



ACCSEE函数汇总

目 录

(一) Abs 函数.....	4
(二) Array 函数.....	4
(三) Asc 函数.....	4
(四) Atn 函数.....	5
(五) CallByName 函数.....	5
(六) 类型转换函数.....	5
(七) Choose 函数.....	7
(九) Chr 函数.....	7
(十) Command 函数.....	8
(十一) Cos 函数.....	8
(十二) CreateObject 函数.....	8
(十三) CurDir 函数.....	10
(十四) CVErr 函数.....	10
(十五) Date 函数.....	10
(十六) DateAdd 函数.....	10
(十七) DateDiff 函数.....	11
(十八) DatePart 函数.....	13
(十九) DateSerial 函数.....	14
(二十) DateValue 函数.....	14
(二十一) Day 函数.....	15
(二十二) DDB 函数.....	15

(二十三) Dir 函数.....	15
(二十四) DoEvents 函数.....	16
(二十五) Environ 函数.....	17
(二十六) EOF 函数.....	17
(二十七) Error 函数.....	17
(二十八) Exp 函数.....	18
(二十九) FileAttr 函数.....	18
(三十) FileDateTime 函数.....	18
(三十一) FileLen 函数.....	19
(三十二) Filter 函数.....	19
(三十五) Int、Fix 函数.....	19
(三十六) Format 函数.....	20
(三十七) FormatCurrency 函数.....	21
(三十八) FormatDateTime 函数.....	22
(三十九) FormatNumber 函数.....	22
(四十) FormatPercent 函数.....	23
(四十一) FreeFile 函数.....	24
(四十二) FV 函数.....	24
(四十三) GetAllSettings 函数.....	24
(四十四) GetAttr 函数.....	25
(四十五) GetObject 函数.....	25
(四十六) GetSetting 函数.....	26
(四十七) Hex 函数.....	27
(四十八) Hour 函数.....	27
(四十九) Iif 函数.....	27
(五十) IMEStatus 函数.....	28
(五十一) Input 函数.....	28
(五十二) InputBox 函数.....	29
(五十三) InStr 函数.....	29
(五十四) InStrRev 函数.....	30
(五十五) IPmt 函数.....	31
(五十六) IRR 函数.....	32
(五十七) IsArray 函数.....	32
(五十八) IsDate 函数.....	32
(五十九) IsEmpty 函数.....	32
(六十) IsError 函数.....	33
(六十一) IsMissing 函数.....	33
(六十二) IsNull 函数.....	33
(六十三) IsNumeric 函数.....	34
(六十四) IsObject 函数.....	34
(六十五) Join 函数.....	34
(六十六) LBound 函数.....	35
(六十七) LCase 函数.....	35
(六十八) Left 函数.....	35
(六十九) Len 函数.....	36
(七十) Loc 函数.....	36

(七十一) LOF 函数.....	36
(七十二) Log 函数.....	37
(七十三) LTrim、RTrim 与 Trim 函数.....	37
(七十四) MacID 函数.....	37
(七十五) MacScript 函数.....	38
(七十六) Mid 函数.....	38
(七十七) Minute 函数.....	38
(七十八) MIRR 函数.....	38
(七十九) Month 函数.....	39
(八十) MonthName 函数.....	39
(八十一) MsgBox 函数.....	39
(八十二) Now 函数.....	41
(八十三) NPer 函数.....	41
(八十四) NPV 函数.....	41
(八十五) Oct 函数.....	42
(八十六) Partition 函数.....	42
(八十七) Pmt 函数.....	43
(八十八) PPmt 函数.....	43
(八十九) PV 函数.....	44
(九十) QBColor 函数.....	45
(九十一) Rate 函数.....	45
(九十二) Replace 函数.....	46
(九十三) RGB 函数.....	46
(九十四) Right 函数.....	47
(九十五) Rnd 函数.....	47
(九十六) Round 函数.....	48
(九十七) Second 函数.....	48
(九十八) Seek 函数.....	48
(九十九) Sgn 函数.....	49
(一百) Shell 函数.....	49
(一百零一) Sin 函数.....	50
(一百零二) SLN 函数.....	50
(一百零三) Space 函数.....	50
(一百零四) Spc 函数.....	51
(一百零五) Split 函数.....	51
(一百零六) Sqr 函数.....	52
(一百零七) Str 函数.....	52
(一百零八) StrComp 函数.....	52
(一百零九) StrConv 函数.....	53
(一百一十) StrReverse 函数.....	53
(一百一十一) String 函数.....	54
(一百一十二) Switch 函数.....	54
(一百一十三) SYD 函数.....	54
(一百一十四) Tab 函数.....	55
(一百一十五) Tan 函数.....	55
(一百一十六) Time 函数.....	56

(一百一十七) Timer 函数.....	56
(一百一十八) TimeSerial 函数.....	56
(一百一十九) TimeValue 函数.....	56
(一百二十) TypeName 函数.....	57
(一百二十一) UBound 函数.....	57
(一百二十二) UCase 函数.....	58
(一百二十三) Val 函数.....	58
(一百二十四) VarType 函数.....	58
(一百二十五) Weekday 函数.....	59
(一百二十六) WeekdayName 函数.....	60
(一百二十七) Year 函数.....	60

(一) Abs 函数

返回参数的绝对值，其类型和参数相同。

语法

`Abs(number)`

必要的 *number* 参数是任何有效的数值表达式，如果 *number* 包含 `Null`，则返回 `Null`，如果 *number* 是未初始化的变量，则返回 0。

说明

一个数的绝对值是将正负号去掉以后的值。例如，`ABS(-1)` 和 `ABS(1)` 都返回 1。

(二) Array 函数

返回一个包含数组的 `Variant`。

语法

`Array(arglist)`

所需的 *arglist* 参数是一个用逗号隔开的值表，这些值用于给 `Variant` 所包含的数组的各元素赋值。如果不提供参数，则创建一个长度为 0 的数组。

说明

用来表示数组元素的符号由变量名、圆括号以及括号中的所需元素的索引号组成。在下面的示例中，第一条语句创建一个 `Variant` 的变量 A。第二条语句将一个数组赋给变量 A。最后一条语句将该数组的第二个元素的值赋给另一个变量。

```
Dim A As Variant
```

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

```
B = A(2)
```

使用 `Array` 函数创建的数组的下界受 `Option Base` 语句指定的下界的决定，除非 `Array` 是由类型库（例如 `VBA.Array`）名称限定。如果是由类型库名称限定，则 `Array` 不受 `Option Base` 的影响。

注意 没有作为数组声明的 `Variant` 也可以表示数组。除了长度固定的字符串以及用户定义类型之外，`Variant` 变量可以表示任何类型的数组。尽管一个包含数组的 `Variant` 和一个元素为 `Variant` 类型的数组在概念上有所不同，但对数组元素的访问方式是相同的。

(三) Asc 函数

返回一个 `Integer`，代表字符串中首字母的字符代码。

语法

`Asc(string)`

必要的 *string* 参数可以是任何有效的字符串表达式。如果 *string* 中没有包含任何字符，则会产生运行时错误。

说明

在非 DBCS 系统下，返回值范围为 0 - 255。在 DBCS 系统下，则为 -32768 - 32767。

注意 AscB 函数作用于包含在字符串中的字节数据，AscB 返回第一个字节的字符代码，而非字符的字符代码。AscW 函数返回 Unicode 字符代码，若平台不支持 Unicode，则与 Asc 函数功能相同。

(四) Atn 函数

返回一个 Double，指定一个数的反正切值。

语法

Atn(*number*)

必要的 *number* 参数是一个 Double 或任何有效的数值表达式。

说明

Atn 函数的参数值 (*number*) 为直角三角形两边的比值并返回以弧度为单位的角。这个比值是角的对边长度除以角的邻边长度之商。

值的范围在 $-\pi/2$ 和 $\pi/2$ 弧度之间。

为了将角度转换为弧度，请将角度乘以 $\pi/180$ 。为了将弧度转换为角度，请将弧度乘以 $180/\pi$ 。

注意 Atn 是 Tan 的反三角函数，Tan 的参数值为角度，返回直角三角形的两条边的比值。不要将 Atn 和余切函数混淆，余切函数值是正切函数值的倒数， $\text{cotangent} = (1/\text{tangent})$ 。

(五) CallByName 函数

执行一个对象的方法，或者设置或返回一个对象的属性。

语法

CallByName(*object*, *procname*, *calltype*, [*args()*])

CallByName 函数的语法有以下命名参数：

部分	描述
<i>object</i>	必需的； 变体型（对象） 。函数将要执行的对象的名称。
<i>procname</i>	必需的； 变体型（字符串） 。一个包含该对象的属性名称或者方法名称的字符串表达式。
<i>calltype</i>	必需的； 常数 。一个 vbCallType 类型的常数，代表正在被调用的过程的类型。
<i>args()</i>	可选的； 变体型（数组） 。

说明

CallByName 函数用于获取或者设置一个属性，或者在运行时使用一个字符串名称来调用一个方法。

在下面的例子中，第一行使用 CallByName 来设置一个文本框的 MousePointer 属性，第二行得到 MousePointer 属性的值，第三行调用 Move 方法来移动文本框：

```
CallByName Text1, "MousePointer", vbLet, vbCrosshair
```

```
Result = CallByName (Text1, "MousePointer", vbGet)
CallByName Text1, "Move", vbMethod, 100, 100
```

(六) 类型转换函数

每个函数都可以强制将一个表达式转换成某种特定数据类型。

语法

```
CBool(expression)
CByte(expression)
CCur(expression)
CDate(expression)
CDBl(expression)
CDec(expression)
CInt(expression)
CLng(expression)
CSng(expression)
CStr(expression)
CVar(expression)
CStr(expression)
```

必要的 *expression* 参数可以是任何字符串表达式或数值表达式。

返回类型

函数名称决定返回类型，如下所示：

函数	返回类型	<i>expression</i> 参数范围
CBool	Boolean	任何有效的字符串或数值表达式。
CByte	Byte	0 至 255。
CCur	Currency	-922, 337, 203, 685, 477. 5808 至 922, 337, 203, 685, 477. 5807。
CDate	Date	任何有效的日期表达式。
CDBl	Double	负数从 -1. 79769313486231E308 至 -4. 94065645841247E-324；正数从 4. 94065645841247E-324 至 1. 79769313486232E308。
CDec	Decimal	零变比数值，即无小数位数值，为 +/-79, 228, 162, 514, 264, 337, 593, 543, 950, 335。对于 28 位小数的数值，范围则为 +/-7. 9228162514264337593543950335；最小的可能非零值是 0. 00000000000000000000000000000001。
CInt	Integer	-32, 768 至 32, 767，小数部分四舍五入。
CLng	Long	-2, 147, 483, 648 至 2, 147, 483, 647，小数部分四舍五入。
CSng	Single	负数为 -3. 402823E38 至 -1. 401298E-45；正数为 1. 401298E-45 至 3. 402823E38。
CStr	String	依据 <i>expression</i> 参数返回 Cstr。
CVar	Variant	若为数值，则范围与 Double 相同；若不为数值，则范围与 String 相同。

说明

如果传递给函数的 *expression* 超过转换目标数据类型的范围，将发生错误。

通常，在编码时可以使用数据类型转换函数，来体现某些操作的结果应该表示为特定的数据类型，而不是缺省的数据类型。例如，当单精度、双精度或整数运算发生的情况下，使用 **CCur** 来强制执行货币运算。

应该使用数据类型转换函数来代替 **Val**，以使国际版的数据转换可以从一种数据类型转换为另一种。例如，当使用 **Ccur** 时，不同的小数点分隔符、千分位分隔符和各种货币选项，依据系统的区域设置都会被妥善识别。

当小数部分恰好为 0.5 时，**Cint** 和 **CLng** 函数会将它转换为最接近的偶数值。例如，0.5 转换为 0、1.5 转换为 2。**Cint** 和 **CLng** 函数不同于 **Fix** 和 **Int** 函数，**Fix** 和 **Int** 函数会将小数部分截断而不是四舍五入。并且 **Fix** 和 **Int** 函数总是返回与传入的数据类型相同的值。

使用 **IsDate** 函数，可判断 *date* 是否可以被转换为日期或时间。**Cdate** 可用于识别日期文字和时间文字，以及落入可接受的日期范围内的数值。当转换一个数字成为日期时，是将整数部分转换为日期，小数部分转换为从午夜起算的时间。

CDate 依据系统上的区域设置来决定日期的格式。如果提供的格式为不可识别的日期设置，则不能正确判断年、月、日的顺序。另外，长日期格式，若包含有星期的字符串，也不能被识别。

CVDate 函数也提供对早期 Visual Basic 版本的兼容性。**CVDate** 函数的语法与 **CDate** 函数是完全相同的，不过，**CVDate** 是返回一个 **Variant**，它的子类型是 **Date**，而不是实际的 **Date** 类型。因为现在已有真正的 **Date** 类型，所以 **CVDate** 也不再需要了。转换一个表达式成为 **Date**，再赋值给一个 **Variant**，也可以达到同样的效果。也可以使用这种技巧将其他真正的数据类型转换为对等的 **Variant** 子类型。

注意 **CDec** 函数不能返回独立的数据类型，而总是返回一个 **Variant**，它的值已经被转换为 **Decimal** 子类型。

(七) Choose 函数

(八) 从参数列表中选择并返回一个值。

语法

Choose(*index*, *choice-1* [, *choice-2*, ... [, *choice-n*]])

Choose 函数的语法具有以下几个部分：

部分	描述
<i>index</i>	必要参数，数值表达式或字段，它的运算结果是一个数值，且界于 1 和可选择的项目数之间。
<i>choice</i>	必要参数，Variant 表达式，包含可选择项目的其中之一。

说明

Choose 会根据 *index* 的值来返回选择项列表中的某个值。如果 *index* 是 1，则 **Choose** 会返回列表中的第 1 个选择项。如果 *index* 是 2，则会返回列表中的第 2 个选择项，以此类推。

可以使用 **Choose** 来查阅一个列表中的项目。例如，如果 *index* 所指定的值为 3，而 *choice-1* = "one"、*choice-2* = "two"、且 *choice-3* = "three"，那么 **Choose** 将返回 "three"。当 *index* 代表一选项组中的值时，则这项功能将会特别有用。

即使它只返回一个选项值，**Choose** 仍然会计算列表中的每个选择项。所以应该注意到这项副作用。例如，当在每个选择项表达式中使用了 **MsgBox** 函数作为其中的一部分时，每计算一个选择项，就会显示一次消息框。

当 *index* 小于 1 或大于列出的选择项数目时，**Choose** 函数返回 Null。
如果 *index* 不是整数，则会先四舍五入为与其最接近的整数。

(九) Chr 函数

返回 String，其中包含有与指定的字符代码相关的字符。

语法

Chr(*charcode*)

必要的 *charcode* 参数是一个用来识别某字符的 Long。

说明

0 到 31 之间的数字与标准的非打印 ASCII 代码相同。例如，**Chr**(10) 可以返回换行字符。*charcode* 的正常范围为 0 - 255。然而，在 DBCS 系统，*charcode* 的实际范围为 -32768 到 65535。

注意 **ChrB** 函数作用于包含在 **String** 中的字节数据。**ChrB** 总是返回一个单字节，而不是返回一个字符，一个字符可能是一个或两个字节。**ChrW** 函数返回包含 Unicode 的 **String**，若在不支持 Unicode 的平台上，则其功能与 **Chr** 函数相同。

注意 Visual Basic for the Macintosh 不支持 Unicode 字符串。因此，当 *n* 值在 128 - 65,535 范围内时，**ChrW**(*n*) 不能像在 Windows 环境中那样返回所有的 Unicode 字符。相反地，当 Unicode 的 *n* 值大于 127 时，**ChrW**(*n*) 会试图做一个“最好的猜测”。因此，在 Macintosh 环境中，不能使用 **ChrW**。

(十) Command 函数

返回命令行的参数部分，该命令行用于装入 Microsoft Visual Basic 或 Visual Basic 开发的可执行程序。Visual Basic **Command** 函数在 Microsoft Office 应用程序中不可用。

语法

Command

说明

当从命令行装入 Visual Basic 时，/cmd 之后的命令行的任何部分作为命令行的参数传递给程序。下面的示例中，*cmdlineargs* 代表 **Command** 函数返回的参数信息。

VB /cmd *cmdlineargs*

对于使用 Visual Basic 开发并编译为 .exe 文件的应用程序，**Command** 返回出现在命令林中应用程序名之后的任何参数。例如：

MyApp *cmdlineargs*

想知道如何在正在使用的应用程序的用户界面中改变命令行参数，请搜寻关于“命令行参数”的帮助。

(十一) Cos 函数

返回一个 **Double**，指定一个角的余弦值。

语法

Cos(*number*)

