**窗体属性**

Autoredraw 图像重绘

 如果为false 最小化窗体，然后再恢复的时候之前显示的文字会消失

 如果为true 文字不会消失

Backcolor 设置背景颜色

Borderstyle 设置窗口的边框

 设置为0 无边框

 设置为1 固定单边框无最大 最小化按钮

 设置为2 可调整的边框

 设置为3 固定边框不包含最大化，最小化按钮

 。。。。。。。。。。（默认设置2就可以）在运行期间不可以改变！

Caption 窗体标题

 可以通过属性窗口设置

 也可以通过代码设置：object.caption=”窗口” object 窗体对象

Controlbox 控制框

 设置为1 有控制图标：无最大 最小化按钮

 设置为0 无控制图标

Enabled 允许

 设置为1 用户可以操作

 设置为0 用户不可以操作

Forecolor 前景颜色

Heighe wideth 高、宽

 object. Heighe=60 object对象

 object. wideth =60 object 对象

maxbutton 最大化按钮

 设置为1 可使用

 设置为0 不可使用

minbutton 最大小化按钮

 设置为1 可使用

 设置为0 不可使用

Icon 设置窗体的图标

Visible 可见属性

 设置true 可见

 设置false 不可见

Windowstate 设置窗口状态

 设置为0时 运行的时候为设计时窗口大小

 设置为1时 窗口最小化

 设置为2时 窗口最大化

Private sub command1\_click（）

 Load form2 加载form2窗体

 Form2.show 显示form2窗体

End sub

显示窗体

 Object.show.style.ownerform

 Object：窗体对象

 Styke：如果为0则窗体是无模式的 也就是说在窗口1 调用窗口2的时候

 不关闭窗口2的情况下可以控制窗口1

 如果设置为1窗体是有模式的 在调用窗口2的时候，不可以控制窗口1。

 Form2.show 无模式调用form2窗体

 Form2.show 1 有模式调用form2窗体

隐藏窗体

 Object.hide

 Object：窗体对象

 Form2.hide

Click 单击事件

 单击鼠标左键触发事件

Dblclick 双击事件

 连点两下鼠标左键，单机事件也会执行一次。

Load 装入事件

 窗口调用的时候自动执行一次装入事件

Unload 卸载事件

 窗体关闭的时候自动执行一次卸载事件

窗体的启动过程

 先是lnitialize事件→load事件→activate事件

程序运行时，如果窗体被调用，首先触发nitialize事件，紧跟着是load事件，将窗体装入内存后，窗体激活，activate事件发生，这三个事件是在一瞬间发生的。

 nitialize事件和load事件都是发生在窗体被显示前，用于放置一些初始化命令，而在里面使用print语句时要将autoredraw属性设置为真（图像重绘）

 **数据类型**

String 字符串 类型声明符：$

String\*3 定长字符串 字符的个数3个 类型声明符：$

Integer 整数 两个字节 类型声明符：%

Long 长整数 四个字节 类型声明符：&

Single 单精度浮点数 四个字节 类型声明符：！

Double 双精度浮点数 八个字节 类型声明符：#

Currency 货币型 八个字节 类型声明符：@

Byte 字节 一个字节

Boolean 布尔 两个字节

Date 日期 八个字节

Variant 变体型

Type 自定义数据类型

 【private | public】 type 数据类型名

 数据类型元素名 as 类型名

 数据类型元素名 as 类型名

 。。。。。。。

 End type

 Type 语句只能在模块级使用或窗体的通用声明部分

 例： 在工程中添加一个模块并在模块中编写一下代码

 Type sell

 Name as string \*20 定义长度为20的字符型变量

 Price as Integer 定义整型

 End type

 在窗体的load事件并编写代码

 Private aa as sell 定义一个sell类型的aa变量

 aa．Name=”11”

 aa.price=12

 end sub

public 在整个工程中都是可见的

private 只在所声明模块中可见

static 静态变量只在所声明模块可见

定义变量：declare 变量名 as 类型

 Declare可以是dim、 public或static，as是关键字，类型是基本数据类型

 例：dim aa as integer

 也可以：dim aa%

定义多个变量时： dim aa as integer，bb as integter，cc as integer

定义定长字符串：dim aa string\*5 长度为5

 **常用内部函数**

转换函数：

 Int(x) 求不大于自变量的最大数

 Fix(x) 截取小数部分

 Hex(x) 十进制数转换成十六进制

 Oct(x） 十进制数转换成八进制

 Asc(x） 返回字串符的第一个字符的ascll码

 chr(x）转换成ascll码

 str(x) 转换成字符型

 val(x) 转换成数值型

 cint(x) 小数转换成整数 并四舍五入

 ccur(x) 转换成货币类型 保留四位小数并四舍五入

 cdbl(x) 转换成双精度

 csng(x) 转换成单精度

数学函数：

 Sin(x) x的正弦值

 Cos(x) x的余弦值

 Tan(x) x的正切值

 Atn(x) x的反正切值

 Abs(x) x的绝对值

 Sgn(x) 返回x的符号，x为负数时返回-1，x为0时返回0，x为正数时返回1。

 Sqr(x) x的平方根 必须大于0

日期函数：

 Day(now) 当前日期

 Weekday(now) 当前星期

 Month(now) 当前月份

 Yeat(now) 当前年份

 Hour(now) 当前小时

 Minute(now) 当前分钟

 Second(now) 当前秒

 **字串符处理**

Unicode类型的字串符：

 “abcd大大” 6个字符 占12个字节 英文中文都占两个字节

Ansl类型的字串符：

 “abcd大大” 4个字符 占8个字节 英文占一个字节，中文占两个字节，如果是abcde大大，还是4个字符

 两个字节一个字符

Strconv转换函数：

 Strconv（待转换字符串，转换格式）

转换格式可以是vbunicode或者是vbfromunicode

 Vbunicode是将ansl格式转换成unicode格式

 Vbfromunicode是将unicode转换成ansl格式

例： dim a as string

 A=“aa啊”

 A=strconv（a,vbunicode） 转换unicode格式

字符串测试函数：

Len（x）测试占几个字符

Lenb（x）测试占几个字节

X可以是变量名 也可以是字串符

删除空白函数：

 Ltrim（x） 删除字符串左边空白字符

 Rtrim（x）删除右边空白字符

 Trim（x）删除两边空白字符

字符串截取函数：

 从字符串中返回指定数目的字符。从1开始计数

 Left（x，n）返回字符串的前n个字符 左侧

 Mid（x，p，n）从p个字符开始，向后截取n个字符

 Right（x，n）返回字符串的后n个字符 右侧

String函数：

 返回由n个指定字符组成的字符串，第二个变量可以是ascll码 也可以是字符串当为ascll码时，返回ascll码对应的n个字符，当为字符串时返回由该字符串第一个字符组成的n个字符的字符串。

 例：string（n，ascll码） n就是个数

 String（n，字符串）

空白函数：

 Space（n）

 返回n个空格

字符串匹配函数：

|  |
| --- |
| InStr([start，]string1,string2[,compare]) |

表达式:

InStr([起始，] 接受搜索的字符串，被搜索的字符串[，匹配模式])

**instr函数参数**

* **start**

可选参数。为[数值表达式](https://baike.baidu.com/item/%E6%95%B0%E5%80%BC%E8%A1%A8%E8%BE%BE%E5%BC%8F)，设置每次搜索的起点。如果省略，将从第一个字符的位置开始。如果 start 包含 Null，将发生错误。**如果指定了 compare 参数，则一定要有 start 参数。**

* **string1**

必要参数。接受搜索的字符串[表达式](https://baike.baidu.com/item/%E8%A1%A8%E8%BE%BE%E5%BC%8F)。

* **string2**

必要参数。被搜索的字符串表达式。

* **compare**

可选参数。指定字符串比较。如果 compare 是 Null，将发生错误。如果省略 compare，Option Compare 的设置将决定比较的类型。

Compare 参数设置为：

|  |  |  |
| --- | --- | --- |
| 常数 | 值 | 描述 |
| **vbUseCompareOption** | -1 | 使用**Option Compare** 语句设置执行一个比较。 |
| **vbBinaryCompare** | 0 | 执行一个二进制比较（即 英文字符区分大小写）。 |
| **vbTextCompare** | 1 | 执行一个按照原文的比较（即 英文字符**不**区分大小写）。 |
| **vbDatabaseCompare** | -2 | 仅适用于Microsoft Access，执行一个基于数据库信息比较。 |

字母转换函数：

 UCase()函数用来将小写字母转化为大写字母。

 Lcase()将大写字母转换成小写字母

插入字符串语句：

 Mid（字串符，位置，l）=子字串符

 从“字串符”的“位置”开始的字符用“子字串符”代替 如果有l自变量，则替换的内容是“子字串符”左部的l个字符，l是可选参数

 例：dim a as string，b as string

 A=“abcdef”

 B=“111”

 Mid（a，2）=b

 A的值是a111ef

 Dim a as string，b as string

 A=“abcdef”

 B=“111”

 Mid（a，2，1）=b

 A的值是a1cdef 这里的1控制的是代替值的个数

算数运算符：

 ^ 幂 3^2 3的2次幂

* 取负

\* 乘法

/ 浮点除法

\ 整数除法

Mod 取模 例：6 mod 4 取余

+ 加

* 减

& 链接字符串

关系运算符：

 = 相等

 <>或>< 不相等

 < 小于

> 大于

<= 小于等于

>= 大于等于

 逻辑运算：

 Not 非 就是取反 例：not（a）

 And 与 例：a and b

 Or 或

 Xor 异或

 Eqv 等价 都为1或者都为0 才为真

 Imp 蕴含 当第一个表达式为true ，且第二个表达式为false 结果为false

 其他的情况都为true。 例： a imp b

 数据的输出

Print：

 例：print”啊啊啊啊啊啊是0”

 如果是数值型字串符 输出的时候前后都有一个空格

 当输出多个表达式或字符串时，各表达式可以用逗号，分号，空格隔开，

 如果输出的各表达式之间用逗号分隔，在这种情况下以14个字符位置为单位把

 一个输出行分为若干个区段，逗号后面的表达式在下个区段输出，

 如果用分号空格作为分隔符则按紧凑的格式输出。

Tab函数：

 Tab（n）

 在print将光标移到n的位置 n的值是整数

 例：print tab（5）“1111”

Spc函数：

 Spc（n）

 在print中添加空格 n的值是整数

Space函数：

 Space（n）

 在print中添加空格 这个函数可以用加法连接

 例：print “aaa”+space(1)+”11”

Format函数：

 Format（数值表达式，格式字符串）

 该函数的功能是按格式字符串指定的格式输出数值表达式的值。如果省略

 格式字符串，则功能与str函数相同，唯一差别是当把正数转换成字符串时

 Str函数会在字符前面留一个空格，而format函数不留空格。

 格式说明符：

 # 数字不在前面或后面补0

 0 数字在前面或后面补0

 . 小数点

 ，千位分隔符

 % 百分比符号

 $ 美元符号

- + 负 正号

E+ e- 指数符号

 例：print format（1111.111，”###.##”）

Cls函数：

 对象.cls

 清除print方法显示的文本或图片框中显示内容，并将光标移到左上角（0，0）

Texheight 和 textwidth函数：

 测试输出字符串的宽度与高度

 例：print texheighe（“1111”），textwidth（”1111”）

 数据的输入

 Inputbox函数：

 InputBox是[Visual Basic](https://baike.baidu.com/item/Visual%20Basic)和[VBS](https://baike.baidu.com/item/VBS)中的函数，功能是弹出一个对话框，在其中显示提示，等待用户输入文字并按下按钮，然后返回用户输入的文字。右图就是InputBox函数的效果，其中展示了InputBox函数的三个重要参数Prompt、Title和Default的位置。该函数与[MsgBox](https://baike.baidu.com/item/MsgBox)函数类似，区别就是InputBox函数用于简单地接受用户输入的一些文本。

InputBox函数效果

## inputbox使用方法

编辑

### inputbox语法

InputBox(Prompt,[Title],[Default],[XPos],[YPos],[Helpfile],[Context])

### inputbox参数

|  |
| --- |
| **InputBox函数参数** |
| **部分** | **描述** |
| Prompt | 必需的。作为对话框消息出现的字符串表达式。Prompt的最大长度大约为1024个字符，由所用字符的宽度决定。如果Prompt的内容超过一行，则可以在每一行之间用回车符（Chr(13)）、换行符（Chr(10)）或是回车与换行符的组合（Chr(13) & Chr(10)，即[vbCrLf](https://baike.baidu.com/item/vbCrLf)）将各行分隔开来。 |
| Title | 可选的。显示对话框标题栏中的字符串表达式。如果省略，则把应用程序名放入标题栏中。 |
| Default | 可选的。显示文本框中的字符串表达式，在用户输入前作为缺省值。如果省略，则文本框为空。 |
| XPos | 可选的。数值表达式，**与YPos一起出现**，指定对话框的左边与屏幕左边的水平距离。如果省略，则对话框会在水平方向居中。 |
| YPos | 可选的。数值表达式，**与XPos一起出现**，指定对话框的顶端与屏幕顶端的距离。如果省略，则对话框被放置在屏幕垂直方向距底端大约三分之一的位置。 |
| Helpfile | 可选的。字符串表达式，识别用来向对话框提供上下文相关帮助的帮助文件。如果提供了Helpfile，则也必须提供Context。 |
| Context | 可选的。数值表达式，由帮助文件的作者指定给适当的帮助主题的帮助上下文编号。如果提供了Context，则也必须提供Helpfile。 |

### inputbox说明

如果同时提供了Helpfile与Context，用户可以按F1来查看与Context相应的帮助主题。某些主应用程序（支持使用Visual Basic for Applications的应用程序），例如[Microsoft Excel](https://baike.baidu.com/item/Microsoft%20Excel)，会在对话框中自动添加一个帮助按钮。如果用户单击确定或按下回车，则InputBox函数返回文本框中的内容。如果用户单击取消，则此函数返回一个长度为零的字符串（""）。

在VB中，如果没有同时提供XPos和YPos，不会提示语法错误，但单独出现的XPos或YPos将被忽略，相当于省略了这两个参数。但在VBS中，单独出现的XPos或YPos会生效，但另一个没有确定的参数将使用默认值。

## inputbox代码示例

编辑

'程序的作用是用InputBox函数输入2个数据之后，把他们连接起来，用MsgBox函数显示。
　　Private Sub Command1\_Click()
　　Dim a As String, b As String
　　a = InputBox("输入第一个数据：")
　　b = InputBox("输入第二个数据：")
　　MsgBox a & b
　　End Sub

弹出窗口：

 MsgBox (Prompt [,Buttons] [,Title] [,Helpfile,Context])

