被调用的程序过程或用户自定义函数中创建局部变量和数组。可用 PARAMETERS 命令创建私有变量和数组。

如果向程序、过程或用户自定义函数传递值、变量或者数组，那么，LPARAMETERS 必须是被调用程序、过程或用户自定义函数的第一条可执行语句。

默认情况下，DO ... WITH 以引用传递方式向过程传递变量和数组。在被调用过程中更改某个值时，新的值将传递回调用程序中相关的变量和数组。如果希望以值传递方式向过程传递某一变量或数组，应在 DO ... WITH 语句的参数列表中用圆括号将该变量或数组括起来。此时，被调用过程中参数的任何更改都不会传递回调用程序。

在默认情况下，变量以引用传递方式传递给过程，以值传递方式传递给用户自定义函数。可以用 SET UDFPARMS TO REFERENCE 指定以引用传递方式向用户自定义函数传递变量。

请参阅

DO, FUNCTION, LOCAL, PARAMETERS, PARAMETERS(), PRIVATE, PROCEDURE, PUBLIC, SET UDFPARMS

LTRIM() 函数

删除指定字符表达式的前导空格后返回。

语法

LTRIM(cExpression)

返值类型

字符型

参数描述

cExpression

指定的字符表达式，LTRIM()函数将删除其前导空格。

说明

当使用STR()函数将数值转换为字符串时，可能会插入前导空格，本函数对于删除这些前导空格很有用。

示例

```
STORE 'Redmond' TO gcCity
STORE 'Washington' TO gcState
CLEAR
? gcCity, gcState  &&显示 Redmond Washington
? gcCity, LTRIM(gcState)  &&显示 Redmond Washington
```

请参阅

ALLTRIM( ), LEFT( ), RIGHT( ), RTRIM( ), SUBSTR( ), TRIM( )

LUPDATE() 函数

返回一个表最近更新的日期。

语法

LUPDATE([nWorkArea | cTableAlias])

返值类型

日期型

参数描述

nWorkArea | cTableAlias

返回在某一工作区中打开的表最近一次更改的日期。*nWorkArea* 指定工作区号，*cTableAlias* 指定表别名。如果省略 *nWorkArea* 和 *cTableAlias*，

LUPDATE( ) 函数返回当前选定工作区中的表最近一次更新的日期。

如果指定的工作区中没有表打开，LUPDATE( )函数返回空日期。如果不存在具有指定别名的表，Visual FoxPro 将产生错误信息。

说明

本函数在更新过程中很有用。

注意，在早期版本的 Visual FoxPro 中，表最后更新的年的后两个数字保存在表头中。由于只保存了后两个数字，所以无法确定该表更新的世纪；默认值为二十世纪。

在 Visual FoxPro 6.0 中，LUPDATE() 询问 Windows，以确定表更新的日期，允许确定表更新的世纪。但是，表最后更新的年的后两个数字仍保存在表头中。

示例

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer && 打开表 Customer
CLEAR
? LUPDATE( ) && 显示最近一次更新的日期
```

请参阅

DIR, FDATE( ), FTIME( )

_MAC 系统变量

如果用户使用的是 FoxPro for Macintosh 或 Microsoft Visual FoxPro for Macintosh，则为“真” (.T.)。

语法

_MAC = lExpression

说明

对于任何其他平台的 FoxPro (Visual FoxPro for Windows 、FoxPro for Windows、FoxPro for MS-DOS 或 FoxPro for UNIX)，_MAC 为“假”。

不能使用 STORE 或 = 更改 _MAC 中包含的值。

请参阅

_DOS, _UNIX, VERSION(), _WINDOWS

MacDesktop 属性

指定表单是否放置在 Visual FoxPro 主窗口中。

语法

Object.MacDesktop[= nValue]

参数描述

nValue

MacDesktop 属性设置为：

设置	说明
0	（默认值）自动。根据 SET MACDESKTOP 命令的设置，确定表单是否包含在 Visual FoxPro 主窗口中。如果 SET MACDESKTOP 为 ON，表单位于 Macintosh 桌面的顶部；如果设置为 OFF，则表单包含在 Visual FoxPro 主窗口中。
1	Macintosh 桌面。表单位于 Macintosh 桌面的顶部，并且可以在移动和调整大小时独立于 Visual FoxPro 主窗口。
2	Visual FoxPro 桌面。表单包含在 Visual FoxPro 主窗口中，并且不能移动到外面去。

说明

设置 MacDesktop 属性可以覆盖使用 SET MACDESKTOP 命令对单个表单进行的设置。SET MACDESKTOP 影响用户自定义窗口（表单）和系统窗口，例如浏览、查看和表单设计器窗口。在默认情况下，用户自定义窗口的行为遵循 SETMACDESKTOP 的设置。

MacDesktop 属性使您能控制您的应用程序更象 Macintosh 还是更像 Windows。

应用于

表单

请参阅

SET KEYCOMP

MainClass 属性

包含项目中设置为主程序的 ActiveDoc 类的名称。只读。

语法

Object.MainClass

说明

在一个 ActiveDoc 类上单击鼠标右键，并且从快捷菜单中选择“设置为主程序”命令，可以将这个 ActiveDoc 类设置为主程序。只有基于 ActiveDoc 基类的类才可以设置为主程序（程序或表单也可以设置为主程序）。使用 MainFile 属性可以确定 .vcx 可视类库，其中包含了设置为主程序的 ActiveDoc 类。

如果主程序没有设置为 ActiveDoc 类，则 MainClass 包含空字符串。

用 SetMain 方法来更改主文件和主类。

应用于

项目对象

请参阅

MainFile 属性, SetMain 方法

MainFile 属性

包含项目中设置为主程序的文件的名称和路径。只读。

语法

Object.MainFile

说明

如果项目没有设置主程序，则 MainFile 包含空字符串。程序、表单或 ActiveDoc 类可以设置为项目的主程序，需要在程序、表单或 ActiveDoc 类上单击鼠标右键，并且从快捷菜单中选择“设置为主程序”命令。也可以使用 SetMain 方法通过编程指定项目的主程序。

如果一个 ActiveDoc 类被设置为主程序，则 MainFile 包含 .vcx 可视类库（其中包含该类）的名称和路径。使用 MainClass 属性可以确定设置为项目的主程序的 ActiveDoc 类。

应用于

项目对象

请参阅

MainClass 属性, SetMain 方法

Margin 属性

为控件的文本部分指定应留的空白宽度。设计和运行时可用。

语法

Control.Margin[= nValue]

参数描述

nValue

为控件的文本部分指定应留的空白。

说明

使用 Margin 属性来指定控件中的文本位置。Margin 属性在控件内部创建空白，它并不影响控件本身的大小。

应用于

组合框，编辑框，微调，文本框

请参阅

Alignment **属性**

MAX() 函数

比较几个表达式的值，并返回其中具有最大值的表达式。

语法

MAX(eExpression1, eExpression2 [, eExpression3 ...])

返值类型

字符型、数值型、货币型、双精度型、浮点型、日期型或日期时间型

参数描述

eExpression1, eExpression2 [, eExpression3 ...]

指定的若干个表达式，MAX( )**返回其中具有最大值的表达式。所有表达式必须为同一数据类型。**

示例

下列示例使用 APPEND BLANK **创建一个包含** 10 **个记录的表，其中的值是任意的，然后使用** **和** MAX() **函数显示表中的最大和最小值。**

```
CLOSE DATABASES
CREATE TABLE Random (cValue N(3))
FOR nItem = 1 TO 10  && Append 10 records,
    APPEND BLANK
    REPLACE cValue WITH 1 + 100 * RAND( ) &&插入任意值
ENDFOR

CLEAR
LIST  && 显示值
gnMaximum = 1  && 初始化最小值
gnMinimum = 100  && 初始化最大值
SCAN
    gnMinimum = MIN(gnMinimum, cValue)
    gnMaximum = MAX(gnMaximum, cValue)
ENDSCAN
? 'The minimum value is: ', gnMinimum  && 显示最小值
? 'The maximum value is: ', gnMaximum  && 显示最大值
```

请参阅

CALCULATE, MIN(), SELECT – SQL

MaxButton 属性

指定表单是否含有最大化按钮。设计和运行时可用。

语法

Object.MaxButton[= lExpr]

参数描述

lExpr

MaxButton **属性的设置为：**

设置	**说明**
“真” (.T.)	**（默认值）表单有最大化按钮。**
“假” (.F.)	**表单没有最大化按钮**

说明

最大化按钮允许用户将表单扩大到整屏。

最大化按钮在窗口最大时自动变为还原按钮。当窗口最小或者还原时，自动将还原按钮变为最大化按钮。

MaxButton、MinBotton、BorderStyle **和** ControlBox **属性的设置值只有在运行时才起作用。**

在运行时最大化表单将触发 Resize **事件。**

注意 WindowState **属性反映了窗口的当前状态。如果将窗口状态设置为** 2 **（最大化），表单将最大化而不管** MaxButton **和** BorderStyle **属性中的设置。**

应用于

表单，_SCREEN

请参阅

BorderStyle 属性，ControlBox 属性，MinButton 属性，Resize 事件，TitleBar 属性，WindowState 属性

MaxHeight 属性

决定表单可能的最大高度。设计和运行时可用。

语法

Object.MaxHeight[= nHeight]

参数描述

nHeight

表单的最大高度，度量单位由表单的 ScaleMode 属性指定。

说明

当调整表单大小时，无论是选择控件菜单的“大小”命令还是拖动表单的边框，表单的高度将不会超过 MaxHeight 属性的设定值。

MaxHeight 的默认设置为 -1；即没有指定最大高度。

应用于

表单，_SCREEN

请参阅

ScaleMode 属性

MaxLeft 属性

指定表单从 Visual FoxPro 主窗口左侧边缘处开始的最大可能距离。设计和运行时可用。

语法

Object.MaxLeft[= nMaxLeft]

参数描述

nMaxLeft

指定 Visual FoxPro 主窗口的左边界与最大化表单左边界间的距离，度量单位由表单的 ScaleMode 属性指定。

说明

用 MaxLeft 属性来保证最大化表单在运行时不会离 Visual FoxPro 主窗口左边界太远。

应用于

表单， _SCREEN

请参阅

MaxTop 属性, MaxWidth 属性, MinHeight 属性, Scalemode 属性

MaxLength 属性

指定允许在编辑框中输入字符的最大长度。设计和运行时可用。

语法

Control.MaxLength[= nMaxLength]

参数描述

nMaxLength

可输入到编辑框中的最大字符数。若 *nMaxLength* 设置为 0，那么输入到编辑框的字符数目没有限制。

说明

可使用 MaxLength 来限制用户输入编辑框的字符数目。当 MaxLength 大于 0 时，MaxLength 只适用于文本框，文本框不使用 InputMask 属性，并且它的 Value 属性是字符类型。

应用于

编辑框，文本框

请参阅

MaxTop 属性

指定最大化表单与 Visual FoxPro 主窗口上边界之间的距离。设计和运行时可用。

语法

Object.MaxTop[= nMaxTop]

参数描述

nMaxTop

指定最大化表单上边界与 Visual FoxPro 主窗口上边界之间的距离。度量单位由表单的 ScaleMode 属性指定。

说明

在 Visual FoxPro for the Macintosh 中忽略此属性。

用 MaxTop 属性来确保在运行时刻，最大化的表单不会移动到与 Visual FoxPro 主窗口的顶部太近的地方。

应用于

表单，_SCREEN

请参阅

MaxLeft 属性, MaxWidth 属性, MinHeight 属性, MinWidth 属性, Scalemode 属性

MaxWidth 属性

指定表单的最大宽度。设计和运行时可用。

语法

Object.MaxWidth[= nMaxWidth]

参数描述

nMaxWidth

指定表单的最大宽度，度量单位由表单的 ScaleMode 属性指定。

说明

调整表单时，无论是选用控件菜单中的“大小”命令还是拖动表单的边框，表单的宽度都不会大于 MaxWidth 属性中设置的值。MaxWidth 的默认设置为 -1；即没有指定最大宽度。

应用于

表单， _SCREEN

请参阅

MaxHeight 属性, MinHeight 属性, MinWidth 属性, ScaleMode 属性

MCOL() 函数

返回鼠标指针在 Visual FoxPro 主窗口或用户自定义窗口中的列位置。

语法

MCOL([cWindowName [, nScaleMode]])

返值类型

数值型

参数描述

cWindowName

指定窗口名，MCOL( ) 函数返回鼠标指针在该窗口中的列位置。

若省略 *cWindowName*，且不存在活动的用户自定义窗口，将返回鼠标指针在 Visual FoxPro 主窗口中的列位置。若省略 *cWindowName*，且存在活动的用户自定义窗口，则返回鼠标指针在活动的用户自定义窗口中的列位置。如果鼠标指针位于用户自定义窗口之外，则返回 -1。如果未安装鼠标驱动程序，且没有输出窗口，则返回 -1。

nScaleMode

设定 MCOL( ) **返回值所使用的度量单位。**

nScaleMode	描述
0	（默认）Foxels。foxel 的大小是当前表单中字型宽与高的平均值。当开发可在字符平台和图形平台上跨平台运行的应用程序时，foxel 十分有用。
3	像素。像素是屏幕和打印机的最小分辨单位。在不同的屏幕中，像素的大小不同。

请参阅

AMOUSEOBJ( ), COL( ), GridHitTest **方法**, INKEY( ), ISMOUSE( ), MROW( ), ROW( ), WCOLS( ), WROWS( )

MD 或 MKDIR 命令

在磁盘上创建一个新目录或子目录。

语法

MD cPath | MKDIR cPath

参数描述

cPath

指定一条路径（含驱动器指示符和目录）或目录。

若 *cPath* 是不含驱动器指示符的目录，将把该目录创建为当前 Visual FoxPro 默认目录的子目录。

说明

如果试图创建一个已存在的目录，Visual FoxPro 将产生错误信息。

示例

下面的示例用 MKDIR 创建了一个新目录 mytstdir，使用 CHDIR 转到新目录。GETDIR( ) 用于显示目录结构，然后用 RMDIR 删除新建目录。再用 GETDIR( ) 来显示目录结构。