必要的 *number* 参数是一 Double 或任何有效的数值表达式，表示一个以弧度为单位的角。

说明

Cos 函数的参数为一个角，并返回直角三角形两边的比值。该比值为角的邻边长度除以斜边长度之商。

结果的取值范围在 -1 到 1 之间。

为了将角度转换成弧度，请将角度乘以 $\pi/180$ 。为了将弧度转换成角度，请将弧度乘以 $180/\pi$ 。

(十二) CreateObject 函数

创建并返回一个对 ActiveX 对象的引用。

语法

CreateObject(*class*, [*servername*])

CreateObject 函数的语法有如下部分：

部分	描述
<i>class</i>	必需的； Variant (String) 。要创建的应用程序名称和类。
<i>servername</i>	可选的； Variant (String) 。要在其上创建对象的网络服务器名称。如果 <i>servername</i> 是一个空字符串(""), 即使用本地机器。

class 参数使用 *appname.objecttype* 这种语法，包括以下部分：

部分	描述
<i>appname</i>	必需的； Variant (字符串) 。提供该对象的应用程序名。
<i>objecttype</i>	必需的； Variant (字符串) 。待创建对象的类型或类。

说明

每个支持自动化的应用程序都至少提供一种对象类型。例如，一个字处理应用程序可能会提供 **Application** 对象，**Document** 对象，以及 **Toolbar** 对象。

要创建 ActiveX 对象，只需将 **CreateObject** 返回的对象赋给一个对象变量：

```
' 声明一个对象变量来存放该对象
```

```
' 的引用。Dim as Object 采用后期绑定方式。
```

```
Dim ExcelSheet As Object
```

```
Set ExcelSheet = CreateObject("Excel.Sheet")
```

上述代码将启动该应用程序创建该对象，在本例中就是创建一个 Microsoft Excel 电子数据表。对象创建后，就可以在代码中使用自定义的对象变量来引用该对象。在下面的示例中，可以使用对象变量 *ExcelSheet* 来访问新建对象的属性和方法，以及访问 Microsoft Excel 的其它对象，包括应用程序对象和单元格集合。

```
' 设置 Application 对象使 Excel 可见
```

```
ExcelSheet.Application.Visible = True
```

```
' 在表格的第一个单元中写些文本
```

```
ExcelSheet.Application.Cells(1, 1).Value = "This is column A, row 1"
```

```
' 将该表格保存到 C:\test.xls 目录
```

```
ExcelSheet.SaveAs "C:\TEST.XLS"
```

```
' 使用应用程序对象的 Quit 方法关闭 Excel。
```

```
ExcelSheet.Application.Quit
```

' 释放该对象变量

```
Set ExcelSheet = Nothing
```

使用 `As Object` 子句声明对象变量，可以创建一个能包含任何类型对象引用的变量。不过，该变量访问对象是后期绑定的，也就是说，绑定在程序运行时才进行。要创建一个使用前期绑定方式的对象变量，也就是说，在程序编译时就完成绑定，则对象变量在声明时应指定类 ID。例如，可以声明并创建下列 Microsoft Excel 引用：

```
Dim xlApp As Excel.Application  
Dim xlBook As Excel.Workbook  
Dim xlSheet As Excel.WorkSheet  
Set xlApp = CreateObject("Excel.Application")  
Set xlBook = xlApp.Workbooks.Add  
Set xlSheet = xlBook.Worksheets(1)
```

前期绑定的变量引用可以提供更好的性能，但该变量只能存放声明中所指定的类的引用。

可以将 `CreateObject` 函数返回的对象传给一个参数为对象的函数。例如，下面的代码创建并传递了一个 `Excel.Application` 对象的引用：

```
Call MySub (CreateObject("Excel.Application"))
```

可以在一个远端连网的计算机上创建一个对象，方法是把计算机的名称传递给 `CreateObject` 的 `servername` 参数。这个名称与共享名称的机器名部份相同：对于一个共享名称为 `"\\MyServer\Public,"` 的 `servername` 参数是 `"MyServer"`。

注意 关于使应用程序在远程网络计算机上可见的详细信息，请参阅 COM 文档（参阅 *Microsoft Developer Network*）。您可能必须给应用程序添加注册号。

下面的代码返回在一个名为 `MyServer` 的远端计算机上运行的 Excel 实例的版本号：

```
Dim xlApp As Object  
Set xlApp = CreateObject("Excel.Application", "MyServer")  
Debug.Print xlApp.Version
```

如果远端服务器不存在或者不可用，则会发生一个运行时错误。

注意 当该对象当前没有实例时，应使用 `CreateObject`。如果该对象已有实例在运行，就会启动一个新的实例，并创建一个指定类型的对象。要使用当前实例，或要启动该应用程序并加载一个文件，可以使用 `GetObject` 函数。

如果对象已登记为单个实例对象，则不管执行多少次 `CreateObject`，都只能创建该对象的一个实例。

(十三) CurDir 函数

返回一个 `Variant (String)`，用来代表当前的路径。

语法

```
CurDir[(drive)]
```

可选的 `drive` 参数是一个字符串表达式，它指定一个存在的驱动器。如果没有指定驱动器，或 `drive` 是零长度字符串（""），则 `CurDir` 会返回当前驱动器的路径。在 Macintosh 上，`CurDir` 忽略任何指定的 `drive`，并只简单地返回当前驱动器的路径。

(十四) CVerErr 函数

返回 **Error** 子类型的 **Variant**，其中包含指定的错误号。

语法

CVerErr(*errornumber*)

必要的 *errornumber* 参数可以是任何有效的错误号代码。

说明

可以在过程中，使用 **CVerErr** 函数来创建用户自定义错误。例如，如果创建一个函数，它可以接受若干个参数，且正常返回一个字符串，则可以让函数来判断输入的参数，确认它们是在可接受的范围内。如果不是的话，此函数将不会返回所要的字符串。在这种情况下，**CVerErr** 可以返回一个错误号，并告知应该采取的行动。

注意，**Error** 的隐式转换是不允许的，例如，不能直接把 **CVerErr** 的返回值赋值给一个非 **Variant** 的变量。然而，可以对 **CVerErr** 的返回值进行显式转换（使用 **CInt**、**Cdbl** 等等），并赋值给适当的数据类型变量。

(十五) Date 函数

返回包含系统日期的 **Variant (Date)**。

语法

Date

说明

为了设置系统日期，请使用 **Date** 语句。

(十六) DateAdd 函数

返回包含一个日期的 **Variant (Date)**，这一日期还加上了一段时间间隔。

语法

DateAdd(*interval*, *number*, *date*)

DateAdd 函数语法中有下列命名参数：

部分	描述
<i>interval</i>	必要。字符串表达式，是所要加上去的时间间隔。
<i>number</i>	必要。数值表达式，是要加上的时间间隔的数目。其数值可以为正数（得到未来的日期），也可以为负数（得到过去的日期）。
<i>date</i>	必要。 Variant (Date) 或表示日期的文字，这一日期还加上了时间间隔。

设置

interval 参数具有以下设定值：

设置	描述
yyyy	年
q	季
m	月

y	一年的日数
d	日
w	一周的日数
ww	周
h	时
n	分钟
s	秒

说明

可以使用 **DateAdd** 函数对日期加上或减去指定的时间间隔。例如，可以用 **DateAdd** 来计算距今天为三十天的日期；或者计算距现在为 45 分钟的时间。

为了对 **date** 加上“日”，可以使用“一年的日数”（“y”），“日”（“d”）或“一周的日数”（“w”）。

DateAdd 函数将不返回有效日期。在以下实例中将 1 月 31 日加上一个月：

```
DateAdd(m, 1, 31-Jan-95)
```

上例中，**DateAdd** 返回 1995 年 2 月 28 日，而不是 1995 年 2 月 31 日。如果 **date** 是 1996 年 1 月 31 日，则由于 1996 年是闰年，返回值是 1996 年 2 月 29 日。

如果计算的日期超前 100 年（减去的年度超过 **date** 中的年份），就会导致错误发生。

如果 **number** 不是一个 Long 值，则在计算时取最接近的整数值来计算。

注意 **DateAdd** 返回值的格式由 **Control Panel** 设置决定，而不是由传递到 **date** 参数的格式决定。

(十七) DateDiff 函数

返回 Variant (Long) 的值，表示两个指定日期之间的时间间隔数目。

语法

```
DateDiff(interval, date1, date2[, firstdayofweek[, firstweekofyear]])
```

DateDiff 函数语法中有下列命名参数：

部分	描述
<i>interval</i>	必要。字符串表达式，表示用来计算 <i>date1</i> 和 <i>date2</i> 的时间差的时间间隔
<i>date1</i> □ <i>date2</i>	必要；Variant (Date)。计算中要用到的两个日期。
<i>Firstdayofweek</i>	可选。指定一个星期的第一天的常数。如果未予指定，则以星期日为第一天。
<i>firstweekofyear</i>	可选。指定一年的第一周的常数。如果未予指定，则以包含 1 月 1 日的星期为第一周。

设置

interval 参数的设定值如下：

设置	描述
yyyy	年
q	季
m	月
y	一年的日数
d	日

w	一周的日数
ww	周
h	时
n	分钟
s	秒

firstdayofweek 参数的设定值如下:

常数	值	描述
vbUseSystem	0	使用 NLS API 设置。
vbSunday	1	星期日 (缺省值)
vbMonday	2	星期一
vbTuesday	3	星期二
vbWednesday	4	星期三
vbThursday	5	星期四
vbFriday	6	星期五
vbSaturday	7	星期六

常数	值	描述
vbUseSystem	0	用 NLS API 设置。
vbFirstJan1	1	从包含 1 月 1 日的星期开始 (缺省值)。
vbFirstFourDays	2	从第一个其大半个星期在新的一年的一周开始。
vbFirstFullWeek	3	从第一个无跨年度的星期开始。

说明

DateDiff 函数用来决定两个日期之间所指定的时间间隔数目。例如, 可以使用 **DateDiff** 来计算两个日期之间相隔几日, 或计算从今天起到年底还有多少个星期。

为了计算 *date1* 与 *date2* 相差的日数, 可以使用“一年的日数”(y) 或“日”(d)。当 *interval* 是“一周的日数”(w) 时, **DateDiff** 返回两日期期间的周数。如果 *date1* 是星期一, **DateDiff** 计算到 *date2* 为止的星期一的个数。这个数包含 *date2* 但不包含 *date1*。不过, 如果 *interval* 是“周”(ww), 则 **DateDiff** 函数返回两日期期间的“日历周”数。由计算 *date1* 与 *date2* 之间星期日的个数而得。如果 *date2* 刚好是星期日, 则 *date2* 也会被加进 **DateDiff** 的计数结果中; 但不论 *date1* 是否为星期日, 都不将它算进去。

如果 *date1* 比 *date2* 来得晚, 则 **DateDiff** 函数的返回值为负数。

firstdayofweek 参数会影响使用时间间隔符号“W”或“WW”计算的结果。

如果 *date1* 或 *date2* 是日期文字, 则指定的年份成为该日期的固定部分。但是, 如果 *date1* 或 *date2* 用双引号(“”)括起来, 且年份略而不提, 则在每次计算表达式 *date1* 或 *date2* 时, 当前年份都会插入到代码之中。这样就可以书写适用于不同年份的程序代码。

在计算 12 月 31 日和来年的 1 月 1 日的年份差时, **DateDiff** 返回 1 表示相差一个年份, 虽然实际上只相差一天而已。

(十八) DatePart 函数

返回一个包含已知日期的指定时间部分的 Variant (Integer)。

语法

DatePart(*interval*, *date*[, *firstdayofweek*[, *firstweekofyear*]])

DatePart 函数语法中有下列命名参数：

部分	描述
<i>interval</i>	必要。字符串表达式，是要返回的时间间隔。
<i>date</i>	必要。要计算的 Variant (Date) 值。
<i>Firstdayofweek</i>	可选。指定一个星期的第一天的常数。如果未予指定，则以星期日为第一天。
<i>firstweekofyear</i>	可选。指定一年第一周的常数。如果未予指定，则以包含 1 月 1 日的星期为第一周。

设置

interval 参数的设定值如下：

设置	描述
yyyy	年
q	季
m	月
y	一年的日数
d	日
w	一周的日数
ww	周
h	时
n	分钟
s	秒

firstdayofweek 参数的设定值如下：

常数	值	描述
vbUseSystem	0	使用 NLS API 设置。
vbSunday	1	星期日（缺省值）
vbMonday	2	星期一
vbTuesday	3	星期二
vbWednesday	4	星期三
vbThursday	5	星期四
vbFriday	6	星期五
vbSaturday	7	星期六

firstweekofyear 参数的设定值如下：

常数	值	描述
vbUseSystem	0	使用 NLS API 设置。
vbFirstJan1	1	从包含 1 月 1 日的星期开始（缺省值）。
vbFirstFourDays	2	从第一个其大半个星期在新的一年的一周开始。
vbFirstFullWeek	3	从第一个无跨年度的星期开始。

说明

DatePart 函数可以用来计算日期并返回指定的时间间隔。例如，可以使用 DatePart 计算某个日期是星期几或目前为几点钟。

firstdayofweek 参数会影响使用时间间隔符号 “W” 或 “WW” 计算的结果。

如果 *date* 是日期文字，则指定的年份成为该日期的固定部分。但是，如果 *date* 用双引号 (“”) 括起来，且年份略而不提，则在每次计算 *date* 表达式时，当前年份都会插入到代码之中。这样就可以书写适用于不同年份的程序代码。

(十九) DateSerial 函数

返回包含指定的年、月、日的 Variant (Date)。

语法

DateSerial(*year*, *month*, *day*)

DateSerial 函数语法有下列的命名参数：

部分	描述
<i>year</i>	必要；Integer。从 100 到 9999 间的整数，或一数值表达式。
<i>month</i>	必要；Integer。任何数值表达式。
<i>day</i>	必要；Integer。任何数值表达式。

说明

为了指定某个日期，如 1991 年 12 月 31 日，DateSerial 函数中的每个参数的取值范围应该是可接受的；即，日的取值范围应在 1-31 之间，而月的取值范围应在 1-12 之间。但是，当一个数值表达式表示某日之前或之后的年、月、日数时，也可以为每个使用这个数值表达式的参数指定相对日期。

以下示例中使用了数值表达式代替绝对日期。这里，DateSerial 函数返回 1990 年 8 月 1 日的十年 (1990 - 10) 零两个月 (8 - 2) 又一天 (1 - 1) 之前的日期；换句话说，就是 1980 年 5 月 31 日。

DateSerial(1990 - 10, 8 - 2, 1 - 1)

year 参数的数值若介于 0 与 29 之间，则将其解释为 2000 - 2029 年，若介于 30 和 99 之间则解释为 1930 - 1999 年。而对所有其它 *year* 参数，则请用四位数值表示（如 1800）。当任何一个参数的取值超出可接受的范围时，它会适时进位到下一个较大的时间单位。例如，如果指定了 35 天，则这个天数被解释成一个月加上多出来的日数，多出来的日数将由其年份与月份来决定。如果一个参数值超出 -32, 768 到 32, 767 的范围，就会导致错误发生。

(二十) DateValue 函数

返回一个 Variant (Date)。

语法

DateValue(*date*)

必要的 *date* 参数 *date* 通常是字符串表达式，表示从 100 年 1 月 1 日到 9999 年 12 月 31 日之间的一个日期。但是，*date* 也可以是任何表达式，其所代表的日期、时间在上述范围内。

说明

如果 *date* 是一个字符串，且其内容只有数字以及分隔数字的日期分隔符，则 DateValue 就会根据系统中指定的短日期格式来识别月、日、年的顺序。DateValue 也识别明确的英文月份名称，全名或缩写均可。例如，除了 12/30/1991 和 12/30/91 之外，DateValue 也识别 December 30, 1991 和 Dec 30, 1991。

如果 *date* 中略去了年这一部分，DateValue 就会使用由计算机系统日期设置的当前年份。如果 *date* 参数包含时间信息，则 DateValue 不会返回它。但是，如果 *date* 包含无效时间信息（如 89:98），则会导致错误发生。

(二十一) Day 函数

返回一个 Variant (Integer)，其值为 1 到 31 之间的整数，表示一个月中的某一日。

语法

Day(*date*)

必要的 *date* 参数，可以是任何能够表示日期的 Variant、数值表达式、字符串表达式或它们的组合。如果 *date* 包含 Null，则返回 Null。