### msgbox参数

|  |
| --- |
| **MsgBox函数参数** |
| **部分** | **描述** |
| Prompt | 必选。字符串[表达式](https://baike.baidu.com/item/%E8%A1%A8%E8%BE%BE%E5%BC%8F)，显示在对话框中的消息。Prompt的最大长度大约为1024个字符，由所用字符的字节大小决定。如果Prompt的内容超过一行，则可以在每一行之间用回车符（Chr(13)）、换行符（Chr(10)）或是回车与换行符的组合（Chr(13) & Chr(10)，即[vbCrLf](https://baike.baidu.com/item/vbCrLf)）将各行分隔开来。 |
| Buttons | 可选。数值表达式，是一些数值的总和，指定所显示的按钮的数目及形式、使用的图标样式（及声音），缺省按钮以及消息框的强制性等。如果省略，则其缺省值为0。具体数值见“参数设置值”表格。 |
| Title | 可选。字符串表达式，在对话框标题栏中显示的内容。如果省略Title，则将应用程序标题（App.Title）放在标题栏中。 |
| Helpfile | 可选。字符串表达式，用来向对话框提供上下文相关帮助的帮助文件。如果提供了Helpfile，则也必须提供Context。 |
| Context | 可选。数值表达式，由帮助文件的作者指定给适当的帮助主题的帮助上下文编号。如果提供了Context，则也必须提供Helpfile。 |

### msgbox参数设置值

|  |
| --- |
| **Buttons参数设置值** |
| **常数** | **值** | **描述** |
| **对话框中显示的按钮的类型与数目** |
| vbOKOnly | 0 | 只显示“确定”按钮。（缺省） |
| vbOKCancel | 1 | 显示“确定”和“取消”按钮。 |
| vbAbortRetryIgnore | 2 | 显示“终止”、“重试”和“忽略”按钮。 |
| vbYesNoCancel | 3 | 显示“是”、“否”和“取消”按钮。 |
| vbYesNo | 4 | 显示“是”和“否”按钮。 |
| vbRetryCancel | 5 | 显示“重试”和“取消”按钮。 |
| **图标的样式（根据系统设置，可能伴有声音）** |
| vbCritical | 16 | 显示“错误信息”图标。 |
| vbQuestion | 32 | 显示“询问信息”图标。 |
| vbExclamation | 48 | 显示“警告消息”图标。 |
| vbInformation | 64 | 显示“通知消息”图标。 |
| **默认按钮** |
| vbDefaultButton1 | 0 | 第一个按钮是默认按钮。（缺省） |
| vbDefaultButton2 | 256 | 第二个按钮是默认按钮。 |
| vbDefaultButton3 | 512 | 第三个按钮是默认按钮。 |
| vbDefaultButton4 | 768 | 第四个按钮是默认按钮。 |
| **对话框的强制返回性** |
| vbApplicationModal | 0 | 应用程序强制返回；应用程序一直被挂起，直到用户对消息框作出响应才继续工作。 |
| vbSystemModal | 4096 | 系统强制返回；全部应用程序都被挂起，直到用户对消息框作出响应才继续工作。 |
| **对话框特殊设置** |
| vbMsgBoxHelpButton | 16384 | 将帮助按钮添加到消息框。 |
| vbMsgBoxSetForeground | 65536 | 指定消息框窗口作为前景窗口。 |
| vbMsgBoxRight | 524288 | 文本为右对齐。 |
| vbMsgBoxRtlReading | 1048576 | 指定文本应为在希伯来和阿拉伯语系统中的从右到左显示。 |

这些常数都是Visual Basic for Applications（[VBA](https://baike.baidu.com/item/VBA)）指定的。所以，可以在程序代码中使用这些常数名称，而不使用实际数值。

### msgbox返回值

|  |
| --- |
| **MsgBox函数返回值** |
| **常数** | **值** | **描述** |
| vbOK | 1 | 单击了“确定”按钮。 |
| vbCancel | 2 | 单击了“取消”按钮。 |
| vbAbort | 3 | 单击了“终止”按钮。 |
| vbRetry | 4 | 单击了“重试”按钮。 |
| vbIgnore | 5 | 单击了“忽略”按钮。 |
| vbYes | 6 | 单击了“是”按钮。 |
| vbNo | 7 | 单击了“否”按钮。 |

### msgbox说明

在提供了Helpfile与Context参数的时候，用户可以按F1来查看由Context参数指定的帮助主题。像[Microsoft Excel](https://baike.baidu.com/item/Microsoft%20Excel)这样的一些主机应用程序就会在

MsgBox函数效果

对话框中自动添加一个帮助按钮。

如果对话框显示“取消”按钮，则按下Esc键与单击“取消”按钮的效果相同。如果对话框中有帮助按钮，则对话框中提供与上下文相关的帮助。但是，直到其它按钮中有一个被单击之前，都不会返回任何值。

标签控件

Fontname=’”宋体” 设置字体

Fontsize=20 设置字体大小

Fontbold=true 设置粗体

Fontitlic=true 设置斜体

Fontunderline=true 设置下划线

Fontstrikethru=true 设置删除线

Aligrnment 值为0时：标题靠左显示

 值为1时：标题靠右显示

 值为2时：标题居中显示

Autosize 设置为true：则按照字体的大小长度自动调整大小

 设置为false：则保持字体设计大小

Boderstyle 值为0时：无边框

 值为1时：有边框

Caption 设置标签中的文本

Backstyle 值为1时：不透明

 值为0时：透明

Wordwrap 值为1时垂直显示，但autosize也要为1 输入的时候空格符换行。

 文本控件

multiline 为true时：可以是多行文本 使用ctrl+enter换行

scrollbars：

 值为0 没有滚动条

 值为1 只有水平滚动条

 值为2 只有垂直滚动条

 值为3 水平垂直滚动条都有

 只有multiline为true时才可以添加滚动条，此属性不能在运行的时候设定

Locked：

 值为true时 不可以对内容进行编辑

 值为false时 可以对内容进行编辑

passwordchar 用于口令输入 输入的时候不显示输入的内容

sellength 定义文本的结束位置 例：x=text.sellength （sellength，selstart，seltext综合使用）

selstart 定义当前文本的起始位置 从0开始

seltext 反馈选中的字符

这几个都跟选区有关的，就是选中一串字符串，选中的会变蓝色
selstart就是选区的第一个字符
sellength就是选中的长度
seltext就是选中的字符了，就是这串字符串变蓝色。
比如：
文本框text1.text="1234567890"
现在选中"456"
就这样
text1.selstart=3
text1.selength=3
就行了
如果你想得到鼠标选中的字符串，那就这样str1=text1.seltext

change 事件：当用户输入信息时改变或改变文本内容时触发

gotfocus 事件：光标在这个控件的时候触发

lostfocus 事件：光标离开这个控件的时候触发

获取焦点：

 例： 控件.setfocus

 不能在窗体的加载事件中对控件使用setfocus方法。

 图形控件

Currentx：移动x坐标 例：对象.currentx =500

Currenty：移动y坐标

Stretch属性：

 自动调整图像框中的图形大小，当为true时 自动放大或缩小到图像框的大小。

Loadpicture函数：

 运行的时候更改图片

例：p1.picture=loadpicture（”c\图片\1.jpg”）

 **直线和图形**

Brdercolor： 设置颜色

Brderstyle：设置线性的样式

Borderwidth： 设定边线的宽度

Backstyle：

 0 透明状态

 1 不透明状态

Fillcolor：

 设定图形内部颜色

Fillstyle：

 设定填充图形

Shape：

 设定形状

 **按钮控件**

Cancel属性：

 按esc和单机命令按钮的效果相同 在同一个窗体中只允许有一个按钮这只为true （值为true启用该功能）

Default属性：

 按回车键和单机命令按钮的效果相同 同一个窗体中只允许有一个按钮这只为true（值为true启用该功能）

Style属性：

 按钮的样式

值为0 标准样式

值为1 图形样式 可以为按钮添加图片来显示

Picture属性：

 当style为1时 设置按钮的图片

Downpicture属性：

 当按下的时候显示的图片 style必须为true

Disabledpicture属性：

 按钮不可用时显示的图片 style必须为true

 **单选按钮和复选框**

Value属性：

 用来表示复选框或单选按钮的状态，对于单选按钮来说 value属性设置为true

 时，该单选按钮时打开的，按钮中心有一个圆点，如果为false 则单选按钮是

 关闭的，单选按钮一组是能选择一个。

 对于复选框来说 value可以设置为0 1 2

 为0时 表示没有选择复选框

 为1时 表示选中复选框

 为2时 复选框为灰选

 判断是否选中：控件名.value=true

Alignment属性：

 值为0 标题在控件右侧显示

 值为1 标题在控件左侧显示

 **列表框**

Columns属性:

 当设置为0时，所有的项目呈单列显示

 当设置为1时，呈多行多列显示

List属性：

 List保存了列表框的所有数组。

 例 x=列表框,list（下标）下标从0开始 （读取表项的值）

Listcount属性：

 列出列表框的表项数目。

Listlndex属性：

 已选中列表框的表项，显示下标。（返回或设置当前项目的索引）

 例 print list1.list（list1.listlndex） 返回当前选择的表项

 List.listlndex=0 设置当前的表项

Multiselect属性：

 值为0 每次只能选择一项

 值为1 可以同时选择多个项，后续的选择不会取消前面所选择的项

 值为2 选择指定范围的表项，单击所选范围的第一项，然后按下shift键

 不要松开，并单击所要选择范围的最后一项，如果按住ctrl键，并单击列表

 框中的项目，则可不连续的选择多个表项。

 如果选择多个表项，listlndex和text属性只表示最后一次的选择值

Selected属性：

 该属性实际上是一个数组，各个元素的值为true或false，每个元素与列表框

 中的一项对应，当元素的值为true，表明选择了该项，如果为false表示未选

 择。

 列表框.selected(索引值)

Selcount属性：

 用于读取列表框中所选，项的数目（表列总共选了几个）

Sorted属性：

 如果sorted的属性设置为true，则按字母，数字升序排列，如果为false则

 按添加表项的先后次序排列，

Style属性：

 列表框样式

 值为0 标准形式

 值为1 复选框形式

Text属性：

 最后一次选中表项的文本

Addltem属性：

 在列表框插入一行表项文本

 Object.addltem item,indx

 Object 列表对象

 Item 字串符

 Index 插入的位置

Clear属性：

 清除列表框的全部内容

 Object.clear

Removeltem属性：

 删除列表框指定的表项

 Object.removeltem 索引值

 **组合框**

Style属性：

 值为0 下拉式组合框，可以输入文本或下拉列表选择

 值为1 简单组合框，如果项目的总高度比组合框高度大，则自动加滚动条

 值为2 下拉式列表框 不可以通过设置来设定text属性

Text属性：

 用户所选择的文本或直接从编辑区输入的文本

只有style为1才能接受dblclick双击事件，组合框的属性与列表框的属性基本一样。

  **滚动条**

Max：滚动条所能表示的最大值

Min：滚动条表示的最小值

Largechange：单击滚动条中的滚动框前面或后面的部位时，增加或减小的增量

Smallchange：单击滚动条两端箭头时，增加或减小的增量值

Value：滚动条的当前位置

Scroll事件：在滚动条内拖动滚动框时触发的事件

Change事件：改变滚动框位置后触发事件

  **计时器**

最大时间间隔不能超过65秒

Interval属性：指定时间间隔，单位是毫秒

Enabled属性：控制定时器的起停

Timer事件：每间隔时间触发一次事件

  **选择控制结构**

单分支if then语句：

 if 条件表达式 then 语句 （）单行形式

 当执行的时候首先检查条件表达式，如果条件为true，则执行then后面的语句，如果为false则不执行任何语句

 If 条件表达式 then

 语句块

 End if （块形式）

语句块可以是单个语句，也可以是多个语句。多个语句可以写在多行中，也可以写在一行中，并用冒号隔开“：”。

双分支 if then else 语句：

 If 条件表达式 then 语块1 else 语块2 （单行形式）

当条件表达式为true时，执行语句块1，否则执行语句块2。

 If 条件表达式 then

 语块1

 Else

 语块2

 End if

块形式中的 end if不可以省略。在某些情况下，可能有多个条件为true，但也只能执行一个语句块。

多分支if then elseif语句：

 If 条件表达式1 then

 语句块1

 Elseif 条件表达式2 then

 语句块2

 Elseif 条件表达式3 then

 语句块3

 Else

 语句块4

 End if

根据不同条件确定执行那个语句，执行顺序是表达式1，表达式2.。。。。。。。一旦条件表达式为true，就执行该条件下的语句块。否则执行else 下的语句块4。

Llf函数：

 Result=llf（条件，true部分，false部分）

 Result是函数的返回值，当条件为真执行true部分，当条件为假执行false部分。

Select case语句：

Select case 测试表达式

Case 表达式列1

 语句块1

Case 表达式列2

 语句块2

 Case else

 语句块3

End select

计算测试表达式的值，用这个值与后面的表达式1 表达式2.。。。。。比较，若有匹配的则执行case表达式后面的语句块，当测试表达式的值与 表达式的值不匹配时，若有case else语句，就执行case else下面的语句块，若没有case else，就结束select case。

表达式的样式可以是：

 Case x+100 表达式列表为表达式

 Case 1，4，7 表示条件在1，4，7的时候内取值

 Case 50 to 60 表示条件取值范围为50-60

 **循环控制结构**

For循环：

 For 循环变量=初值 to 终值 [step 步长]

 循环体

 Next 循环变量

 For i=1 to 10 step 1

 A=a+i

 Next a

以指定次数重复执行一组语句。

**For** counter = start To end [Step step]
[statements]
[Exit For]
[statements]
**Next**

#### 参数

counter

用做循环计数器的数值变量。这个变量不能是数组元素或用户自定义类型的元素。

start

counter 的初值。

end

counter 的终值。

step

**counter** 的步长。如果没有指定，则 **step** 的默认值为 1。

statements

**For** 和 **Next** 之间的一条或多条语句，将被执行指定次数。

#### 说明

**step** 参数可以是正数或负数。**step** 参数值决定循环的执行情况，如下所示：

|  |  |
| --- | --- |
| **值** | **如果 ... 则循环执行** |
| 正数或 0  | counter <= end |
| 负数 | counter >= end |

当循环启动并且所有循环中的语句都执行后，**step** 值被加到 **counter** 中。这时，或者循环中的语句再次执行（基于循环开始执行时同样的测试），或者退出循环并从 **Next** 语句之后的语句继续执行。

**注意** 在循环体内改变 **counter** 的值，将会使程序代码的阅读和调试变得更加困难。

**Exit For** 只能用于 **For Each...Next** 或 **For...Next** 结构中，提供另一种退出循环的方法。可在语句中的任意位置放置任意个 **Exit For** 语句。**Exit For** 经常和条件判断语句一起使用（例如 **If...Then**），并立即将控制权转移到 **Next** 之后的语句。