```
SET DEFAULT TO HOME( ) && 恢复 Visual FoxPro 目录

MKDIR mytstdir &&创建一个新目录
CHDIR mytstdir && 进入新目录
= GETDIR( ) && 显示选择目录对话框
SET DEFAULT TO HOME( ) && 恢复 Visual FoxPro 目录
RMDIR mytstdir && 移动新目录
= GETDIR( ) && 显示选择目录对话框
```

请参阅

CD | CHDIR, DIRECTORY, DIRECTORY( ), GETDIR( ), HOME( ), RD | RMDIR, SET DEFAULT, SET PATH, SYS(5), SYS(2003), SYS(2004)

MDIForm 属性

指定表单是否为 MDI 界面（多文档界面）。包含此命令是为了提供向后兼容性。

MDOWN() 函数

确定鼠标键是否按下，并返回“真”(.T.)或“假”(.F.)。

MDX() 函数

根据指定的索引编号，返回打开的 .CDX 复合索引文件名。

语法

MDX(nIndexNumber [, nWorkArea | cTableAlias])

返值类型

字符型

参数描述

nIndexNumber

指定的索引编号，返回复合索引文件名。若表含有一个结构复合索引文件并且 *nIndexNumber* 为 1，将返回结构复合索引文件名（它总是与表的名称相同）；若 *nIndexNumber* 为 2，将返回 USE 或 SET INDEX 命令指定的第一个复合索引文件名；若 *nIndexNumber* 为 3，将返回第二个复合索引文件名，依次类推。若 *nIndexNumber* 大于打开的复合索引文件数，则返回空字符串。

如果表没有结构复合索引文件并且 *nIndexNumber* 为 1，则返回 USE 或 SET INDEX 命令指定的第一个复合索引文件。若 *nIndexNumber* 为 2，将返回第二个复合索引文件名，依次类推。若 *nIndexNumber* 大于打开的复合索引文件数，将返回空字符串。

nWorkArea

指定不在当前工作区打开的复合索引文件所在的工作区号。若省略这个可选参数，将返回当前工作区中的复合索引文件名。

cTableAlias

指定不在当前工作区中打开的复合索引文件的表别名。若省略该可选参数，将返回当前工作区中打开的复合索引文件名。

说明

MDX( ) 等同于 CDX( ) 。

使用含 INDEX 子句的 USE 命令或 SET INDEX 命令均可为表打开索引文件。结构复合索引文件自动随表打开。MDX( ) 忽略用 USE 或 SET INDEX 指定的任何 .IDX 索引

文件。
TAG( )函数可以返回复合索引文件中的标识名；用 NDX( )返回打开的 .IDX索引文件名。
当 SET FULLPATH为 ON时，MDX( )返回含 .CDX文件名的路径。当 SET FULLPATH为 OFF时，MDX( )返回 .CDX文件所在的驱动器指示符及 .CDX文件名。

请参阅

CDX( ), INDEX, NDX( ), SET INDEX, SET FULLPATH, SYS(14), TAG( ), USE

MDY()函数

以“月-日-年”格式返回指定日期或日期时间表达式，其中月份名不缩写。

语法

MDY(dExpression | tExpression)

返值类型

字符型

参数描述

dExpression
指定要返回的日期表达式。表达式用“月-日-年”格式表示。
tExpression
指定要返回的日期时间表达式。表达式用“月-日-年”格式表示。

说明

若 SET CENTURY 为 OFF，按 month dd,yy 格式返回字符表达式；若 SET CENTURY 为 ON，返回格式为 month dd, yyyy。

示例

下面的示例创建了一个用户自定义函数，该函数返回星期的相应日期。

```
SET CENTURY OFF

CLEAR
? Longdate({^1998-02-16}) && 显示星期一，1998 年 2 月 16 日

SET CENTURY ON
? Longdate({^1998-02-16}) && 显示星期一，1998 年 2 月 16 日

*** LongDate ***

FUNCTION longdate
PARAMETERS gdDate
RETURN CDOW(gdDate) + ', ' + MDY(gdDate)
```

请参阅

DMY(), SET CENTURY, SET DATE

MEMLINES() 函数

返回备注字段中的行数。

语法

MEMLINES(MemoFieldName)

返值类型

数值型

参数描述

MemoFieldName

指定的备注字段名。若备注字段所在表不在当前工作区中打开，则应在字段名前加上表的别名和一个句点。

说明

备注字段中的行数由 SET MEMOWIDTH 的当前值决定。

示例

下面的示例从 employee 表中扫描 3 个记录，并使用 MEMLINES( ) 确定 notes 备注字段中是否有数据，并确定何时分页。显示记录 last_name 字段中的数据，并显示 notes 字段（如果此备注字段有数据）或者显示消息指示该记录没有 notes。

```
CLOSE DATABASES
CLEAR
```

```
SET TALK OFF
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')
USE employee  && 打开 Employee 表

SET MEMOWIDTH TO 65
gnLine = 1
GOTO 2
SCAN NEXT 3
    gnMemoSize = MEMLINES(notes)
    IF gnMemoSize = 0
        STORE .T. TO glNoMemo
        STORE 1 TO gnMemoSize
    ELSE
        STORE .F. TO glNoMemo
    ENDIF
    IF gnLine + gnMemoSize > 65
        EJECT
        gnLine = 1
    ENDIF
    @ gnLine,2 SAY 'Last Name: '+ last_name
    gnLine = gnLine +1
    @ gnLine ,2 SAY 'notes: '
    ?? IIF(glNoMemo, 'No notes ',notes)
    gnLine = gnLine + gnMemoSize + 2
    IF gnLine > 24
        gnLine = 1
        CLEAR
    ENDIF
```

ENDSCAN

请参阅

ALINES( ), MLINE( ), SET MEMOWIDTH

MEMORY( ) 函数

返回可供外部程序运行的内存大小。

语法

MEMORY( )

返值类型

数值型

说明

在 Visual FoxPro **中，** MEMORY( ) **总是返回** 640K。

MEMORY( ) **类似于** SYS(12)，**但有两点区别：**

- MEMORY( ) **以千字节为单位返回可用内存数量；** SYS(12) **以字节为单位返回内存数量。**
- MEMORY( ) **返回数值表达式；** SYS(12)**以字符串形式返回它的值。**

请参阅

SYS(12), SYS(1001)

MemoWindow 属性

包含此命令是为了提供向后兼容性。可用编辑框控件代替。

MENU 命令

创建菜单系统。包含此命令是为了提供向后兼容性。可使用菜单设计器创建菜单。

MENU() 函数

以大写字符串形式返回活动菜单栏的名称。

语法

MENU( )

返值类型

字符型

说明

没有活动菜单时，MENU( )返回空字符串。用菜单设计器可创建并激活菜单。

示例

下面的示例用 MENU( ) 将菜单栏的名称传递到过程中。用 SET SYSMENU SAVE 在内存中保存当前的系统菜单栏，而用 SET SYSMENU TO 删除所有的系统菜单标题。用 DEFINE PAD 创建了几个系统菜单标题。选定了一个菜单标题时，MENU( ) 将 Microsoft Visual FoxPro 的系统菜单栏的名称，_MSYSMENU 传递到 choice 过程中。choice 过程显示所选的菜单标题和系统菜单栏的名称。如果选择了“退出”菜单，将恢复最初的 Visual FoxPro 系统菜单。

```
*** 在默认 VFP 目录下将此程序保存为 MENUEXAM.PRG***
CLEAR
SET SYSMENU SAVE
SET SYSMENU TO
DEFINE PAD padSys OF _MSYSMENU PROMPT '\<System' COLOR SCHEME 3 ;
    KEY ALT+S, ''
DEFINE PAD padEdit OF _MSYSMENU PROMPT '\<Edit' COLOR SCHEME 3 ;
    KEY ALT+E, ''
DEFINE PAD padRecord OF _MSYSMENU PROMPT '\<Record' COLOR SCHEME 3 ;
    KEY ALT+R, ''
DEFINE PAD padWindow OF _MSYSMENU PROMPT '\<Window' COLOR SCHEME 3 ;
    KEY ALT+W, ''
```

```
DEFINE PAD padReport OF _MSYSMENU PROMPT 'Re\<ports' COLOR SCHEME 3 KEY ALT+P, ''
DEFINE PAD padExit OF _MSYSMENU PROMPT 'E\<xit' COLOR SCHEME 3 ;
    KEY ALT+X, ''
ON SELECTION MENU _MSYSMENU ;
    DO choice IN menuexam WITH PAD(), MENU()
PROCEDURE choice
PARAMETER gcPad, gcMenu
WAIT WINDOW 'You chose ' + gcPad + ;
    ' from menu ' + gcMenu NOWAIT
IF gcPad = 'PADEXIT'
    SET SYSMENU TO DEFAULT
ENDIF
```

请参阅

ACTIVATE MENU, CREATE MENU, DEFINE MENU

MENU TO 命令

包含此命令是为了提供向后兼容性。可用 Visual FoxPro **的菜单设计器来创建菜单。**

Message 事件

在屏幕底部的状态栏中显示一条消息。包含此命令是为了提供向后兼容性。可用 StatusBarText 属性代替。

MESSAGE() 函数

以字符串形式返回当前错误消息，或者返回导致这个错误的程序行内容。

语法

MESSAGE([1])

返值类型

字符型

参数描述

1

当在 ON ERROR 过程中使用 MESSAGE( ) 时，包含这个参数可以返回导致这个错误的程序源代码。当不能取得程序源代码时，MESSAGE(1) 返回下列

内容之一：

- 当此行是宏替换时，返回整个程序行。
- 当此行是不含附加子句的命令时，返回该命令。
- 当程序行是含附加子句的命令时，返回命令以及三个句点(...)。

说明

与 ERROR( ) 不同，MESSAGE( ) 不被 RETURN 或 RETRY 重置。

示例

下面的示例显示了 MESSAGE( ) 和 MESSAGE(1) 的输出数值。

```
ON ERROR DO Errhand

*** 下一行可能出现错误 ***

USE Nodatabase
ON ERROR   &&恢复系统错误处理
PROCEDURE Errhand
? 'Line of code with error: ' + MESSAGE(1)
? 'Error number: ' + STR(ERROR( ))
? 'Error message: ' + MESSAGE( )
```

请参阅

ERROR( ), ON ERROR

MESSAGEBOX() 函数

显示一个用户自定义对话框。

语法

MESSAGEBOX(cMessageText [, nDialogBoxType [, cTitleBarText]])

返值类型

数值型

参数描述

cMessageText

指定在对话框中显示的文本。在 *cMessageText* 包含回车符 (CHR(13)) 可以使消息移到下一行显示。对话框的高度和宽度会根据 *cMessageText* 适当增大，以包含全部消息。

nDialogBoxType

指定对话框中的按钮和图标、显示对话框时的默认按钮以及对话框的行为。

在下面的表中，对话框按钮值从 0 到 5 指定了对话框中显示的按钮。图标值 16、32、64 指定了对话框中的图标。默认值 0、256、512 指定对话框中哪个按钮为默认按钮。当显示对话框时，此默认按钮被首先选中。

当省略 nDialagBoxType 时，等同于指定 nDialogBoxType 值为 0。

数值	对话框按钮
0	仅有确定按钮
1	“确定”和“取消”按钮
2	“放弃”、“重试”和“忽略”按钮
3	“是”、“否”和“取消”按钮
4	“是”、“否”按钮
5	“重试”和“取消”按钮

数值	图标
16	“停止”图标
32	问号
48	惊叹号
64	信息 (i) 图标

数值	默认按钮
0	第一个按钮
256	第二个按钮
512	第三个按钮

nDialogBoxType 可以是三个值的和-从上面每个表中选一个值。例如，若 *nDialogBoxType* 为 290（2+32+256），则指定的对话框含有如下特征：

- “放弃”、“重试”或“忽略”按钮。
- 消息框显示问号图标。
- 第二个按钮，“重试”为默认按钮。

cTitleBarText

指定对话框标题栏中的文本。若省略 *cTitleBarText*，标题栏中将显示“Microsoft Visual FoxPro”。

说明

MESSAGEBOX( ) 的返回值标明选取了对话框中的哪个按钮。在含有取消按钮的对话框中，如果按下 ESC 键退出对话框，则与选取取消按钮一样，返回值 (2)。

注意本函数的最短缩写为 MESSAGEB( )。

下表列出了 MESSAGEBOX( ) 对应每个按钮的返回值。

返回值	按钮
1	确定 (OK)
2	取消 (CANCEL)
3	放弃 (ABORT)
4	重试 (RETRY)
5	忽略 (IGNORE)
6	是 (YES)
7	否 (NO)

示例

下面的示例显示了一个用户自定义的对话框。该用户自定义的对话框的标题是

“Record not found. Would you like to search again?”，并且标题栏是“My Application”。

这个用户自定义的对话框含有 Yes 和 No 按钮，还有问号图标，并且第二个按钮 (No) 是默认按钮。当您选定了其中一个按钮时，您的选择会被显示出来。

```
cMessageTitle = 'My Application'
cMessageText = 'Record not found. Would you like to search again?'
nDialogType = 4 + 32 + 256
*  4 = Yes and No buttons
*  32 = Question mark icon
*  256 = Second button is default

nAnswer = MESSAGEBOX(cMessageText, nDialogType, cMessageTitle)

DO CASE
    CASE nAnswer = 6
        WAIT WINDOW 'You chose Yes'
    CASE nAnswer = 7
        WAIT WINDOW 'You chose No'
ENDCASE
```

请参阅

WAIT

MiddleClick 事件

当用户在一个控件上单击一个三键鼠标的中央鼠标时发生。

语法

PROCEDURE Control.MiddleClick
[LPARAMETERS nIndex]

参数描述

nIndex

如果一个控件在一个控件数组中，则本参数唯一标识该控件。

应用于

复选框，组合框，命令按钮，命令组，容器，控件，编辑框，表单，表格，标头，图像，标签，线条，列表框，选项按钮，选项按钮组，页面，页框，形状，微调，文本框，工具栏

请参阅

Click 事件, DblClick 事件, DragDrop 事件, DragOver 事件, DropDown 事件, Enabled 属性, KeyPress 事件, MouseDown 事件, MouseMove 事件, MouseUp 事件, Scrolled 事件, Undock 事件

MIN() 函数

比较一组表达式，并返回其中具有最小值的表达式。

语法

MIN(eExpression1, eExpression2 [, eExpression3 ...])

返值类型

字符型、数值型、货币型、双精度型、浮点型、日期型、日期时间型

参数描述

eExpression1, eExpression2 [, eExpression3 ...]

指定的一组表达式，MIN( )返回其中具有最小值的表达式。所有的表达式必须是相同的类型。

示例

下面的示例用 APPEND BLANK 创建了一个包含随机值的 10 个记录的表，再用 MIN( ) 和 MAX( ) 来显示表中的最大值和最小值。

```
CLOSE DATABASES
CREATE TABLE Random (cValue N(3))
FOR nItem = 1 TO 10 &&增加 10 个记录
    APPEND BLANK
    REPLACE cValue WITH 1 + 100 * RAND( ) &&插入任意值
```