(二十二) DDB 函数

返回一个 Double，指定一笔资产在一特定期间的折旧。可使用双下落收复平衡方法或其它指定的方法进行计算。

语法

DDB(*cost*, *salvage*, *life*, *period*[, *factor*])

DDB 函数具有下列命名参数：

部分	描述
<i>cost</i>	必要。Double 指定资产的初始成本。
<i>salvage</i>	必要。Double. 指定使用年限结束时的资产价值。
<i>life</i>	必要。Double 指定资产可用的可用年限。
<i>period</i>	必要。Double 指定计算资产折旧所用的那一期间。
<i>factor</i>	可选。Variant 指定收复平衡下落时的速度。如果省略的话，2（双下落方法）为缺省值。

说明

双下落收复平衡方法用加速利率法计算折旧。在第一段时期，折旧为最高，而在接下来的期间内降低。

life 和 *period* 参数必须用相同的单位表示。例如，如果 *life* 用月份表示，则 *period* 也必须用月份表示。所有参数都必须是正值。

DDB 函数使用下列公式计算在一定时期后的折旧：

折旧 / *period* = ((*cost* - *alvage*) * *factor*) / *life*

(二十三) Dir 函数

返回一个 String，用以表示一个文件名、目录名或文件夹名称，它必须与指定的模式或文件属性、或磁盘卷标相匹配。

语法

Dir[(*pathname*[, *attributes*])]

Dir 函数的语法具有以下几个部分：

部分	描述
<i>pathname</i>	可选参数。用来指定文件名的字符串表达式，可能包含目录或文件夹、以及驱动器。如果没有找到 <i>pathname</i> ，则会返回零长度字符串 ("")。
<i>attributes</i>	可选参数。常数或数值表达式，其总和用来指定文件属性。如果省略，则会返

	回匹配 <i>pathname</i> 但不包含属性的文件。
--	--------------------------------

设置值

attributes 参数的设置可为：

常数	值	描述
vbNormal	0	(缺省) 指定没有属性的文件。
vbReadOnly	1	指定无属性的只读文件
vbHidden	2	指定无属性的隐藏文件
VbSystem	4	指定无属性的系统文件 在 Macintosh 中不可用。
vbVolume	8	指定卷标文件；如果指定了其它属性，则忽略 vbVolume 在 Macintosh 中不可用。
vbDirectory	16	指定无属性文件及其路径和文件夹。
vbAlias	64	指定的文件名是别名，只在 Macintosh 上可用。

注意 这些常数是 由 VBA 所指定的，在程序代码中的任何位置，可以使用这些常数来替换真正的数值。

说明

在 Microsoft Windows 中，**Dir** 支持多字符 (*) 和单字符 (?) 的通配符来指定多重文件。在 Macintosh 中，这些字符作为合法文件名字符并且不能作为通配符来指定多个文件。由于 Macintosh 不支持通配符，使用文件类型指定文件组。可以使用 **MacID** 函数指定文件类型而不用文件名。比如，下列语句返回当前文件夹中第一个 TEXT 文件的名称：

```
Dir("SomePath", MacID("TEXT"))
```

为选中文件夹中所有文件，指定一空串：

```
Dir("")
```

在 Microsoft Windows 中，如果在 **Dir** 函数中使用 **MacID** 函数，将产生错误。

任何大于 256 的 *attribute* 值都被认为是 **MacID** 函数的值。

在第一次调用 **Dir** 函数时，必须指定 *pathname*，否则会产生错误。如果也指定了文件属性，那么就必须包括 *pathname*。

Dir 会返回匹配 *pathname* 的第一个文件名。若想得到其它匹配 *pathname* 的文件名，再一次调用 **Dir**，且不要使用参数。如果已没有合乎条件的文件，则 **Dir** 会返回一个零长度字符串 ("")。一旦返回值为零长度字符串，并要再次调用 **Dir** 时，就必须指定 *pathname*，否则会产生错误。不必访问到所有匹配当前 *pathname* 的文件名，就可以改变到一个新的 *pathname* 上。但是，不能以递归方式来调用 **Dir** 函数。以 **vbDirectory** 属性来调用 **Dir** 不能连续地返回子目录。

提示 由于文件名并不会以特别的次序来返回，所以可以将文件名存储在一个数组中，然后再对这个数组排序。

(二十四) DoEvents 函数

转让控制权，以便让操作系统处理其它的事件。

语法

```
DoEvents ( )
```

说明

DoEvents 函数会返回一个 Integer，以代表 Visual Basic 独立版本中打开的窗体数目，例如，Visual Basic，专业版，在其它的应用程序中，**DoEvents** 返回 0。

DoEvents 会将控制权传给操作系统。当操作系统处理完队列中的事件，并且在 **SendKeys** 队列中的所有键也都已送出之后，返回控制权。

DoEvents 对于简化诸如允许用户取消一个已启动的过程 — 例如搜寻一个文件 — 特别有用。对于长时间过程，放弃控制权最好使用定时器或通过委派任务给 ActiveX EXE 部件来完成。以后，任务还是完全独立于应用程序，多任务及时间片由操作系统来处理。

小心 确保以 **DoEvents** 放弃控制权的过程，在第一次 **DoEvents** 返回之前，不能再次被其他部分的代码调用；否则会产生不可预料的结果。此外，如果其它的应用程序可能会和本过程以不可预知的方式进行交互操作，那么也不要使用 **DoEvents**，因为此时不能放弃控制权。

(二十五) Environ 函数

返回 **String**，它关连于一个操作系统环境变量。在 Macintosh 中不可用

语法

Environ({*envstring* | *number*})

Environ 函数的语法含有以下这些命名参数：

部分	描述
<i>envstring</i>	可选参数。包含一个环境变量名的字符串表达式。
<i>number</i>	可选参数。数值表达式，用来表示环境字符串在环境字符串表格中的数值顺序。 <i>number</i> 参数可以是任意的数值表达式，不过在计算前，它会先转换为一个整数。

说明

如果在环境字符串表格中找不到 *envstring*，则会返回一个零长度字符串 (“”)。如果找到，则 **Environ** 会返回一段文本，文本是赋值给指定的 *envstring* 的，也就是说，在环境字符串表格中对应那个环境变量的等号 (=) 后面的那段文本。

如果指定了 *number*，则在环境字符串表格中相应位置上的字符串会返回。在这种情况下，**Environ** 会返回整个文本，包括 *envstring*。如果在指定位置上没有环境字符串，那么 **Environ** 会返回一个零长度字符串。

(二十六) EOF 函数

返回一个 **Integer**，它包含 **Boolean** 值 **True**，表明已经到达为 **Random** 或顺序 **Input** 打开的文件的结尾。

语法

EOF(*filenumber*)

必要的 *filenumber* 参数是一个 **Integer**，包含任何有效的文件号。

说明

使用 **EOF** 是为了避免因试图在文件结尾处进行输入而产生的错误。

直到到达文件的结尾，**EOF** 函数都返回 **False**。对于为访问 **Random** 或 **Binary** 而打开的文件，直到最后一次执行的 **Get** 语句无法读出完整的记录时，**EOF** 都返回 **False**。

对于为访问 **Binary** 而打开的文件，在 **EOF** 函数返回 **True** 之前，试图使用 **Input** 函数读出整个文件的任何尝试都会导致错误发生。在用 **Input** 函数读出二进制文件时，要用 **LOF** 和 **Loc** 函数来替换 **EOF** 函数，或者将 **Get** 函数与 **EOF** 函数配合使用。对于为 **Output** 打开的文件，**EOF** 总是返回 **True**。

(二十七) Error 函数

返回对应于已知错误号的错误信息。

语法

Error[(*errornumber*)]

这个可选的 *errornumber* 参数可以为任何有效的错误号。如果 *errornumber* 是有效的错误号，但尚未被定义，则 **Error** 将返回字符串“应用程序定义的错误或对象定义的错误”。如果 *errornumber* 不是有效的错误号，则会导致错误发生。如果省略 *errornumber*，就会返回与最近一次运行时错误对应的消息。如果没有发生运行时错误，或者 *errornumber* 是 0，则 **Error** 返回一个长度为零的字符串 (“”)。

说明

请检查 **Err** 对象的属性设置，以便认定最近一次运行时错误。**Error** 函数的返回值对应于 **Err** 对象的 **Description** 属性。

(二十八) Exp 函数

返回 **Double**，指定 *e*（自然对数的底）的某次方。

语法

Exp(*number*)

必要的 *number* 参数 *number* 是 **Double** 或任何有效的数值表达式。

说明

如果 *number* 的值超过 709.782712893，则会导致错误发生。常数 *e* 的值大约是 2.718282。

注意 **Exp** 函数的作用和 **Log** 的作用互补，所以有时也称做反对数。

(二十九) FileAttr 函数

返回一个 **Long**，表示使用 **Open** 语句所打开文件的文件方式。

语法

FileAttr(*filenumber*, *returntype*)

FileAttr 函数的语法具有以下几个命名参数：

部分	描述
<i>filenumber</i>	必要。 Integer 类型，任何有效的文件号。
<i>returntype</i>	必要。 Integer 类型。它是数字，指出返回信息的类型。指定 1 则可返回一个代表文件方式的数值。而仅仅在 16 位系统中，指定 2 才可以恢复操作系统的文件句柄。在 32 位系统中不支持 <i>Returntype</i> 2，它会导致错误发生。

返回值

当 *returntype* 参数值为 1 时，下列返回值指出文件访问方式：

方式	值
Input	1
Output	2

Random	4
Append	8
Binary	32

(三十) FileDateTime 函数

返回一个 Variant (Date)，此为一个文件被创建或最后修改后的日期和时间。

语法

FileDateTime(*pathname*)

必要的 *pathname* 参数是用来指定一个文件名的字符串表达式。*pathname* 可以包含目录或文件夹、以及驱动器。

(三十一) FileLen 函数

返回一个 Long，代表一个文件的长度，单位是字节。

语法

FileLen(*pathname*)

必要的 *pathname* 参数是用来指定一个文件名的字符串表达式。*pathname* 可以包含目录或文件夹、以及驱动器。

说明

当调用 FileLen 函数时，如果所指定的文件已经打开，则返回的值是这个文件在打开前的大小。

注意 若要取得一个打开文件的长度大小，使用 LOF 函数。

(三十二) Filter 函数

(三十三) 描述

(三十四) 返回一个下标从零开始的数组，该数组包含基于指定筛选条件的一个字符串数组的子集。

语法

Filter(*sourcearray*, *match*[, *include*[, *compare*]])

Filter 函数语法有如下的命名参数：

部分	描述
<i>sourcearray</i>	必需的。要执行搜索的一维字符串数组。
<i>match</i>	必需的。要搜索的字符串。
<i>include</i>	可选的。Boolean 值，表示返回子串包含还是不包含 <i>match</i> 字符串。如果 <i>include</i> 是 True，Filter 返回的是包含 <i>match</i> 子字符串的数组子集。如果 <i>include</i> 是 False，Filter 返回的是不包含 <i>match</i> 子字符串的数组子集。
<i>compare</i>	可选的。数字值，表示所使用的字符串比较类型。有关其设置，请参阅下面的“设置值”部分。

设置值

Compare 参数的设置值如下:

常数	值	描述
<code>vbUseCompareOption</code>	- 1	使用 <code>Option Compare</code> 语句的设置值来执行比较。
<code>vbBinaryCompare</code>	0	执行二进制比较。
<code>vbTextCompare</code>	1	执行文字比较。
<code>vbDatabaseCompare</code>	2	只用于 Microsoft Access。基于您的数据库信息来执行比较。

说明

如果在 *sourcearray* 中没有发现与 *match* 相匹配的值, `Filter` 返回一个空数组。如果 *sourcearray* 是 `Null` 或不是一个一维数组, 则产生错误。

`Filter` 函数所返回的数组, 其元素数目刚好是所找到的匹配项目数。

(三十五) Int、Fix 函数

返回参数的整数部分。

语法

`Int(number)`

`Fix(number)`

必要的 *number* 参数是 `Double` 或任何有效的数值表达式。如果 *number* 包含 `Null`, 则返回 `Null`。

说明

`Int` 和 `Fix` 都会删除 *number* 的小数部份而返回剩下的整数。

`Int` 和 `Fix` 的不同之处在于, 如果 *number* 为负数, 则 `Int` 返回小于或等于 *number* 的第一个负整数, 而 `Fix` 则会返回大于或等于 *number* 的第一个负整数。例如, `Int` 将 -8.4 转换成 -9, 而 `Fix` 将 -8.4 转换成 -8。

`Fix(number)` 等于:

`Sgn(number) * Int(Abs(number))`

(三十六) Format 函数

返回 `Variant (String)`, 其中含有一个表达式, 它是根据格式表达式中的指令来格式化的。

语法

`Format(expression[, format[, firstdayofweek[, firstweekofyear]])`

`Format` 函数的语法具有下面几个部分:

部分	说明
<i>expression</i>	必要参数。任何有效的表达式。
<i>format</i>	可选参数。有效的命名表达式或用户自定义格式表达式。
<i>firstdayofweek</i>	可选参数。常数, 表示一星期的第一天。
<i>firstweekofyear</i>	可选参数。常数, 表示一年的第一周。

设置值

firstdayofweek 参数有下面设置:

常数	值	说明
vbUseSystem	0	使用 NLS API 设置。
VbSunday	1	星期日 (缺省)
vbMonday	2	星期一
vbTuesday	3	星期二
vbWednesday	4	星期三
vbThursday	5	星期四
vbFriday	6	星期五
vbSaturday	7	星期六

firstweekofyear 参数有下面设置:

常数	值	说明
vbUseSystem	0	使用 NLS API 设置。
vbFirstJan1	1	从包含一月一日的那一周开始 (缺省)。
vbFirstFourDays	2	从本年第一周开始, 而此周至少有四天在本年中。
VbFirstFullWeek	3	从本年第一周开始, 而此周完全在本年中。

说明

格式化	作法
数字	使用预先定义的命名数值格式或创建用户自定义数值格式。
日期和时间	使用预先定义的命名日期/时间格式或创建用户自定义日期/时间格式。
日期和时间序数	使用日期和时间格式或数值格式。
字符串	创建自定义的字符串格式。

如果在格式化数字时没有指定 *format*, **Format** 会提供与 **Str** 函数类似的功能, 尽管它是国际化的。然而, 以 **Format** 作用在正数上不会保留正负号空间, 而以 **Str** 的话则会。如果要格式化一个没有本地化的数值字符串, 应该使用一个用户自定义的数值格式, 以保证得到需要的外观。

注意 如果 **Calendar** 属性设置是 Gregorian, 并且 *format* 指定了日期格式, 那么, 提供的 *expression* 必须是 Gregorian。如果 Visual Basic **Calendar** 属性设置是 Hijri, 则提供的 *expression* 必须是 Hijri。

如果日历是 Gregorian, 则 *format* 表达式的意义没有改变。如果日历是 Hijri, 则所有的日期格式符号 (例如, *dddd*, *mmmm*, *yyyy*) 有相同的意义, 这些意义只应用于 Hijri 日历。格式符号保持英文, 用于文本显示的符号 (例如, AM 和 PM) 显示与该符号有关的字符串 (英文或阿拉伯数字)。当日历是 Hijri 时, 一些符号的范围会改变。

符号	范围
<i>d</i>	1-30
<i>dd</i>	1-30
<i>ww</i>	1-51
<i>mmm</i>	显示完整的月份名称 (Hijri 月份名称无缩写形式)
<i>y</i>	1-355
<i>yyyy</i>	100-9666

(三十七) FormatCurrency 函数

描述

返回一个货币值格式的表达式，它使用系统控制面板中定义的货币符号。

语法

FormatCurrency (*Expression* [, *NumDigitsAfterDecimal* [, *IncludeLeadingDigit* [, *UseParensForNegativeNumbers* [, *GroupDigits*]]]])

FormatCurrency 函数语法有如下几部分：

部分	描述
<i>Expression</i>	必需的。要格式化的表达式。
<i>NumDigitsAfterDecimal</i>	可选的。数字值，表示小数点右边的显示位数。缺省值为 -1，表示使用计算机的区域设置值。
<i>IncludeLeadingDigit</i>	可选的。三态常数，表示小数点前是否显示一个零。关于其值，请参阅“设置值”部分。
<i>UseParensForNegativeNumbers</i>	可选的。三态常数，表示是否把负数值放在圆括号内。关于其值，请参阅“设置值”部分。
<i>GroupDigits</i>	可选的。三态常数，表示是否用组分隔符对数字进行分组，组分隔符由计算机的区域设置值指定。关于其值，请参阅“设置值”部分。