可以将一个 **For...Next** 循环放置在另一个 **For...Next** 循环中，组成嵌套循环。每个循环中的 **counter** 要使用不同的变量名。下面的结构是正确的：

**For** I = 1 **To** 10

 **For** J = 1 **To** 10

 **For** K = 1 **To** 10

 **. . .**

 **Next**

 **Next**

Next

While...Wend 语句

当指定的条件为 **True** 时，执行一系列的语句。

**While** condition
&nbsp；  Version [**statements**]
**Wend**

#### 参数

condition

数值或字符串表达式，其计算结果为 **True** 或 **False**。如果 **condition** 为 Null，则 **condition** 被当作 **False**。

statements

在条件为 **True** 时执行的一条或多条语句。

#### 说明

如果 **condition** 为 **True**，则 **statements** 中所有 **Wend** 语句之前的语句都将被执行，然后控制权将返回到 **While** 语句，并且重新检查 **condition**。如果 **condition** 仍为 **True**，则重复执行上面的过程。如果不为 **True**，则从 **Wend** 语句之后的语句继续执行程序。

**While...Wend** 循环可以是多层嵌套结构。每个 **Wend** 与最近的 **While** 语句对应。

**注意** **Do...Loop** 语句提供一种结构化与适应性更强的方法以执行循环。

下面例子举例说明如何使用 **While...Wend** 语句:

Dim Counter

Counter = 0 '初始化变量。

**While** Counter < 20 ' 测试计数器的值。

 Counter = Counter + 1 ' 增加计数器。

 Alert Counter

**Wend** ' 计数器大于 19 时终止循环。

DO while loop循环：

 Do while condition

 Statements

 Loop

 当执行这个do循环时，首先测试condition表达式，如果为false，则跳过所有语句，只要为true，就会多次执行statenents语句，直到condition为假false。

 Do until e condition

 Statements

 Loop

 当执行这个do循环时，首先测试condition表达式，如果为true，则跳过所有语句，只要为false，就会多次执行statenents语句，直到condition为假true。

 Do

 Statements

 Loop while codition

 先执行一次statements语句，然后进行判断，如果为真就继续执行，否则跳出。

 Do

 Statements

 Loop until codition

 先执行一次statements语句，然后进行判断，如果为假就继续执行，否则跳出。

Exit do 退出当前do循环。

  **数组**

定义数组的格式：

 声明固定大小的数组

  有三种方法声明固定大小的数组，用哪一种方法取决于数组应有的有效范围：

  建立公用数组，在模块的声明段用   Public   语句声明数组。

  建立模块级数组，在模块的声明段用   Private   语句声明数组。

  建立局部数组，在过程中用   Dim  语句声明数组。

  设定上下界

  声明数组时，在数组名之后跟一个用括号括起来的上界。上界不得超过   Long   数据类型的范围（-2,147,483,648   到   2,147,483,647）。例如，下列数组声明可出现在模块的声明段：

  Dim   Counters   (14)   As   Integer                   '15   个元素。

  Dim   Sums   (20)   As   Double                         '21   个元素。

  为建立公用数组，直接用   Public   取代   Dim。

  Public   Counters   (14)   As   Integer

  Public   Sums   (20)   As   Double

  在过程之中同样的声明使用   Dim：

  Dim   Counters   (14)   As   Integer

  Dim   Sums   (20)   As   Double

  第一个声明建立了一个有   15   个元素的数组，其索引号从   0   到   14   。第二个声明建立了一个有   21   个元素的数组，其索引号从   0   到   20   。缺省的下界为   0。

  为了规定下界，用关键字   To   显式提供下界（为   Long   数据类型）：

  Dim   Counters   (1   To   15)   As   Integer

  Dim   Sums   (100   To   120)   As   String

  在前述声明中，Counters   的索引值范围从   1   到   15，而   Sums   的索引值范围从   100   到   200。

  包含其它数组的数组

  有可能建立   Variant   数据类型数组，并与不同数据类型的数组共居一处。以下代码建立两个数组，一个包含整数，而另一个包含   字符串。然后声明第三个   Variant   数组，并将整数和字符串数组放置其中：

  Private   Sub   Command1\_Click   ()

        Dim   intX   As   Integer       '声明计数器变量。

        '声明并放置整数数组。

        Dim   countersA   (5)   As   Integer

              For   intX   =   0   To   4

                    countersA   (intX)   =   5

              Next   intX

        '声明并放置字符串数组。

              Dim   countersB   (5)   As   String

                    For   intX   =   0   To   4

                          countersB   (intX)   =   "hello"

                    Next   intX

        Dim   arrX   (2)   As   Variant                   '声明拥有两个成员的新数组。

              arrX   (1)   =   countersA   ()             '将其它数组移居到数组。

              arrX   (2)   =   countersB   ()

              MsgBox   arrX   (1)   (2)                   '显示每一个数组的成员。

              MsgBox   arrX   (2)   (3)

  End   Sub

  多维数组

  有时需要追踪记录数组中的相关信息。例如，为了追踪记录计算机屏幕上的每一个像素，需要引用它的   X、Y   坐标。这时应该用多维数组存储值。

  可用   Visual   Basic   声明多维数组。例如，下面的语句声明了一个过程内的   10   ×   10   的二维数组。

  Static   MatrixA   (9,   9)   As   Double

  可用显式下界来声明两个维数或两个维数中的任何一个：

  Static   MatrixA   (1   To   10,   1   To   10)   As   Double

  可以将所有这些推广到二维以上的数组。例如：

  Dim   MultiD   (3,   1   To   10,   1   To   15)

  这个声明建立了三维数组，大小为   4   ×   10   ×   15。元素总数为三个维数的乘积，为   600。

  注意   在增加数组的维数时，数组所占的存储空间会大幅度增加，所以要慎用多维数组。使用   Variant   数组时更要格外小心，因为他们需要更大的存储空间。

  用循环操作数组

  可以用   For   循环嵌套有效的处理多维数组。例如，在   MatrixA   中基于每个元素在数组中的位置为其赋值：

  Dim   I   As   Integer,   J   As   Integer

  Static   MatrixA(1   To   10,   1   To   10)   As   Double

  For   I   =   1   To   10

        For   J   =   1   To   10

              MatrixA   (I,   J)   =   I   \*   10   +   J

        Next   J

  Next   I

  要创建动态数组，请按照以下步骤执行：

  （如果希望数组为公用数组，则）用   Public   语句声明数组，或者，（如果希望数组为模块级，则）在模块级用   Dim

语句声明数组，或者（如果希望数组为局部数组，则）在过程中用   Static   或   Dim

语句声明数组。给数组附以一个空维数表，这样就将数组声明为动态数组。

  Dim   DynArray   ()

  用   ReDim   语句分配实际的元素个数。

  ReDim   DynArray   (X   +   1)

  ReDim   语句只能出现在过程中。与   Dim   语句、Static   语句不同，ReDim   语句是一个可执行语句，由于这一语句，应用程序在运行时执行一个操作。

  ReDim   语句支持这样的语法，它与固定数组中使用的语法相同。对于每一维数，每个   ReDim   语句都能改变元素数目以及上下界。但是，数组的维数不能改变。

  ReDim   DynArray   (4   to   12)

  例如，用第一次声明在模块级所建立的动态数组   Matrix1：

  Dim   Matrix1   ()   As   Integer

  然后，在过程中给数组分配空间：

  Sub   CalcValuesNow   ()

        .

        .

        .

        ReDim   Matrix1   (19,   29)

  End   Sub

  这里的   ReDim   语句给   Matrix   分配一个   20   ×   30   的整数矩阵（元素总大小为   600）。还有一个办法，用变量设置动态数组的边界：

  ReDim   Matrix1   (X,   Y)

  注意   您可以将字符串赋值给大小可变的字节数组。一个字节数组也可以被赋值给一个可变长的字符串。一定要注意字符串中的字节数会随平台而变化。同一个字符串在   Unicode   平台上的字节数是它在非   Unicode   平台上的两倍。

  保留动态数组的内容

  每次执行   ReDim   语句时，当前存储在数组中的值都会全部丢失。Visual   Basi   重新将数组元素的值置为

Empty（对   Variant   数组）、置为   0（对   Numeric   数组）、置为   零长度字符串（对   String

  数组）或者置为   Nothing（对于对象的数组）。

在为新数据准备数组，或者要缩减数组大小以节省内存时，这样做是非常有用的。有时希望改变数组大小又不丢失数组中的数据。使用具有

Preserve   关键字的   ReDim   语句就可做到这点。例如，使用   UBound

函数引用上界，使数组扩大、增加一个元素，而现有元素的值并未丢失：

  ReDim   Preserve   DynArray   (UBound   (DynArray)   +   1)

  在用   Preserve   关键字时，只能改变多维数组中最后一维的上界；如果改变了其它维或最后一维的下界，那么运行时就会出错。

所以可这样编程：

  ReDim   Preserve   Matrix   (10,   UBound   (Matrix,   2)   +   1)

  而不可这样编程：

  ReDim   Preserve   Matrix   (UBound   (Matrix,   1)   +   1,   10)

定义数组下标起始值：

 Option base 1 下标只能是从1或0开始，不能出现在过程当中。

测试数组函数：

 Lbounf （数组，维） 测试数组的下标

 Ubound（数组，维） 测试数组的上标

 例：Dim a（4）as integer ，b as integer

 b=Lbounf （a）

 例：Dim a（2，1）as integer ，b as integer

 b=Lbounf （a，0）

数组的清除：

 在一个程序中，同一数组只能用Dim语句定义一次。但有时可能需要清除数组的内容或对数组重新定义，这可以用：Erase语句来实现。
　　格式：Erase(数组名)[，(数组名)]
　　功能：用于重新初始化静态数组的元素，或者释放动态数组的存储空间。
　　例如：
　　Dim T(10)As Integer
　　Erase T
　　examda说明：
　　(1)在Erase语句中，只需给出数组名，不带括号和下标。
　　(2)在Erase语句用于静态数组时，如果这个数组是数值数组，则把数组中的所有数组元素置为0；如果是字符串数组，则把所有数组元素置为空字符串。
　　(3)当把Erase语句用于动态数组时，将删除整个数组结构并释放该数组所占用的内存区。也就是说，动态数组经Erase语句执行后，即不复存在；而静态数组经Erase后，仍然存在，只是其内容被清空。
　　(4)Erase语句释放动态数组所使用的内存，在下一次引用该动态数组之前，必须用ReDim语句，重新定义该数组。

Dim NumArray(9)

Dim DynamicArray()

ReDim DynamicArray(9) '分配存储空间。

**Erase** NumArray ' 每一元素都被重新初始化。

**Erase** DynamicArray '释放数组占用的内存。

数组的赋值：

 **Array(***arglist***)**

**arglist** 参数是赋给包含在 **Variant** 中的数组元素的值的列表（用逗号分隔）。如果没有指定此参数，则将会创建零长度的数组。

#### 说明

用于引用数组元素的表示符，由跟随有括号的变量名组成，括号中包含指示所需元素的索引号。在下面的示例中，第一条语句创建名为 A 的变量。第二条语句将一个数组赋值给变量 A。最后一条语句将包含在第二个数组元素中的值赋值给另一个变量。

Dim A

A = **Array(**10**,**20**,**30**)**

B = A(2) ' B is now 30。

**注意** 未作为数组声明的变量仍可以包含数组。虽然包含数组的 **Variant** 变量与包含 **Variant** 元素的数组变量有概念上的不同，但访问数组元素的方法是相同的。

**Sub和函数过程**

Sub过程建立：

 声明 **Sub** 过程的名称、参数以及构成其主体的代码。

 [Public [Default]| Private] **Sub** name [(
  arglist
)]
[statements]
[Exit Sub]
[statements]
**End Sub**

#### 参数

Public

表示 **Sub** 过程可被所有脚本中的所有其他过程访问。

Default

只与 **类**块中的**Public** 关键字连用，用来表示 **Sub** 过程是类的默认方法。 如果在类中指定了不止一个**Default**过程，就会出错。

Private

表示 **Sub** 过程只可被声明该过程的脚本中的其他过程访问。

name

**Sub** 的名称，遵循标准变量命名约定。

arglist

代表在调用时要传递给 **Sub** 过程的参数的变量列表。用逗号隔开多个变量。

statements

在 **Sub** 过程主体内所执行的任何语句组。

**arglist** 参数包含下列语法和部分：

[ByVal | ByRef] varname[( )]

#### 参数

ByVal

表示该参数是按值传递的。

ByRef

表示该参数按引用传递。

varname

代表参数的变量名称，遵循标准变量命名约定。

#### 说明

如没有显式地指定使用 Public 或 Private，则 **Sub** 过程默认为公用，即它们对于脚本中的所有其他过程都是可见的。**Sub** 过程中局部变量的值在调用过程中不被保留。

不能在任何其他过程（例如，**Function** 或 **Property Get**）中定义 **Sub** 过程。

使用 **Exit Sub** 语句可以立即从 **Sub** 过程中退出。程序继续执行调用 **Sub** 过程的语句之后的语句。可以在 **Sub** 过程中任意位置出现任意个 **Exit Sub** 语句。

与 **Function** 过程相似之处是：**Sub** 过程是一个可以获取参数，执行一系列语句以及可改变其参数的值的独立过程。而与 **Function** 过程不同之处是：Function 过程可以返回值，而 **Sub** 过程不能用于表达式中。

可以使用过程名并跟随相应的参数列表来调用 **Sub** 过程。关于如何调用 **Sub** 过程的详细说明信息，请参阅 **Call** 语句。

**小心** **Sub** 过程可以是递归的，即该过程可以调用自己来完成某个给定的任务。但是递归可能会导致堆栈溢出。

在 **Sub** 过程中使用的变量分为两类：一类是在过程内显式声明的，另一类则不是。在过程内显式声明的变量（使用 **Dim** 或等效方法）总是局部变量。对于那些没有在过程中显式声明的变量也是局部的，除非在该过程外更高级别的位置显式地声明它们。

**小心** 过程可以使用没有在过程内显式声明的变量，但只要有任何 script 级定义的名称与之同名，就会产生名称冲突。如果过程中引用的未声明的变量与其他的过程、常数或变量的名称相同，则会认为过程引用的是脚本级的名称。要避免这类冲突，请使用 **Option Explicit** 语句可强制显式声明变量。

调用 Sub 过程：

与 Sub 过程不同，在表达式中，Sub 过程不能用其名字调用。调用 Sub 过程的是一个独立的语句。Sub 过程还有一点与函数不一样，它不会用名字返回一个值。但是，与 Function过程一样，Sub 过程也可以修改传递给它们的任何变量的值。