```
ENDFOR

CLEAR
LIST  && 显示数值
gnMaximum = 1  && 初始化最小值
gnMinimum = 100  && 初始化最大值
SCAN
    gnMinimum = MIN(gnMinimum, cValue)
    gnMaximum = MAX(gnMaximum, cValue)
ENDSCAN
? 'The minimum value is: ', gnMinimum  && 显示最小值
? 'The maximum value is: ', gnMaximum  && 显示最大值
```

请参阅

CALCULATE, MAX()

MinButton 属性

指定表单是否有最小化按钮。设计和运行时可用。

语法

Object.MinButton[= lExpr]

参数描述

lExpr

MinButton 属性的设置为：

设置	描述
“真” (.T.)	（默认值）表单具有最小化按钮
“假” (.F.)	表单不具有最小化按钮

说明

最小化按钮能够将表单窗口最小化为图标。

MaxButton、MinButton、BorderStyle 和 ControlBox 的设置值，只有运行时才起作用。在 Microsoft Windows 3.0 或更高版本中，MDI 子表单窗口总是具有最小化按钮，而不管 MinButton 的设置如何。但 MinButton 设置为“假” (.F.) 时并不响应单击，并且对应的最小化命令也不出现在表单的控件框菜单中。

运行时将表单最小化为图标，将产生 Resize 事件。

注意 WindowState 属性反应了窗口的当前状态。把 WindowState 属性设为 1（最小化）时，不管 MinButton 和 BorderStyle 属性的有效设置如何，表单都将最小化。

应用于

表单、_SCREEN

请参阅

BorderStyle 属性，ControlBox 属性，MaxButton 属性，TitleBar 属性，

MinHeight 属性

指定表单可被调整到的最小高度。设计和运行时可用。

语法

Object.MinHeight[= nHeight]

参数描述

nHeight

指定表单可被调整到的最小高度，度量单位由表单的 ScaleMode 属性指定。

说明

当用户从控件菜单中选取“大小”命令或者拖动表单的边框调整表单大小时，表单的高度不会小于 MinHeight 属性中的设置值。

MinHeight 属性的默认设置为 -1；即没有指定最小高度。

应用于

表单, _SCREEN

请参阅

MaxHeight 属性, MaxWidth 属性, MinWidth 属性, ScaleMode 属性

MINUTE() 函数

返回日期时间表达式中的分钟部分。

语法

MINUTE(tExpression)

返值类型

数值型

参数描述

tExpression

指定的日期时间表达式。MINUTE( ) 返回其中的分钟部分。

示例

下面的示例显示了当前时间的分钟部分和指定时间的分钟部分。

```
CLEAR
? MINUTE(DATETIME( ))
? MINUTE({^1998-02-16 10:42a}) && 显示 42
```

请参阅

CTOT( ), DATE( ), DATETIME( ), DTOT( ), HOUR( ), SEC( ), SECONDS( ),

MinWidth 属性

指定表单可调整到的最小宽度。设计和运行时可用。

语法

Object.MinWidth[= nWidth]

参数描述

nWidth

指定表单可调整到的最小宽度，度量单位由表单的 ScaleMode 属性指定。

说明

当用户选择表单控件菜单中的“大小”命令或者拖动表单的边框调整表单时，表单的宽度不会小于 MinWidth 中的设置值。

MinWidth 属性的默认设置为 -1；即没有指定最小宽度。

应用于

Form, _SCREEN

请参阅

MaxHeight 属性, MaxWidth 属性, MinHeight 属性, ScaleMode 属性

MLINE() 函数

以字符串形式返回备注字段中的指定行。

语法

MLINE(MemoFieldName, nLineNumber [, nNumberOfCharacters])

返值类型

字符型

参数描述

MemoFieldName

指定的备注字段名，MLINE( )**从该字段中返回一行。若备注字段是在非当前工作区打开的表中，则应在备注字段名前加上表别名和一个句点。**

nLineNumber

指定从备注字段中返回行的行号，当 *nLineNumber* **为负数或者大于备注字段的行数时，则返回空字符串。**

nNumberOfCharacters

指定从备注字段开始的字符数，从这些字符之后 MLINE( ) **返回指定行的内容。**

_MLINE **系统变量通常可用作** *nNumberOfCharacters*。**每次调用** MLINE( ) **后，** _MLINE

将自动调整。

在从较大备注字段中返回行的递归过程中，把 _MLINE 作为 *nNumberOfCharacters* 将会取得较好效果。

说明

MLINE( ) 将裁剪掉 *nLineNumber* 指定行的后续空格。

备注字段的行长度和行数由 SET MEMOWIDTH 的当前值决定（一行的默认长度为 50 个字符）。遇到回车符时，不再返回更多字符。_WRAP 的当前设置决定备注字段行的显示方式。

在备注字段中查找字符串时，可以使用 ATLINE( ) 或 ATCLINE( ) 返回找到的字符串所在行的行号，然后在 MLINE( ) 中使用这个行号返回备注字段中此行的内容。

示例

下面的示例中，使用两种方法从备注字段中返回行内容。使用 MLINE( ) 在两个循环中返回备注字段的行。注意在第二个循环中使用系统变量 -MLINE 作为 -MLINE( ) 的参数，从而增强了性能。

```
CLEAR
SET TALK OFF
SET MEMOWIDTH TO 50
CLOSE DATABASES
CREATE TABLE tmemo (name c(10), notes m)
APPEND BLANK                    && 添加一个记录
WAIT WINDOW 'Filling memo field - takes several seconds' NOWAIT
*** 填充备注字段 ***
FOR gnOuterLoop = 1 TO 5        && 循环 5 次
    FOR gnAlphabet = 65 TO 75   && 从字符 A 到 H
        REPLACE notes WITH REPLICATE(CHR(gnAlphabet), 10) ;
            + CHR(13) ADDITIVE
```

```
    NEXT
NEXT

*** 显示备注字段的所有行***

STORE MEMLINES(notes) TO gnNumLines  && 备注字段的行数
STORE SECONDS( ) TO gnBegin     && 开始时间
FOR gnCount = 1 TO gnNumLines  &&执行备注字段行数次循环 Z
    ? MLINE(notes, gnCount)    &&显示每一行
NEXT
? STR(SECONDS( ) - gnBegin, 4, 2) + ' seconds'  && 总时间

*** 更好的方法是在-MLINE( )中使用 -MLINE***
*** 显示备注字段的所有行***

WAIT 'Press a key to see the preferred method' WINDOW
CLEAR
STORE 0 TO _MLINE          && 设置 MLINE 为 0
STORE SECONDS( ) TO gnBegin     && 开始时间
FOR count = 1 TO gnNumLines     && 执行备注字段行数次循环
    ? MLINE(notes, 1, _MLINE)    && 显示每一行
NEXT
? STR(SECONDS( ) - gnBegin, 4, 2) + ' seconds'  && 总时间
SET TALK ON
CLOSE DATABASES
ERASE tmemo.dbf
ERASE tmemo.fpt
```

请参阅

ALINES( ), ATCLINE( ), ATLINE( ), COPY MEMO, MEMLINES( ), _MLINE, MODIFY MEMO, SET MEMOWIDTH, _WRAP

_MLINE 系统变量

包含 MLINE( )函数中使用的备注字段偏移量。

语法

_MLINE = nNumberOfCharacters

参数描述

nNumberOfCharacters

指定备注字段偏移量。有关使用 -MLINE 的详细内容和使用的示例，请参阅 MLINE( )。

说明

MLINE( )从备注字段中返回一行文本。MLINE( )将备注字段中行的偏移量保存在 _MLINE 系统变量中。把 _MLINE 作为 MLINE( )函数的第二个数值参数，可以提高 MLINE( )的性能。

_MLINE 的初始默认值为 0。确保在再次使用 MLINE( )函数前，将 _MLINE 的值重置为 0。

请参阅

ALINES( ), MEMLINES( ), MLINE( ), SET MEMOWIDTH

MOD() 函数

用一个数值表达式去除另一个数值表达式，返回余数。

语法

MOD(nDividend, nDivisor)

返值类型

数值型

参数描述

nDividend

指定的被除数。*nDividend* 中的小数位数决定了返回值中的小数位。

nDivisor

指定除数。如果 nDivisor 为正，返回正值，如果 nDivisor 为负，返回负值。

说明

取余函数 MOD() 和 % 操作符返回的值相等。

示例

```
CLEAR
? MOD(36,10) && 显示 6
```

```
? MOD((4*9), (90/9)) && 显示 6
? MOD(25.250,5.0) && 显示 0.250
? IIF(MOD(YEAR(DATE( )), 4) = 0, 'Summer Olympics this year';
    , 'No Summer Olympics this year')
```

请参阅

% 操作符

Modify 方法

打开项目中的一个文件，以便在相应的设计器或编辑器中修改该文件。

语法

Object.Modify([cClassName])

参数描述

cClassName

当 Object 是一个 .vcx 可视类库时，本参数指定要打开修改的可视类的名称。

说明

如果该文件成功打开，则 Modify 方法返回“真” (.T.)。否则返回“假” (.F.)。

当打开该文件之前，发生 QueryModifyFile 事件。如果 QueryModifyFile 事件返回

“假” (.F.)，则不打开该文件，并且 Modify 方法返回“假” (.F.)。如果 QueryModifyFile 方法返回“真” (.T.)，则打开该文件，并且 Modify 方法返回“真” (.T.)。

应用于

文件对象

请参阅

Add 方法, Remove 方法, QueryModifyFile 事件

返回总目录

MODIFY CLASS 命令
MODIFY COMMAND 命令
MODIFY CONNECTION 命令
MODIFY DATABASE 命令
MODIFY FILE 命令
MODIFY FORM 命令
MODIFY GENERAL 命令
MODIFY LABEL 命令
MODIFY MEMO 命令
MODIFY MENU 命令
MODIFY PROCEDURE 命令
MODIFY PROJECT 命令
MODIFY QUERY 命令
MODIFY REPORT 命令
MODIFY SCREEN 命令
MODIFY STRUCTURE 命令
MODIFY VIEW 命令
MODIFY WINDOW 命令

MONTH（）函数
MOUSE 命令
MouseDown 事件
MouseIcon 属性
MouseMove 事件
MousePointer 属性
MouseUp 事件
MouseWheel 事件
Movable 属性
Move 方法
MOVE POPUP 命令
MOVE WINDOW 命令
Moved 事件
MoverBars 属性
MRKBAR（）函数
MRKPAD（）函数
MROW（）函数
MTON（）函数
MultiSelect 属性
MWINDOW（）函数
Name 属性

MODIFY CLASS 命令

打开类设计器，让用户修改已有的类定义或创建新的类定义。

语法

MODIFY CLASS ClassName [OF ClassLibraryName1]
[AS cBaseClassName [FROM ClassLibraryName2]]
[METHOD MethodName] [NOWAIT] [SAVE]

参数描述

ClassName

指定要修改或创建的类定义名称。

OF ClassLibraryName1

指定含有这个类定义的 .VCX 可视类库名称。若创建新类，并且 .VCX 可视类库已经存在，那么把类定义添加到库中。

VCX 是可视类库的默认扩展名。当指定的可视类库扩展名不是 .VCX 时，应指定扩展名。

如果指定的 .VCX 可视类库位于当前 SET CLASSLIB 命令设置的搜索列表中，那么将把

这个类库从搜索列表中移去。

AS cBaseClassName

指定类定义的基类。*cBaseClassName* 可以是 Visual FoxPro 基类中除 Column、Cursor、DataEnvironment、Header、Page 和 Relation 之外的任何类。

若省略 AS *cBaseClassName* 参数，则类定义派生于 Visual FoxPro 的 FormSet 基类。

FROM ClassLibraryName2

指定 .VCX 可视类库名，该库中包含 *cBaseClassName* 指定的用户自定义类。

METHOD MethodName

指定在“类设计器”中打开“代码窗口”的事件或方法。METHOD 子句让您在“类设计器”中立即开始编辑事件或方法代码。

MethodName 支持 Visual FoxPro 对象语法。例如，用下面的命令可以立即编辑可视类库 MyClassLibrary 中 MyClass 类的文本框 txtFirstName 的 Click 事件代码：

```
MODIFY CLASS MyClass OF MyClassLibrary;
    METHOD txtFirstName.Click
```

如果只在 METHOD 子句中包含了一个事件或方法，则为类的事件或方法打开“代码”窗口。例如，用下面的命令可以立即编辑可视类库 MyClassLibrary 中 MyClass 类的 Click 事件代码：

```
MODIFY CLASS MyClass OF MyClassLibrary METHOD Click
```

NOWAIT

在打开类设计器后程序继续执行。程序不必等待类设计器关闭，而是继续执行 MODIFY CLASS NOWAIT 行后面的语句。若在程序中使用 MODIFY CLASS 命令时省略 NOWAIT 子句，那么打开类设计器后，程序暂停执行，直到关闭类设计器。

NOWAIT 仅在程序中才有效。当在命令窗口中发出 MODIFY CLASS 时，该项无效。如果包含了 NOWAIT 和 METHOD 子句，一定要将 NOWAIT 放在 METHOD 子句之前，否则会忽略 NOWAIT。

SAVE

激活另一窗口后，仍保持类设计器打开。如果省略 SAVE，则当另一窗口激活时，类设计器随之关闭。从命令窗口中发出包含 SAVE 子句的命令时，将无此作用。

说明

可利用 MODIFY CLASS 修改已有的类定义或创建新的类定义，并存入一个 .VCX 可视类库中。可以用 SET CLASSLIB 打开 .VCX 可视类库，并访问 .VCX 可视类库中的类定义。

请参阅

ADD CLASS, CREATE CLASS, CREATE CLASSLIB, RELEASE CLASSLIB, SETCLASSLIB

MODIFY COMMAND 命令

打开一个编辑窗口，从中可以修改或创建程序文件。

语法