设置值

IncludeLeadingDigit、*UseParensForNegativeNumbers* 和 *GroupDigits* 参数的设置值如下：

常数	值	描述
vbTrue	-1	True
vbFalse	0	False
vbUseDefault	-2	使用计算机区域设置中的设置值。

说明

当忽略一个或多个选项参数时，被忽略的参数值由计算机的区域设置值提供。

货币符号相对货币值的位置由计算机的区域设置值确定。

注意 除起始的零外，所有设置值信息都来自“区域设置”的“货币”选项卡，起始的零来自“数字”选项卡。

(三十八) FormatDateTime 函数

描述

返回一个日期或时间格式的表达式。

语法

FormatDateTime (*Date* [, *NamedFormat*])

FormatDateTime 函数语法有如下几部分：

部分	描述
<i>Date</i>	必需的。要被格式化的日期表达式。
<i>NamedFormat</i>	可选的。数字值，表示日期/时间所使用的格式。如果忽略该值，则使用

	vbGeneralDate。
--	----------------

设置值

NamedFormat 参数的设置值如下：

常数	值	描述
vbGeneralDate	0	显示日期和/或时间。如果有日期部分，则用短日期格式显示。如果有时间部分，则用长时间格式显示。如果都有，两部分都显示。
vbLongDate	1	用计算机区域设置值中指定的长日期格式显示日期。
vbShortDate	2	用计算机区域设置值中指定的短日期格式显示日期。
vbLongTime	3	用计算机区域设置值中指定的时间格式显示时间。
vbShortTime	4	用 24 小时格式 (hh: mm) 显示时间。

(三十九) FormatNumber 函数

描述

返回一个数字格式的表达式。

语法

FormatNumber (*Expression* [, *NumDigitsAfterDecimal* [, *IncludeLeadingDigit* [, *UseParensForNegativeNumbers* [, *GroupDigits*]]]])

FormatNumber 函数语法有如下几部分：

部分	描述
<i>Expression</i>	必需的。要被格式化的表达式。
<i>NumDigitsAfterDecimal</i>	可选的。数字值，表示小数点右边的显示位数。缺省值为 -1，表示使用计算机的区域设置值。
<i>IncludeLeadingDigit</i>	可选的。三态常数，表示小数点前是否显示零。关于其值，请参阅“设置值”部分。
<i>UseParensForNegativeNumbers</i>	可选的。三态常数，表示是否把负数值放在圆括号内。关于其值，请参阅“设置值”部分。
<i>GroupDigits</i>	可选的。的三态常数，表示是否用组分隔符对数字分组，组分隔符在计算机的区域设置值中指定。关于其值，请参阅“设置值”部分。

设置值

IncludeLeadingDigit、*UseParensForNegativeNumbers* 和 *GroupDigits* 参数的设置值如下：

常数	值	描述
vbTrue	-1	True
vbFalse	0	False
vbUseDefault	-2	用计算机区域设置值中的设置值。

说明

当忽略一个或多个选项参数时，被忽略的参数值由计算机的区域设置值提供。

注意 所有设置值信息都来自“区域设置”的“数字”选项卡。

(四十) FormatPercent 函数

描述

返回一个百分比格式（乘以 100）的表达式，后面有%符号。

语法

FormatPercent (*Expression* [, *NumDigitsAfterDecimal* [, *IncludeLeadingDigit* [, *UseParensForNegativeNumbers* [, *GroupDigits*]]]])

FormatPercent 函数语法有如下几部分：

部分	描述
<i>Expression</i>	必需的。要格式化的表达式。
<i>NumDigitsAfterDecimal</i>	可选的。表示小数点右边的显示位数。缺省值为 - 1，表示使用计算机的区域设置值。
<i>IncludeLeadingDigit</i>	可选的。三态常数，表示小数点前是否显示零。关于其值，请参阅“设置值”部分。
<i>UseParensForNegativeNumbers</i>	可选的。三态常数，表示是否把负数放在圆括号内。关于其值，请参阅“设置值”部分。
<i>GroupDigits</i>	可选的。三态常数，表示是否用组分隔符对数字进行分组，组分隔符在计算机的区域设置值中指定。关于其值，请参阅“设置值”部分。

设置值

IncludeLeadingDigit、*UseParensForNegativeNumbers* 和 *GroupDigits* 参数的设置值如下：

常数	值	描述
vbTrue	- 1	True
vbFalse	0	False
vbUseDefault	- 2	使用计算机区域设置值中的设置值。

说明

当忽略一个或多个选项参数时，被忽略的参数值由计算机的区域设置值提供。

注意 所有的设置值信息都来自“区域设置”的“数字”选项卡。

(四十一) FreeFile 函数

返回一个 Integer，代表下一个可供 **Open** 语句使用的文件号。

语法

FreeFile [(*rangenumber*)]

可选的参数 *rangenumber* 是一个 Variant，它指定一个范围，以便返回该范围内的下一个可用文件号。指定 0（缺省值）则返回一个介于 1 - 255 之间的文件号。指定 1 则返回一个介于 256 - 511 之间的文件号。

说明

使用 **FreeFile** 提供一个尚未使用的文件号。

(四十二) FV 函数

返回一个 Double，指定未来的定期定额支付且利率固定的年金。

语法

FV(*rate*, *nper*, *pmt*[, *pv*[, *type*]])

FV 函数有下列命名参数：

部分	描述
<i>rate</i>	必要。Double，指定每一期的利率。例如，如果有一笔贷款年百分率（APR）为百分之十且按月付款的汽车贷款，则利率为 0.1/12 或 0.0083。
<i>nper</i>	必要。Integer，指定一笔年金的付款总期限。例如，如果对一笔为期四年的汽车贷款选择按月付款方式，则贷款期限共有 4 * 12（或 48）个付款期。
<i>pmt</i>	必要。Double 指定每一期的付款金额。付款金额通常包含本金和利息，而且此付款金额在年金的有效期间是不会改变的。
<i>pv</i>	可选。Variant，指定未来一系列付款（或一次付清款项）的现值。例如，当借钱买一辆汽车时时，向贷方所借的金额为未来每月付款给贷方的现值。如果省略的话，缺省值为 0。
<i>type</i>	可选。Variant，指定贷款到期时间。如果贷款在贷款周期结束时到期，请使用 0。如果贷款在周期开始时到期，请使用 1。如果省略的话，缺省值为 0。

说明

年金是一段时间内一系列固定现金支付。年金可以是贷款（如房屋抵押贷款），也可以是一笔投资（如按月储蓄计划）。

在支付期间，必须用相同的单位来计算 *rate* 和 *nper* 参数。例如，如果 *rate* 用月份来计算，则 *nper* 也必须用月份来计算。

对所有参数，用负数表示现金支出（如储蓄存款），而用正数表示现金收入（如红利支票）。

(四十三) GetAllSettings 函数

从 Windows 注册表 或 (Macintosh 中)应用程序初始化文件中的信息中返回应用程序项目的所有注册表项设置及其相应值（开始是由 SaveSetting 产生）。

语法

GetAllSettings(*appname*, *section*)

GetAllSettings 函数的语法具有下列命名参数：

部分	描述
<i>appname</i>	必要。字符串表达式，包含应用程序或工程的名称，并要求这些应用程序或工程有注册表项设置 在 Macintosh 中，这是 System 文件夹中 Preferences 文件夹中初始化文件的文件名。
<i>section</i>	必要。字符串表达式，包含区域名称，并要求该区域有注册表项设置。 GetAllSettings 返回 Variant，其内容为字符串的二维数组，该二维数组包含指定区域中的所有注册表项设置及其对应值。

说明

如果 *appname* 或 *section* 不存在，则 GetAllSettings 返回未初始化的 Variant。

(四十四) GetAttr 函数

返回一个 Integer，此为一个文件、目录、或文件夹的属性。

语法

GetAttr(*pathname*)

必要的 *pathname* 参数是用来指定一个文件名的字符串表达式。*pathname* 可以包含目录或文件夹、以及驱动器。

返回值

由 GetAttr 返回的值，是下面这些属性值的总和：

常数	值	描述
vbNormal	0	常规
vbReadOnly	1	只读
vbHidden	2	隐藏
vbSystem	4	系统文件 在 Macintosh 中不可用。
vbDirectory	16	目录或文件夹
vbArchive	32	上次备份以后 在 Macintosh 中不可用.，文件已经改变
vbalias	64	指定的文件名是别名。只在 Macintosh 中可用。

注意 这些常数是由 VBA 指定的，在程序代码中的任何位置，可以使用这些常数来替换真正的值。

说明

若要判断是否设置了某个属性，在 GetAttr 函数与想要得知的属性值之间使用 And 运算符与逐位比较。如果所得的结果不为零，则表示设置了这个属性值。例如，在下面的 And 表达式中，如果档案 (Archive) 属性没有设置，则返回值为零：

```
Result = GetAttr(FName) And vbArchive
```

如果文件的档案属性已设置，则返回非零的数值。

(四十五) GetObject 函数

返回文件中的 ActiveX 对象的引用。

语法

GetObject([*pathname*] [, *class*])

GetObject 函数的语法包含下面几个命名参数：

部分	描述
<i>pathname</i>	可选的；Variant (String)。包含待检索对象的文件的全路径和名称。如果省略 <i>pathname</i> ，则 <i>class</i> 是必需的。
<i>class</i>	可选的；Variant (String)。代表该对象的类的字符串。

其中，*class* 参数的语法格式为 *appname.objecttype*，且语法的各个部分如下：

部分	描述
<i>appname</i>	必需的；Variant (String)。提供该对象的应用程序名称。
<i>objecttype</i>	必需的；Variant (String)。待创建对象的类型或类。

说明

使用 **GetObject** 函数可以访问文件中的 ActiveX 对象，而且可以将该对象赋给对象变量。可以使用 **Set** 语句将 **GetObject** 返回的对象赋给对象变量。例如：

```
Dim CADObject As Object
Set CADObject = GetObject("C:\CAD\SCHEMA.CAD")
```

当执行上述代码时，就会启动与指定的 *pathname* 相关联的应用程序，同时激活指定文件中的对象。

如果 *pathname* 是一个零长度的字符串 ("")，则 **GetObject** 返回指定类型的新的对象实例。如果省略了 *pathname* 参数，则 **GetObject** 返回指定类型的当前活动的对象。如果当前没有指定类型的对象，就会出错。

有些应用程序允许只激活文件的一部分，其方法是在文件名后加上一个感叹号 (!) 以及用于标识想要激活的文件部分的字符串。关于如何创建这种字符串的信息，请参阅有关应用程序创建对象的文档。

例如，在绘图应用程序中，一个存放在文件中的图可能有多层。可以使用下述代码来激活图中被称为 SCHEMA.CAD 的层：

```
Set LayerObject = GetObject("C:\CAD\SCHEMA.CAD!Layer3")
```

如果不指定对象的 *class*，则自动化会根据所提供的文件名，来确定被启动的应用程序以及被激活的对象。不过，有些文件可能不止支持一种对象类。例如，图片可能支持三种不同类型的对象：**Application** 对象，**Drawing** 对象，以及 **Toolbar** 对象，所有这些都是同一个文件中的一部分。为了说明要具体激活文件中的哪种对象，就应使用这个可选的 *class* 参数。例如：

```
Dim MyObject As Object
Set MyObject = GetObject("C:\DRAWINGS\SAMPLE.DRW", "FIGMENT.DRAWING")
```

在上述例子中，FIGMENT 是一个绘图应用程序的名称，而 DRAWING 则是它支持的一种对象类型。

对象被激活之后，就可以在代码中使用所定义的对象变量来引用它。在前面的例子中，可以使用对象变量 MyObject 来访问这个新对象的属性和方法。例如：

```
MyObject.Line 9, 90
MyObject.InsertText 9, 100, "Hello, world."
MyObject.SaveAs "C:\DRAWINGS\SAMPLE.DRW"
```

注意 当对象当前已有实例，或要创建已加载的文件的对象时，就使用 **GetObject** 函数。如果对象当前还没有实例，或不想启动已加载文件的对象，则应使用 **CreateObject** 函数。如果对象已注册为单个实例的对象，则不管执行多少次 **CreateObject**，都只能创建该对象的一个实例。若使用单个实例对象，当使用零长度字符串 ("") 语法调用时，**GetObject** 总是返回同一个实例，而若省略 *pathname* 参数，就会出错。不能使用 **GetObject** 来获取 Visual Basic 创建的类的引用。

(四十六) GetSetting 函数

从 Windows 注册表中 或 (Macintosh 中)应用程序初始化文件中的信息的应用程序项目返回注册表项设置值。

语法

```
GetSetting(appname, section, key[, default])
```

GetSetting 函数的语法具有下列命名参数：

部分	描述
----	----

appname	必要。字符串表达式，包含应用程序或工程的名称，要求这些应用程序或工程有注册表项设置。在 Macintosh 中，这是 System 文件夹中 Preferences 文件夹中初始化文件的文件名。
section	必要。字符串表达式，包含区域名称，要求该区域有注册表项设置。
key	必要。字符串表达式，返回注册表项设置的名称。
default	可选。表达式，如果注册表项设置中没有设置值，则返回缺省值。如果省略，则 <i>default</i> 取值为长度为零的字符串 (“”)。

说明

如果 `GetSetting` 的参数中的任何一项都不存在，则 `GetSetting` 返回 *default* 的值。

(四十七) Hex 函数

返回代表十六进制数值的 String。

语法

`Hex(number)`

必要的 *number* 参数为任何有效的数值表达式或字符串表达式。

说明

如果 *number* 还不是一个整数，那么在执行前会先被四舍五入成最接近的整数。

如果 <i>number</i> 为	所得为
Null	Null
Empty	零 (0)
任何其他数字	最多可到八个十六进制字符。

适当范围内的数字，前缀以 &H，可以直接表示十六进制数字。例如，十六进制表示法的 &H10 代表十进制的 16。

(四十八) Hour 函数

返回一个 Variant (Integer)，其值为 0 到 23 之间的整数，表示一天之中的某一钟点。

语法

`Hour(time)`

必要的 *time* 参数，可以是任何能够表示时刻的 Variant、数值表达式、字符串表达式或它们的组合。如果 *time* 包含 Null，则返回 Null。

(四十九) IIf 函数

根据表达式的值，来返回两部分中的其中一个。

语法

`IIf(expr, truepart, falsepart)`

IIf 函数的语法含有下面这些命名参数：

部分	描述
<i>expr</i>	必要参数。用来判断真伪的表达式。

<i>truepart</i>	必要参数。如果 <i>expr</i> 为 True，则返回这部分的值或表达式。
<i>falsepart</i>	必要参数。如果 <i>expr</i> 为 False，则返回这部分的值或表达式。

说明

由于 IIf 会计算 *truepart* 和 *falsepart*，虽然它只返回其中的一个。因此要注意到这个副作用。例如，如果 *falsepart* 产生一个被零除错误，那么程序就会发生错误，即使 *expr* 为 True。

(五十) IMEStatus 函数

返回一个 Integer，用来指定当前 Microsoft Windows 的输入法 (IME) 方式；只对东亚区版本有效。

语法

IMEStatus

返回值

下面是日本区域的返回值：

常数	值	描述
vbIMEModeNoControl	0	不控制 IME (缺省)
vbIMEModeOn	1	打开 IME
vbIMEModeOff	2	关闭 IME
vbIMEModeDisable	3	IME 无效
vbIMEModeHiragana	4	完整宽度 Hiragana 模式
vbIMEModeKatakana	5	完整宽度 Katakana 片假名模式
vbIMEModeKatakanaHalf mode	6	半宽 Katakana 模式
vbIMEModeAlphaFull mode	7	完整宽度 Alphanumeric 模式
vbIMEModeAlpha mode	8	半宽 Alphanumeric 模式

下面是韩国地区的返回值：

常数	值	描述
vbIMEModeAlphaFull	7	完整宽度 Alphanumeric 模式
vbIMEModeAlpha	8	半宽 Alphanumeric 模式
vbIMEModeHangulFull	9	完整宽度 Hangul 模式
vbIMEModeHangul	10	半宽 Hangul 模式

下面是中文地区的返回值：

常数	值	描述
vbIMEModeNoControl	0	不控制 IME (缺省)
vbIMEModeOn	1	打开 IME
vbIMEModeOff	2	关闭 IME