调用 Sub 过程有两种方法：

'以下两个语句都调用了名为 MyProc 的 Sub 过程。

Call MyProc (FirstArgument, SecondArgument)

MyProc FirstArgument, SecondArgument

注意，当使用 Call 语法时，参数必须在括号内。若省略 Call 关键字，则也必须省略参数两边的括号。

函数过程建立：

 函数过程的语法是：

Private|Public][Static] Function procedurename (arguments) [As type]
statements

End Function

与 Sub 过程一样，Function 过程也是一个独立的过程，可读取参数、执行一系列语句并改变其参数的值。与子过程不同，Function 过程可返回一个值到调用的过程。在Sub 过程与Function 过程之间有三点区别：

一般说来，让较大的语句或表达式的右边包含函数过程名和参数 (returnvalue = function)，这就调用了函数。

与变量完全一样，函数过程有数据类型。这就决定了返回值的类型。（如果没有 As 子句，缺省的数据类型为 Variant。）

给 procedurename 自身赋一个值，就可返回这个值。Function 过程返回一个值时，该值可成为较大表达式的一部分。

例如，下面是已知直角三角形两直角边的值，计算第三边（斜边）的函数：

Function Hypotenuse (A As Integer, B As Integer) As String

 Hypotenuse = Sqr (A ^ 2 + B ^ 2)

End Function

在 Visual Basic 中调用 Function 过程的方法和调用任何内部函数的方法是一样的：

Label1.Caption = Hypotenuse(CInt(Text1.Text), \_

CInt(Text2.Text))

strX = Hypotenuse (Width, Height)

指名传送：

 Sub qw（a as integer,b asinteger）

 Print a-b

 End sub

 Qw b:=3,a:=5 指名传送的时候可以不按照顺序来传送值。

引用：

 向过程传递参数的方式之一，即将参数的地址而不是值传递给过程。“引用方式”允许过程访问和更改实际变量。ByRef

 例：

 Sub cc (x as integer,v as integer) 默认是引用的方式

 X=x+100

 Y=y+100

 End sub

Dim a as integer,b as integer

 A=10

 B=20

 Cc a,b

 Print a,b

最后输出的结果是 110 120，引用是采取传地址的方式可以去改变地址的值，所以a跟b的值不再是10，20了。

传值：

向过程传递参数的方式之一，即将参数的值而不是地址传递给过程。“值方式”允许过程访问变量的一个副本，但是不能更改变量的实际值。ByVal

Sub cc (ByVal x as integer, ByVal v as integer)

 X=x+100

 Y=y+100

 End sub

可选参数：

 可选的。表示参数不是必需的。如果使用该选项，则 arglist 中的后续参数都必需是可选的，而且必须都使用 Optional 关键字声明。如果使用了 ParamArray，则任何参数都不能使用 Optional。

 Sub cc (ByVal x as integer, ByVal v as integer,optional c)

 X=x+100

 Y=y+100

 End sub

 调用的时候c可以不用赋值。

# IsMissing 函数

返回 **Boolean** 值，指出一个可选的 **Variant** 参数是否已经传递给过程。

**语法**

**IsMissing(***argname***)**

必要的 *argname* 参数包含一个可选的 **Variant** 过程参数名。

**说明**

使用 **IsMissing** 函数来检测在调用一个程序时是否提供了可选 **Variant** 参数。如果对特定参数没有传递值过去，则 **IsMissing** 返回 **True**；否则返回 **False**。如果 **IsMissing** 对某个参数返回 **True**，则在其它代码中使用这个丢失的参数将产生一个用户自定义的错误。如果对 **ParamArray** 参数使用 **IsMissing**，则函数总是返回 **False**。为了检测空的 **ParamArray**，可试看一下数组的上界是否小于它的下界。

**注意** **IsMissing** 对简单数据类型（例如 **Integer**或**Double）**不起作用，因为与**Variants**不同，它们没有“丢失”标志位的前提。正由于此，对于可选参数类型，可以指定缺省值。如果调用过程时，参数被忽略，则该参数将具有该缺省值，如下列示例中所示：

Sub MySub(Optional MyVar As String = "specialvalue")

 If MyVar = "specialvalue" Then

 ' MyVar 被忽略。

 Else

 ...

End Sub

在许多情况下，如果用户从函数调用中忽略，则可以通过使缺省值等于希望 MyVar 所包含的值来完全忽略 If MyVar 测试。这将使您的代码更简洁有效。

**Function 语句**

声明 **Function** 过程的名称，参数以及构成其主体的代码。

**语法**

[**Public** | **Private | Friend**] [**Static**] **Function** *name* [**(***arglist***)**] [**As** *type*]
[*statements*]
[*name* **=** *expression*]
[**Exit Function**]
[*statements*]
[*name* **=** *expression*]

**End Function**

**Function** 语句的语法包含下面部分：

|  |  |
| --- | --- |
| **部分** | **描述** |
| **Public** | 可选的。表示所有模块的所有其它过程都可访问这个 **Function** 过程。如果是在包含 **Option Private** 的模块中使用，则这个过程在该工程外是不可使用的。 |
| **Private** | 可选的。表示只有包含其声明的模块的其它过程可以访问该 **Function** 过程。 |
| **Friend** | 可选的。只能在类模块中使用。表示该 **Function** 过程在整个工程中都是可见的，但对于对象实例的控制者是不可见的。 |
| **Static** | 可选的。表示在调用之间将保留 **Function** 过程的局部变量值。**Static** 属性对在该 **Function** 外声明的变量不会产生影响，即使过程中也使用了这些变量。 |
| *name* | 必需的。**Function** 的名称；遵循标准的变量命名约定。 |
| *arglist* | 可选的。代表在调用时要传递给 **Function** 过程的参数变量列表。多个变量应用逗号隔开。 |
| *type* | 可选的。**Function** 过程的返回值的数据类型，可以是 Byte、布尔、Integer、Long、Currency、Single、Double、Decimal（目前尚不支持）、Date、String（除定长）、Object、Variant或任何用户定义类型。 |
| *statements* | 可选的。在 **Function** 过程中执行的任何语句组。 |
| *expression* | 可选的。**Function** 的返回值。 |

其中的 *arglist* 参数的语法以及语法各个部分如下：

[**Optional**] [**ByVal** | **ByRef**] [**ParamArray**] *varname*[**( )**] [**As** *type*] [**=** *defaultvalue*]

|  |  |
| --- | --- |
| **部分** | **描述** |
| **Optional** | 可选的。表示参数不是必需的。如果使用了该选项，则 *arglist* 中的后续参数都必须是可选的，而且必须都使用 **Optional** 关键字声明。如果使用了 **ParamArray**，则任何参数都不能使用 **Optional** 声明。 |
| **ByVal** | 可选的。表示该参数按值传递。 |
| **ByRef** | 可选的。表示该参数按地址传递。**ByRef** 是 Visual Basic 的缺省选项。 |
| **ParamArray** | 可选的。只用于 *arglist* 的最后一个参数，指明最后这个参数是一个 **Variant** 元素的 **Optional** 数组。使用 **ParamArray** 关键字可以提供任意数目的参数。**ParamArray** 关键字不能与 **ByVal**，**ByRef**，或 **Optional** 一起使用。 |
| *varname* | 必需的。代表参数的变量的名称；遵循标准的变量命名约定。 |
| *type* | 可选的。传递给该过程的参数的数据类型；可以是 **Byte、Boolean、Integer、Long、Currency、Single、Double、Decimal**（目前尚不支持）、**Date、String**（只支持变长）、**Object** 或 **Variant**。如果参数不是 **Optional**，则也可以是用户定义类型，或对象类型。 |
| *defaultvalue* | 可选的。任何常数或常数表达式。只对于 **Optional** 参数时是合法的。如果类型为 **Object**，则显式缺省值只能是 **Nothing**。 |

**说明**

如果没有使用 **Public、Private** 或 **Friend** 显式指定，则 **Function** 过程缺省为公用。如果没有使用 **Static**，则局部变量的值在调用之后不会保留。**Friend** 关键字只能在类模块中使用。但 **Friend** 过程可以被工程的任何模块中的过程访问。**Friend** 过程不会在其父类的类型库中出现，且 **Friend** 过程不能被后期绑定。

**小心** **Function** 过程可以是递归的；也就是说，该过程可以调用自己来完成某个特定的任务。不过，递归可能会导致堆栈上溢。通常 **Static** 关键字和递归的 **Function** 过程不在一起使用。

所有的可执行代码都必须属于某个过程。不能在另外的 **Function、Sub** 或 **Property** 过程中定义 **Function** 过程。

**Exit Function** 语句使执行立即从一个 **Function** 过程中退出。程序接着从调用该 **Function** 过程的语句之后的语句执行。在 **Function** 过程的任何位置都可以有 **Exit Function** 语句。

**Function** 过程与 **Sub** 过程的相似之处是： **Function** 过程是一个可以获取参数，执行一系列语句，以及改变其参数值的独立过程，而与**子**过程不同的是：当要使用该函数的返回值时，可以在表达式的右边使用 **Function** 过程，这与内部函数，诸如 **Sqr、Cos** 或 **Chr** 的使用方式一样。

在表达式中，可以通过使用函数名，并在其后用圆括号给出相应的参数列表来调用一个 **Function** 过程。请参阅 **Call** 语句关于如何调用 **Function** 过程的详细说明。

要从函数返回一个值，只需将该值赋给函数名。在过程的任意位置都可以出现这种赋值。如果没有对 *name* 赋值，则过程将返回一个缺省值：数值函数返回 0，字符串函数返回一个零长度字符串 ("")，**Variant** 函数则返回 Empty。如果在返回对象引用的 **Function** 过程中没有将对象引用赋给 *name* （通过 **Set**），则函数返回 **Nothing**。

下面的示例说明如何给一个名为 BinarySearch 的函数赋返回值。在这个示例中，将 **False** 赋给了该函数名，表示没有找到某个值。

Function BinarySearch(. . .) As Boolean

**. . .**

 '值未找到，返回一个 False 值。

 If lower > upper Then

 BinarySearch = False

 Exit Function

 End If

**. . .**

End Function

在 **Function** 过程中使用的变量分为两类：一类是在过程内显式声明的，另一类则不是。在过程内显式声明的变量（使用 **Dim** 或等效方法）都是局部变量。对于那些没有在过程中显式声明的变量，除非它们在该过程外更高级别的位置有显示地声明，否则也是局部的。

**小心** 过程可以使用没有在过程内显式声明的变量，但只要有任何在模块级别中定义的名称与之相同，就会产生名称冲突。如果过程中使用的未声明的变量与另一个过程，常数，或变量的名称相同，则会认为过程使用的是模块级的名称。显式声明变量就可以避免这类冲突。可以使用 **Option Explicit** 语句来强制显式声明变量。

**小心** Visual Basic 可能会重新安排数学表达式以提高内部效率。若 **Function** 过程会改变某个数学表达式中变量的值，则应避免在此表达式中使用该函

**Sub 语句**

声明子过程的名称，参数，以及构成其主体的代码。

**语法**

[**Private** | **Public** | **Friend**] [**Static**] **Sub** *name* [**(***arglist***)**]
[*statements*]
[**Exit Sub**]
[*statements*]

**End Sub**

**Sub** 语句的语法包含下面部分：

|  |  |
| --- | --- |
| **部分** | **描述** |
| **Public** | 可选的。表示所有模块的所有其它过程都可访问这个 **Sub** 过程。 如果在包含 **Option Private** 的模块中使用，则这个过程在该工程外是不可使用的。 |
| **Private** | 可选的。表示只有在包含其声明的模块中的其它过程可以访问该 **Sub** 过程。 |
| **Friend** | 可选的。只能在类模块中使用。表示该 **Sub** 过程在整个工程中都是可见的，但对对象实例的控制者是不可见的。 |
| **Static** | 可选的。表示在调用之间保留 **Sub** 过程的局部变量的值。**Static** 属性对在 **Sub** 外声明的变量不会产生影响，即使过程中也使用了这些变量。 |
| *name* | 必需的。**Sub** 的名称；遵循标准的变量命名约定。  |
| *arglist* | 可选的。代表在调用时要传递给 **Sub** 过程的参数的变量列表。多个变量则用逗号隔开。 |
| *statements* | 可选的。**Sub** 过程中所执行的任何语句组。 |

其中的 *arglist* 参数的语法以及语法各个部分如下：

[**Optional**] [**ByVal** | **ByRef**] [**ParamArray**] *varname*[**( )**] [**As** *type*] [**=** *defaultvalue*]

|  |  |
| --- | --- |
| **部分** | **描述** |
| **Optional** | 可选的。表示参数不是必需的关键字。如果使用了该选项，则 *arglist* 中的后续参数都必须是可选的，而且必须都使用 **Optional** 关键字声明。如果使用了 **ParamArray**，则任何参数都不能使用 **Optional**。 |
| **ByVal** | 可选的。表示该参数按值传递。 |
| **ByRef** | 可选的。表示该参数按地址传递。**ByRef** 是 Visual Basic 的缺省选项。 |
| **ParamArray** | 可选的。只用于 *arglist* 的最后一个参数，指明最后这个参数是一个 **Variant** 元素的 **Optional** 数组。使用 **ParamArray** 关键字可以提供任意数目的参数。**ParamArray** 关键字不能与 **ByVal**，**ByRef**，或 **Optional** 一起使用。 |
| *varname* | 必需的。代表参数的变量的名称；遵循标准的变量命名约定。 |
| *type* | 可选的。传递给该过程的参数的数据类型，可以是 Byte、Boolean、Integer、Long、Currency、Single、Double、Decimal（目前尚不支持）、Date、String（只支持变长）、Object 或 Variant。如果没有选择参数 **Optional**，则可以指定用户定义类型，或对象类型。 |
| *defaultvalue* | 可选的。任何常数或常数表达式。只对 **Optional** 参数合法。如果类型为 **Object**，则显式的缺省值只能是 **Nothing**。 |

**说明**

如果没有使用 **Public、Private** 或 **Friend** 显式指定，**Sub** 过程按缺省情况就是公用的。如果没有使用 **Static**，则在调用之后不会保留局部变量的值。**Friend** 关键字只能在类模块中使用。不过 **Friend** 过程可以被工程的任何模块中的过程访问。**Friend** 过程不会在其父类的类型库中出现，且 **Friend** 过程不能被后期绑定。