```
MODIFY COMMAND [FileName | ?]
  [NOEDIT]
  [NOMENU]
  [NOWAIT]
  [RANGE nStartCharacter, nEndCharacter]
  [[WINDOW WindowName1]
  [IN [WINDOW] WindowName2 | IN SCREEN]]
  [AS nCodePage]
  [SAME]
  [SAVE]
```

参数描述

FileName

指定打开或创建的程序文件名。若不指定新建程序文件的扩展名， Visual FoxPro **自动指定** .PRG **为扩展名。** MODIFY COMMAND **支持含有星号（***）

和问号（？）通配符的文件梗概。名称与这个文件梗概匹配的每一个文件都在编辑窗口中打开。

若省略文件名，将给打开的编辑窗口赋以一个初始名称“程序 1.PRG”。当关闭编辑窗口时，可以用另外的文件名保存该文件。

?

显示“打开”对话框。可以从中选择一个已有程序或键入要创建的新程序名。

NOEDIT

指定不能对程序文件作更改，但是可以查看程序文件或者将程序文件复制到剪贴板中。

NOMENU

从系统菜单栏中移去“格式”菜单标题，可以防止用户更改字体、字体大小、行间距和行缩进。

NOWAIT

在打开编辑窗口后继续程序的执行。程序不必等待编辑窗口关闭，而是继续执行命令 MODIFY COMMAND NOWAIT 所在行的后继语句行。若在程序中使用 MODIFY COMMAND 时省略 NOWAIT 子句，打开编辑窗口后，程序暂停执行，直到关闭编辑窗口。

NOWAIT 仅在程序中有效。在命令窗口中发出 MODIFY COMMAND NOWAIT 时，无效。

当用一条 MODIFY COMMAND 命令打开多个窗口时，隐含使用 NOWAIT，例如：

MODIFY COMMAND *.PRG.

RANGE nStartCharacter, nEndCharacter

指定打开编辑窗口时，选择的字符范围。从 nStartCharacter 位置开始一直到（但不包括） nEndCharacter；若 nStartCharacter 等于 nEndCharacter，则不选择字符，光标停在 nStartCharacter 指定的位置处。

WINDOW WindowName1

指定一个窗口名，编辑窗口继承其特性。例如，如果创建该窗口时 DEFINE WINDOWS 含有 FLOAT 选项，则编辑窗口可移动。该窗口不必为活动或可见的，但必须是已定义的。

IN [WINDOW] WindowName2

指定编辑窗口打开时所在的父窗口名。编辑窗口不继承父窗口的特性并且不能移出父窗口之外。父窗口移动时，编辑窗口随之移动。

要访问编辑窗口，必须先用 DEFINE WINDOW 定义父窗口，并且父窗口必须是可见的。

IN SCREEN

把编辑窗口放入父窗口后，在 Visual FoxPro 主窗口中打开它。通过包含 IN WINDOW 子句，可以将编辑窗口放在父窗口中。

AS nCodePage

自动转换在其他 Visual FoxPro 平台上所创建程序文件中的重音字符。数值表达式 nCodePage 指定 Visual FoxPro 平台的代码页，在此代码页上创建程序文

件。文件以此代码页保存，除非从“文件”菜单中选择“另存为”命令，将文件以其他的代码页保存。

SAME

防止编辑窗口成为活动窗口。若编辑窗口为隐藏状态，它将被显示但不会成为活动窗口。

SAVE

在激活其他窗口后，保持编辑窗口打开。若省略 SAVE，则其他窗口激活后将关闭编辑窗口。在命令窗口中发出命令时包含 SAVE，则无此作用。

说明

修改程序文件之后，更新过的文件将写到磁盘上。选取“编辑”菜单上的“属性”，若在其中的“编辑属性”对话框中选取“制作备份”复选框，则创建具有 .BAK 扩展名的备份文件。在早期版本的 FoxPro 中，是通过在“编辑”菜单中选取“参数选择”项，并在出现的“选择”对话框中选取“备份”复选框，创建一个带 .BAK 扩展名的备份文件。

除非在配置文件中用 TEDIT 指定一个外部编辑器，否则，一般使用 Visual FoxPro 的内置编辑器。

请参阅

*, &&, DO, MODIFY FILE, NOTE

MODIFY CONNECTION 命令

显示连接设计器，让您能够交互地修改当前数据库中已有的命名连接。

语法

MODIFY CONNECTION [ConnectionName | ?]

参数描述

ConnectionName

指定要修改的连接名。

?

显示“打开”对话框，从中可以选取已有的命名连接进行修改。

说明

若省略可选参数，将显示“打开”对话框，从中可以指定已有的命名连接进行修改。在选定要修改的命名连接后将打开连接设计器。

示例

下面的示例假定 ODBC 数据源 MyFoxSQLNT 可用，数据源的用户标识为“sa”。先打开 testdata 数据库，创建名为 Myconn 的连接，然后用 MODIFY CONNECTION 来显示连接设计器并修改连接。

CLOSE DATABASES

```
OPEN DATABASE (HOME(2) + 'data\testdata')
CREATE CONNECTION Myconn DATASOURCE "MyFoxSQLNT" USERID "sa"
MODIFY CONNECTION Myconn && 显示数据库中的命名连接
```

请参阅

CREATE CONNECTION, DELETE CONNECTION, OPEN DATABASE, RENAME CONNECTION

MODIFY DATABASE 命令

打开数据库设计器，让您能够交互地修改当前数据库。

语法

```
MODIFY DATABASE [DatabaseName | ?]
  [NOWAIT] [NOEDIT]
```

参数描述

DatabaseName

指定要修改的数据库名。

?

显示“打开”对话框，可从中选取数据库进行修改。

NOWAIT

在打开数据库设计器后继续程序的执行。程序不必等待数据库设计器关闭，而是继续执行 MODIFY DATABASE NOWAIT 之后的程序行。若在程序中使用 MODIFY DATABASE 时省略 NOWAIT，则打开数据库设计器后，程序暂停执行，直到关闭数据库设计器。

NOWAIT 仅在程序中有效。在命令窗口中发出 MODIFY DATABASE NOWAIT 时，无效。

NOEDIT

禁止修改数据库。

说明

有关用“数据库设计器”交互式地修改数据库的详细内容，请参阅帮助中的“数据库设计器”和“数据库设计器工具栏”，与《用户指南》的第三章“将表加入数据库”。

示例

下面示例在数据库设计器中显示 testdata 数据库中的表。

```
CLOSE DATABASES
SET PATH TO (HOME(2) + 'data\')    && 设置数据库的路径。

MODIFY DATABASE testdata    && 打开 testdata 数据库
```

请参阅

ADD TABLE, CLOSE DATABASES, CREATE DATABASE, DBC（）, DBGETPROP（）, DBSETPROP（）, DELETE DATABASE, DISPLAY TABLES, OPEN DATABASE, REMOVE TABLE

MODIFY FILE 命令

打开编辑窗口，从中可以修改或创建文本文件。

语法

```
MODIFY FILE [FileName | ?]
  [NOEDIT]
  [NOMENU]
  [NOWAIT]
  [RANGE nStartCharacter, nEndCharacter]
  [[WINDOW WindowName1]
  [IN [WINDOW] WindowName2 | IN SCREEN]]
  [AS nCodePage]
  [SAME]
  [SAVE]
```

参数描述

FileName

指定文本文件的文件名。若未给新文本文件指定扩展名，Visual FoxPro自动指定扩展名为.TXT。MODIFY FILE支持包含星号（*）和问号（?）通配符的文件梗概。每个匹配的文本文件都将出现在打开的编辑窗口中。

若省略文件名，打开的编辑窗口将赋给初始文件名1。当关闭编辑窗口时，可以用另外一个文件名保存该文件。

?

显示打开对话框，从中可以选取一个文本文件。

NOEDIT

指定该文本文件不能更改，但可以查看，或者将文本文件复制到剪贴板上。

NOMENU

将“格式”菜单标题从系统菜单栏中移去，可防止更改字体、字体大小、行间距和缩进。

NOWAIT

在编辑窗口打开后继续程序执行。程序不必等待关闭编辑窗口，而是继续执行MODIFY FILE NOWAIT之后的程序行。如果在程序中发出MODIFY FILE命令时省略NOWAIT，编辑窗口打开后，暂停执行程序直至编辑窗口关闭为止。

NOWAIT仅在程序中有效。在命令窗口中发出MODIFY FILE NOWAIT时，无效。

如果用单个 MODIFY FILE 命令打开多个编辑窗口，将隐含地发生 NOWAIT。例如：
MODIFY FILE *.TXT

RANGE nStartCharacter, nEndCharacter
指定打开编辑窗口时选定的字符范围。字符范围从 nStartCharacter 指定位置到（但不包括） nEndCharacter 指定位置。如果 nStartCharacter 等于 nEndCharacter，则没有选定值，光标定位于 nStartCharacter 指定的位置。

WINDOW WindowName1
指定一个窗口，编辑窗口继承它的特性。例如，如果窗口是由 DEFINE WINDOW 的 FLOAT 子句创建的，那么编辑窗口能够移动。窗口不必是活动的或可见的，但必须是已定义的。

IN [WINDOW] WindowName2
指定一个父窗口，编辑窗口从中打开。编辑窗口不继承父窗口的特性并且不能移出父窗口之外。如果父窗口移动，编辑窗口随之移动。
要访问编辑窗口，必须先用 DEFINE WINDOW 命令定义父窗口，并且父窗口是可见的。

IN SCREEN
把编辑窗口放入父窗口后，在 Visual FoxPro 主窗口中打开它。包含 IN WINDOW 子句可以把编辑窗口放入父窗口。

AS nCodePage
自动转换其他 Visual FoxPro 平台上创建的文本文件中的重音字符。数值表达

式 nCodePage 指定了文本文件创建时所在的 Visual FoxPro 平台的代码页。文件以该代码页保存，除非选择“文件”菜单中的“另存为”选项将文件以其他代码页保存。

SAME

防止编辑窗口成为活动窗口。如果编辑窗口被隐藏，它仍显示但不成为活动窗口。

SAVE

激活另外一个窗口后保持编辑窗口打开。若省略 SAVE，激活另外一个窗口后，编辑窗口将关闭。从命令窗口发出命令时包含 SAVE 将不起作用。

说明

对文本文件进行修改后，更新的文件将写到磁盘上。在 Visual FoxPro 中，当在“工具”菜单中选择“选项”时，可以在“选项”对话框的“编辑”选项卡中选定“制作备份”复选框，从而创建一个带有 .BAK 扩展名的备份文件。在 FoxPro 早期版本中，当在“编辑”菜单中选择“参数选择”时，可以在“参数选择”对话框中选定“备份”复选框，从而创建一个带有 .BAK 扩展名的备份文件。

除非在配置文件中用 TEDIT 指定一个外部编辑器，否则，使用 Visual FoxPro 编辑器。

请参阅

*, &&, DO, MODIFY COMMAND, NOTE

MODIFY FORM 命令

打开表单设计器，从中可以修改或创建表单。

语法

MODIFY FORM
[FormName | ?] [METHOD MethodName]
[NOENVIRONMENT] [NOWAIT] [SAVE]
[[WINDOW WindowName1]
[IN [WINDOW] WindowName2 | IN SCREEN]]

参数描述

FormName

指定表单的文件名。若未给文件名指定扩展名， Visual FoxPro **自动指定扩展名为** .SCX**。**

?

显示“打开”对话框，从中可以选取已有的表单或者输入新建表单的名称。

METHOD MethodName

指定在“表单设计器”中打开“代码窗口”的事件或方法。 METHOD **子句让您在“表单设计器”中立即开始编辑事件或方法代码。**

MethodName 支持 Visual FoxPro 对象语法。例如，用下面的命令可以立即编辑表单 frmAddress 中文本框 txtFirstName 的 Click 事件代码：

```
MODIFY FORM frmAddress METHOD txtFirstName.Click
```

如果只在 METHOD 子句中包含了一个事件或方法，则为表单的事件或方法打开“代码”窗口。例如，用下面的命令可以立即编辑表单 frmAddress 的 Click 事件代码：

```
MODIFY FORM frmAddress METHOD Click
```

NOENVIRONMENT

包含此子句是为了提供与 2.x 屏幕的向后兼容性，用以防止恢复与屏幕一起保存的环境。

在 Visual FoxPro 中，通过把 AutoOpenTables 属性设置为“真”(.T.)，可以恢复与 Visual FoxPro 表单相联系的数据环境。要确保释放表单后关闭表单环境，可以把数据环境 AutoCloseTables 属性设置为“真”(.T.)。此属性的默认设置值为“真”(.T.)。

创建或修改表单时，可以将当前 Visual FoxPro 数据环境与表单定义文件一起保存。保存数据环境将在表单定义表中放置附加记录，这些记录指出所有打开的表和索引文件、索引顺序以及任何表间的关系。

NOWAIT

在表单设计器打开后继续程序执行。程序不必等待关闭表单设计器，而是继续执行 MODIFY FORM NOWAIT 之后的程序行。如果在程序中发出 MODIFY FORM 命令时省略 NOWAIT，表单设计器打开后，暂停程序执行，直至表单

设计器关闭为止。

NOWAIT 仅在程序中有效。在命令窗口中发出 MODIFY FORM NOWAIT 时无效。如果包含了 NOWAIT 和 METHOD 子句，一定要将 NOWAIT 放在 METHOD 子句之前，否则会忽略 NOWAIT。

SAVE

在程序中发出。指出激活另外一个窗口后保持表单设计器。从命令窗口发出时，SAVE 将不起作用。

WINDOW WindowName1

指定一个窗口，表单设计器采用它的特性。例如，如果窗口是由 DEFINE WINDOW 的 FLOAT 子句创建的，那么表单设计器能够移动。窗口不必是活动的或可见的，但必须是已定义的。

表单设计器的默认大小可能比它的特性窗口大。这种情况下，表单设计器仍然采用该窗口的特性。表单设计器的左上角放置在特性窗口左上角坐标处，但其尺寸超出窗口的边界。

IN [WINDOW] WindowName2

指定一个父窗口，表单设计器从中打开。表单设计器不继承父窗口的特性并且不能移出父窗口之外。如果父窗口移动，表单设计器随之移动。

要访问表单设计器，必须先用 DEFINE WINDOW 命令定义父窗口，并且父窗口是可见的。

IN SCREEN

把表单设计器放入父窗口后，在 Visual FoxPro 主窗口中打开它。包含 IN

WINDOW 子句可以把表单设计器放在父窗口中。

说明

发出不带有任何参数的 MODIFY FORM 命令，将显示打开对话框。关闭表单设计器时可以用其他名称保存表单。

示例

下面示例打开了表单设计器中的计时器 (SWATCH.SCX) 控制示例。

```
MODIFY FORM (HOME(2) + 'solution\controls\timer\swatch.scx')
```

请参阅

COMPILE FORM, CREATE FORM, DO FORM

MODIFY GENERAL 命令

在编辑窗口中打开当前记录中的通用字段。

语法