(五十一) Input 函数

返回 String，它包含以 Input 或 Binary 方式打开的文件中的字符。

语法

`Input(number, [#] filename)`

`Input` 函数的语法具有以下几个部分：

部分	描述
<i>number</i>	必要。任何有效的数值表达式，指定要返回的字符个数。
<i>filename</i>	必要。任何有效的文件名。

说明

通常用 `Print #` 或 `Put` 将 `Input` 函数读出的数据写入文件。`Input` 函数只用于以 `Input` 或 `Binary` 方式打开的文件。

与 `Input #` 语句不同，`Input` 函数返回它所读出的所有字符，包括逗号、回车符、空白列、换行符、引号和前导空格等。

对于 `Binary` 访问类型打开的文件，如果试图用 `Input` 函数读出整个文件，则会在 `EOF` 返回 `True` 时产生错误。在用 `Input` 读出二进制文件时，要用 `LOF` 和 `LOC` 函数代替 `EOF` 函数，而在使用 `EOF` 函数时要配合以 `Get` 函数。

注意 对于文本文件中包含的字节数据要使用 `InputB` 函数。对于 `InputB` 来说，*number* 指定的是要返回的字节个数，而不是要返回的字符个数。

(五十二) InputBox 函数

在一对话框中显示提示，等待用户输入正文或按下按钮，并返回包含文本框内容的 `String`。

语法

`InputBox(prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context])`

`InputBox` 函数的语法具有以下几个命名参数：

部分	描述
<i>Prompt</i>	必需的。作为对话框消息出现的字符串表达式。 <i>prompt</i> 的最大长度大约是 1024 个字符，由所用字符的宽度决定。如果 <i>prompt</i> 包含多个行，则可在各行之间用回车符 (<code>Chr(13)</code>)、换行符 (<code>Chr(10)</code>) 或回车换行符的组合 (<code>Chr(13) & Chr(10)</code>) 来分隔。
<i>Title</i>	可选的。显示对话框标题栏中的字符串表达式。如果省略 <i>title</i> ，则把应用程序名放入标题栏中。
<i>Default</i>	可选的。显示文本框中的字符串表达式，在没有其它输入时作为缺省值。如果省略 <i>default</i> ，则文本框为空。
<i>Xpos</i>	可选的。数值表达式，成对出现，指定对话框的左边与屏幕左边的水平距离。如果省略 <i>xpos</i> ，则对话框会在水平方向居中。
<i>Ypos</i>	可选的。数值表达式，成对出现，指定对话框的上边与屏幕上边的距离。如果省略 <i>ypos</i> ，则对话框被放置在屏幕垂直方向距下边大约三分之一的位置。
<i>Helpfile</i>	可选的。字符串表达式，识别帮助文件，用该文件为对话框提供上下文相关的帮助。如果已提供 <i>helpfile</i> ，则也必须提供 <i>context</i> 。
<i>Context</i>	可选的。数值表达式，由帮助文件的作者指定给某个帮助主题的帮助上下文编号。如果已提供 <i>context</i> ，则也必须提供 <i>helpfile</i> 。

说明

如果同时提供了 *helpfile* 与 *context*，用户可以按 `F1` (Windows) 或 `HELP` (Macintosh) 来查看与 *context* 相应的帮助主题。某些主应用程序，例如，Microsoft Excel，会在对话框中自动添加一个 `Help` 按钮。如果用户单击 `OK` 或按下 `ENTER`，则 `InputBox` 函数返回文本框中的内容。如果用户单击 `Cancel`，则此函数返回一个长度为零的字符串 (“”)。

注意 如果还要指定第一个命名参数以外的参数，则必须在表达式中使用 `InputBox`。如果要省略某些位置参数，则必须加入相应的逗号分界符。

(五十三) InStr 函数

返回 `Variant (Long)`，指定一字符串在另一字符串中最先出现的位置。

语法

`InStr([start,]string1, string2[, compare])`

`InStr` 函数的语法具有下面的参数：

部分	说明
<i>start</i>	可选参数。为数值表达式，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 <i>start</i> 包含 <code>Null</code> ，将发生错误。如果指定了 <i>compare</i> 参数，则一定要有 <i>start</i> 参数。
<i>string1</i>	必要参数。接受搜索的字符串表达式。
<i>string2</i>	必要参数。被搜索的字符串表达式。
<i>Compare</i>	可选参数。指定字符串比较。如果 <i>compare</i> 是 <code>Null</code> ，将发生错误。如果省略 <i>compare</i> ， <code>Option Compare</code> 的设置将决定比较的类型。指定一个有效的 LCID (LocaleID) 以在比较中使用与区域有关的规则。

设置

compare 参数设置为：

常数	值	描述
<code>vbUseCompareOption</code>	-1	使用 <code>Option Compare</code> 语句设置执行一个比较。
<code>vbBinaryCompare</code>	0	执行一个二进制比较。
<code>vbTextCompare</code>	1	执行一个按照原文的比较。
<code>vbDatabaseCompare</code>	2	仅适用于 Microsoft Access，执行一个基于数据库中信息的比较。

返回值

如果	InStr 返回
<i>string1</i> 为零长度	0
<i>string1</i> 为 <code>Null</code>	<code>Null</code>
<i>string2</i> 为零长度	<i>Start</i>
<i>string2</i> 为 <code>Null</code>	<code>Null</code>
<i>string2</i> 找不到	0
在 <i>string1</i> 中找到 <i>string2</i>	找到的位置
<i>start</i> > <i>string2</i>	0

说明

`InStrB` 函数作用于包含在字符串中的字节数据。所以 `InStrB` 返回的是字节位置，而不是字符位置。

(五十四) InStrRev 函数

描述

返回一个字符串在另一个字符串中出现的位置，从字符串的末尾算起。

语法

`InStrRev(stringcheck, stringmatch[, start[, compare]])`

InStrRev 函数语法有如下命名参数：

部分	描述
<i>stringcheck</i>	必需的。要执行搜索的字符串表达式。
<i>stringmatch</i>	必需的。要搜索的字符串表达式。
<i>start</i>	可选的。数值表达式，设置每次搜索的开始位置。如果忽略，则使用 -1，它表示从上一个字符位置开始搜索。如果 <i>start</i> 包含 Null，则产生一个错误。
<i>compare</i>	可选的。数字值，指出在判断子字符串时所使用的比较方法。如果忽略，则执行二进制比较。关于其值，请参阅“设置值”部分。

设置值

compare 参数值如下：

常数	值	描述
<code>vbUseCompareOption</code>	-1	用 <code>Option Compare</code> 语句的设置值来执行比较。
<code>vbBinaryCompare</code>	0	执行二进制比较。
<code>vbTextCompare</code>	1	执行文字比较。
<code>vbDatabaseCompare</code>	2	只用于 Microsoft Access。基于您的数据库信息执行比较。

返回值

InStrRev 返回值如下：

如果	InStrRev 返回
<i>stringcheck</i> 长度为零。	0
<i>stringcheck</i> 为 Null。	Null
<i>stringmatch</i> 长度为零	<i>Start</i>
<i>stringmatch</i> 为 Null	Null
<i>stringmatch</i> 没有找到。	0
<i>stringmatch</i> 在 <i>stringcheck</i> 中找到。	找到匹配字符串的位置。
<i>start</i> > Len(<i>stringmatch</i>)	0

说明

请注意，InStrRev 函数的语法和 InStr 函数的语法不相同。

(五十五) IPmt 函数

返回一个 Double，指定在一段时间内对定期定额支付且利率固定的年金所支付的利息值。

语法

IPmt(*rate*, *per*, *nper*, *pv*[, *fv*[, *type*]])

IPmt 函数有下列命名参数:

部分	描述
<i>rate</i>	必要。Double 指定每一期的利率。例如, 如果有一笔贷款年百分率 (APR) 为百分之十且按月付款的汽车贷款, 则每一期的利率为 0.1/12, 或 0.0083。
<i>per</i>	必要。Double 指定在 <i>nper</i> 间范围 1 中的付款周期。
<i>nper</i>	必要。Double 指定一笔年金的付款总期数。例如, 如果在一笔为期四年的汽车贷款中选择按月付款方式, 则贷款共有 4 * 12 (或 48) 个付款期。
<i>pv</i>	必要。Double, 指定未来一系列付款或收款的现值。例如, 当借钱买汽车时, 向贷方所借金额为将来每月偿付给贷方款项的现值。
<i>fv</i>	可选。Variant 指定在付清贷款后所希望的未来值或现金结存。例如, 贷款的未来值在贷款付清后为 0 美元。但是, 如果想要在 18 年间存下 50,000 美元作为子女教育基金, 那么 50,000 美元为未来值。如果省略的话, 缺省值为 0。
<i>type</i>	可选。Variant 指定贷款到期时间。如果贷款在贷款周期结束时到期, 请使用 0。如果贷款在周期开始时到期, 请使用 1。如果省略的话, 缺省值为 0。

说明

年金是指在一段时间内的一系列固定现金支付。年金可以是贷款 (如房屋抵押贷款), 也可以是一笔投资 (如按月储蓄计划)。

在支付期间必须用相同的单位计算 *rate* 和 *nper* 参数。例如, 如果 *rate* 用月份计算, 则 *nper* 也必须用月份计算。

对所有参数, 用负数表示现金支出 (如储蓄存款), 而用正数表示现金收入 (如红利支票)。

(五十六) IRR 函数

返回一个 Double, 指定一系列周期性现金流 (支出或收入) 的内部利率。

语法

IRR(*values*()[, *guess*])

IRR 函数有下列命名参数:

部分	描述
<i>values</i> ()	必要。Double 数组, 指定现金流值。此数组必须至少含有一个负值 (支付) 和一个正值 (收入)。
<i>Guess</i>	可选。Variant, 指定 IRR 返回的估算值。如果省略, <i>guess</i> 为 0.1 (10%)。

说明

返回的内部利率是在正常的时间间隔内, 一笔含有支出及收入的投资得到的利率。

IRR 函数使用数组中数值的顺序来解释支付和收入的顺序。要确保支付和收入的顺序正确。

每一时期的现金流不必像年金那样固定不变。

IRR 是利用叠代进行计算。先从 *guess* 的值开始, IRR 反复循环进行计算, 直到精确度达到 0.00001%。如果经过 20 次反复叠代测试还不能得到结果, 则 IRR 计算失败。

(五十七) IsArray 函数

返回 Boolean 值, 指出变量是否为一个数组。

语法

IsArray(*varname*)

必要的 *varname* 参数是一个指定变量的标识符。

说明

如果变量是数组，则 **IsArray** 返回 **True**；否则返回 **False**。对于包含数组的 *variant* 表达式来说，**IsArray** 尤为有用。

(五十八) IsDate 函数

返回 **Boolean** 值，指出一个表达式是否可以转换成日期。

语法

IsDate(*expression*)

必要的 *expression* 参数是一个 *Variant*，包含日期表达式或字符串表达式，这里的字符串表达式是可以作为日期或时间来认定的。

说明

如果表达式是一个日期，或可以作为有效日期识别，则 **IsDate** 返回 **True**；否则返回 **False**。在 Microsoft Windows 中，有效日期的范围介于公元 100 年 1 月 1 日与公元 9999 年 12 月 31 日之间；其有效范围随操作系统不同而不同。

(五十九) IsEmpty 函数

返回 **Boolean** 值，指出变量是否已经初始化。

语法

IsEmpty(*expression*)

必要的 *expression* 参数是一个 *Variant*，包含一个数值或字符串表达式。但是，因为 **IsEmpty** 被用来确定个别变量是否已初始化，所以 *expression* 参数通常是单一变量名。

说明

如果变量未初始化或已明确设置为 *Empty*，则 **IsEmpty** 返回 **True**；否则返回 **False**。如果 *expression* 含有多个变量，则 **IsEmpty** 总是返回 **False**。**IsEmpty** 只返回对 *variant* 表达式有意义的信息。

(六十) IsError 函数

返回 **Boolean** 值，指出表达式是否为一个错误值。

语法

IsError(*expression*)

必需的 *expression* 参数，可以是任何有效表达式。

说明

利用 **CVErr** 函数将实数转换成错误值就会建立错误值。**IsError** 函数被用来确定一个数值表达式是否表示一个错误。如果 *expression* 参数表示一个错误，则 **IsError** 返回 **True**；否则返回 **False**。

(六十一) 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。

(六十二) IsNull 函数

返回 **Boolean** 值，指出表达式是否不包含任何有效数据 (**Null**)。

语法

IsNull(*expression*)

必要的 *expression* 参数是一个 **Variant**，其中包含数值表达式或字符串表达式。

说明

如果 *expression* 为 **Null**，则 **IsNull** 返回 **True**；否则 **IsNull** 返回 **False**。如果 *expression* 由多个变量组成，则表达式的任何作为变量组成成分的 **Null** 都会使整个表达式返回 **True**。

Null 值指出 **Variant** 不包含有效数据。**Null** 与 **Empty** 不同，后者指出变量尚未初始化。

Null 与长度为零的字符串（“ ”）也不同，长度为零的字符串指的是空串。

重要 使用 **IsNull** 函数是为了确定表达式是否包含 **Null** 值的。在某些情况下，希望表达式取值为 **True**，比如希望 **If Var = Null** 和 **If Var <> Null** 取值为 **True**，而它们总取值为 **False**。这是因为任何包含 **Null** 的表达式本身就是 **Null**，所以为 **False**。

(六十三) IsNumeric 函数

返回 Boolean 值，指出表达式的运算结果是否为数。

语法

`IsNumeric(expression)`

必要的 *expression* 参数是一个 Variant，包含数值表达式或字符串表达式。

说明

如果整个 *expression* 的运算结果为数字，则 `IsNumeric` 返回 `True`；否则返回 `False`。

如果 *expression* 是日期表达式，则 `IsNumeric` 返回 `False`。

(六十四) IsObject 函数

返回 Boolean 值，指出标识符是否表示对象变量。

语法

`IsObject(identifier)`

必要的 *identifier* 参数是一个变量名。

说明

`IsObject` 只用于确定 Variant 是否属于 `VarType vbObject`。如果 Variant 实际引用（或曾经引用过）一个对象，或者如果 Variant 包含 `Nothing`，则可能出现这种情况。

如果 *identifier* 是 Object 类型或任何有效的类类型，或者，如果 *identifier* 是 `VarType vbObject` 的 Variant 或用户自定义的对象，则 `IsObject` 返回 `True`；否则返回 `False`。即使变量已设置成 `Nothing`，`IsObject` 也仍返回 `True`。

使用错误捕获方法可以确认对象引用是否有效。

(六十五) Join 函数

描述

返回一个字符串，该字符串是通过连接某个数组中的多个子字符串而创建的。

语法

`Join(sourcearray[, delimiter])`

`Join` 函数语法有如下命名参数：

部分	描述
<i>sourcearray</i>	必需的。包含被连接子字符串的一维数组。
<i>delimiter</i>	可选的。在返回字符串中用于分隔子字符串的字符。如果忽略该项，则使用空格(" ")来分隔子字符串。如果 <i>delimiter</i> 是零长度字符串(""), 则列表中的所有项目都连接在一起，中间没有分隔符。

(六十六) LBound 函数

返回一个 Long 型数据，其值为指定数组维可用的最小下标。

语法

LBound(*arrayname*[, *dimension*])

LBound 函数的语法包含下面部分：

部分	描述
<i>arrayname</i>	必需的。数组变量的名称，遵循标准的变量命名约定。
<i>dimension</i>	可选的； Variant (Long) 。指定返回哪一维的下界。1 表示第一维，2 表示第二维，如此类推。如果省略 <i>dimension</i> ，就认为是 1。

说明

LBound 函数与 **UBound** 函数一起使用，用来确定一个数组的大小。**UBound** 用来确定数组某一维的上界。

对具有下述维数的数组而言，**LBound** 的返回值见下表：

Dim A(1 To 100, 0 To 3, -3 To 4)

语句	返回值
LBound(A, 1)	1
LBound(A, 2)	0
LBound(A, 3)	-3

所有维的缺省下界都是 0 或 1，这取决于 **Option Base** 语句的设置。使用 **Array** 函数创建的数组的下界为 0；它不受 **Option Base** 的影响。

对于那些在 **Dim** 中用 **To** 子句来设定维数的数组而言，**Private**、**Public**、**ReDim** 或 **Static** 语句可以用任何整数作为下界。

(六十七) LCase 函数

返回转成小写的 String。

语法

LCase(*string*)

必要的 *string* 参数可以是任何有效的字符串表达式。如果 *string* 包含 Null，将返回 Null。

说明

只有大写的字母会转成小写；所有小写字母和非字母字符保持不变。

(六十八) Left 函数

返回 **Variant (String)**，其中包含字符串中从左边算起指定数量的字符。

语法

`Left(string, length)`

`Left` 函数的语法有下面的命名参数:

部分	说明
<i>string</i>	必要参数。字符串表达式其中最左边的那些字符将被返回。如果 <i>string</i> 包含 Null, 将返回 Null。
<i>length</i>	必要参数; 为 Variant (Long) 。数值表达式, 指出将返回多少个字符。如果为 0, 返回零长度字符串 ("")。如果大于或等于 <i>string</i> 的字符数, 则返回整个字符串。

说明

欲知 *string* 的字符数, 使用 `Len` 函数。

注意 `LeftB` 函数作用于包含在字符串中的字节数据。所以 *length* 指定的是字节数, 而不是要返回的字符数。

(六十九) Len 函数

返回 Long, 其中包含字符串内字符的数目, 或是存储一变量所需的字节数。

语法

`Len(string | varname)`

`Len` 函数的语法有下面这些部分:

部分	说明
<i>string</i>	任何有效的字符串表达式。如果 <i>string</i> 包含 Null, 会返回 Null。
<i>Varname</i>	任何有效的变量名称。如果 <i>varname</i> 包含 Null, 会返回 Null。如果 <i>varname</i> 是 Variant, <code>Len</code> 会视其为 String 并且总是返回其包含的字符数。

说明

两个可能的参数必须有其一 (而且只能有其一)。如为用户定义类型, `Len` 会返回其写至文件的大小。

注意 `LenB` 函数作用于字符串中的字节数据, 如同在双字节字符集 (DBCS) 语言中一样。所以 `LenB` 返回的是用于代表字符串的字节数, 而不是返回字符串中字符的数量。如为用户自定义类型, `LenB` 返回在内存中的大小, 包括元素之间的衬垫。对于使用 `LenB` 的示例代码, 请参阅示例主题中的第二个示例。

注意 当在用户自定义数据类型中使用变长字符串时, `Len` 可能不能确定实际存储所需的字节数目。

(七十) Loc 函数

返回一个 Long, 在已打开的文件中指定当前读/写位置。

语法

`Loc(filenum)`

必要的 *filenum* 参数是任何一个有效的 Integer 文件号。

说明

`Loc` 函数对各种文件访问方式的返回值如下:

方式	返回值
Random	上一次对文件进行读出或写入的记录号。

Sequential	文件中当前字节位置除以 128 的值。但是，对于顺序文件而言，不会使用 Loc 的返回值，也不需要使用 Loc 的返回值。
Binary	上一次读出或写入的字节位置。

(七十一) LOF 函数

返回一个 Long，表示用 **Open** 语句打开的文件的大小，该大小以字节为单位。

语法

LOF(*filenumber*)

必要的 *filenumber* 参数是一个 Integer，包含一个有效的文件号。

注意 对于尚未打开的文件，使用 **FileLen** 函数将得到其长度。

(七十二) Log 函数

返回一个 Double，指定参数的自然对数值。

语法

Log(*number*)

必要的 *number* 参数是 Double 或任何有效的大于 0 的数值表达式。

说明

自然对数是以 *e* 为底的对数。常数 *e* 的值大约是 2.718282。

如下所示，将 *x* 的自然对数值除以 *n* 的自然对数值，就可以对任意底 *n* 来计算数值 *x* 的对数值：

$$\text{Log}_n(x) = \text{Log}(x) / \text{Log}(n)$$

下面的示例说明如何编写一个函数来求以 10 为底的对数值：

```
Static Function Log10(X)
    Log10 = Log(X) / Log(10#)
End Function
```

(七十三) LTrim、RTrim 与 Trim 函数

返回 Variant (String)，其中包含指定字符串的拷贝，没有前导空白 (**LTrim**)、尾随空白 (**RTrim**) 或前导和尾随空白 (**Trim**)。

语法

LTrim(*string*)

RTrim(*string*)

Trim(*string*)

必要的 *string* 参数可以是任何有效的字符串表达式。如果 *string* 包含 Null，将返回 Null。

(七十四) MacID 函数

此函数用在 Macintosh 上, 将长为 4 个字符的常量 转换成被 Dir, Kill, Shell, 和 AppActivate. 使用的值。

语法

MacID(常量)

所需的常量参数包含 4 个字符, 用来说明一个资源类型、文件类型、数字签名或 Apple Event, 例如: “TEXT”、 “OBIN”, Excel 文件用 “XLS5” 说明(Excel 97 用 “XLS8” 说明), Microsoft Word 用 “W6BN” 说明(Word 97 用 “W8BN” 说明), 如此等等。

注释

MacID 与 Dir 和 Kill 一起被用来说明一个 Macintosh 的文件类型。尽管 Macintosh 不支持 * 和 ? 作为通配符, 您仍可以用 4 个字符常量替代说明一组文件。例如, 下面的指令从当前的文件夹中返回 TEXT 类型的文件:

```
Dir("SomePath", MacID("TEXT"))
```

MacID 与 Shell 和 AppActivate 一起被用来说明一个应用使用此应用的唯一签名。

(七十五) MacScript 函数

执行一个脚本并返回由此脚本返回的值, 如果脚本有返回值的话。

语法

MacScript 脚本

参数脚本是一个字符串表达式。此字符串表达式既可以是一系列 AppleScript 的命令, 也可以说明成 AppleScript 脚本或一个脚本文件的名称。

Remarks

多行脚本可由嵌入回车字符 (Chr (13)) 生成。

(七十六) Mid 函数

返回 Variant (String), 其中包含字符串中指定数量的字符。

语法

```
Mid(string, start[, length])
```

Mid 函数的语法具有下面的命名参数:

部分	说明
<i>string</i>	必要参数。字符串表达式, 从中返回字符。如果 <i>string</i> 包含 Null, 将返回 Null。
<i>start</i>	必要参数。为 Long。 <i>string</i> 中被取出部分的字符位置。如果 <i>start</i> 超过 <i>string</i> 的字符数, Mid 返回零长度字符串 (“”)。
<i>length</i>	可选参数; 为 Variant (Long)。要返回的字符数。如果省略或 <i>length</i> 超过文本的字符数 (包括 <i>start</i> 处的字符), 将返回字符串中从 <i>start</i> 到尾端的所有字符。

说明

欲知 *string* 的字符数, 可用 Len 函数。

注意 MidB 函数作用于字符串中包含的字节数据，如同在双字节字符集(DBCS)语言中一样。因此其参数指定的是字节数，而不是字符数。对于使用 MidB 的示例代码，请参阅示例主题中的第二个示例。

(七十七) Minute 函数

返回一个 Variant (Integer)，其值为 0 到 59 之间的整数，表示一小时中的某分钟。

语法

Minute(*time*)

必要的 *time* 参数，可以是任何能够表示时刻的 Variant、数值表达式、字符串表达式或它们的组合。如果 *time* 包含 Null，则返回 Null。

(七十八) MIRR 函数

返回一个 Double，指定一系列修改过的周期性现金流（支出或收入）的内部利率。

语法

MIRR(*values()*, *finance_rate*, *reinvest_rate*)

MIRR 函数有下列命名参数：

部分	描述
<i>values()</i>	必要。Double 数组，指定现金流值。此数组至少要包含一个负值（支付）和一个正值（收入）。
<i>finance_rate</i>	必要。Double 指定财务成本上的支付利率。
<i>reinvest_rate</i>	必要。Double 指定由现金再投资所得利率。

说明

修改过的返回内部利率是指在用不同的利率计算支出和收入时的内部利率。MIRR 函数既考虑投资成本 (*finance_rate*)，也考虑现金再投资所得利率 (*reinvest_rate*)。

finance_rate 和 *reinvest_rate* 参数是用十进制数值表示的百分比。例如，0.12 表示百分之十二。

MIRR 函数用数组中的数值顺序来解释支付和收入的顺序。要确保支付和收入的输入顺序正确。

(七十九) Month 函数

返回一个 Variant (Integer)，其值为 1 到 12 之间的整数，表示一年中的某月。

语法

Month(*date*)

必要的 *date* 参数，可以是任何能够表示日期的 Variant、数值表达式、字符串表达式或它们的组合。如果 *date* 包含 Null，则返回 Null。

(八十) MonthName 函数

描述

返回一个表示指定月份的字符串。

语法

MonthName(*month*[, *abbreviate*])

MonthName 函数语法有如下几部分：

部分	描述
<i>month</i>	必需的。月份的数值表示。例如一月是 1，二月是 2，等等。
<i>abbreviate</i>	可选的。Boolean 值，表示月份名是否缩写。如果忽略，缺省值为 False，表明月份名不能被缩写。

(八十一) MsgBox 函数

在对话框中显示消息，等待用户单击按钮，并返回一个 Integer 告诉用户单击哪一个按钮。

语法

MsgBox(*prompt*[, *buttons*] [, *title*] [, *helpfile*, *context*])

MsgBox 函数的语法具有以下几个命名参数：

部分	描述
<i>Prompt</i>	必需的。字符串表达式，作为显示在对话框中的消息。 <i>prompt</i> 的最大长度大约为 1024 个字符，由所用字符的宽度决定。如果 <i>prompt</i> 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)) 将各行分隔开来。
<i>Buttons</i>	可选的。数值表达式是值的总和，指定显示按钮的数目及形式，使用的图标样式，缺省按钮是什么以及消息框的强制回应等。如果省略，则 <i>buttons</i> 的缺省值为 0。
<i>Title</i>	可选的。在对话框标题栏中显示的字符串表达式。如果省略 <i>title</i> ，则将应用程序名放在标题栏中。
<i>Helpfile</i>	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 <i>helpfile</i> ，则也必须提供 <i>context</i> 。
<i>Context</i>	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 <i>context</i> ，则也必须提供 <i>helpfile</i> 。

设置值

buttons 参数有下列设置值：

常数	值	描述
vbOKOnly	0	只显示 OK 按钮。
VbOKCancel	1	显示 OK 及 Cancel 按钮。
VbAbortRetryIgnore	2	显示 Abort、Retry 及 Ignore 按钮。
VbYesNoCancel	3	显示 Yes、No 及 Cancel 按钮。
VbYesNo	4	显示 Yes 及 No 按钮。
VbRetryCancel	5	显示 Retry 及 Cancel 按钮。

VbCritical	16	显示 Critical Message 图标。
VbQuestion	32	显示 Warning Query 图标。
VbExclamation	48	显示 Warning Message 图标。
VbInformation	64	显示 Information Message 图标。
vbDefaultButton 1	0	第一个按钮是缺省值。
vbDefaultButton 2	256	第二个按钮是缺省值。
vbDefaultButton 3	512	第三个按钮是缺省值。
vbDefaultButton 4	768	第四个按钮是缺省值。

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。
vbMsgBoxHelpButton	16384	将 Help 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	指定文本应在希伯来和阿拉伯语系统中的从右到左显示

第一组值 (0 - 5) 描述了对话框中显示的按钮的类型与数目；第二组值 (16, 32, 48, 64) 描述了图标的样式；第三组值 (0, 256, 512) 说明哪一个按钮是缺省值；而第四组值 (0, 4096) 则决定消息框的强制返回性。将这些数字相加以生成 *buttons* 参数值的时候，只能由每组值取用一个数字。

注意 这些常数都是 Visual Basic for Applications (VBA) 指定的。结果，可以在程序代码中到处使用这些常数名称，而不必使用实际数值。

返回值

常数	值	描述
vbOK	1	OK
vbCancel	2	Cancel
vbAbort	3	Abort
vbRetry	4	Retry
vbIgnore	5	Ignore
vbYes	6	Yes
vbNo	7	No

说明

在提供了 *helpfile* 与 *context* 的时候，用户可以按 F1 (Windows) 或 HELP (Macintosh) 来查看与 *context* 相应的帮助主题。像 Microsoft Excel 这样一些主应用程序也会在对话框中自动添加一个 Help 按钮。

如果对话框显示 **Cancel** 按钮，则按下 ESC 键与单击 **Cancel** 按钮的效果相同。如果对话框中有 **Help** 按钮，则对话框中提供有上下文相关的帮助。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。

注意 如果还要指定第一个命名参数以外的参数，则必须在表达式中使用 **MsgBox**。为了省略某些位置参数，必须加入相应的逗号分界符。

(八十二) Now 函数

返回一个 **Variant (Date)**，根据计算机系统设置的日期和时间来指定日期和时间。

语法

Now

(八十三) NPer 函数

返回一个 **Double**，指定定期定额支付且利率固定的总期数。

语法

NPer(*rate*, *pmt*, *pvt*[, *fv*[, *type*]])

NPer 函数有下列命名参数：

部分	描述
<i>rate</i>	必要。 Double 指定每一期的利率。例如，如果有一笔贷款年百分率 (APR) 为百分之十并按月付款的汽车贷款，则每一期的利率为 0.1/12 或 0.0083。
<i>pmt</i>	必要。 Double 指定每一期所付金额。付款金额通常包含本金和利息，且付款金额在年金的有效期间不变。
<i>pvt</i>	必要。 Double 指定未来一系列付款或收款的现值。例如，当贷款买一辆汽车时，向贷方所借贷的金额为将来每月偿付给贷方款项的现值。
<i>fv</i>	可选。 Variant 指定在付清贷款后所希望的未来值或现金结存。例如，贷款的未来值在贷款付清后为 0 美元。但是，如果想要在 18 年间存下 50,000 美元作为子女教育基金，那么 50,000 美元为未来值。如果省略的话，缺省值为 0。
<i>type</i>	可选。 Variant 指定贷款到期时间。如果贷款是在贷款周期结束时到期，请使用 0，如果贷款是在周期开始时到期，请使用 1。如果省略的话，缺省值为 0。

说明

年金是在一段时间内一系列固定现金支付。年金可以是贷款（如房屋抵押贷款），也可以是一笔投资（如按月储蓄计划）。

对所有参数，用负数表示现金支出（如储蓄存款），而用正数表示现金收入（如红利支票）。

(八十四) NPV 函数

返回一个 **Double**，指定根据一系列定期的现金流（支付和收入）和贴现率而定的投资净现值。

语法

NPV(*rate*, *values*())

NPV 函数有下列命名参数：

部分	描述
<i>rate</i>	必要。 Double 指定在一期间的贴现率，用十进制表示。
<i>values()</i>	必要。 Double 数组 指定现金流值。此数组至少要包含一个负值（支付）和一个正值（收入）。

说明

投资的净现值是未来一系列支付或收入的当前价值。

NPV 函数使用数组中数值的顺序来解释支付和收入的顺序。要确保支付和收入值是用正确的顺序输入的。

NPV 投资在第一笔现金流值之前开始计算周期，而结束于数组中最后的现金流值。

净现值是根据未来的现金流进行计算的。如果第一笔现金流在第一期开始时发生，那么 **NPV** 返回的值必须加上第一笔值才是净现值。而且 *values()* 数组不可包含第一笔值。

NPV 函数与 **PV** 函数（现值）相似，只是 **PV** 函数在一个期间的开始或结束时才允许有现金流。与可变的 **NPV** 现金流值不同，**PV** 的现金流在整个投资期间必须固定。

(八十五) Oct 函数

返回 **Variant (String)**，代表一数值的八进制值。

语法

Oct(*number*)

必要的 *number* 参数为任何有效的数值表达式或字符串表达式。

说明

如果 *number* 尚非整数，那么在执行前会先四舍五入成最接近的整数。

如果 <i>number</i> 为	Oct 返回
Null	Null
Empty	零 (0)
任何其他数字	最多可到 11 个八进制字符。

可以将适当范围的数前缀以 **&0** 来直接表示八进制数字。例如，八进制表示法的 **&010** 代表十进制的 8。

(八十六) Partition 函数

返回一个 **Variant (String)**，指定一个范围，在一系列计算的范围中指定的数字出现在这个范围内。

语法

Partition(*number, start, stop, interval*)

Partition 函数的语法含有下面这些命名参数：

部分	描述
<i>number</i>	必要参数。整数，在所有范围中判断这个整数是否出现。
<i>start</i>	必要参数。整数，数值范围的开始值，这个数值不能小于 0。
<i>stop</i>	必要参数。整数，数值范围的结束值，这个数值不能等于或小于

<i>start</i> 。

说明

Partition 函数会标识 *number* 值出现的特定范围，并返回一个 **Variant (String)** 来描述这个范围。**Partition** 函数在查询中是最有用的。可以创建一个选择查询显示有多少定单落在几个变化的范围内，例如，定单数从 1 到 1000、1001 到 2000，以此类推。

下面的表格使用三组 *start*, *stop* 以及 *interval* 部分，来显示怎样决定这个范围。第一个范围和最后一个范围两列显示 **Partition** 的返回值，此范围的低端 (*lowvalue*) 和高端 (*uppervalue*) 是以冒号分开的。