**小心** **Sub** 过程可以是递归的；也就是说，该过程可以调用自己来完成某个特定的任务。不过，递归可能会导致堆栈上溢。通常 **Static** 关键字和递归的 **Sub** 过程不在一起使用。

所有的可执行代码都必须属于某个过程。不能在别的 **Sub、Function** 或 **Property** 过程中定义 **Sub** 过程。

**Exit Sub** 语句使执行立即从一个 **Sub** 过程中退出。程序接着从调用该 **Sub** 过程的语句下一条语句执行。在 **Sub** 过程的任何位置都可以有 **Exit Sub** 语句。

**Sub** 过程与 **Function** 过程的相似之处是：它们都一个可以获取参数，执行一系列语句，以及改变其参数的值的独立过程。而与 **Function** 过程不同的是：带返回值的 **Sub** 过程不能用于表达式。

可以使用过程名并后跟相应的参数列表来调用一个 **Sub** 过程。请参阅 **Call** 语句关于如何调用 **Sub** 过程的详细的说明信息。

在 **Sub** 过程中使用的变量分为两类：一类是在过程内显式定义的，另一类则不是。在过程内显式定义的变量（使用 **Dim** 或等效方法）都是局部变量。对于使用了但又没有在过程中显式定义的变量，除非其在该过程外更高级别的位置有显示地定义，否则也是局部的。

**小心** 过程可以使用没有在过程内显式定义的变量，但只要有任何在模块级别定义的名称与之同名，就会产生名称冲突。如果过程中使用的未定义的变量与别的过程，常数，或变量的名称相同，则认为过程使用的是模块级的名称。显式定义变量就可以避免这类冲突。可以使用 **Option Explicit** 语句来强制显式定义变量。

**注意** 不能使用 **GoSub、GoTo** 或 **Return** 来进入或退出 **Sub** 过程。

  **键盘事件**

# KeyPress 事件

此事件当用户按下和松开一个 ANSI 字符集 中的键时发生。

**语法**

**Private Sub Form\_KeyPress(***keyascii* **As Integer)**

**Private Sub** *object***\_KeyPress(**[*index* **As Integer**,]*keyascii* **As Integer)**

KeyPress 事件语法包含下列部分：

|  |  |
| --- | --- |
| **部分** | **描述** |
| *object* | 一个对象表达式，其值是“应用于”列表中的一个对象。 |
| *index* | 一个整数，它用来唯一标识一个在控件数组中的控件。 |
| *keyascii* | 是返回一个标准数字 ANSI 键代码的整数。*Keyascii* 通过引用传递，对它进行改变可给对象发送一个不同的字符。将 *keyascii* 改变为 0 时可取消击键，这样一来对象便接收不到字符。 |

**说明**

具有焦点的对象接收该事件。一个窗体仅在它没有可视和有效的控件或 **KeyPreview** 属性被设置为 **True** 时才能接收该事件。一个 KeyPress 事件可以引用任何可打印的键盘字符，一个来自标准字母表的字符或少数几个特殊字符之一的字符与 CTRL 键的组合，以及 ENTER 或 BACKSPACE 键。KeyPress 事件过程在截取 **TextBox** 或 **ComboBox** 控件所输入的击键时是非常有用的。它可立即测试击键的有效性或在字符输入时对其进行格式处理。改变 *keyascii* 参数的值会改变所显示的字符。

可使用下列表达式将 *keyascii* 参数转变为一个字符：

Chr(KeyAscii)

然后执行字符串操作，并将该字符反译成一个控件可通过该表达式解释的 ANSI 数字：

KeyAscii = Asc(char)

应当使用 KeyDown 和 KeyUP 事件过程来处理任何不被 KeyPress 识别的击键，诸如：功能键、编辑键、定位键以及任何这些键和键盘换档键的组合等。与 KeyDown 和 KeyUp 事件不同的是，KeyPress 不显示键盘的物理状态，而只是传递一个字符。

KeyPress 将每个字符的大、小写形式作为不同的键代码解释，即作为两种不同的字符。而 KeyDown 和 KeyUp 用两种参数解释每个字符的大写形式和小写形式：*keycode* — 显示物理的键（将 A 和 a 作为同一个键返回）和 *shift* —指示 *shift* + *key* 键的状态而且返回 A 或 a 其中之一。

如果 **KeyPreview** 属性被设置为 **True**，窗体将先于该窗体上的控件接收此事件。可用 **KeyPreview** 属性来创建全局键盘处理例程。

**注意** CTRL+@ 的键盘组合的 ANSI 编号是 0。因为 Visual Basic 将一个零值的 *keyascii*

# KeyDown、KeyUp 事件

这些事件是当一个对象具有焦点时按下 ( KeyDown ) 或松开 ( KeyUp ) 一个键时发生的。（要解释 ANSI 字符集 中的字符，应使用 KeyPress 事件。） 不区分大小写但支持识别shift，ctrl，alt

**语法**

**Private Sub** **Form\_KeyDown(***keycode* **As Integer**, *shift* **As Integer)**

**Private Sub** *object***\_KeyDown(**[*index* **As Integer**,]*keycode* **As Integer**, *shift* **As Integer)**

**Private Sub** **Form\_KeyUp(***keycode* **As Integer**, *shift* **As Integer)**

**Private Sub** *object***\_KeyUp(**[*index* **As Integer**,]*keycode* **As Integer**, *shift* **As Integer)**

KeyDown 和 KeyUp 事件包括下列部分：

|  |  |
| --- | --- |
| **部分** | **描述** |
| *object* | 一个对象表达式，其值是“应用于”列表中的一个对象。 |
| *index* | 是一个整数，它用来唯一标识一个在控件数组中的控件。 |
| *keycode* | 是一个键代码，诸如 **vbKeyF1** （ F1 键）或 **vbKeyHome** （ HOME 键）。要指定键代码，可使用对象浏览器中的 Visual Basic ( VB ) 对象库中的常数。 |
| *shift* | 是在该事件发生时响应 SHIFT 、CTRL和 ALT 键的状态的一个整数。*shift* 参数是一个位域，它用最少的位响应 SHIFT 键（位 0 ）、CTRLL 键（位 1 ）和 ALT 键（位 2 ）。这些位分别对应于值 1、2 和 4。可通过对一些、所有或无位的设置来指明有一些、所有或零个键被按下。例如，如果 CTRL 和 ALT 这两个键都被按下，则 *shift* 的值为 6。 |

**说明**

对于这两个事件来说，带焦点的对象都接收所有击键。一个窗体只有在不具有可视的和有效的控件时才可以获得焦点。虽然 KeyDown 和 KeyUp 事件可应用于大多数键，它们最经常地还是应用于：

* 扩展的字符键如功能键等。
* 定位键。
* 键盘修饰键和按键的组合。
* 区别数字小键盘和常规数字键。

在需要对按下和松开一个键都响应时，可使用 KeyDown 和 KeyUp 事件过程。

下列情况不能引用 KeyDown 和 KeyUp 事件：

* 窗体有一个 **CommandButton** 控件，并且 **Default** 属性设置为 **True** 时的 ENTER 键。
* 窗体有一个 **CommandButton** 控件，并且 **Cancel** 属性设置为 **True** 时的 ESC 键。
* TAB 键。

KeyDown 和 KeyUp 用两种参数解释每个字符的大写形式和小写形式：*keycode* —显示物理的键（将 A 和 a 作为同一个键返回）和 *shift* —显示 *shift* + *key* 键的状态而且返回 A 或 a 其中之一。

如果需要测试 *shift* 参数，可使用该参数中定义各位的 *shift* 常数。该常数有下列值：

|  |  |  |
| --- | --- | --- |
| **常数** | **值** | **描述** |
| **vbShiftMask** | 1 | SHIFT 键的位屏蔽。 |
| **VbCtrlMask** | 2 | CTRL 键的位屏蔽。 |
| **VbAltMask** | 4 | ALT 键的位屏蔽。 |

该常数用作位屏蔽。它可被用来测试任何键组合。

测试一个条件时，首先将每个结果分配给一个临时整数变量，然后将 *shift* 与一个位屏蔽进行对比。如下例，可用 **And** 运算符和 *shift* 参数一起来测试条件是否大于 0。该条件说明该修正键被按下：

ShiftDown = (Shift And vbShiftMask) > 0

可按此例在一个过程中测试任何条件的组合：

If ShiftDown And CtrlDown Then

**注意** 如果 **KeyPreview** 属性被设置为 **True**，则一个窗体先于该窗体上的控件接收到此事件。可用 **KeyPreview** 属性来创建全局键盘处理例程。

*Keycode的返回值含义：*

|  |  |  |
| --- | --- | --- |
| **常数** | **值** | **描述** |
| **vbKeyLButton** | 0x1 | 鼠标左键 |
| **vbKeyRButton** | 0x2 | 鼠标右键 |
| **vbKeyCancel** | 0x3 | CANCEL 键 |
| **vbKeyMButton** | 0x4 | 鼠标中键 |
| **vbKeyBack** | 0x8 | BACKSPACE 键 |
| **vbKeyTab** | 0x9 | TAB 键 |
| **vbKeyClear** | 0xC | CLEAR 键 |
| **vbKeyReturn** | 0xD | ENTER 键 |
| **vbKeyShift** | 0x10 | SHIFT 键 |
| **vbKeyControl** | 0x11 | CTRL 键 |
| **vbKeyMenu** | 0x12 | MENU 键 |
| **vbKeyPause** | 0x13 | PAUSE 键 |
| **vbKeyCapital** | 0x14 | CAPS LOCK 键 |
| **vbKeyEscape** | 0x1B | ESC 键 |
| **vbKeySpace** | 0x20 | SPACEBAR 键 |
| **vbKeyPageUp** | 0x21 | PAGE UP 键 |
| **vbKeyPageDown** | 0x22 | PAGE DOWN 键 |
| **vbKeyEnd** | 0x23 | END 键 |
| **vbKeyHome** | 0x24 | HOME 键 |
| **vbKeyLeft** | 0x25 | LEFT ARROW 键 |
| **vbKeyUp** | 0x26 | UP ARROW 键 |
| **vbKeyRight** | 0x27 | RIGHT ARROW 键 |
| **vbKeyDown** | 0x28 | DOWN ARROW 键 |
| **vbKeySelect** | 0x29 | SELECT 键 |
| **vbKeyPrint** | 0x2A | PRINT SCREEN 键 |
| **vbKeyExecute** | 0x2B | EXECUTE 键 |
| **vbKeySnapshot** | 0x2C | SNAPSHOT 键 |
| **vbKeyInsert** | 0x2D | INSERT 键 |
| **vbKeyDelete** | 0x2E | DELETE 键 |
| **vbKeyHelp** | 0x2F | HELP 键 |
| **vbKeyNumlock** | 0x90 | NUM LOCK 键 |

A 至 Z 键与 A – Z 字母的 ASCII 码相同：

|  |  |  |
| --- | --- | --- |
| **常数** | **值** | **描述** |
| **vbKeyA** | 65 | A 键 |
| **vbKeyB** | 66 | B 键 |
| **vbKeyC** | 67 | C 键 |
| **vbKeyD** | 68 | D 键 |
| **vbKeyE** | 69 | E 键 |
| **vbKeyF** | 70 | F 键 |
| **vbKeyG** | 71 | G 键 |
| **vbKeyH** | 72 | H 键 |
| **vbKeyI** | 73 | I 键 |
| **vbKeyJ** | 74 | J 键 |
| **vbKeyK** | 75 | K 键 |
| **vbKeyL** | 76 | L 键 |
| **vbKeyM** | 77 | M 键 |
| **vbKeyN** | 78 | N 键 |
| **vbKeyO** | 79 | O 键 |
| **vbKeyP** | 80 | P 键 |
| **vbKeyQ** | 81 | Q 键 |
| **vbKeyR** | 82 | R 键 |
| **vbKeyS** | 83 | S 键 |
| **vbKeyT** | 84 | T 键 |
| **vbKeyU** | 85 | U 键 |
| **vbKeyV** | 86 | V 键 |
| **vbKeyW** | 87 | W 键 |
| **vbKeyX** | 88 | X 键 |
| **vbKeyY** | 89 | Y 键 |
| **vbKeyZ** | 90 | Z 键 |
|  |  |  |

0 至 9 键与数字 0 – 9 的 ASCII 码相同：

|  |  |  |
| --- | --- | --- |
| **常数** | **值** | **描述** |
| **vbKey0** | 48 | 0 键 |  |
| **vbKey1** | 49 | 1 键 |
| **vbKey2** | 50 | 2 键 |
| **vbKey3** | 51 | 3 键 |
| **vbKey4** | 52 | 4 键 |
| **vbKey5** | 53 | 5 键 |
| **vbKey6** | 54 | 6 键 |
| **vbKey7** | 55 | 7 键 |
| **vbKey8** | 56 | 8 键 |
| **vbKey9** | 57 | 9 键 |

下列常数代表数字键盘上的键：

|  |  |  |
| --- | --- | --- |
| **常数** | **值** | **描述** |
| **vbKeyNumpad0** | 0x60 | 0 键 |
| **vbKeyNumpad1** | 0x61 | 1 键 |  |
| **vbKeyNumpad2** | 0x62 | 2 键 |  |
| **vbKeyNumpad3** | 0x63 | 3 键 |  |
| **vbKeyNumpad4** | 0x64 | 4 键 |  |
| **vbKeyNumpad5** | 0x65 | 5 键 |  |
| **vbKeyNumpad6** | 0x66 | 6 键 |  |
| **vbKeyNumpad7** | 0x67 | 7 键 |  |
| **vbKeyNumpad8** | 0x68 | 8 键 |  |
| **vbKeyNumpad9** | 0x69 | 9 键 |  |
| **vbKeyMultiply** | 0x6A | MULTIPLICATION SIGN (**\***) 键 |  |
| **vbKeyAdd** | 0x6B | PLUS SIGN (**+**) 键 |  |
| **vbKeySeparator** | 0x6C | ENTER 键 |  |
| **vbKeySubtract** | 0x6D | MINUS SIGN (**–**) 键 |  |
| **vbKeyDecimal** | 0x6E | DECIMAL POINT (**.**) 键 |  |
| **vbKeyDivide** | 0x6F | DIVISION SIGN (**/**) 键 |  |

下列常数代表功能键：

|  |  |  |
| --- | --- | --- |
| **常数** | **值** | **描述** |
| **vbKeyF1** | 0x70 | F1 键 |  |
| **vbKeyF2** | 0x71 | F2 键 |
| **vbKeyF3** | 0x72 | F3 键 |
| **vbKeyF4** | 0x73 | F4 键 |
| **vbKeyF5** | 0x74 | F5 键 |
| **vbKeyF6** | 0x75 | F6 键 |
| **vbKeyF7** | 0x76 | F7 键 |
| **vbKeyF8** | 0x77 | F8 键 |
| **vbKeyF9** | 0x78 | F9 键 |
| **vbKeyF10** | 0x79 | F10 键 |
| **vbKeyF11** | 0x7A | F11 键 |
| **vbKeyF12** | 0x7B | F12 键 |
| **vbKeyF13** | 0x7C | F13 键 |
| **vbKeyF14** | 0x7D | F14 键 |
| **vbKeyF15** | 0x7E | F15 键 |
| **vbKeyF16** | 0x7F | F16 键 |