```
MODIFY GENERAL GeneralField1 [, GeneralField2 ...]
  [NOMODIFY]
  [NOWAIT]
  [[WINDOW WindowName1]
```

[IN [WINDOW] WindowName2 | IN SCREEN]]

参数描述

GeneralField1 [, GeneralField2 ...]

指定打开的通用字段名称。若要打开非当前工作区中表的通用字段的编辑窗口，可以在字段名称中包含表别名。通过包含逗号分隔的通用字段列表，可以打开当前记录中的多个通用字段。

NOMODIFY

指定不能更改通用字段中包含的 OLE 对象，但可以查看，或者将 OLE 对象复制到剪贴板上。

NOWAIT

在通用字段编辑窗口打开后继续程序执行。程序不必等待关闭编辑窗口，而是继续执行 MODIFY GENERAL NOWAIT 之后的程序行。如果在程序中发出 MODIFY GENERAL 时省略 NOWAIT，编辑窗口打开后，暂停执行程序，直至编辑窗口关闭为止。

NOWAIT 仅在程序中有效。在命令窗口中发出 MODIFY GENERAL 时，无效。

WINDOW WindowName1

指定一个窗口，通用字段编辑窗口采用它的特性。例如，如果窗口是由 DEFINE WINDOW 的 FLOAT 子句创建的，那么通用字段编辑窗口能够移动。窗口不必是活动的或可见的，但必须是已定义的。

IN [WINDOW] WindowName2

指定一个父窗口，通用字段编辑窗口从中打开。通用字段编辑窗口不继承父窗

口的特性并且不能移出父窗口之外。如果父窗口移动，通用字段编辑窗口随之移动。

要访问通用字段编辑窗口，必须先用 DEFINE WINDOW 命令定义父窗口，并且父窗口是可见的。

IN SCREEN

把通用字段窗口放入父窗口后，在 Visual FoxPro 主窗口中打开它。包含 IN WINDOW 子句可以把通用字段编辑窗口放入父窗口。

说明

当打开编辑窗口后，不能插入、修改或者删除一个 OLE 对象。

有关 Visual FoxPro 中的 OLE 对象的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第十六章“添加 OLE”中的”在表中添加 OLE 对象”。

请参阅 **@ ... SAY-Pictures & OLE Objects,APPEND GENERAL**

MODIFY LABEL 命令

打开标签设计器，从中可以修改或创建标签。

语法

MODIFY LABEL [FileName | ?]

[[WINDOW WindowName1]
[IN [WINDOW] WindowName2 | IN SCREEN]]
[NOENVIRONMENT]
[NOWAIT]
[SAVE]

参数描述

FileName

指定标签的文件名。若未给文件名指定扩展名，Visual FoxPro**自动指定扩展名为** .LBX**；如果指定的文件不存在或者未找到，则创建一个新标签文件。**

?

显示“打开”对话框，从中可以选取已有的标签或者输入新建标签的名称。

WINDOW WindowName1

指定一个窗口，标签设计器采用它的特性。例如，如果窗口是由 DEFINE WINDOW **的** FLOAT **子句创建的，那么标签设计器能够移动。窗口不必是活动的或可见的，但必须是已定义的。**

标签设计器的默认尺寸可能比它的特性窗口大。这种情况下，标签设计器仍然采用它所在的窗口的特性。标签设计器的左上角放置在特性窗口左上角坐标处，但其尺寸超出窗口的边界。

IN [WINDOW] WindowName2

指定一个父窗口，标签设计器从中打开。标签设计器不继承父窗口的特性并且不能移出父窗口之外。如果父窗口移动，标签设计器随之移动。

要访问标签设计器，必须先用 DEFINE WINDOW 命令定义父窗口，并且父窗口是可见的。

IN SCREEN

把标签设计器放入父窗口后，在 Visual FoxPro 主窗口中打开它。包含 IN WINDOW 子句可以把标签设计器放在父窗口中。

NOENVIRONMENT

包含此子句是为了提供与 2.x 屏幕的向后兼容性。用以防止恢复与屏幕一起保存的环境。

通过把 AutoOpenTables 属性设置为“真”(.T.)，可以恢复与 Visual FoxPro 标签相联系的数据环境。要确保打印标签结束后关闭标签环境，可以把数据环境 AutoCloseTables 属性设置为“真”(.T.)。此属性的默认设置值为“真”(.T.)。

创建或修改标签时，可以将当前 Visual FoxPro 数据环境与标签定义文件一起保存。保存 Visual FoxPro 数据环境将在标签定义表中放置附加记录，这些记录指出所有打开的表和索引文件、索引顺序以及任何表间的关系。

NOWAIT

在标签设计器打开后继续程序执行。程序不必等待关闭标签设计器，而是执行 CREATE LABEL NOWAIT 之后的程序行。如果在程序中发出 CREATE LABEL 时省略 NOWAIT，标签设计器打开后，暂停执行程序，直至标签设计器关闭为止。

在命令窗口中发出 CREATE LABLE 时，无效。

SAVE

激活另外一个窗口后保持标签设计器打开。若省略 SAVE，激活另外一个窗口后，标签设计器将关闭。从命令窗口发出时，SAVE 将不起作用。

请参阅

CREATE LABEL, LABEL

MODIFY MEMO 命令

打开当前记录中备注字段的编辑窗口。

语法

```
MODIFY MEMO MemoField1 [, MemoField2 ...]
  [NOEDIT]
  [NOMENU]
  [NOWAIT]
  [RANGE nStartCharacter, nEndCharacter]
  [[WINDOW WindowName1]
  [IN [WINDOW] WindowName2 | IN SCREEN]]
```

[SAME]

[SAVE]

参数描述

MemoField1 [, MemoField2 ...]

指定要编辑的备注字段名。若要打开非当前工作区中表备注字段的编辑窗口，可在字段名称中包含表别名。

NOEDIT

指定打开的备注字段不能被更改，但是可以查看，或者将备注字段复制到剪贴板上。

NOMENU

从系统菜单栏中移去“格式”菜单标题，可以防止更改字体、字体大小、行间距和缩进。

NOWAIT

在编辑窗口打开后继续程序执行。程序不必等待关闭编辑窗口，而是继续执行 MODIFY MEMO NOWAIT 之后的程序行。如果在程序中发出 MODIFY GENERAL 时省略 NOWAIT，编辑窗口打开后，暂停执行程序，直至编辑窗口关闭为止。

NOWAIT 仅在程序中有效。在命令窗口中发出 MODIFY MEMO 时，无效。

RANGE nStartCharacter, nEndCharacter

指定打开编辑窗口时，选定的字符范围。字符范围从 *nStartCharacter* 指定位置到（但不包括） *nEndCharacter* 指定位置；若 *nStartCharacter* 等于

nEndCharacter，则不选择字符，光标定位于 *nStartCharacter* 指定的位置处。

WINDOW WindowName1

指定一个窗口，编辑窗口采用它的特性。例如，如果窗口是由 DEFINE WINDOW 的 FLOAT 子句创建的，那么编辑窗口能够移动。窗口不必是活动的或可见的，但必须是已定义的。

IN [WINDOW] WindowName2

指定一个父窗口，编辑窗口从中打开。编辑窗口不继承父窗口的特性并且不能移出父窗口之外。如果父窗口移动了，编辑窗口随之移动。

要访问编辑窗口，必须先用 DEFINE WINDOW 命令定义父窗口，并且父窗口是可见的。

IN SCREEN

把编辑窗口放入父窗口后，在 Visual FoxPro 主窗口中打开它。包含 IN WINDOW 子句可以把编辑窗口放在父窗口中。

SAME

防止编辑窗口成为活动窗口。如果编辑窗口隐藏，它仍显示但不活动。

SAVE

激活另外一个窗口后保持编辑窗口打开。若省略 SAVE，激活另外一个窗口后，编辑窗口将关闭。从命令窗口发出时，SAVE 将不起作用。

说明

在编辑窗口中，可以查看或者更改备注字段的内容。

对于在网络上以共享方式打开的表，一旦开始编辑某一记录的备注字段即锁定该记录。

注意，在发布的运行时刻应用程序中，废止备注字段编辑窗口中的语法着色。

若要在开发时启用编辑窗口中的语法着色

1. 在编辑窗口中单击鼠标右键，从快捷菜单中选择“属性”命令。
2. 选中“语法着色”复选框。

示例

下面的示例在编辑窗口中打开了 employee 表中第一条记录的 notes 备注字段，并且突出显示选定范围的字符。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE employee && 打开 Employee 表

MODIFY MEMO NOTES NOEDIT RANGE 1,10 && 选定前 10 个字符
USE
```

请参阅

CLOSE MEMO

MODIFY MENU 命令

打开菜单设计器，从中可以修改或创建菜单系统。

语法

```
MODIFY MENU [FileName | ?]
  [[WINDOW WindowName1]
  [IN [WINDOW] WindowName2 | IN SCREEN]]
  [NOWAIT]
  [SAVE]
```

参数描述

FileName

指定菜单的文件名。如果没有指定文件的扩展名， Visual FoxPro **自动指定扩展名为** .MNX**。**

?

显示“打开”对话框，从中可以选择一个已存在的菜单文件，或者输入要创建的新菜单名。

WINDOW WindowName1

指定一个窗口，菜单设计器采用它的特性。例如，如果该窗口是由 DEFINE

WINDOW 命令的 FLOAT 选项创建的，则菜单设计器可以移动。此窗口不必是活动的或可见的，但必须是已定义的。

IN [WINDOW] WindowName2

指定一个父窗口，菜单设计器从中打开。菜单设计器不具有父窗口的特性也不能移出父窗口。如果父窗口移动，菜单设计器随之移动。

要访问菜单设计器，必须首先用 DEFINE WINDOW 命令定义父窗口，并且父窗口必须是可见的。

IN SCREEN

把菜单设计器放入父窗口后，在 Visual FoxPro 主窗口中打开它。包含 IN WINDOW 子句，可以把菜单设计器放入一个父窗口。

NOWAIT

打开菜单设计器之后继续执行程序。程序不必等待关闭菜单设计器，而是继续执行 MODIFY MENU NOWAIT 命令后面的程序行。如果在程序中发出 MODIFY MENU 命令时不含有 NOWAIT 子句，打开菜单设计器后，程序暂停执行，直到关闭菜单设计器为止。

NOWAIT 子句只在程序中有效。从命令窗口发出 MODIFY MENU NOWAIT 时，无效。如果在“命令”窗口中发出 MODIFY MENU 时，没有菜单名称和包含 NOWAIT，则不能显示“打开”对话框。可用“新建菜单”对话框指定所创建的菜单的类型（标准或快捷）。

SAVE

在激活另一个窗口之后，菜单设计器仍保持打开。如果省略 SAVE，在激活另

一个窗口之后，将关闭菜单设计器。从命令窗口发出 MODIFY MENU 命令时，SAVE 子句不起作用。

说明

有关创建菜单的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第十一章“设计菜单与工具栏”中的“创建菜单系统”。

请参阅

DEFINE BAR, DEFINE MENU, DEFINE PAD, DEFINE POPUP, SET SYSMENU

MODIFY PROCEDURE 命令

打开 Visual FoxPro 文本编辑器，可在其中为当前数据库创建新的内部存储过程，或修改数据库已有的内部存储过程。

语法

MODIFY PROCEDURE

说明

在创建或修改内部存储过程之前必须打开一个数据库。内部存储过程通常是在删除、插

入或更新触发器中指定的，可以用 CREATE TRIGGER 命令为一个数据库创建触发器。当前数据库的内部存储过程可以像其他已打开的过程文件或程序中的 Visual FoxPro 过程一样执行。有关 Visual FoxPro 搜索过程的顺序和位置的说明，请参阅 PROCEDURE。

示例

下面的示例打开 testdata 数据库，并用 MODIFY PROCEDURE 命令打开 Visual FoxPro 文本编辑器，在编辑器中可以创建新的内部存储过程或者修改已有的内部存储过程。

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'Data\testdata')

MODIFY PROCEDURE &&打开 Visual FoxPro 文本编辑器
```

请参阅

CREATE TRIGGER, DISPLAY PROCEDURES, OPEN DATABASE, PROCEDURE

MODIFY PROJECT 命令

打开项目管理器，从中可以修改或创建一个项目文件。

语法

```
MODIFY PROJECT [FileName | ?]
  [IN SCREEN] [NOWAIT] [SAVE]
  [NOSHOW] [NOPROJECTHOOK]
```

参数描述

FileName

指定项目的文件名。如果未指定文件的扩展名，Visual FoxPro 自动指定扩展名为 .PIX。

?

显示“打开”对话框，从中可以打开一个已存在的项目文件，或者输入要创建的新项目名称。

IN SCREEN

把项目管理器放入父窗口之后，在 Visual FoxPro 主窗口中打开项目管理器。包含 IN WINDOW 子句可以把项目管理器放进一个父窗口。

NOWAIT

打开项目管理器之后继续执行程序。程序不必等待关闭项目管理器，而是继续执行 MODIFY PROJECT NOWAIT 命令后面的程序行。如果在程序中发出 MODIFY PROJECT 命令时不含有 NOWAIT 子句，打开项目管理器后，程序暂停执行，直到关闭项目管理器为止。

NOWAIT 子句只在程序中起作用。从命令窗口发出 MODIFY PROJECT 时，无效。

SAVE

在激活另一个窗口之后，仍然保持项目管理器打开。如果省略 SAVE 子句，在激活另一个窗口之后，将关闭项目管理器。从命令窗口发出 MODIFY PROJECT 命令时，SAVE 子句不起作用。

NOSHOW

指定当打开项目管理器（Visible 属性设置为“假” (.F.)）时，隐藏它。为了显示项目管理器，可将项目管理器的 Visible 属性设置为“真” (.T.)。

NOSHOW 允许您在项目管理器中显示一个项目之前管理它。注意，为了避免与 NOSHADOW 关键字混淆，不可以将 NOSHOW 缩减为少于 5 个字符。

NOPROJECTHOOK

指定当打开项目管理器时，不创建一个项目管理器对象。对于由项目管理器挂接程序通过编程管理的项目，可以包含 NOPROJECTHOOK 关键字。

说明

在一个项目中，为应用程序指定它所需要的所有源文件，Visual FoxPro 将可以确保用最新的源文件生成应用程序。

项目文件是一个表，它跟踪所有的源文件（如程序、表单、菜单、库、报表、标签、表、索引和格式文件）。项目文件也跟踪文件之间的依赖、引用和连接关系。项目文件扩展名为 .PIX，相关备注文件的扩展名为 .PJT。发出不带任何参数的 MODIFY PROJECT 命令会显示“打开”对话框。

注意 项目中不能包含共享库 (.fll、 .mlb、 .dll 和 CFM 文件)。

一个项目文件有 .pjx 文件扩展名和一个有 .pjt 扩展名的相关联备注文件。

不带任何参数描述的 MODIFY PROJECT 命令显示“打开”对话框。

有关项目管理器的详细内容，请参阅《Microsoft Visual FoxPro 6.0 中文版程序员指南》的第十三章“编译应用程序”。

请参阅

BUILD APP, BUILD EXE, BUILD PROJECT, COMPILE, CREATE PROJECT

MODIFY QUERY 命令

打开查询设计器，从中可以修改或创建一个查询。

语法

```
MODIFY QUERY [FileName | ?]
  [[WINDOW WindowName1]
```

[IN SCREEN]
[NOWAIT]
[SAVE]
[AS nCodePage]

参数描述

FileName

指定查询的文件名。如果未指定扩展名，Visual FoxPro自动指定扩展名为 .QPR。

?

显示打开对话框，从中可以选择一个已存在的查询，或者输入要创建的新查询的名称。

WINDOW WindowName1

指定“查询设计器”接受的特性。例如，如果用 DEFINE WINDOW 的 FLOAT 选项创建窗口，就可以移动“查询设计器”。窗口不一定是活动或可见的，但必须被定义。

IN SCREEN

把查询设计器放入父窗口之后，在 Visual FoxPro 主窗口中打开查询设计器。包含 IN WINDOW 子句，可以把查询设计器放进一个父窗口。

NOWAIT

打开查询设计器之后继续执行程序。程序不必等待查询设计器关闭，而是继续执行 MODIFY QUERY NOWAIT 命令后面的程序行。如果在程序中发出

MODIFY QUERY 命令时不含有 NOWAIT 子句，打开查询设计器后，程序暂停执行，直到关闭查询设计器为止。

NOWAIT 子句只在程序中起作用。从命令窗口发出 MODIFY QUERY NOWAIT 时，无效。

SAVE

激活另一个窗口之后，仍然保持查询设计器打开。如果省略 SAVE，在激活另一个窗口之后，将关闭查询设计器。从命令窗口发出 MODIFY QUERY 命令时，SAVE 子句不起作用。

AS nCodePage

指定查询的代码页。当查询由 Visual FoxPro 的非当前代码页创建时，需要包含 AS nCodePage。打开查询时，Visual FoxPro 自动把查询转换到 Visual FoxPro 当前代码页。可以用 GETCP() 函数为 nCodePage 显示“代码页”对话框，从中可以为查询指定一个代码页。

查询关闭时，它以原始代码页保存。

如果省略 AS *nCodePage* 子句或者 *nCodePage* 等于 0，则查询并不转换到 Visual FoxPro 当前代码页。如果不支持 *nCodePage* 指定的值，Visual FoxPro 产生错误信息。

在 Visual FoxPro 中，可以把查询添加进一个项目，也可以在项目容器中指定查询的代码页。项目容器跟踪查询的代码页。但是，如果用 MODIFY QUERY 命令在项目容器以外打开查询，则应该包含 AS *nCodePage* 子句来指定查询的代码页。

说明

有关创建查询的详细内容，请参阅帮助中的《用户指南》第四章“检索数据”中的“创

建查询”。

如果发出不带任何参数的 MODIFY QUERY 命令，将显示打开对话框。如果从中选择了新建按钮，则指定查询名为“查询 1”。关闭查询设计器时可以用另外的名字保存该查询。

创建一个查询后，它将存贮为带 .QPR 扩展名的 Visual FoxPro 程序文件。可以用 DO 命令执行一个查询程序，但查询文件名必须包含 .QPR 扩展名。

请参阅

CREATE QUERY, DO, GETCP（）

MODIFY REPORT 命令

打开报表设计器，从中可以修改或创建报表。

语法