<i>start</i>	<i>stop</i>	<i>interval</i>	第一个范围之前	第一个范围	最后一个范围	最后一个范围之后
0	99	5	" :-1"	" 0: 4"	" 95: 99"	" 100: "
20	199	10	" : 19"	" 20: 29"	" 190: 199"	" 200: "
100	1010	20	" : 99"	" 100: 119"	" 1000: 1010"	" 1011: "

从上面的表格中得知，在第三行中，由 *start* 和 *stop* 所定义的数值范围不能以 *interval* 来均分。所以，即使 *interval* 是 20，最后一个范围也只能扩展到 *stop* (11 个数)。

如果需要的话，**Partition** 会在返回的范围中加上足够的空白，以便让返回值在冒号的左右两侧有相同的字符数，其值就是 *stop* 中的字符数再加一。如此可确保当要使用 **Partition** 与其它数值作运算时，所得的字符串，可以在之后的排序操作中得到正确的结果。

如果 *interval* 是 1，则范围便是 *number:number*，而不管 *start* 和 *stop* 参数如何。比如说，如果 *interval* 是 1，*number* 是 100，而 *stop* 是 1000，则 **Partition** 会返回 "100:100"。

如果任何部分是 Null，则 **Partition** 会返回一个 Null。

(八十七) Pmt 函数

返回一个 Double，指定根据定期定额支付且利率固定的年金支付额。

语法

Pmt(*rate*, *nper*, *pv*[, *fv*[, *type*]])

Pmt 函数有下列命名参数：

部分	描述
<i>rate</i>	必要。Double 指定每一期的利率。例如，如果有一笔贷款年百分比率 (APR) 为百分之十且按月付款的汽车贷款，则每一期的利率为 0.1/12 或 0.0083。
<i>nper</i>	必要。Integer 指定一笔年金的付款总期数。例如，如果对一笔为期四年的汽车贷款选择按月付款，则贷款共有 4 * 12 (或 48) 个付款期。
<i>pv</i>	必要。Double 指定未来一系列付款或收款的现值。例如，当贷款买一辆汽车时，向贷方所借贷的金额为将来每月偿付给贷方款项的现值。
<i>fv</i>	可选。Variant 指定在付清贷款后所希望的未来值或现金结存。例如，贷款的未来值在贷款付清后为 0 美元。但是，如果想要在 18 年间存下 50,000 美元作为子女教育基金，那么 50,000 美元为未来值。如果省略的话，缺省值为 0。
<i>type</i>	可选。Variant，指定贷款到期时间。如果贷款是在贷款周期结束时到期，请使用 0。如果贷款是在周期开始时到期，则请使用 1。如果省略的话，缺省值为 0。

说明

年金是在一段时间内一系列固定现金支付，年金可以是贷款（如房屋抵押贷款），也可以是一笔投资（如按月储蓄计划）。

在支付期间必须用相同的单位计算 *rate* 和 *nper* 参数。例如，如果 *rate* 用月份计算，则 *nper* 也必须用月份计算。

对所有参数，用负数表示现金支出（如储蓄存款），而用正数表示现金收入（如红利支票）。

(八十八) PPmt 函数

返回一个 Double，指定在定期定额支付且利率固定的年金的指定期间内的本金偿付额。

语法

PPmt(*rate*, *per*, *nper*, *pv*[, *fv*[, *type*]])

PPmt 函数有下列命名参数：

部分	描述
<i>rate</i>	必要。Double 指定每一期的利率。例如，如果有一笔贷款年百分比率（APR）为百分之十且按月付款的汽车贷款，则每一期的利率为 0.1/12 或 0.0083。
<i>per</i>	必要。Integer 指定在 <i>nper</i> 间范围 1 中的付款周期。
<i>nper</i>	必要。Integer 指定一笔年金的付款总期数。例如，如果对一笔为期四年的汽车贷款选择按月付款，则贷款共有 4 * 12（或 48）个付款期。
<i>pv</i>	必要。Double 指定未来一系列付款或收款的现值。例如，当贷款买一辆汽车时，向贷方所借贷的金额为将来每月偿付给贷方款项的现值。
<i>fv</i>	可选。Variant 指定在付清贷款后所希望的未来值或现金结存值。例如，贷款的未未来值在贷款付清后为 0 美元。但是，如果想要在 18 年间存下 50,000 美元作为子女教育基金，那么 50,000 美元为未来值。如果省略的话，缺省值为 0。
<i>type</i>	可选。Variant 指定贷款到期时间。如果贷款是在贷款周期结束时到期，则请使用 0。如果贷款是在周期开始时到期，则请使用 1。如果省略的话，缺省值为 0。

说明

年金是在一段时间内一系列固定现金支付。年金可以是贷款（如房屋抵押贷款），也可以是一笔投资（如按月储蓄计划）。

在支付期间必须用相同的单位计算 *rate* 和 *nper* 参数。例如，如果 *rate* 用月份计算，则 *nper* 也必须用月份计算。

对所有参数，用负数表示现金支出（如储蓄存款），而用正数表示现金收入（如红利支票）。

(八十九) PV 函数

返回一个 Double 指定在未来定期、定额支付且利率固定的年金现值。

语法

PV(*rate*, *nper*, *pmt*[, *fv*[, *type*]])

PV 函数有下列命名参数：

部分	描述
<i>rate</i>	必要。Double 指定每一期的利率。例如，如果有一笔贷款年百分比率（APR）为百分之十且按月付款的汽车贷款，则每一期的利率为 0.1/12 或 0.0083。
<i>nper</i>	必要。Integer 指定一笔年金的付款总期数。例如，如果对一笔为期四年的汽车贷款选择按月付款，则贷款共有 4 * 12（或 48）个付款期。
<i>pmt</i>	必要。Double 指定每一期的付款金额。付款金额通常包含本金和利息，且此付款金

	额在年金的有效期间不变。
<i>fv</i>	可选。Variant，指定在付清贷款后所希望的未来值或现金结存。例如，贷款的未来值在贷款付清后为 0 美元。但是，如果想要在 18 年间存下 50,000 美元作为子女教育基金，那么 50,000 美元为未来值。如果省略的话，缺省值为 0。
<i>type</i>	可选。Variant 指定贷款到期时间。如果贷款是在贷款周期结束时到期，则请使用 0。如果贷款是在周期开始时到期，则请使用 1。如果省略的话，缺省值为 0。

说明

年金是在一段时间内一系列固定现金支付。年金可以是贷款（如房屋抵押贷款），也可以是一笔投资（如按月储蓄计划）。

在支付期间必须用相同的单位计算 *rate* 和 *nper* 参数。例如，如果 *rate* 用月份计算，则 *nper* 也必须用月份计算。

对所有参数，现金支出（如储蓄存款）用负数表示，而现金收入（如红利支票）用正数表示。

(九十) QBColor 函数

返回一个 Long，用来表示所对应颜色值的 RGB 颜色码。

语法

QBColor(*color*)

必要的 *color* 参数是一个介于 0 到 15 的整型。

设置值

color 参数有以下这些设置：

值	颜色	值	颜色
0	黑色	8	灰色
1	兰色	9	亮兰色
2	绿色	10	亮绿色
3	青色	11	亮青色
4	红色	12	亮红色
5	洋红色	13	亮洋红色
6	黄色	14	亮黄色
7	白色	15	亮白色

说明

color 参数代表使用于早期版本的 Basic（诸如 Microsoft Visual Basic for MS-DOS 以及 Basic Compiler）的颜色值。始于最低有效字节，返回值指定了红、绿、蓝三原色的值，用于设置成 VBA 中 RGB 系统的对应颜色。

(九十一) Rate 函数

返回一个 Double，指定每一期的年金利率。

语法

Rate(*nper*, *pmt*, *pv*[, *fv*[, *type*[, *guess*]]])

Rate 函数有下列命名参数：

部分	描述
<i>nper</i>	必要。 Double 指定一笔年金的付款总期数。例如，如果对一笔为期四年的汽车贷款选择按月付款，则贷款共有 4 * 12（或 48）个付款期。
<i>pmt</i>	必要。 Double ，指定每一期的付款金额。付款金额通常包含本金和利息，且此付款金额在年金的有效期间不变。
<i>pv</i>	必要。 Double 指定未来一系列付款或收款的现值。例如，当贷款买一辆汽车时，向贷方所借贷的金额为将来每月偿付给贷方款项的现值。
<i>fv</i>	可选。 Variant 指定在付清贷款后所希望的未来值或现金结存。例如，贷款的未来值在贷款付清后为 0 美元。但是，如果想要在 18 年间存下 50,000 美元作为子女教育基金，那么 50,000 美元为未来值。如果省略的话，缺省值为 0。
<i>type</i>	可选。 Variant ，指定贷款到期时间，如果贷款是在贷款周期结束时到期，则请使用 0。如果贷款是在周期开始时到期，则请使用 1。如果省略的话，缺省值为 0。
<i>guess</i>	可选。 Variant 指定 Rate 返回的估算值。如果省略，则 <i>guess</i> 为 0.1 (10%)。

说明

年金是在一段时间内的一系列固定现金支付，年金可以是贷款（如房屋抵押贷款）或是一笔投资（如按月储蓄计划）。

对所有参数，现金支出（如储蓄存款）用负数表示，而现金收入（如红利支票）用正数表示。**Rate** 是叠代计算的。先从 *guess* 的值开始，**Rate** 反复循环计算，直到精确度达到 0.00001%。如果经过 20 次叠代测试还不能得到结果，则 **Rate** 计算失败。如果猜测是 10% 而 **Rate** 计算失败，则请试用不同的 *guess* 值。

(九十二) Replace 函数

描述

返回一个字符串，该字符串中指定的子字符串已被替换成另一子字符串，并且替换发生的次数也是指定的。

语法

Replace(*expression*, *find*, *replace*[, *start*[, *count*[, *compare*]])

Replace 函数语法有如下命名参数：

部分	描述
<i>expression</i>	必需的。字符串表达式，包含要替换的子字符串。
<i>find</i>	必需的。要搜索到的子字符串。
<i>replace</i>	必需的。用来替换的子字符串。
<i>start</i>	可选的。在表达式中子字符串搜索的开始位置。如果忽略，假定从 1 开始。
<i>count</i>	可选的。子字符串进行替换的次数。如果忽略，缺省值是 -1，它表明进行所有可能的替换。
<i>compare</i>	可选的。数字值，表示判别子字符串时所用的比较方式。关于其值，请参阅“设置值”部分。

设置值

compare 参数的设置值如下：

常数	值	描述
<i>vbUseCompareOption</i>	- 1	使用 Option Compare 语句的设置值来执行比较。

vbBinaryCompare	0	执行二进制比较。
vbTextCompare	1	执行文字比较。
vbDatabaseCompare	2	仅用于 Microsoft Access。基于您的数据库的信息执行比较。

返回值

Replace 的返回值如下：

如果	Replace 返回值
<i>expression</i> 长度为零	零长度字符串(“”)。
<i>expression</i> 为 Null	一个错误。
<i>find</i> 长度为零	<i>expression</i> 的复本。
<i>replace</i> 长度为零	<i>expression</i> 的复本，其中删除了所有出现的 <i>find</i> 的字符串。
<i>start</i> > Len(<i>expression</i>)	长度为零的字符串。
<i>count</i> is 0	<i>expression</i> 的复本。

说明

Replace 函数的返回值是一个字符串，但是，其中从 *start* 所指定的位置开始，到 *expression* 字符串的结尾处的一段子字符串已经发生过替换动作。并不是原字符串从头到尾的一个复制。

(九十三) RGB 函数

返回一个 Long 整数，用来表示一个 RGB 颜色值。

语法

RGB(*red*, *green*, *blue*)

RGB 函数的语法含有以下这些命名参数：

部分	描述
<i>red</i>	必要参数；Variant (Integer)。数值范围从 0 到 255，表示颜色的红色成份。
<i>green</i>	必要参数；Variant (Integer)。数值范围从 0 到 255，表示颜色的绿色成份。
<i>blue</i>	必要参数；Variant (Integer)。数值范围从 0 到 255，表示颜色的蓝色成份。

说明

可以接受颜色说明的应用程序的方法和属性期望这个说明是一个代表 RGB 颜色值的数值。一个 RGB 颜色值指定红、绿、蓝三原色的相对亮度，生成一个用于显示的特定颜色。

传给 RGB 的任何参数的值，如果超过 255，会被当作 255。

下面的表格显示一些常见的标准颜色，以及这些颜色的红、绿、蓝三原色的成份：

颜色	红色值	绿色值	蓝色值
黑色	0	0	0
蓝色	0	0	255
绿色	0	255	0
青色	0	255	255
红色	255	0	0
洋红色	255	0	255
黄色	255	255	0
白色	255	255	255

该函数返回的 RGB 颜色值与 Macintosh 操作系统使用的不兼容，这些值可能在 Macintosh 上的 Microsoft 应用程序上下文中使用，但当通信色直接改变到 Macintosh 操作系统时，不能使用。

(九十四) Right 函数

返回 Variant (String)，其中包含从字符串右边取出的指定数量的字符。

语法

Right(*string*, *length*)

Right 函数的语法具有下面的命名参数：

部分	说明
<i>string</i>	必要参数。字符串表达式，其中最右边的字符将被返回。如果 <i>string</i> 包含 Null，将返回 Null。
<i>length</i>	必要参数；为 Variant (Long)。为数值表达式，指出想返回多少字符。如果为 0，返回零长度字符串 ("")。如果大于或等于 <i>string</i> 的字符数，则返回整个字符串。

说明

欲知 *string* 的字符数，用 Len 函数。

注意 RightB 函数作用于包含在字符串中的字节数据。所以 *length* 指定的是字节数，而不是指定返回的字符数。

(九十五) Rnd 函数

返回一个包含随机数值的 Single。

语法

Rnd[(*number*)]

可选的 *number* 参数是 Single 或任何有效的数值表达式。

返回值

如果 <i>number</i> 的值是	Rnd 生成
小于 0	每次都使用 <i>number</i> 作为随机数种子得到的相同结果。
大于 0	序列中的下一个随机数。
等于 0	最近生成的数。
省略	序列中的下一个随机数。

说明

Rnd 函数返回小于 1 但大于或等于 0 的值。

number 的值决定了 Rnd 生成随机数的方式。

对最初给定的种子都会生成相同的数列，因为每一次调用 Rnd 函数都用数列中的前一个数作为下一个数的种子。

在调用 Rnd 之前，先使用无参数的 Randomize 语句初始化随机数生成器，该生成器具有根据系统计时器得到的种子。

为了生成某个范围内的随机整数，可使用以下公式：

$\text{Int}((\text{upperbound} - \text{lowerbound} + 1) * \text{Rnd} + \text{lowerbound})$

这里，*upperbound* 是随机数范围的上限，而 *lowerbound* 则是随机数范围的下限。

注意 若想得到重复的随机数序列，在使用具有数值参数的 Randomize 之前直接调用具有负参数值的 Rnd。使用具有同样 *number* 值的 Randomize 是不会得到重复的随机数序列的。

(九十六) Round 函数

描述

返回一个数值，该数值是按照指定的小数位数进行四舍五入运算的结果。

语法

`Round(expression [, numdecimalplaces])`

Round 函数语法有如下几部分：

部分	描述
<i>expression</i>	必需的。要进行四舍五入运算的数值表达式。
<i>numdecimalplaces</i>	可选的。数字值，表示进行四舍五入运算时，小数点右边应保留的位数。如果忽略，则 Round 函数返回整数。

(九十七) Second 函数

返回一个 Variant (Integer)，其值为 0 到 59 之间的整数，表示一分钟之中的某个秒。

语法

`Second(time)`

必要的 *time* 参数，可以是任何能够表示时刻的 Variant、数值表达式、字符串表达式或它们的组合。如果 *time* 包含 Null，则返回 Null。

(九十八) Seek 函数

返回一个 Long，在 Open 语句打开的文件中指定当前的读/写位置。

语法

`Seek(filenumber)`

必要的 *filenumber* 参数是一个包含有效文件号的 Integer。

说明

Seek 函数返回介于 1 和 2, 147, 483, 647 (相当于 $2^{31} - 1$) 之间的值。

对各种文件访问方式的返回值如下：

方式	返回值
Random	下一个读出或写入的记录号。
Binary, Output, Append, Input	下一个操作将要发生时所在的字节位置。文件中的第一个字节位于位置 1，第二个字节位于位置 2，依此类推。

(九十九) Sgn 函数

返回一个 Variant (Integer)，指出参数的正负号。

语法

Sgn(*number*)