  **ASCll码表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bin(二进制) | Oct(八进制) | Dec(十进制) | Hex(十六进制) | 缩写/字符 | 解释 |
| 0000 0000 | 0 | 0 | 00 | NUL(null) | 空字符 |
| 0000 0001 | 1 | 1 | 01 | SOH(start of headline) | 标题开始 |
| 0000 0010 | 2 | 2 | 02 | STX (start of text) | 正文开始 |
| 0000 0011 | 3 | 3 | 03 | ETX (end of text) | 正文结束 |
| 0000 0100 | 4 | 4 | 04 | EOT (end of transmission) | 传输结束 |
| 0000 0101 | 5 | 5 | 05 | ENQ (enquiry) | 请求 |
| 0000 0110 | 6 | 6 | 06 | ACK (acknowledge) | 收到通知 |
| 0000 0111 | 7 | 7 | 07 | BEL (bell) | 响铃 |
| 0000 1000 | 10 | 8 | 08 | BS (backspace) | 退格 |
| 0000 1001 | 11 | 9 | 09 | HT (horizontal tab) | 水平制表符 |
| 0000 1010 | 12 | 10 | 0A | LF (NL line feed, new line) | 换行键 |
| 0000 1011 | 13 | 11 | 0B | VT (vertical tab) | 垂直制表符 |
| 0000 1100 | 14 | 12 | 0C | FF (NP form feed, new page) | 换页键 |
| 0000 1101 | 15 | 13 | 0D | CR (carriage return) | 回车键 |
| 0000 1110 | 16 | 14 | 0E | SO (shift out) | 不用切换 |
| 0000 1111 | 17 | 15 | 0F | SI (shift in) | 启用切换 |
| 0001 0000 | 20 | 16 | 10 | DLE (data link escape) | 数据链路转义 |
| 0001 0001 | 21 | 17 | 11 | DC1 (device control 1) | 设备控制1 |
| 0001 0010 | 22 | 18 | 12 | DC2 (device control 2) | 设备控制2 |
| 0001 0011 | 23 | 19 | 13 | DC3 (device control 3) | 设备控制3 |
| 0001 0100 | 24 | 20 | 14 | DC4 (device control 4) | 设备控制4 |
| 0001 0101 | 25 | 21 | 15 | NAK (negative acknowledge) | 拒绝接收 |
| 0001 0110 | 26 | 22 | 16 | SYN (synchronous idle) | 同步空闲 |
| 0001 0111 | 27 | 23 | 17 | ETB (end of trans. block) | 结束传输块 |
| 0001 1000 | 30 | 24 | 18 | CAN (cancel) | 取消 |
| 0001 1001 | 31 | 25 | 19 | EM (end of medium) | 媒介结束 |
| 0001 1010 | 32 | 26 | 1A | SUB (substitute) | 代替 |
| 0001 1011 | 33 | 27 | 1B | ESC (escape) | 换码(溢出) |
| 0001 1100 | 34 | 28 | 1C | FS (file separator) | 文件分隔符 |
| 0001 1101 | 35 | 29 | 1D | GS (group separator) | 分组符 |
| 0001 1110 | 36 | 30 | 1E | RS (record separator) | 记录分隔符 |
| 0001 1111 | 37 | 31 | 1F | US (unit separator) | 单元分隔符 |
| 0010 0000 | 40 | 32 | 20 | (space) | 空格 |
| 0010 0001 | 41 | 33 | 21 | ! | 叹号 |
| 0010 0010 | 42 | 34 | 22 | " | 双引号 |
| 0010 0011 | 43 | 35 | 23 | # | 井号 |
| 0010 0100 | 44 | 36 | 24 | $ | 美元符 |
| 0010 0101 | 45 | 37 | 25 | % | 百分号 |
| 0010 0110 | 46 | 38 | 26 | & | 和号 |
| 0010 0111 | 47 | 39 | 27 | ' | 闭单引号 |
| 0010 1000 | 50 | 40 | 28 | ( | 开括号 |
| 0010 1001 | 51 | 41 | 29 | ) | 闭括号 |
| 0010 1010 | 52 | 42 | 2A | \* | 星号 |
| 0010 1011 | 53 | 43 | 2B | + | 加号 |
| 0010 1100 | 54 | 44 | 2C | , | 逗号 |
| 0010 1101 | 55 | 45 | 2D | - | 减号/破折号 |
| 0010 1110 | 56 | 46 | 2E | . | 句号 |
| 00101111 | 57 | 47 | 2F | / | 斜杠 |
| 00110000 | 60 | 48 | 30 | 0 | 数字0 |
| 00110001 | 61 | 49 | 31 | 1 | 数字1 |
| 00110010 | 62 | 50 | 32 | 2 | 数字2 |
| 00110011 | 63 | 51 | 33 | 3 | 数字3 |
| 00110100 | 64 | 52 | 34 | 4 | 数字4 |
| 00110101 | 65 | 53 | 35 | 5 | 数字5 |
| 00110110 | 66 | 54 | 36 | 6 | 数字6 |
| 00110111 | 67 | 55 | 37 | 7 | 数字7 |
| 00111000 | 70 | 56 | 38 | 8 | 数字8 |
| 00111001 | 71 | 57 | 39 | 9 | 数字9 |
| 00111010 | 72 | 58 | 3A | : | 冒号 |
| 00111011 | 73 | 59 | 3B | ; | 分号 |
| 00111100 | 74 | 60 | 3C | < | 小于 |
| 00111101 | 75 | 61 | 3D | = | 等号 |
| 00111110 | 76 | 62 | 3E | > | 大于 |
| 00111111 | 77 | 63 | 3F | ? | 问号 |
| 01000000 | 100 | 64 | 40 | @ | 电子邮件符号 |
| 01000001 | 101 | 65 | 41 | A | 大写字母A |
| 01000010 | 102 | 66 | 42 | B | 大写字母B |
| 01000011 | 103 | 67 | 43 | C | 大写字母C |
| 01000100 | 104 | 68 | 44 | D | 大写字母D |
| 01000101 | 105 | 69 | 45 | E | 大写字母E |
| 01000110 | 106 | 70 | 46 | F | 大写字母F |
| 01000111 | 107 | 71 | 47 | G | 大写字母G |
| 01001000 | 110 | 72 | 48 | H | 大写字母H |
| 01001001 | 111 | 73 | 49 | I | 大写字母I |
| 01001010 | 112 | 74 | 4A | J | 大写字母J |
| 01001011 | 113 | 75 | 4B | K | 大写字母K |
| 01001100 | 114 | 76 | 4C | L | 大写字母L |
| 01001101 | 115 | 77 | 4D | M | 大写字母M |
| 01001110 | 116 | 78 | 4E | N | 大写字母N |
| 01001111 | 117 | 79 | 4F | O | 大写字母O |
| 01010000 | 120 | 80 | 50 | P | 大写字母P |
| 01010001 | 121 | 81 | 51 | Q | 大写字母Q |
| 01010010 | 122 | 82 | 52 | R | 大写字母R |
| 01010011 | 123 | 83 | 53 | S | 大写字母S |
| 01010100 | 124 | 84 | 54 | T | 大写字母T |
| 01010101 | 125 | 85 | 55 | U | 大写字母U |
| 01010110 | 126 | 86 | 56 | V | 大写字母V |
| 01010111 | 127 | 87 | 57 | W | 大写字母W |
| 01011000 | 130 | 88 | 58 | X | 大写字母X |
| 01011001 | 131 | 89 | 59 | Y | 大写字母Y |
| 01011010 | 132 | 90 | 5A | Z | 大写字母Z |
| 01011011 | 133 | 91 | 5B | [ | 开方括号 |
| 01011100 | 134 | 92 | 5C | \ | 反斜杠 |
| 01011101 | 135 | 93 | 5D | ] | 闭方括号 |
| 01011110 | 136 | 94 | 5E | ^ | 脱字符 |
| 01011111 | 137 | 95 | 5F | \_ | 下划线 |
| 01100000 | 140 | 96 | 60 | ` | 开单引号 |
| 01100001 | 141 | 97 | 61 | a | 小写字母a |
| 01100010 | 142 | 98 | 62 | b | 小写字母b |
| 01100011 | 143 | 99 | 63 | c | 小写字母c |
| 01100100 | 144 | 100 | 64 | d | 小写字母d |
| 01100101 | 145 | 101 | 65 | e | 小写字母e |
| 01100110 | 146 | 102 | 66 | f | 小写字母f |
| 01100111 | 147 | 103 | 67 | g | 小写字母g |
| 01101000 | 150 | 104 | 68 | h | 小写字母h |
| 01101001 | 151 | 105 | 69 | i | 小写字母i |
| 01101010 | 152 | 106 | 6A | j | 小写字母j |
| 01101011 | 153 | 107 | 6B | k | 小写字母k |
| 01101100 | 154 | 108 | 6C | l | 小写字母l |
| 01101101 | 155 | 109 | 6D | m | 小写字母m |
| 01101110 | 156 | 110 | 6E | n | 小写字母n |
| 01101111 | 157 | 111 | 6F | o | 小写字母o |
| 01110000 | 160 | 112 | 70 | p | 小写字母p |
| 01110001 | 161 | 113 | 71 | q | 小写字母q |
| 01110010 | 162 | 114 | 72 | r | 小写字母r |
| 01110011 | 163 | 115 | 73 | s | 小写字母s |
| 01110100 | 164 | 116 | 74 | t | 小写字母t |
| 01110101 | 165 | 117 | 75 | u | 小写字母u |
| 01110110 | 166 | 118 | 76 | v | 小写字母v |
| 01110111 | 167 | 119 | 77 | w | 小写字母w |
| 01111000 | 170 | 120 | 78 | x | 小写字母x |
| 01111001 | 171 | 121 | 79 | y | 小写字母y |
| 01111010 | 172 | 122 | 7A | z | 小写字母z |
| 01111011 | 173 | 123 | 7B | { | 开花括号 |
| 01111100 | 174 | 124 | 7C | | | 垂线 |
| 01111101 | 175 | 125 | 7D | } | 闭花括号 |
| 01111110 | 176 | 126 | 7E | ~ | 波浪号 |
| 01111111 | 177 | 127 | 7F | DEL (delete) | 删除 |

 **鼠标事件**

# MouseDown、MouseUp 事件

这些事件是当按下 ( MouseDown ) 或者释放 ( MouseUp ) 鼠标按钮时发生。

**语法**

**Private Sub Form\_MouseDown(***button* **As Integer**,*shift* **As Integer**,*x* **As Single**,*y* **As Single)**

**Private Sub MDIForm\_MouseDown(***button* **As Integer**,*shift* **As Integer**,*x* **As Single**,*y* **As Single)**

**Private Sub** *object***\_MouseDown(**[*index* **As Integer**,]*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**

**Private Sub** **Form\_MouseUp(***button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**

**Private Sub MDIForm\_MouseUp(***button* **As Integer**,*shift* **As Integer**,*x* **As Single**,*y* **As Single)**

**Private Sub** *object* **\_MouseUp(**[*index* **As Integer**,]*button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**

MouseDown 和 MouseUp 事件各种语法包含下列部分：

|  |  |
| --- | --- |
| **部分** | **描述** |
| *object* | 返回一个对象表达式，其值是“应用于”列表中的一个对象。 |
| *index* | 返回一个整数，用来唯一地标识一个在控件数组中的控件。 |
| *button* | 返回一个整数，用来标识该事件的产生是按下 ( MouseDown ) 或者释放 ( MouseUp ) 按钮引起的。*button* 参数是具有相应于左按钮（位 0），右按钮（位 1），以及中间按钮（位 2）的一个位字段。这些位的值分别等于 1，2，和 4。其中仅有一位被设置，指示出引起该事件的那个按钮。 |
| *shift* | 返回一个整数，在 *button* 参数指定的按钮被按下或者被释放的情况下，该整数相应于 SHIFT,CTRL,和 ALT 键的状态。某键被按下使得一个二进制位被设置。*shift* 参数是具有相应于 SHIFT 键（位 0），CTRL 键（位 1），以及 ALT 键（位 2）最少二进制位的一个位字段。这些位的值分别等于 1，2，和 4。*shift* 参数指示这些键的状态。这些位中可能有一些，全部，或者一个也没有被设置，指示这些键中的一些、全部，或者一个也没有被按下。例如，CTRL 和 ALT 键都被按下，则 *shift* 的值就是 6。 |
| *x*, *y* | 返回一个指定鼠标指针当前位置的数。*x* 和 *y* 的值所表示的总是通过该对象 **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, 和**ScaleTop** 属性所建立的坐标系统的方式。 |

**说明**

为了在给定的一个鼠标按钮按下或释放时指定将引起的一些操作，应当使用 MouseDown 或者 MouseUp 事件过程。不同于 Click 和 DblClick 事件的是，MouseDown 和 MouseUp 事件能够区分出鼠标的左、右、和中间按钮。也可以为使用 SHIFT, CTRL, 和 ALT 等键盘换挡键编写用于鼠标—键盘组合操作的代码。

下列情况对 Click 和 DblClick 事件都适用：

* 如果鼠标按钮是当其指针在窗体或控件之上时被按下，则该对象将“捕获”鼠标并接收包括最后 MouseUp 事件在内的全部鼠标事件。这暗示了通过鼠标事件所返回的 *x*，*y* 鼠标指针坐标值，可以不总是在接收它们的对象的内部区域之内。
* 如果鼠标被持续地按下，则第一次按下之后捕获鼠标的对象将接收全部鼠标事件直至所有按钮被释放为止。

如果要测试 *button* 或 *shift* 参数，可以使用对象浏览器中的 Visual Basic (VB) 对象库中所列出的常数，用来定义该参数中的各个二进制位：

|  |  |  |
| --- | --- | --- |
| **常数（按钮）** | **值** | **描述** |
| **vbLeftButton** | 1 | 左按钮被按下 |
| **vbRightButton** | 2 | 右按钮被按下 |
| **vbMiddleButton** | 4 | 中间按钮被按下 |

|  |  |  |
| --- | --- | --- |
| **常数（换挡）** | **值** | **描述** |
| **vbShiftMask** | 1 | SHIFT 键被按下。 |
| **vbCtrlMask** | 2 | CTRL 键被按下。 |
| **vbAltMask** | 4 | ALT 键被按下。 |

随后这些常数作为位屏蔽，对于按钮的各种组合，无须计算各个组合的唯一的位字段即可进行测试。

**注意** 可使用 MouseMove 事件过程对由于鼠标移动而引起的事件进行响应。MouseDown 和 MouseUp 所使用的 *button* 参数与 MouseMove 所使用的 *button* 参数是不同的。对于 MouseDown 和 MouseUp 来说，*button* 参数要精确地指出每个事件的一个按钮，而对于 MouseMove 来说，它指示的是所有按钮的当前状态。