```
MODIFY REPORT [FileName | ?]
  [[WINDOW WindowName1]
  [IN [WINDOW] WindowName2 | IN SCREEN]]
  [NOENVIRONMENT]
  [NOWAIT]
```

会自动地赋给 .frx 扩展名。

?

显示“打开”对话框，从中可以选择一个已存在的报表文件，或者输入要创建的新报表名称。

WINDOW WindowName1

指定一个窗口，报表设计器采用它的特性。例如，如果窗口是由 DEFINE WINDOW 命令的 FLOAT 选项创建的，则报表设计器可以移动。窗口不必是活动的或可见的，但必须是已定义的。

IN [WINDOW] WindowName2

指定一个父窗口，报表设计器将在这个窗口中打开。报表设计器并不具有父窗口的特性，并且不能移出父窗口。如果父窗口移动，报表设计器随之移动。要访问报表设计器，必须首先用 DEFINE WINDOW 命令定义父窗口，并且父窗口必须是可见的。

IN SCREEN

把报表设计器放入父窗口之后，在 Visual FoxPro 主窗口中打开报表设计器。包含 IN WINDOW 子句，可以把报表设计器放进一个父窗口。

NOENVIRONMENT

包含此命令是为了提供对 2.x 报表的向后兼容性，此命令防止保存报表的环境。

在 Visual FoxPro 中，数据环境的 AutoOpenTables 属性设置为（默认值）“真”(.T.)时，可以恢复与 Visual FoxPro 报表相关的数据环境。要想确保报表打印完毕之后关闭报表环境，可以把数据环境的 AutoCloseTables 属性设置为“真”(.T.)。此属性默认值为“真”(.T.)。

当创建或修改报表时，可以用报表定义文件保存当前的 Visual FoxPro 数据环境。保存 Visual FoxPro 数据环境时，将在“报表定义”表中追加记录，这些记录指出所有打开的表、索引文件和索引排序以及表之间的任何关系。

NOWAIT

打开报表设计器之后继续执行程序。程序不必等待报表设计器关闭，而是继续执行 MODIFY REPORT NOWAIT 命令后面的程序行。如果在程序中发出 MODIFY REPORT 命令时不含有 NOWAIT 子句，打开报表设计器后，程序暂停，直到关闭报表设计器为止。

NOWAIT 子句只在程序中起作用。从命令窗口发出 MODIFY REPORT NOWAIT 时无效。

SAVE

激活另一个窗口之后，仍然保持报表设计器打开。如果省略 SAVE，在激活另一个窗口之后，将关闭报表设计器。从命令窗口发出 MODIFY REPORT 命令时，SAVE 子句将不起作用。

说明

如果发出 MODIFY REPORT 命令时不带任何参数，将显示“打开”对话框。如果从中选择了“新建”按钮，则指定报表名为“报表 1”。关闭报表设计器时可以用另外的名字保存报表。

有关创建和修改报表的详细内容，请参阅帮助中《用户指南》的第七章“设计报表和标签”。

请参阅

_ASCIICOLS, _ASCIIROWS, CREATE REPORT, REPORT

MODIFY SCREEN 命令

包含此命令是为了提供向后兼容性。请使用 MODIFY FORM 命令。

MODIFY STRUCTURE 命令

显示表设计器，从中可以修改表的结构。

语法

MODIFY STRUCTURE

说明

在 FoxPro 的早期版本中，MODIFY STRUCTURE 命令打开“表结构”对话框。

如果在当前选定的工作区中没有打开表，则显示“打开”对话框以允许您选择要修改的表。

对表结构可做的更改包括添加和删除字段；修改字段名称、大小和数据类型；添加、删除或修改索引标识；指定支持 null 值的字段。

还可以通过界面修改表结构。详细内容，请参阅帮助中《用户指南》的第二章“创建表和索引”中的“修改表”。

注意 把字段从一种数据类型更改为另一种数据类型并不完全转换字段的内容，或者根本不转换。例如，如果将日期类型的字段转换成数值类型，则字段内容不转换。

在更改表结构之前，Visual FoxPro 自动备份当前表。当修改完之后，将备份表中包含的数据追加到新修改的表结构中。如果表有一个备注字段，也将创建一个备注备份文件。

表备份文件的扩展名为 .BAK，备注备份文件的扩展名是 .TBK。
如果接受对结构的更改，然后中断数据复制过程，则新表不包含原表的所有记录。
记住：Visual FoxPro 为原表文件创建一个 .BAK 文件，并且为原备注文件（如果存在）创建一个 .TBK 备份文件。如果使用 MODIFY STRUCTURE 命令时出现问题，可以删除新文件，并且把 .BAK 文件和 .TBK 文件改回为原文件扩展名（.DBF 和 .FPT）。
当修改一个具有备注字段的表结构时，备注文件的块大小设置为当前的块大小。可以用 SET BLOCKSIZE 命令指定备注文件的块大小。

请参阅

ALTER TABLE-SQL, CREATE, CREATE TABLE-SQL, SET BLOCKSIZE

MODIFY VIEW 命令

显示视图设计器，从中可以修改已存在的 SQL 视图。

语法

MODIFY VIEW ViewName [REMOTE]

参数描述

ViewName
指定要修改的视图名称。
REMOTE
指定该视图是一个使用远程表的远程视图。如果省略 REMOTE 子句，可以修改一个基于本地表的视图。

说明

使用 CREATE SQL VIEW 命令创建 SQL 视图。

请参阅

CREATE SQL VIEW, OPEN DATABASE

MODIFY WINDOW 命令

修改用户自定义窗口或 Visual FoxPro 主窗口。

语法

```
MODIFY WINDOW WindowName | SCREEN
  [FROM nRow1, nColumn1 TO nRow2, nColumn2
  | AT nRow3, nColumn3 SIZE nRow4, nColumn4]
  [FONT cFontName [, nFontSize]]
```

[STYLE cFontStyle]
[TITLE cTitleText]
[HALFHEIGHT]
[DOUBLE | PANEL | NONE | SYSTEM]
[CLOSE | NOCLOSE]
[FLOAT | NOFLOAT]
[GROW | NOGROW]
[MINIMIZE | NOMINIMIZE]
[ZOOM | NOZOOM]
[ICON FILE FileName1]
[FILL FILE FileName2]
[COLOR SCHEME nSchemeNumber
| COLOR ColorPairList]

参数描述

WindowName

指定要修改的用户自定义窗口。要修改的窗口必须先用 DEFINE WINDOW **命令创建。**

SCREEN

指定要修改 Visual FoxPro **主窗口。** SCREEN **不能缩写，否则** Visual FoxPro **会产生错误信息。要想将** Visual FoxPro **主窗口返回到它启动时的配置，发出下面的命令，并且不带任何附加子句：**

MODIFY WINDOW SCREEN

提示 可以用 MODIFY WINDOW SCREEN NOCLOSE 命令防止意外地过早终止 Visual FoxPro。

有关 MODIFY WINDOW 子句的详细内容，请参阅 DEFINE WINDOW 命令。

说明

MODIFY WINDOW 命令更改一个已存在的用户自定义窗口（该窗口用 DEFINE WINDOW 命令创建）或 Visual FoxPro 主窗口的属性。MODIFY WINDOW 命令不能用来更改 Visual FoxPro 系统窗口（例如命令窗口和浏览窗口）的属性。

可以使用 MODIFY WINDOW 命令更改用户自定义的窗口或 Visual FoxPro 主窗口的位置、默认字体、标题、边框、控制、图标、壁纸或颜色。可以包含前面列出的子句从而更改这些属性（注意，如果您想更改颜色，必须先用 CLEAR 命令使颜色的更改生效）。

例如，要想指定用户自定义窗口或 Visual FoxPro 主窗口的新位置和大小，需要包含 FROM 和 TO 或者 AT 和 SIZE 子句。要想防止用户自定义窗口或 Visual FoxPro 主窗口被移动，需要包含 NOFLOAT 关键字。

示例

下面的示例更改 Visual FoxPro 主窗口标题栏的内容。

```
MODIFY WINDOW SCREEN TITLE '应用程序'
```

请参阅

MONTH（）函数

返回给定日期表达式中的月份。

语法

MONTH(dExpression | tExpression)

返值类型

数值型

参数描述

dExpression

指定的日期表达式，用 MONTH（）函数返回其月份值。

tExpression

指定的日期时间表达式，用 MONTH（）函数返回其月份值。

说明

MONTH（）函数返回从 1 到 12 的一个数。一月是 1，十二月是 12。

示例

```
CLEAR
? DATE（） && 显示今天的日期
? MONTH(DATE（）) && 显示月份的序号
STORE {^1998-05-03} TO gdBuy
STORE MONTH(gdBuy + 31) TO gdMonth
? gdMonth && 显示 6
```

请参阅

CMONTH（）,DAY（）,SYS（）函数概览,YEAR（）

MOUSE 命令

单击、双击、移动或拖动鼠标。

语法

MOUSE [CLICK | DBLCLICK] [AT nRow1, nColumn1]
| DRAG TO nRow2, nColumn2, nRow3, nColumn3 ...]
[PIXELS]
[WINDOW cWindowName]
[LEFT | MIDDLE | RIGHT]
[SHIFT] [CONTROL] [ALT]

参数描述

CLICK | DBLCLICK

指定单击或双击鼠标。如果省略 AT 子句，则在鼠标指针的当前位置单击或双击鼠标。

AT nRow1, nColumn1

指定在哪里单击或双击鼠标，或者是把鼠标指针移动到哪里。如果省略 CLICK 或 DBLCLICK，则鼠标指针移动到 nRow1，nColumn1 指定的位置。

除非指定一个窗口，否则 *nRow1*，*nColumn1* 指定的位置相对于 Visual FoxPro 主窗口，并且由 Visual FoxPro 主窗口的字体决定。大部分字体可以按一系列大小显示，有一些能按比例安排间隔。一行对应于当前字体的高度，一列对应于当前字体的字母平均宽度。

不能用 AT 子句选择一个 Visual FoxPro 菜单标题。使用 SYS(1500) – 激活系统目录项命令代替。

DRAG TO nRow2, nColumn2, nRow3, nColumn3 ...]

指定把鼠标指针拖动到一个或一系列位置。

当拖动鼠标指针时，按下一个鼠标键并保持不放，直到鼠标指针到达目标位置后释放鼠标键。如果省略 LEFT，MIDDLE 和 RIGHT 子句，则默认按下左鼠标键（主键）并保持不放。

DRAG 子句接受多组的 *nRow*，*nColumn* 坐标，从而将鼠标指针拖动到多个位置。如果包含了 CLICK 或 DBLCLICK 子句，则在当前位置单击或双击鼠标，然后将鼠标指针拖动到指定位置。

PIXELS

指定在 AT 和 DRAG TO 子句中包含的位置是以像素为单位。

如果省略 PIXELS 子句，则位置由 Visual FoxPro 主窗口或用 *cWindowName* 指定窗口的字体决定。大部分字体可以按一系列尺寸显示，有一些能按比例安排间隔。一个行对应于当前字体的高度，一个列对应于当前字体的字母平均宽度。

WINDOW cWindowName

指定一个窗口，AT 和 DRAG TO 子句的坐标相对于这个窗口。如果不包含 WINDOW 子句和一个活动窗口名，则 AT 和 DRAG TO 子句的坐标相对于 Visual FoxPro 主窗口。

若想指定一个系统窗口和一个工具栏，应把整个系统窗口或工具栏名称括在引号中。

LEFT | MIDDLE | RIGHT

指定当单击、双击或拖动鼠标时，按下哪个鼠标键。如果省略 LEFT、MIDDLE 和 RIGHT 子句，默认按下左鼠标键（主键）。

[SHIFT] [CONTROL] [ALT]

指定当单击、双击或拖动鼠标时按下的一个键。SHIFT 指定 SHIFT 键，CTRL 指定 CTRL 键，ALT 指定 ALT 键。

可以指定 SHIFT、CTRL 和 ALT 键的任意组合。

说明

MOUSE 命令通常用来自动测试交互的应用程序或者创建演示程序。

示例

下面的示例中，第一个命令在 Visual FoxPro 主窗口中的第三行和列单击鼠标。第二个

主窗口中的第十行和列。第三个命令从当前位置拖动鼠标指针到第二十行和列。第四个命令在当前位置双击鼠标，再拖动鼠标指针到第三十行和列。最后一个命令从当前位置拖动鼠标指针到第十行和列，然后是第二十行和列，再到第三十行和列。

```
MOUSE CLICK AT 3,3
MOUSE CLICK AT 3,3 DRAG TO 10,10
MOUSE DRAG TO 20,20
MOUSE DBLCLICK DRAG TO 30,30
MOUSE DRAG TO 10,10,20,20,30,30
```

请参阅

MCOL（），MDOWN（），MROW（），SYS(1500)

MouseDown 事件

当用户按下一个鼠标键时发生。

语法

PROCEDURE Object.MouseDown
[LPARAMETERS nIndex, nButton, nShift, nXCoord, nYCoord]
–或者–

LPARAMETERS nButton, nShift, nXCoord, nYCoord

参数描述

必须在事件过程中包含 LPARAMETERS 或 PARAMETERS 语句，并且为每个参数指定一个名称。Visual FoxPro 按下列顺序把 5 个参数中的 4 个传送给 MouseDown 事件。

nIndex

存放一个数，它唯一标识控件数组中的一个控件。仅当控件是控件数组的一部分时，才传送 nIndex 参数。

nButton

存放一个数，它指定为触发事件需要按下哪个键：1 (左)，2 (右) 或 4 (中)。

nShift

存放一个数，它指定当按下用 nButton 设置的键时，SHIFT，CTRL，和 ALT 键的状态。

下表列出了单独修改键在 nShift 中返回的值。

nShift 的修改键值

Windows 键	值
SHIFT	1
CTRL	2
ALT	4

如果按下了一个键，就设置一个位，*nShift* 参数是这些位的和。低位对应于 SHIFT 键（0 位）、CTRL 键（1 位）、ALT 键（2 位），这些位分别对应于 1，2 和 4。*nShift* 参数表明这

些键的状态。部分、全部或没有设置三个位，表明部分、全部或没有按下三个键。例如，如果 CTRL 和 ALT 键全部按下，那么 *nShift* 的值为 6。

nXCoord, nYCoord

存放表单中鼠标指针当前的水平 (nXCoord) 和垂直 (nYCoord) 位置。这些坐标总是以 ScaleMode 属性设置值为度量单位，按照指定表单的坐标系统表达的。

说明

用 MouseDown 过程指定按下鼠标键时发生的动作。与 Click 和 DblClick 事件不同，可以用 MouseDown 事件区别左、右、中鼠标键，也可以为使用 SHIFT，CTRL 和 ALT 键的鼠标-键盘组合编写代码。

注意 可以用 MouseMove 事件响应由鼠标移动引起的事件。MouseDown 和 MouseUp 使用的 *nButton* 参数与 MouseMove 使用的 *nButton* 参数不同。对于 MouseDown 或 MouseUp，每个事件中 *nButton* 参数确切地指明一个键，对于 MouseMove，它表明了所有键的当前状态。

应用于

复选框，组合框，命令按钮，命令组，容器对象，控件对象，编辑框，表单，表格，标头，图像，标签，线条，列表框，选项按钮，选项组，页面，页框，形状，微调，文本框，工具栏

请参阅

Click 事件, DblClick 事件, MiddleClick 事件, MouseMove 事件, MousePointer

MouseIcon 属性

指定当鼠标指针位于一个对象上时，所显示的鼠标指针图标。设计和运行时可用。

语法

Object.MouseIcon[= cFileName]

参数描述

cFileName

指定当鼠标指针位于一个对象上时，所显示的文件。可以指定鼠标图标 (.ico) 文件、一个鼠标指针 (.cur) 文件或一个动画鼠标指针 (.ani) 文件。

说明

设置 MousePointer 属性为 99（自定义）可以显示鼠标图标。

应用于

复选框、组合框、命令按钮、命令按钮组、容器对象、控件对象、编辑框、表单、表格、图片、标签、线条、列表框、OLE 绑定型控件、OLE 容器控件、选项按钮、选项按钮组、_SCREEN、图形、微调、文本框、工具栏

请参阅

MousePointer **属性**

MouseMove **事件**

当用户在一个对象上移动鼠标时发生。

语法

PROCEDURE Object.MouseMove, [LPARAMETERS nIndex, nButton, nShift, nXCoord, nYCoord]

–**或者**–

LPARAMETERS nButton, nShift, nXCoord, nYCoord

参数描述

必须在事件过程中包含 LPARAMETERS **或** PARAMETERS **语句，并且为每个参数指定一个名称。** Visual FoxPro **按下列顺序把** 5 **个参数中的** 4 **个传送给** MouseMove **事件。**

nIndex

存放一个数，它唯一标识控件数组中的一个控件。仅当控件是控件数组的一部分时，才传送 nIndex **参数。**

nButton

存放一个数，它以位总和的形式指定鼠标键的状态。如下表所示：

对 nButton 的鼠标按钮值

Windows	nButton Value
左鼠标键	1
右鼠标键	2
中鼠标键	4

如果按下了一个键，则设置一个位。*nButton* 参数表明设置了哪个位，0 位相应于左鼠标键，1 位相应于右鼠标键，2 位相应于中鼠标键，对于三种情况 *nButton* 分别取值为 1，2，4。它表明了鼠标键的各种状态：部分、全部、或不设置三个位，表明部分、全部或没有鼠标键按下。可能的取值从 0 到 7。单个鼠标键相应于下列值：1 (左)，2 (右)，和 4 (中)。例如，如果左和右鼠标键都按下了，则 *nButton* 的值为 3。

nShift

存放一个数，它指定在 Visual FoxPro for Windows 中移动鼠标时，SHIFT，CTRL，和 ALT 键的状态。

下表列出了单独修改键在 nShift 中返回的值。

nShift 的修改键值

Windows 键	值
SHIFT	1
CTRL	2

ALT 4

如果按下鼠标时，有多于一个的修改键也被按下，则 *nShift* 参数是这些修改键的和。例如，在 Visual FoxPro for Windows 中，如果按下鼠标按钮时，也按下 CTRL 键，那么 *nShift* 的值为 2。但是如果 CTRL 和 ALT 键全部按下，那么 *nShift* 的值为 6。

nXCoord, nYCoord

存放表单中鼠标指针当前的水平 (nXCoord) 和垂直 (nYCoord)位置。这些坐标总是以 ScaleMode 属性的设置值为度量单位，按照指定的表单坐标系统表达的。

当鼠标指针在对象之间移动时，连续触发 MouseMove 事件。

注意 可以用 MouseDown 和 MouseUp 事件响应由按下和释放鼠标键引起的事件。

MouseMove 使用的 *nButton* 参数与 MouseDown 和 MouseUp 使用的 *nButton* 参数不同。对于 MouseMove，*nButton* 参数表明了所有键的当前状态；一个单独的 MouseMove 事件可以表明部分、全部或没有按下任何键；对于 MouseDown 或 MouseUp，每个事件中 *nButton* 参数确切地指明一个键。

要避免在 MouseMove 事件过程中移动一个窗口，这会引起级联事件并产生运行错误，例如堆栈溢出。当窗口在指针下移动时会触发 MouseMove 事件。这样即使鼠标静止时也会触发一个 MouseMove 事件。

应用于

复选框，列，组合框，命令按钮，命令组，容器对象，控件对象，编辑框，表单，表

格，标头，图像，标签，线条，列表框，选项按钮，选项组，页面，页框，形状，微调，文本框，工具栏

请参阅

Click 事件，DblClick 事件，MiddleClick 事件，MousePointer 属性，MouseDown 事件，MouseUp 事件，MouseWheel 事件

MousePointer 属性

指定运行时，鼠标在一个对象的特定位置之上时，鼠标指针的形状。设计和运行时该属性均可用。

语法

Object.MousePointer[= nType]

参数描述

nType

MousePointer 属性的设置有：

设置	说明
0	(默认值)由对象决定的形式。

1	箭头。
2	十字标。一个十字指针
3	I型标。
4	图标。一个黑块中的一个小白块。
5	尺寸调整标。指向东西南北的四箭头。
6	东北-西南方向尺寸调整标。指向东北和西南的双箭头。
7	南北方向尺寸调整标。指向南北的双箭头。
8	西北-东南方向尺寸调整标。指向西北和东南的双箭头。
9	东西方向尺寸调整标。指向东西的双箭头。
10	向上的箭头。
11	沙漏。
12	禁止停放。
13	隐藏指针
14	箭头

注意 用属性单设置 MousePointer 属性时，下拉列表显示了应用于 Visual FoxPro for Windows的设置名称。

说明

MousePointer属性可以用来表明当鼠标指针经过表单或对话框上的控件时功能的变化。例如可以把 MousePointer 属性设置为 11 (沙漏)，表明用户需要等待某过程的结束。

应用于

复选框，组合框，命令按钮，命令组，容器对象，控件对象，编辑框，表单，表格，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，选项组，_SCREEN，形状，微调，文本框，工具栏

请参阅

MouseMove 事件

MouseUp 事件

当用户释放一个鼠标键时发生。

语法

PROCEDURE Object.MouseUp
[LPARAMETERS nButton, nShift, nXCoord, nYCoord]
–或者 –
LPARAMETERS nIndex, nButton, nShift, nXCoord, nYCoord

参数描述

必须在事件过程中包含 LPARAMETERS 或 PARAMETERS 语句，并且为每个参数指定一个名称。Visual FoxPro 按下列顺序把 5 个参数中的 4 个传送给 MouseUp 事件。

nIndex

存放一个数，它唯一标识控件数组中的一个控件。

nButton

存放一个数，它指定为引发事件而释放哪个键：1(左)，2(右)和4(中)。

nShift

存放一个数，它指定当释放nButton参数指定的键时SHIFT、CTRL和ALT键的状态：1(SHIFT)，2(CTRL)，4(ALT)。

下表列出了单独修改键在nShift中返回的值。

nShift的修改键值

Windows键	值
SHIFT	1
CTRL	2
ALT	4

如果按下鼠标时，有多于一个的修改键也被按下，则*nShift*参数是这些修改键的和。例如，在Visual FoxPro for Windows中，如果按下鼠标按钮时，也按下CTRL键，那么*nShift*的值为2。但是如果CTRL和ALT键全部按下，那么*nShift*的值为6。

nXCoord, nYCoord

存放鼠标指针在表单中当前的水平(nXCoord)和垂直(nYCoord)位置。这些坐标总是以ScaleMode属性的设置值为度量单位，按照指定的表单坐标系统表达的。

说明

可用 MouseUp 过程指定释放鼠标键时发生的动作。与 Click 和 DblClick 事件不同，可以用 MouseUp 事件区别左、右、中鼠标键，也可以为使用 SHIFT，CTRL 和 ALT 键的鼠标-键盘组合编写代码。

可以用 MouseMove 过程响应由移动鼠标引起的事件。

注意 MouseDown 和 MouseUp 使用的 *nButton* 参数与 MouseMove 使用的 *nButton* 参数不同。对于 MouseDown 或 MouseUp，每个事件中的 *nButton* 参数确切地指明一个键；对于 MouseMove，它表明了所有键的当前状态。

应用于

复选框，组合框，命令按钮，命令组，容器对象，控件对象，编辑框，表单，表格，标头，图像，标签，线条，列表框，选项按钮，选项组，页面，页框，形状，微调，文本框，工具栏

请参阅

Click 事件, DblClick 事件, MiddleClick 事件, MouseDown 事件, MouseMove 事件, MousePointer 属性, MouseWheel 事件, ScaleMode 属性

MouseWheel 事件

当用户在一个具有轮的鼠标设备上滚动鼠标轮时发生。

语法

PROCEDURE Object.MouseWheel
LPARAMETERS [nIndex,] nDirection, nShift, nXCoord, nYCoord

参数描述

在事件过程中必须包含一个 LPARAMETERS 或 PARAMETERS 语句，并且指定每个参数的名称。Visual FoxPro 按以下顺序向 MouseWheel 事件传递四或五个参数。

nIndex

如果一个控件在一个控件数组中，则本参数唯一标识该控件。只有当一个控件在一个控件数组中时才传递 nIndex 参数。

nDirection

包含了一个依靠鼠标设备的数，它表示了鼠标球转动的方向。负值表示鼠标球向后转动，而正值表示鼠标球向前转动。

nShift

包含一个指定当鼠标轮滚动时修饰键的状态。有效的修饰键有 SHIFT、CTRL 和 ALT 键。

下表列出了 nShift 的单个修饰键的返回值。

值	键
1	SHIFT
2	CTRL
4	ALT

如果当鼠标轮滚动时按下了多个修饰键，则 nShift 参数包含各修饰键的值的和。例如，在 Visual FoxPro for Windows 中，如果按下鼠标按钮时，也按下 CTRL 键，那么 *nShift* 的值为 2。但是如果 C 中文版中文版 TRL 和 ALT 键全部按下，那么 *nShift* 的值为 6。

nXCoord, nYCoord

包含鼠标指针在表单中的当前水平 (nXCoord) 和垂直 (nYCoord) 位置。这些坐标是按表单的坐标系统表达的，度量单位由 ScaleMode 属性的设置确定。

应用于

复选框，列，组合框，命令按钮，命令按钮组，容器对象，控件对象，编辑框，表单，表格，标头，图像，标签，线条，列表框，选项按钮，选项按钮组，页面，页框，形状，微调，文本框，工具栏

请参阅

Click 事件, MiddleClick 事件, MousePointer 属性, MouseDown 事件, MouseUp 事件

Movable 属性

指定用户是否可以在运行时移动一个对象。设计时可用，运行时可读写。

语法

Object.Movable[= lExpr]

参数描述

lExpr

Movable **属性的设置有：**

设置	说明
“真”(.T.)	(默认值)可以移动对象。对 Visual FoxPro for Windows 中的表单，在“控件”菜单中添加 Move 命令。
“假”(.F.)	不能移动对象。对于表格列，不能人工直接移动表格列，但是，如果更改了列的 ColumnOrder 属性或者把另一个列移动到它上面，就可以移动列。

应用于

列，表单，_SCREEN，工具栏

请参阅

Move 方法

移动一个对象。

语法

Object.Move (nLeft [, nTop [, nWidth [, nHeight]]])

参数描述

nLeft

指定对象左边沿的水平坐标，nLeft 是一个单精度值。

nTop

指定对象上边沿的垂直坐标，nTop 是一个单精度值。

nWidth

指定对象的新宽度，nWidth 是一个单精度值。

nHeight

指定对象的新高度，nHeight 是一个单精度值。

说明

只有 *nLeft* 参数是必需的。但是，如果想要包含任何其他的参数，也必需包含语法中该

参数前的所有参数。例如，不能指定了 *nWidth* 而不指定 *nLeft* 和 *nTop*。任何未指定的后继参数将保持不变。

在屏幕上移动表单，或在表单中移动控件时，都是相对于原点 (0，0)，原点位于左上角。当在容器中移动控件时，使用容器的坐标系。

应用于

复选框，组合框，命令按钮，命令组，容器对象，控件对象，编辑框，表单，表格，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，选项组，页框，_SCREEN，形状，微调，文本框，工具栏

请参阅

Top 属性, Width 属性

MOVE POPUP 命令

把用 DEFINE POPUP 命令创建的用户自定义菜单移动到新位置。

语法

MOVE POPUP MenuName TO nRow1, nColumn1 | BY nRow2, nColumn2

参数描述

MenuName

指定要移动的菜单。

在 Visual FoxPro 中不能移动系统菜单。

TO nRow1, nColumn1

在用户自定义窗口或 Visual FoxPro 主窗口中，把菜单移动到由 nRow1，nColumnl 指定的新位置。

BY nRow2, nColumn2

相对于当前位置移动菜单。数值表达式 nRow2 指定了移动菜单的行数（如果 nRow2 是正值就向下移动，如果是负值就向上移动），数值表达式 nColumn2 指定了移动菜单的列数（如果 nColumn2 是正值，就向右移动，如果是负值就向左移动）。

说明

可以把菜单移动到指定位置，或者相对它的当前位置移动菜单。只要菜单已定义，就可以移动它，它不必是活动或可见的。

示例

下面的示例定义并激活了一个菜单，然后移动和更改其大小。

```
CLOSE DATABASE
CLEAR
DEFINE POPUP popMovIn FROM 2,2 TO 7, 14 PROMPT FILES LIKE *.PRG ;
    TITLE 'Programs'
ACTIVATE POPUP popMovIn NOWAIT
=CHRSAW(2)
MOVE POPUP popMovIn BY 5,5    && 下移菜单
```

```
=CHRSAW(2)
SIZE POPUP popMovIn BY 5,5    && 扩大菜单
=CHRSAW(2)
SIZE POPUP popMovIn BY -5,-5    && 收缩菜单
=CHRSAW(2)
MOVE POPUP popMovIn BY -5,-5    && 上移菜单
=CHRSAW(2)
DEACTIVATE POPUP popMovIn
RELEASE POPUP popMovIn
```

请参阅

ACTIVATE POPUP, DEFINE POPUP, SIZE POPUP

MOVE WINDOW 命令

把窗口移动到新位置，该窗口可以是用 DEFINE WINDOW 命令创建的用户自定义窗口，或者是 Visual FoxPro 系统窗口（例如命令窗口或浏览窗口）。

语法