# MouseMove 事件

此事件在移动鼠标时发生。

**语法**

**Private Sub Form\_MouseMove(***button* **As Integer**,*shift* **As Integer**,*x* **As Single**,*y* **As Single)**

**Private Sub MDIForm\_MouseMove(***button* **As Integer**,*shift* **As Integer**,*x* **As Single**,*y* **As Single)**

**Private Sub** *object***\_MouseMove(**[*index* **As Integer**,] *button* **As Integer**, *shift* **As Integer**, *x* **As Single**, *y* **As Single)**

MouseMove 事件语法包含下列部分：

|  |  |
| --- | --- |
| **部分** | **描述** |
| *object* | 一个对象表达式，其值是“应用于”列表中的一个对象。 |
| *index* | 一个整数，用来唯一地标识一个在控件数组中的控件。 |
| *button* | 一个整数，它对应鼠标各个按钮的状态，如果某个按钮按下，其中就有一个二进制位被设置。*button* 参数是具有相应于左按钮（位 0），右按钮（位 1），以及中间按钮（位 2）的一个位字段。这些位的值分别等于 1，2，和 4。它指示这些鼠标按钮的整体状态；三个二进制位中的一些、全部、或一个也没有被设置，指示这些按钮中的一些、全部、或一个也没有被按下。 |
| *shift* | 一个整数，该整数相应于 SHIFT,CTRL,和 ALT 键的状态。某键被按下使得一个二进制位被设置。*shift* 参数是具有相应于 SHIFT 键（位 0），CTRL 键（位 1），以及 ALT 键（位 2）最少二进制位的一个位字段。这些位的值分别等于 1，2，和 4。*shift* 参数指示这些键的状态。这些位中可能有一些、全部、或者一个也没有被设置，指示这些键中的一些、全部、或者一个也没有被按下。例如，CTRL 和 ALT 键都被按下，则 *shift* 的值就是 6。 |
| *x*, *y* | 一个指定鼠标指针当前位置的数。*x* 和 *y* 的值所表示的总是通过该对象 **ScaleHeight**, **ScaleWidth**, **ScaleLeft**,和 **ScaleTop** 属性所建立的坐标系统的方式。 |

**说明**

MouseMove 事件伴随鼠标指针在对象间移动时连续不断地产生。除非有另一个对象捕获了鼠标，否则，当鼠标位置在对象的边界范围内时该对象就能接收 MouseMove 事件。

要测试 *button* 或 *shift* 参数，可使用对象浏览器中的 Visual Basic (VB) 对象库中所列出的常数，用来定义该参数中的各个位：

|  |  |  |
| --- | --- | --- |
| **常数（按钮）** | **值** | **描述** |
| **vbLeftButton** | 1 | 左按钮按下。 |
| **vbRightButton** | 2 | 右按钮按下。 |
| **vbMiddleButton** | 4 | 中间按钮按下。 |

|  |  |  |
| --- | --- | --- |
| **常数（换挡）** | **值** | **描述** |
| **vbShiftMask** | 1 | SHIFT 键按下。 |
| **vbCtrlMask** | 2 | CTRL 键按下。 |
| **vbAltMask** | 4 | ALT 键按下。 |

然后这些常数用作位屏蔽，对于按钮的各种组合，无须计算出各个组合的唯一的位字段值即可进行检测。

要测试某一条件，首先将各个结果赋给一个临时整型变量然后再与一个位屏蔽的 *button* 或 *shift* 参数进行比较。测试应当用各个参数进行 **And** 运算，若结果大于零，则说明该键或按钮被按下。其操作如下：

LeftDown = (Button And vbLeftButton) > 0

CtrlDown = (Shift And vbCtrlMask) > 0

然后，接下去可对结果的各种组合进行检测，其操作如下：

If LeftDown And CtrlDown Then

**注意** 为了对鼠标按钮按下和释放所引起的事件进行处理，可使用 MouseDown 和 MouseUp 事件过程。

MouseMove 事件的 *button* 参数与 MouseDown 和 MouseUp 事件的 *button* 参数是不同的。对于 MouseMove 事件来说，*button* 参数指示的是所有按钮当前的状态；一个 MouseMove 事件可指示某些、全部或没有一个按钮被按下。对于 MouseDown 和 MouseUp 事件来说，*button* 参数在每个事件精确地指示一个按钮。

在 MouseMove 事件中任何时侯移动窗口，都能引起层叠事件。当该窗口移动到指针下面时 MouseMove 事件将产生。即使是鼠标完全不动 MouseMove 事件也能产生。

# MousePointer 属性

返回或设置一个值，该值指示在运行时当鼠标移动到对象的一个特定部分时，被显示的鼠标指针的类型。

**语法**

*object***.MousePointer** [**=** *value*]

**MousePointer** 属性语法包含下面部分：

|  |  |
| --- | --- |
| **部分** | **描述** |
| *object* | 对象表达式，其值是“应用于”列表中的一个对象。 |
| *value* | 整数，按照设置值中的描述指定被显示的鼠标指针类型。 |

**设置值**

*value* 的设置值为：

|  |  |  |
| --- | --- | --- |
| **常数** | **值** | **描述** |
| **vbDefault** | 0 | （缺省值）形状由对象决定。 |
| **VbArrow** | 1 | 箭头。 |
| **VbCrosshair** | 2 | 十字线（crosshair 指针）。 |
| **VbIbeam** | 3 | I 型 |
| **VbIconPointer** | 4 | 图标（矩形内的小矩形）。 |
| **VbSizePointer** | 5 | 尺寸线（指向东、南、西和北四方向的箭头）。 |
| **VbSizeNESW** | 6 | 右上-左下尺寸线（指向东北和西南方向的双箭头）。 |
| **VbSizeNS** | 7 | 垂-直尺寸线（指向南和北的双箭头）。 |
| **VbSizeNWSE** | 8 | 左上-右下尺寸线（指向东南和西北方向的双箭头）。 |
| **VbSizeWE** | 9 | 水-平尺寸线（指向东和西两个方向的双箭头）。 |
| **VbUpArrow** | 10 | 向上的箭头。 |
| **VbHourglass** | 11 | 沙漏（表示等待状态）。 |
| **VbNoDrop** | 12 | 不允许放下。 |
| **VbArrowHourglass** | 13 | 箭头和沙漏。 |
| **VbArrowQuestion** | 14 | 箭头和问号。 |
| **VbSizeAll** | 15 | 四向尺寸线。 |
| **VbCustom** | 99 | 通过 **MouseIcon** 属性所指定的自定义图标。 |

**说明**

在鼠标指针越过窗体或对话框上的控件时，为了指出功能上的改变，可以使用该属性。沙漏标形状设置值 (11) 是很有用的，用来指示用户需要等待过程或操作的完成。

**注意** 如果应用程序调用 DoEvents，那么 **MousePointer** 属性在经过 ActiveX 部件时可能暂时地改变。

# MouseIcon 属性

返回或设置自定义的鼠标图标。

**语法**

*object***.MouseIcon** **=** **LoadPicture(***pathname***)**

*object***.MouseIcon** **[=** *picture***]**

**MouseIcon** 属性的语法包含下面部分：

|  |  |
| --- | --- |
| **部分** | **描述** |
| *object* | 对象表达式，其值是“应用于”列表中的一个对象。 |
| *pathname* | 字符串表达式，指定包含自定义图标文件的路径和文件名。 |
| *picture* | **Form** 对象**、PictureBox** 控件、或 **Image** 控件的 **Picture** 属性。 |

**说明**

**MouseIcon** 属性提供一个自定义图标，它在 **MousePointer** 属性设为 99 时使用。

**MouseIcon** 属性使程序能够很容易地访问自定义光标，它可以是任意大小并具有任何热点位置的光标。Visual Basic 不能装入动画光标 (.ani) 文件，即使 32-位版的 Windows 支持这些光标。

 **拖放**

# DragMode 属性

返回或设置一个值，确定在拖放操作中所用的是手动还是自动拖动方式。

**语法**

*object*.**DragMode** [= *number*]

**DragMode** 属性语法包含下面部分：

|  |  |
| --- | --- |
| **部分** | **描述** |
| *object* | 对象表达式，其值是“应用于”列表中的一个对象。 |
| *Number* | 整数，指定拖动方式，“设置值”中有详细描述。 |

**设置值**

*number* 的设置值如下：

|  |  |  |
| --- | --- | --- |
| **常数** | **设置值** | **描述** |
| **VbManual** | 0 | （缺省值）手动方式－ 需要在源控件中用 **Drag** 方法来启动拖放操作。 |
| **VbAutomatic** | 1 | 自动方式 － 单击源控件自动启动拖放操作。 只有在 **OLE** 容器控件没有焦点时，它才能自动地被拖动。 |

**说明**

当 **DragMode** 属性设置为 1（自动方式）时，控件不能正常响应鼠标事件。可以用方式 0（手动方式）来确定拖放操作何时开始或何时结束；用该设置值可以启动一个拖放操作以响应键盘或菜单命令，或允许源控件在拖放操作之前识别 MouseDown 事件。

在拖放操作过程中，当鼠标器指针在一个目标对象或窗体上时，单击鼠标会对目标对象产生 DragDrop 事件。这将终止拖放操作。拖放操作也可能产生 DragOver 事件。

**注意** 当拖动控件时，该控件不能识别用户发出的其它鼠标或键盘事件（KeyDown、KeyPress 或 KeyUp，MouseDown、MouseMove 或 MouseUp)。但是，控件可以接收程序或 DDE 链接启动的事件。

# DragIcon 属性

返回或设置图标，它将在拖放操作中作为指针显示。

**语法**

*object*.**DragIcon** [= *icon*]

**DragIcon** 属性语法包含下面部分：

|  |  |
| --- | --- |
| **部分** | **描述** |
| *Object* | 对象表达式，其值是“应用于”列表中的一个对象。 |
| *Icon* | 任何返回图标的程序引用，例如引用窗体图标 (Form1.Icon)，引用另外控件的 **DragIcon** 属性 (Text1.DragIcon)，或是 **LoadPicture** 函数。 |

**设置值**

*icon* 的设置值如下:

|  |  |
| --- | --- |
| **设置值** | **描述** |
| (none) | （缺省值）矩形框内的箭头指针。 |
| Icon | 自定义的鼠标指针。在设计时用属性窗口设置图标。运行时可以用 **LoadPicture** 函数。所加载的文件必须有 .ico 文件扩展名和格式。 |

**说明**

在拖放操作中，用 **DragIcon** 属性可以提供可见的信息反馈─例如，它可以指定原控件在一个适当的目标之上。**DragIcon** 属性在用户开始拖放操作时起作用。典型做法是把 **DragIcon** 设置为 MouseDown 或 DragOver 事件过程的一部分。

**注意** 运行时，**DragIcon** 属性可以设置为任何对象的 **DragIcon** 或 **Icon** 属性，或者可以用 **LoadPicture** 函数返回的图标给它赋值。

在运行中，把一个控件的 **Picture** 属性值赋给另一个控件的 **DragIcon** 属性时**，** **Picture** 属性必须包含一个 .ico 文件，而不是 .bmp 文件。

# DragDrop 事件

在一个完整的拖放动作（即将一个控件拖动到一个对象上，并释放鼠标按钮）完成，或使用 **Drag** 方法，并将其 *action* 参数被设置为 2 (Drop) 时，该事件发生。

**语法**

**Private Sub** **Form\_DragDrop(***source* **As** **Control**, *x* **As** **Single**, *y* **As** **Single)**

**Private Sub** **MDIForm\_DragDrop(***source* **As** **Control**, *x* **As** **Single**, *y* **As** **Single)**

**Private Sub** *object***\_DragDrop(**[*index* **As** **Integer**,]*source* **As** **Control**, *x* **As** **Single**, *y* **As** **Single)**

DragDrop 事件语法包含下列部分：

|  |  |
| --- | --- |
| **部分** | **描述** |
| *object* | 一个对象表达式，其值是“应用于”列表中的一个对象。 |
| *index* | 一个整数，用来唯一地标识一个在控件数组中的控件。 |
| *source* | 正在被拖动的控件。可用此参数将属性和方法包括在事件过程中 — 例如，Source.Visible = 0。 |
| *x*, *y* | 是一个指定当前鼠标指针在目标窗体或控件中水平 (*x*) 和垂直 (*y*)位置的数字。这些坐标值通常用目标坐标系统来表示，该坐标系是通过 **ScaleHeight、ScaleWidth、ScaleLeft** 和 **ScaleTop** 属性而设置的。 |

**说明**

DragDrop 事件过程用来控制在一个拖动操作完成时将会发生的情况。例如，可将源控件移到一个新的位置或将一个文件从一个位置复制到另一个位置。

当 *source* 参数中可能使用多个控件时：

* 应使用 **TypeOf** 关键字和 **If** 语句一起确定与 *source* 表示的控件的类型。
* 应使用该控件的 **Tag** 属性来标识一个控件，然后使用 DragDrop 事件过程。

**注意** 应使用 **DragMode** 属性和 **Drag** 方法来指定开始拖动的方法。一旦开始拖动，可使用 DragOver 事件过程来处理位于 DragDrop 事件前面的事件

# DragOver 事件

它在拖放操作正在进行时发生。可使用此事件对鼠标指针在一个有效目标上的进入、离开或停顿等进行监控。鼠标指针的位置决定接收此事件的目标对象。

**语法**

**Private Sub** **Form\_DragOver(***source* **As** **Control**, *x* **As** **Single**, *y* **As** **Single**, *state* **As** **Integer)**

**Private Sub** **MDIForm\_DragOver(***source* **As** **Control**, *x* **As** **Single**, *y* **As** **Single**, *state* **As** **Integer)**

**Private Sub** *object***\_DragOver(**[*index* **As** **Integer**,]*source* **As** **Control**, *x* **As** **Single**, *y* **As** **Single**, *state* **As** **Integer)**

DragOver 事件语法包括下列部分：

|  |  |
| --- | --- |
| **部分** | **描述** |
| *object* | 一个对象表达式，其值是“应用于”列表中的一个对象。 |
| *index* | 一个整数，用来唯一地标识一个在控件数组中的控件。 |
| *source* | 正在被拖动的控件。可用此参数在事件过程中引用各属性和方法—例如， Source.Visible = False。 |
| *x*, *y* | 是一个指定当前鼠标指针在目标窗体或控件中水平 (*x*) 和垂直 (*y*) 位置的数字。这些坐标值通常用目标坐标系统来表示，该坐标系是通过 **ScaleHeight、ScaleWidth、ScaleLeft** 和 **ScaleTop** 属性而设置的。 |
| *state* | 是一个整数，它相应于一个控件的转变状态，该控件在相关目标窗体或控件中正在被拖动： |
|  | 0 = 进入（源控件正被向一个目标范围内拖动）。 |
|  | 1 = 离去（源控件正被向一个目标范围外拖动）。 |
|  | 2 = 跨越（源控件在目标范围内从一个位置移到了另一位置）。 |