```
MOVE WINDOW WindowName TO nRow1, nColumn1
  | BY nRow2, nColumn2 | CENTER
```

参数描述

WindowName
指定要移动的窗口名。

TO nRow1, nColumn1
在 Visual FoxPro 主窗口或用户自定义窗口中，把窗口移动到由 nRow1, nColumn1 指定的位置。

BY nRow2, nColumn2
相对当前位置移动窗口，数值表达式 nRow2 指定了移动窗口的行数（如果 nRow2 是正值就向下移动，如果是负值就向上移动）。数值表达式 nColumn2 指定了移动窗口的列数（如果 nColumn2 是正值就向右移动，如果是负值就向左移动）。

CENTER
在 Visual FoxPro 主窗口或父窗口中居中放置窗口。

说明

可以把窗口移动到指定位置，也可以相对它的当前位置移动窗口。只要窗口已定义，就可以移动它，不必是活动或可见的。

若要移动一个系统窗口和/或一个工具栏（在 Visual FoxPro 中），应把整个系统窗口名称或工具栏名称括在引号中。例如，若要在 Visual FoxPro 中移动报表控制工具栏（还未停放），可以用下面命令：

```
MOVE WINDOW "Report Controls" BY 1,1
```

示例

在下面的示例中，定义并激活了窗口 wEnter 后，移动窗口。

```
DEFINE WINDOW wEnter FROM 10,4 TO 15,54 SYSTEM ;
    TITLE "Nomadic Window"
ACTIVATE WINDOW wEnter
WAIT WINDOW 'Press any key to move the window'
MOVE WINDOW wEnter TO 20,15
WAIT WINDOW 'Press any key to center the window'
MOVE WINDOW wEnter CENTER
WAIT WINDOW 'Press any key to release the window'
RELEASE WINDOW wEnter
```

请参阅

ACTIVATE WINDOW

Moved 事件

当对象移动到新位置时，或者以编程方式更改容器对象的 Top 或 Left 属性设置时发生。

语法

PROCEDURE Object.Moved

说明

当对象移动到新位置时，或者在代码中更改 Top 或 Left 属性时发生。

应用于

列，容器对象，控件对象，表单，表格，OLE 绑定型控件，OLE 容器控件，页框，工具栏

请参阅

ColumnOrder 属性, Left 属性, Top 属性

MoverBars 属性

指定是否在 ListBox 控件中显示移动钮一栏。设计时可用，运行时可读写。

语法

ListBox.MoverBars[= lExpr]

参数描述

lExpr

MoverBars 属性的设置有：

设置	说明
“真”(.T.)	显示移动钮栏，用户可以交互地重新排序控件中的内容。
“假”(.F.)	(默认值)不显示移动钮栏。

说明

仅当 RowSourceType 属性设置为 0 或 1 时，MoverBars 属性才可用。

应用于

列表框

请参阅

RowSourceType 属性，ScrollBars 属性

MRKBAR() 函数

确定是否已标记用户自定义菜单或 Visual FoxPro 系统菜单中的一个菜单项。

语法

MRKBAR(cMenuName, nMenuItemNumber | cSystemMenuItemName)

返值类型

逻辑值

参数描述

cMenuName

指定包含该菜单项的菜单名称。菜单可以是 Vsiaul FoxPro 系统菜单（例如 _MFILE，MEDIT 或 _MDATA）。

nMenuItemNumber

指定用户自定义菜单中菜单项的编号。用 DEFINE BAR 命令创建菜单项时，就确定了菜单项的编号。

cSystemMenuItemName

指定 Visual FoxPro 系统菜单项名称。例如，下面的命令显示一个逻辑值，该值指定是否标记“文件”菜单上的“新建”菜单项：

```
? MRKBAR('_MFILE', _MFI_NEW)
```

说明

可用 SET MARK OF 命令添加或移去一个菜单项的标记。

如果标记了指定菜单项，MRKBAR（）函数返回“真”(.T.)；否则，MRKBAR（）函数返回“假”(.F.)。

有关 MRKBAR（）函数的例子，请参阅 CNTBAR（）。

请参阅

MRKPAD（）, SET MARK OF, SET MARK TO

MRKPAD（）函数

确定是否已标记了用户自定义菜单或 Visual FoxPro 系统菜单栏中的一个菜单标题。

语法

MRKPAD(cMenuBarName, cMenuTitleName)

返值类型

逻辑值

参数描述

cMenuBarName

指定包含该菜单标题的菜单栏名称。

cMenuTitleName

指定菜单标题名称。

说明

用 SET MARK OF 命令可以添加或移去一个菜单标题的标记。

如果标记了指定的菜单标题，MRKPAD（）函数返回“真”(.T.)；否则 MRKPAD（）函数返回“假”(.F.)。

示例

下面的程序示例 MARKPAD.PRG，在选择了菜单标题时，使用 MRKPAD（）来切换其标记字符。

用 SET SYSMENU SAVE 将当前的系统菜单栏保存到内存中，并且用 SET SYSMENU TO 删除所有的系统菜单项。

用 DEFINE PAD 创建几个系统菜单项。当选中了一个菜单项时，执行 choice 过程。choice 显示所选的菜单项和菜单栏名称。用 SET MARK OF 和 MRKPAD（）来显示或删除菜单项的标记字符。如果选择了“退出”菜单标题，则恢复最初的 Visual FoxPro 系统菜单。

```
*** 给该程序命名为 MARKPAD.PRG ***
CLEAR
SET SYSMENU SAVE
SET SYSMENU TO
SET MARK OF MENU _MSYSMENU TO CHR(4)
PUBLIC glMarkPad
glMarkPad = .T.
DEFINE PAD padSys OF _MSYSMENU PROMPT '\<System' COLOR SCHEME 3 ;
    KEY ALT+S, ''
DEFINE PAD padEdit OF _MSYSMENU PROMPT '\<Edit' COLOR SCHEME 3 ;
    KEY ALT+E, ''
DEFINE PAD padRecord OF _MSYSMENU PROMPT '\<Record' COLOR SCHEME 3 ;
    KEY ALT+R, ''
DEFINE PAD padWindow OF _MSYSMENU PROMPT '\<Window' COLOR SCHEME 3 KEY ALT+W, ''
```

```
DEFINE PAD padReport OF _MSYSMENU PROMPT 'Re\<ports' COLOR SCHEME 3 ;
    KEY ALT+P, ''
DEFINE PAD padExit OF _MSYSMENU PROMPT 'E\<xit'  COLOR SCHEME 3 ;
    KEY ALT+X, ''
ON SELECTION MENU _MSYSMENU ;
    DO choice IN markpad WITH PAD（）, MENU（）

PROCEDURE choice
PARAMETER gcPad, gcMenu
WAIT WINDOW 'You chose ' + gcPad + ;
    ' from menu ' + gcMenu NOWAIT
SET MARK OF PAD (gcPad) OF _MSYSMENU TO ;
    !MRKPAD('_MSYSMENU', gcPad)
glMarkPad= ! glMarkPad
IF gcPad = 'PADEXIT'
    SET SYSMENU TO DEFAULT
ENDIF
```

请参阅

MRKBAR（），SET MARK OF

MROW（）函数

返回 Visual FoxPro 主窗口或用户自定义窗口中鼠标指针的行位置。

语法

MROW([cWindowName [, nScaleMode]])

返值类型

数值型

参数描述

cWindowName

指定一窗口名，MROW（）函数返回鼠标指针在其中的行位置。

nScaleMode

设定 MROW（）返回值所使用的度量单位。nScaleMode 的设置是

nScaleMode	说明
0	（默认）Foxels。foxel 的大小是当前表单中字型宽与高的平均值。当开发可在字符平台和图形平台上跨平台运行的应用程序时，foxel 十分有用。

续表

3	像素。像素是屏幕和打印机的最小分辨单位。在不同的屏幕中，像素的大小不同。

说明

如果没有活动的用户自定义窗口并且省略了 *cWindowName* 可选参数，则 MROW（）函数返回鼠标指针在 Visual FoxPro 主窗口中的行位置。

如果有一个活动的用户自定义窗口，并且省略了 *cWindowName* 可选参数，MROW（）函数返回相对于该用户自定义窗口的鼠标指针行坐标。如果鼠标指针指在用户自定义窗口之外，MROW（）函数返回 -1。如果没有加载鼠标驱动程序，并且没有输出窗口，MROW（）函数返回 -1。

请参阅

AMOUSEOBJ（），COL（），GridHitTest 方法，ISMOUSE（），MCOL（），ROW（），WCOLS（），WROWS（）

MTON（）函数

由货币型表达式返回一个数值型的值。

语法

MTON(mExpression)

返值类型

数值型

参数描述

mExpression

指定的一个货币型表达式，MTON（）函数返回它的值，mExpression 计算结果必须是有效的货币值，否则 Visual FoxPro 产生错误。

在数值型值前面加一个美元符号 ($) 的前缀，就可以创建一个货币型值。

说明

MTON（）函数返回一个带四位小数的数值型值。

示例

下面的示例创建了一个货币型变量 gyMoney。TYPE（）显示 Y，表示此变量是货币型。用 MTON（）转换变量为数值型，现在 TYPE（）显示 N，表示此变量转换后是数值型。

```
STORE $24.95 TO gyMoney && 创建一个货币型变量
CLEAR
? "gyMoney is type: "
?? TYPE('gyMoney') && 显示 Y, 货币型值

gyMoney = MTON(gyMoney)   && 将 gyMoney 转化为数值型值
? "gyMoney is now type: "
?? TYPE('gyMoney') && 显示 N, 数值型值
```

请参阅

NTOM（）

MultiSelect 属性

指定用户是否可以在一个列表框控件中作多项选择，以及如何选择。设计时可用，运行时可读写。

语法

ListBox.MultiSelect[= nChoice]

参数描述

nChoice

下表列出了 MultiSelect **属性的设置：**

设置	说明
0	(默认值)不允许做多项选择。
1	允许做多项选择。若要做多项选择，按下 CTRL 键的同时单击各项

说明

为使用户能在列表中作多项选择，需把 MultiSelect **设置为“真”** (.T.)。**可以用** Selected

属性确定选择了哪些项。

示例

在下面的示例中创建了一个列表框，它的 MultiSelect 属性设置为“真”(.T.)，允许在列表框中作多项选择。出现在列表框上的项来源于一个数组，这个数组是用 RowSourceType 和 RowSource 属性指定的。

在 FOR ... ENDFOR 循环中，用 ListCount 属性来显示列表框中已选择的一个或多个选项，用 Selected 属性确定要选择的项，用 List 属性返回这些项。

```
CLEAR

DIMENSION gaMyListArray(10)
FOR gnCount = 1 to 10 && 用字母填充数组
    STORE REPLICATE(CHR(gnCount+64),6) TO gaMyListArray(gnCount)
ENDFOR

frmMyForm = CREATEOBJECT('Form') && 创建一个表单
frmMyForm.Closable = .f. && 使“控件”菜单框失效

frmMyForm.Move(150,10) && 移动表单

frmMyForm.AddObject('cmbCommand1','cmdMyCmdBtn') && 添加“退出”命令按钮
frmMyForm.AddObject('lstListBox1','lstMyListBox') && 添加列表框控件

frmMyForm.lstListBox1.RowSourceType = 5 && 指定一个数组
frmMyForm.lstListBox1.RowSource = 'gaMyListArray' && 数组包含列表框选项
```

```
frmMyForm.cmbCommand1.Visible =.T. && “退出”命令按钮可见
frmMyForm.lstListBox1.Visible =.T. && “列表框”可见

frmMyForm.SHOW && 显示表单
READ EVENT && 开始事件过程

DEFINE CLASS cmdMyCmdBtn AS CommandButton && 创建命令按钮
    Caption = '\<Quit' && 命令按钮上的标题
    Cancel = .T. && 默认的取消命令按钮(ESC)
    Left = 125 && 命令按钮的列
    Top = 210 && 命令按钮的行
    Height = 25 && 命令按钮的高度

    PROCEDURE Click
        CLEAR EVENT && 结束事件过程，关闭表单
        CLEAR && 清除 Visual FoxPro 主窗口
ENDDEFINE

DEFINE CLASS lstMyListBox AS ListBox && 创建列表框控件
    Left = 10 && 列表框的列
    Top = 10 && 列表框的行
    MultiSelect = .T. && 允许选择多项

PROCEDURE Click
```

```
    ACTIVATE SCREEN
    CLEAR
    ? "Selected items:"
    ? "--------------"
    FOR nCnt = 1 TO ThisForm.lstListBox1.ListCount
        IF ThisForm.lstListBox1.Selected(nCnt)  && 选择了这个项吗?
            ? SPACE(5) + ThisForm.lstListBox1.List(nCnt) && 显示该选项
        ENDIF
    ENDFOR

ENDDEFINE
```

应用于

列表框

请参阅

AddItem **方法**, Clear **方法**, List **属性**, ListCount **属性**, NewItemID **属性**, RemoveItem **方法**, Selected **属性**, TopItemID **属性**

MWINDOW（）函数

返回鼠标指针所在的窗口名称。

语法

MWINDOW([cWindowName])

返值类型

字符型,逻辑值

参数描述

cWindowName

指定一窗口名。如果鼠标指针在它上面，MWINDOW（）函数返回“真”(.T.)；否则MWINDOW（）函数返回“假”(.F.)。

说明

如果省略可选的窗口名称，MWINDOW（）函数返回鼠标指针所在的窗口名称，或者一个空字符串（如果鼠标指针位于Visual FoxPro主窗口上）。

请参阅

MCOL（）,MDOWN（）,MROW（）,WONTOP（）,WOUTPUT（）

Name 属性

指定在代码中引用对象时所用的名称。设计时可用，运行时可读写。

语法

Object.Name[= cName]

参数描述

cName

指定在代码中引用对象时所用的名称。

说明

新建对象的默认名称是该对象的类型加上一个唯一的整数。例如：第一个新建表单对象的名称是 Forml，而表单上创建的第三个文本框为 Text3。

注意 若对象为分层结构中的第一个对象(即最外层容器)，应使用对象变量代替 Name 属性来引用该对象。

对于一个项目对象，Name 属性包含该项目的名称和路径，并且在设计时和运行时只读。

对于一个文件对象，Name 属性包含该文件的名称和路径，并且在设计时和运行时只读。

应用于

ActiveDoc对象，复选框，列，组合框，命令按钮，命令组，容器对象，控件对象，临时表，自定义，数据环境，文件对象，编辑框，表单，表单集，表格，标头，图像，标签，线条，列表框，OLE 绑定型控件，OLE 容器控件，选项按钮，选项组，页面，页框，项目对象，项目挂接对象，关系，_SCREEN，形状，微调，文本框，计时器，工具栏

请参阅

Caption 属性, NewIndex 属性, NewItemID 属性, ServerName 属性, TypeLibName 属性