**说明**

为了确定在拖动开始后和控件放在目标上之前发生些什么，应使用 DragOver 事件过程。例如，通过加亮目标（由代码设置 **BackColor** 或 **ForeColor** 属性）或者显示一个特定的拖动指针（由代码设置 **DragIcon** 或 **MousePointer** 属性）可验证有效的目标范围。

为了确定一些关键转变点处的操作，应使用 *state* 参数。例如，当 *state* 的设置为 0（进入）时可使一个可能的目标加亮，而当 *state* 的设置为 1（离去）时可恢复该对象先前的外观。

在 *state* 的设置为 0（进入）对象接收 DragOver 事件的情况：

* 如果源控件被放在该对象上，则该对象就接收一个 DragDrop 事件。
* 如果源控件没有被放在该对象上，则当 *state* 的设置为 1（离去）时该对象就接收另一个 DragOver 事件。

**注意** 应使用 **DragMode** 属性和 **Drag** 方法指定开始拖动的方式。关于 *source* 参数的使用技巧，请参阅 DragDrop 事件说明主题部分。

# Move 方法

用以移动 **MDIForm**、**Form** 或控件。不支持命名参数。移动控件的位置跟大小

**语法**

*object***.Move** *left*, *top*, *width*, *height*

**Move** 方法的语法包含下列部分：

|  |  |
| --- | --- |
| **部分** | **描述** |
| *object* | 可选的。一个对象表达式，其值为“应用于”列表中的一个对象。如果省略 *object*，带有焦点的窗体缺省为 *object*。 |
| *left* | 必需的。单精度值，指示 *object* 左边的水平坐标 (x-轴)。 |
| *top* | 可选的。单精度值，指示 *object* 顶边的垂直坐标 (y-轴)。  |
| *width* | 可选的。单精度值，指示 *object* 新的宽度。 |
| *height* | 可选的。单精度值，指示 *object* 新的高度。 |

**说明**

只有 *left* 参数是必须的。但是，要指定任何其它的参数，必须先指定出现在语法中该参数前面的全部参数。例如，如果不先指定 *left* 和 *top 参数，*则无法指定 *width* 参数。任何没有指定的尾部的参数则保持不变。

对于 **Frame** 控件中的窗体和控件，坐标系统总是用缇 。移动屏幕上的窗体或移动 **Frame** 中的控件总是相对于左上角的原点 (0,0)。移动 **Form** 对象或 **PictureBox** 中的控件（或 **MDIForm** 对象中的 MDI 子窗体）时，则使用该容器对象的坐标系统。坐标系统或度量单位是在设计时 用 **ScaleMode** 属性设置。在运行时 使用 **Scale** 方法可以更改该坐标系统。

# LoadPicture 函数

将图形载入到窗体的 **Picture** 属性、**PictureBox** 控件或 **Image** 控件。

**语法**

**LoadPicture(**[*filename*], [*size*], [*colordepth*],[*x,y*]**)**

**LoadPicture** 函数语法有下列部分：

|  |  |
| --- | --- |
| **部分** | **描述** |
| *filename* | 可选的。字符串表达式指定一个文件名。可以包括文件夹和驱动器。如果未指定文件名，LoadPicture 清除图象或 PictureBox 控件。 |
| *size* | 可选变体。如果 *filename* 是光标或图标文件，指定想要的图象大小。 |
| *colordepth* | 可选变体。如果 *filename* 是一个光标或图标文件，指定想要的颜色深度。 |
| *x* | 可选变体，如果使用 *y* ，则必须使用。如果 *filename* 是一个光标或图标文件，指定想要的宽度。在包含多个独立图象的文件中，如果那样大小的图象不能得到时，则使用可能的最好匹配。只有当 *colordepth* 设为 **vbLPCustom** 时，才使用X 和 Y值。For icon files 255 is the maximum possible value |
| *y* | 可选变体，如果使用 *x，* 则必须使用。如果 *filename* 是一个光标或图标文件，指定想要的高度。在包含多个独立图象的文件中，如果那样大小的图象不能得到时，则使用可能的最好匹配。For icon files 255 is the maximum possible value |

**设置值**

*size* 的设置值为：

|  |  |  |
| --- | --- | --- |
| **常量**  | **值** | **描述** |

|  |  |  |
| --- | --- | --- |
| **vbLPSmall** | 0 | 系统小图标。 |
| **vbLPLarge** | 1 | 系统大图标，由视频驱动程序决定。 |
| **vbLPSmallShell** | 2 | 外壳小图标大小，由“控制面板”中的 **Display Properties** 对话框中 **Appearance** 选项卡上的 Caption Buttons 的大小设置决定。 |
| **vbLPLargeShell** | 3 | 外壳大图标大小，由“控制面板”中的 **Display Properties** 对话框中 **Appearance** 选项卡上的图标大小设置决定。 |
| **vbLPCustom** | 4 | 自定义大小，由 *x 和* *y* 参数提供值。 |

*colordepth* 的设置值为：

|  |  |  |
| --- | --- | --- |
| **常量** | **值** | **描述** |

|  |  |  |
| --- | --- | --- |
| **vbLPDefault** | 0 | 如果使用指定文件，则为最佳可用匹配。 |
| **vbLPMonochrome** | 1 | 2 色。 |
| **vbLPVGAColor** | 2 | 16 色。 |
| **vbLPColor** | 3 | 256 色。 |

**说明**

Visual Basic 可以识别的图形格式有：位图 (.bmp) 文件、图标 (.ico) 文件、光标 (.cur) 文件，行程编码 (.rle) 文件、元 (.wmf) 文件、增强的元文件 (.emf), GIF(.gif) 文件以及 JPEG (.jpg) 文件。

赋值不带参数的 **LoadPicture** 将清除窗体、图片框及图象控件中的图形。

为了加载在 **PictureBox** 控件和 **Image** 控件中显示的图形或加载作为窗体背景的图形，必须将 **LoadPicture** 的返回值赋给要显示该图片的对象的 **Picture** 属性。例如：

Set Picture = LoadPicture("PARTY.BMP")

Set Picture1.Picture = LoadPicture("PARTY.BMP")

如果要将图标赋予窗体，则要把 **LoadPicture** 函数的返回值赋给 **Form** 对象的 **Icon** 属性：

Set Form1.Icon = LoadPicture("MYICON.ICO")

图标也可以被赋予除 **Timer** 控件和 **Menu** 控件外的其它控件的 **DragIcon** 属性，例如：

Set Command1.DragIcon = LoadPicture("MYICON.ICO")

使用 **LoadPicture** 可将图形文件载入到系统剪贴板，如下所示：

Clipboard.SetData LoadPicture("PARTY.BMP")

 **菜单栏**

# Checked 属性

返回或设置一个值，该值用来确定是否在一个菜单项后显示复选标记。

**语法**

*object*.**Checked** [= *boolean*]

**Checked** 属性的语法包含下面部分：

|  |  |
| --- | --- |
| **部分** | **描述** |
| *object* | 对象表达式，其值是“应用于”列表中的一个对象。 |
| *boolean* | 一个用来指定是否在一个菜单项后显示复选标记的布尔表达式。 |

**设置**

*boolean* 的设置为：

|  |  |
| --- | --- |
| **设置** | **描述** |
| **True** | 在菜单项之后放置复选标记。 |
| **False** | （缺省）不在紧接菜单项之后放置一个检查标记。 |

**说明**

在设计时可以使用“菜单编辑器”将 **Checked** 设置为 **True**。在运行时，作为 **Menu** 控件所附加的单击事件过程的一部分，能够将 **Checked** 在开和关的状态之间转换。同样可以在启动过程中或在窗体的装入事件过程中设置 **Checked** 的值。

对于一个 **Menu** 控件，**Checked** 在运行时通常是可读/写的。但是，对于那些被 Visual Basic 的加载宏遗弃或提供的菜单项来说，**Checked** 是只读的，例如在“外接程序”菜单中的“外接程序管理器”命令。

# PopupMenu 方法

用以在 **MDIForm** 或 **Form** 对象上的当前鼠标位置或指定的坐标位置显示弹出式菜单。不支持命名参数。

**语法**

*object***.PopupMenu** *menuname*,*flags*,*x*,*y, boldcommand*

**PopupMenu** 方法的语法包含下列部分：

|  |  |
| --- | --- |
| **部分** | **描述** |
| *object* | 可选的。一个对象表达式，其值为“应用于”列表中的一个对象。如果省略 *object*，则带有焦点的 **Form** 对象缺省为*object*。 |
| *Menuname* | 必需的。要显示的弹出式菜单名。指定的菜单必须含有至少一个子菜单。 |
| *Flags* | 可选的。一个数值或常数，按照下列设置中的描述，用以指定弹出式菜单的位置和行为。 |
| *X* | 可选的。指定显示弹出式菜单的 x 坐标。如果该参数省略，则使用鼠标的坐标。 |
| *Y* | 可选的。指定显示弹出式菜单的 y 坐标。如果该参数省略，则使用鼠标的坐标。 |
| *boldcommand* | 可选的。指定弹出式菜单中的菜单控件的名字，用以显示其黑体正文标题。如果该参数省略，则弹出式菜单中没有以黑体字出现的控件。 |

**设置值**

用于 *flag* 的设置值有：

|  |  |  |
| --- | --- | --- |
| **常数位置** | **值** | **描述** |
| **vbPopupMenuLeftAlign** | 0 | （缺省值）。弹出式菜单的左边定位于 *x*。 |
| **vbPopupMenuCenterAlign** | 4 | 弹出式菜单的于 *x* 居中位。 |
| **vbPopupMenuRightAlign** | 8 | 弹出式菜单的右边定位于 *x*。 |

|  |  |  |
| --- | --- | --- |
| **常数行为** | **值** | **描述** |
| **vbPopupMenuLeftButton** | 0 | （缺省值）。 仅当使用鼠标左按钮时， 弹出式菜单中的项目才响应鼠标单击。 |
| **vbPopupMenuRightButton** | 2 | 不论使用鼠标右按钮还是左按钮， 弹出式菜单中的项目都响应鼠标单击。 |

**说明**

上述常数在对象浏览器中的 Visual Basic (VB) 对象库里列出。

可使用 **ScaleMode** 属性指定x和y坐标的度量单位。x 和 y 坐标定义弹出式菜单相对于指定窗体显示的位置。如果没有包括 x 和 y 坐标，则弹出式菜单就显示在鼠标指针当前的位置。

在显示弹出式菜单时，调用 **PopupMenu** 方法后面的代码直到用户或者从菜单中选择了命令（这时，该命令的 Click 事件的代码比 **PopupMenu** 语句后面的代码先执行）或者取消该菜单时才能执行。此外，每次只能显示一个弹出式菜单，因此，如果已经显示了一个弹出式菜单或打开了一个下拉式菜单时，该方法的其它调用将被忽略。

# PopupMenu 方法

用以在 **MDIForm** 或 **Form** 对象上的当前鼠标位置或指定的坐标位置显示弹出式菜单。不支持命名参数。

**语法**

*object***.PopupMenu** *menuname*,*flags*,*x*,*y, boldcommand*

**PopupMenu** 方法的语法包含下列部分：

|  |  |
| --- | --- |
| **部分** | **描述** |
| *object* | 可选的。一个对象表达式，其值为“应用于”列表中的一个对象。如果省略 *object*，则带有焦点的 **Form** 对象缺省为*object*。 |
| *Menuname* | 必需的。要显示的弹出式菜单名。指定的菜单必须含有至少一个子菜单。 |
| *Flags* | 可选的。一个数值或常数，按照下列设置中的描述，用以指定弹出式菜单的位置和行为。 |
| *X* | 可选的。指定显示弹出式菜单的 x 坐标。如果该参数省略，则使用鼠标的坐标。 |
| *Y* | 可选的。指定显示弹出式菜单的 y 坐标。如果该参数省略，则使用鼠标的坐标。 |
| *boldcommand* | 可选的。指定弹出式菜单中的菜单控件的名字，用以显示其黑体正文标题。如果该参数省略，则弹出式菜单中没有以黑体字出现的控件。 |

**设置值**

用于 *flag* 的设置值有：

|  |  |  |
| --- | --- | --- |
| **常数位置** | **值** | **描述** |
| **vbPopupMenuLeftAlign** | 0 | （缺省值）。弹出式菜单的左边定位于 *x*。 |
| **vbPopupMenuCenterAlign** | 4 | 弹出式菜单的于 *x* 居中位。 |
| **vbPopupMenuRightAlign** | 8 | 弹出式菜单的右边定位于 *x*。 |

|  |  |  |
| --- | --- | --- |
| **常数行为** | **值** | **描述** |
| **vbPopupMenuLeftButton** | 0 | （缺省值）。 仅当使用鼠标左按钮时， 弹出式菜单中的项目才响应鼠标单击。 |
| **vbPopupMenuRightButton** | 2 | 不论使用鼠标右按钮还是左按钮， 弹出式菜单中的项目都响应鼠标单击。 |

**说明**

上述常数在对象浏览器中的 Visual Basic (VB) 对象库里列出。

可使用 **ScaleMode** 属性指定x和y坐标的度量单位。x 和 y 坐标定义弹出式菜单相对于指定窗体显示的位置。如果没有包括 x 和 y 坐标，则弹出式菜单就显示在鼠标指针当前的位置。

在显示弹出式菜单时，调用 **PopupMenu** 方法后面的代码直到用户或者从菜单中选择了命令（这时，该命令的 Click 事件的代码比 **PopupMenu** 语句后面的代码先执行）或者取消该菜单时才能执行。此外，每次只能显示一个弹出式菜单，因此，如果已经显示了一个弹出式菜单或打开了一个下拉式菜单时，该方法的其它调用将被忽略。

# PopupMenu 方法示例

本示例显示当用户在窗体上单击鼠标右键时在光标处出现的一个弹出式菜单。要检验此示例，创建一个窗体，它包含一个称为 mnuFile（mnuFile 必需至少有一个 submenu）的 **Menu** 控件。将本例代码粘贴到一个窗体的声明部分，然后按 F5 键。

Private Sub Form\_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)

 If Button = 2 Then

 PopupMenu mnuFile

 End If

End Sub

您的朋友 相思成灾 为这封邮件插入了背景音乐 - 下载  播放

播放器加载中...

正在发送...

此邮件已成功发送。再回一封