必要的 *number* 参数是任何有效的数值表达式。

返回值

如果 <i>number</i> 为	Sgn 返回
大于 0	1
等于 0	0
小于 0	-1

说明

number 参数的符号决定了 Sgn 函数的返回值。

(一百) Shell 函数

执行一个可执行文件，返回一个 **Variant (Double)**，如果成功的话，代表这个程序的任务 ID，若不成功，则会返回 0。

语法

Shell(*pathname*[, *windowstyle*])

Shell 函数的语法含有下面这些命名参数：

部分	描述
<i>pathname</i>	必要参数。 Variant (String) ，要执行的程序名，以及任何必需的参数或命令行变量，可能还包括目录或文件夹，以及驱动器。在 Macintosh 中，可以使用 MacID 函数来指定一个应用程序的署名而不是名称。下面的例子使用了 Microsoft Word 的署名： Shell MacID("MSWD")
<i>Windowstyle</i>	可选参数。 Variant (Integer) ，表示在程序运行时窗口的样式。如果 <i>windowstyle</i> 省略，则程序是以具有焦点的最小化窗口来执行的。在 Macintosh（系统 7.0 或更高）中， <i>windowstyle</i> 仅决定当应用程序运行时是否获得焦点。

windowstyle 命名参数有以下这些值：

常量	值	描述
vbHide	0	窗口被隐藏，且焦点会移到隐式窗口。常数 vbHide 在 Macintosh 平台不可用。
VbNormalFocus	1	窗口具有焦点，且会还原到它原来的大小和位置。
VbMinimizedFocus	2	窗口会以一个具有焦点的图标来显示。
VbMaximizedFocus	3	窗口是一个具有焦点的最大化窗口。
VbNormalNoFocus	4	窗口会被还原到最近使用的大小和位置，而当前活动的窗口仍然保持活动。
VbMinimizedNoFocus	6	窗口会以一个图标来显示。而当前活动的窗口仍然保持活动。

说明

如果 Shell 函数成功地执行了所要执行的文件，则它会返回程序的任务 ID。任务 ID 是一个唯一的数值，用来指明正在运行的程序。如果 Shell 函数不能打开命名的程序，则会产生错误。

在 Macintosh 中, `vbNormalFocus`、`vbMinimizedFocus` 和 `vbMaximizedFocus` 都将应用程序置于前台; `vbHide`、`vbNoFocus`、`vbMinimizeFocus` 都将应用程序置于后台。

注意 缺省情况下, `Shell` 函数是以异步方式来执行其它程序的。也就是说, 用 `Shell` 启动的程序可能还没有完成执行过程, 就已经执行到 `Shell` 函数之后的语句。

(一百〇一) Sin 函数

返回一 `Double`, 指定参数的 `sine` (正弦) 值。

语法

`Sin(number)`

必要的 `number` 参数是 `Double` 或任何有效的数值表达式, 表示一个以弧度为单位的角。

说明

`Sin` 函数取一角度为参数值, 并返回角的对边长度除以斜边长度的比值。

结果的取值范围在 -1 到 1 之间。

为了将角度转换为弧度, 请将角度乘以 $\pi / 180$ 。为了将弧度转换为角度, 请将弧度乘以 $180/\pi$ 。

(一百〇二) SLN 函数

返回一个 `Double`, 在一期里指定一项资产的直线折旧。

语法

`SLN(cost, salvage, life)`

`SLN` 函数有下列命名参数:

部分	描述
<code>cost</code>	必要。 <code>Double</code> 指定资产的初始成本。
<code>salvage</code> <code>e</code>	必要。 <code>Double</code> 指定资产在可用年限结束后的价值。
<code>life</code>	必要。 <code>Double</code> 指定资产的可用年限。

说明

折旧期间必须用与 `life` 参数相同的单位表示。所有参数都必须是正数。

(一百〇三) Space 函数

返回特定数目空格的 `Variant (String)`。

语法

`Space(number)`

必要的 `number` 参数为字符串中想要的空格数。

说明

`Space` 函数在格式输出或清除固定长度字符串数据时很有用。

(一百〇四) Spc 函数

与 `Print #` 语句或 `Print` 方法一起使用，对输出进行定位。

语法

`Spc(n)`

必要的 n 参数是在显示或打印列表中的下一个表达式之前插入的空白数。

说明

如果 n 小于输出行的宽度，则下一个打印位置将紧接在数个已打印的空白之后。如果 n 大于输出行的宽度，则 `Spc` 利用下列公式计算下一个打印位置：

$currentprintposition + (n \text{ Mod } width)$

例如，如果当前输出位置为 24，而输出行的宽度为 80，并指定了 `Spc(90)`，则下一个打印将从位置 34 开始（当前打印位置 + 90/80 的余数）。如果当前打印位置和输出行宽度之间的差小于 n （或 $n \text{ Mod } width$ ），则 `Spc` 函数会跳到下一行的开头，并产生数量为 $n - (width - currentprintposition)$ 的空白。

注意 要确保表格栏宽度足以容纳较宽的字符串。

当 `Print` 方法与间距字体一起使用时，使用 `Spc` 函数打印的空格字符的宽度总是等于选用的字体中以磅数为单位的所有字符的平均宽度。但是，在已打印字符的个数与那些字符所占据的定宽列的数目之间不存在任何关系。例如，大写英文字母 W 占据超过一个定宽的列，而小写字母 i 则占据少于一个定宽的列。

(一百〇五) Split 函数

描述

返回一个下标从零开始的一维数组，它包含指定数目的子字符串。

语法

`Split(expression[, delimiter[, limit[, compare]])`

`Split` 函数语法有如下命名参数：

部分	描述
<i>expression</i>	必需的。包含子字符串和分隔符的字符串表达式。如果 <i>expression</i> 是一个长度为零的字符串("")， <code>Split</code> 则返回一个空数组，即没有元素和数据的数组。
<i>delimiter</i>	可选的。用于标识子字符串边界的字符串字符。如果忽略，则使用空格字符(" ")作为分隔符。如果 <i>delimiter</i> 是一个长度为零的字符串，则返回的数组仅包含一个元素，即完整的 <i>expression</i> 字符串。
<i>limit</i>	可选的。要返回的子字符串数，-1 表示返回所有的子字符串。
<i>compare</i>	可选的。数字值，表示判别子字符串时使用的比较方式。关于其值，请参阅“设置值”部分。

设置值

compare 参数的设置值如下：

常数	值	描述
<code>vbUseCompareOption</code>	-1	用 <code>Option Compare</code> 语句中的设置值执行比较。
<code>vbBinaryCompare</code>	0	执行二进制比较。

vbTextCompare	1	执行文字比较。
vbDatabaseCompare	2	仅用于 Microsoft Access。基于您的数据库的信息执行比较。

(一百〇六) Sqr 函数

返回一个 Double，指定参数的平方根。

语法

Sqr(*number*)

必要的 *number* 参数 *number* 是 Double 或任何有效的大于或等于 0 的数值表达式。

(一百〇七) Str 函数

返回代表一数值的 Variant (String)。

语法

Str(*number*)

必要的 *number* 参数为一 Long，其中可包含任何有效的数值表达式。

说明

当一数字转成字符串时，总会在前头保留一空位来表示正负。如果 *number* 为正，返回的字符串包含一前导空格暗示有一正号。

使用 Format 函数可将数值转成必要的格式，如日期、时间、货币或其他用户自定义格式。与 Str 不同的是，Format 函数不包含前导空格来放置 *number* 的正负号。

注意 Str 函数只视句点 (.) 为有效的小数点。如果使用不同的小数点 (例如，国际性的应用程序)，可使用 CStr 将数字转成字符串。

(一百〇八) StrComp 函数

返回 Variant (Integer)，为字符串比较的结果。

语法

StrComp(*string1*, *string2*[, *compare*])

StrComp 函数的语法有下面的命名参数：

部分	说明
<i>string1</i>	必要参数。任何有效的字符串表达式。
<i>string2</i>	必要参数。任何有效的字符串表达式。
<i>Compare</i>	可选参数。指定字符串比较的类型。如果 <i>compare</i> 参数是 Null，将发生错误。如果省略 <i>compare</i> ，Option Compare 的设置将决定比较的类型。

设置

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。
vbTextCompare	1	执行一个按照原文的比较。
vbDatabaseCompare	2	仅适用于 Microsoft Access, 执行一个基于数据库信息的比较。

返回值

StrComp 函数有下列返回值:

如果	StrComp 返回
<i>string1</i> 小于 <i>string2</i>	-1
<i>string1</i> 等于 <i>string2</i>	0
<i>string1</i> 大于 <i>string2</i>	1
<i>string1</i> 或 <i>string 2</i> 为 Null	Null

(一百〇九) StrConv 函数

返回按指定类型转换的 Variant (String)。

语法

StrConv(*string*, *conversion*, *LCID*)

StrConv 函数的语法有下面的命名参数:

部分	说明
<i>string</i>	必要参数。要转换的字符串表达式。
<i>conversion</i>	必要参数。Integer。其值的和决定转换的类型。
<i>LCID</i>	可选的。如果与系统 LocaleID 不同, 则为 LocaleID (系统 LocaleID 为缺省值。)

设置值

conversion 参数的设置值为:

常数	值	说明
vbUpperCase	1	将字符串文字转成大写。
vbLowerCase	2	将字符串文字转成小写。
vbProperCase	3	将字符串中每个字的开头字母转成大写。
vbWide*	4*	将字符串中单字节字符转成双字节字符。
vbNarrow*	8*	将字符串中双字节字符转成单字节字符。
vbKatakana**	16**	将字符串中平假名字符转成片假名字符。
vbHiragana**	32**	将字符串中片假名字符转成平假名字符。
vbUnicode	64	根据系统的缺省码页将字符串转成 Unicode。(在 Macintosh 中不可用。)
vbFromUnicode	128	将字符串由 Unicode 转成系统的缺省码页。(在 Macintosh 中不可用。)

*应用到远东区域。

**仅应用到日本。

注意 这些常数是由 VBA 指定的。可以在程序中使用它们来替换真正的值。其中大部分是可以组合的, 例如 vbUpperCase + vbWide, 互斥的常数不能组合, 例如 vbUnicode +

vbFromUnicode。当在不适用的区域使用常数 vbWide、vbNarrow、vbKatakana, 和 vbHiragana 时, 就会导致运行时错误。

下面是一些一般情况下的有效分界符: Null (Chr\$(0)), 水平制表符 (Chr\$(9)), 换行 (Chr\$(10)), 垂直制表符 (Chr\$(11)), 换页 (Chr\$(12)), 回车 (Chr\$(13)), 空白 (SBCS) (Chr\$(32))。在 DBCS 中, 空白的实际值会随国家/地区而不同。

说明

在把 ANSI 格式的 Byte 数组转换为字符串时, 您应该使用 StrConv 函数。当您转换 Unicode 格式的这种数组时, 使用赋值语句。

(一百一十) StrReverse 函数

描述

返回一个字符串, 其中一个指定子字符串的字符顺序是反向的。

语法

StrReverse(*expression*)

参数 *expression* 是一个字符串, 它的字符顺序要被反向。如果 *expression* 是一个长度为零的字符串(""), 则返回一个长度为零的字符串。如果 *expression* 为 Null, 则产生一个错误。

(一百一十一) String 函数

回 Variant (String), 其中包含指定长度重复字符的字符串。

语法

String(*number*, *character*)

String 函数的语法有下面的命名参数:

部分	说明
<i>number</i>	必要参数; Long。返回的字符串长度。如果 <i>number</i> 包含 Null, 将返回 Null。
<i>character</i>	必要参数; Variant。为指定字符的字符码或字符串表达式, 其第一个字符将用于建立返回的字符串。如果 <i>character</i> 包含 Null, 就会返回 Null。

说明

如果指定 *character* 的数值大于 255, String 会按下面的公式将其转为有效的字符码:

character Mod 256

(一百一十二) Switch 函数

计算一组表达式列表的值, 然后返回与表达式列表中最先为 True 的表达式所相关的 Variant 数值或表达式。

语法

Switch(*expr-1*, *value-1* [, *expr-2*, *value-2* _ [, *expr-n*, *value-n*]])

Switch 函数的语法具有以下几个部分:

部分	描述
<i>expr</i>	必要参数。要加以计算的 Variant 表达式。

<i>value</i>	必要参数。如果相关的表达式为 True ，则返回此部分的数值或表达式。
--------------	--

说明

Switch 函数的参数列表由多对表达式和数值组成。表达式是由左至右加以计算的，而数值则会在第一个相关的表达式为 **True** 时返回。如果其中有部分不成对，则会产生一个运行时错误。如果 *expr-1* 为 **True** 则 **Switch** 返回 *value-1*，如果 *expr-1* 为 **False**，但 *expr-2* 为 **True**，则 **Switch** 返回 *value-2*，以此类推。

Switch 会返回一个 **Null** 值，如果：

- 没有一个表达式为 **True**。
- 第一个为 **True** 的表达式，其相对应的值为 **Null**。

虽然它只返回其中的一个值，但是 **Switch** 会计算所有的表达式。因此应该注意到所产生的副作用。例如，只要其中一个表达式导致被零除错误，就会发生错误。

(一百一十三) SYD 函数

返回一个 **Double**，指定某项资产在一指定期间用年数总计法计算的折旧。

语法

SYD(*cost*, *salvage*, *life*, *period*)

SYD 函数有下列命名参数：

部分	描述
<i>cost</i>	必要。 Double 指定资产的初始成本。
<i>salvage</i>	必要。 Double 指定资产在可用年限结束后的价值。
<i>life</i>	必要。 Double 指定资产的可用年限。
<i>period</i>	必要。 Double 指定计算资产折旧所用的那一期间。

说明

必须用相同的单位表示 *life* 和 *period* 参数。例如，如果 *life* 用月份表示，则 *period* 也必须用月份表示。所有参数都必须是正数。

(一百一十四) Tab 函数

与 **Print #** 语句或 **Print** 方法一起使用，对输出进行定位。

语法

Tab[(*n*)]

可选的 *n* 参数是在显示或打印列表中的下一个表达式之前移动的列数。若省略此参数，则 **Tab** 将插入点移动到下一个打印区的起点。这就使 **Tab** 可用来替换区域中的逗号，此处，逗号是作为十进制分隔符使用的。

说明

如果当前行上的打印位置大于 *n*，则 **Tab** 将打印位置移动到下一个输出行的第 *n* 列上。如果 *n* 小于 1，则 **Tab** 将打印位置移动到列 1。如果 *n* 大于输出行的宽度，则 **Tab** 函数使用以下公式计算下一个打印位置：

$n \text{ Mod } width$

例如，如果 *width* 是 80，并指定 **Tab(90)**，则下一个打印将从列 10 开始 (90/80 的余数)。如果 *n* 小于当前打印位置，则从下一行中计算出来的打印位置开始打印。如果计算后的打印位置大于当前打印位置，则从同一行中计算出来的打印位置开始打印。输出行最左端的打印位置总是 1。在使用 **Print #** 语句将数据写入文件时，最右端的打印位置是输出文件的当前宽度，这一宽度可用 **Width #** 语句设置。

注意 要确保表格列的宽度足以容纳较宽的字符串。

当 **Print** 方法与 **Tab** 函数一起使用时，打印的外观将会被分割为均匀、定宽的列。各列的宽度等于选用字体内以磅数为单位的所有字符的平均宽度。但是，在已打印字符的个数与那些字符所占据的定宽列的数目之间不存在任何关系。例如，大写字母 W 占据超过一个定宽的列，而小写字母 i 则占据少于一个定宽的列。

(一百一十五) Tan 函数

返回一个 **Double** 的值，指定一个角的正切值。

语法

Tan(*number*)

必要的 *number* 参数是 **Double** 或任何有效的数值表达式，表示一个以弧度为单位的角度。

说明

Tan 取一角度为参数值，并返回直角的两条邻边的比值。该比值是角的对边长度除以角的邻边长度的商。

为了将角度转换为弧度，请将角度乘以 $\pi/180$ 。为了将弧度转换为角度，请将弧度乘以 $180/\pi$ 。

(一百一十六) Time 函数

返回一个指明当前系统时间的 **Variant (Date)**。

语法

Time

说明

为了设置系统时间，请使用 **Time** 语句。

(一百一十七) Timer 函数

返回一个 **Single**，代表从午夜开始到现在经过的秒数。

语法

Timer

说明

Microsoft Windows 中，**Timer** 函数返回一秒的小数部分。在 Macintosh 上，计时器的精度是 1 秒。

(一百一十八) TimeSerial 函数

返回一个 Variant (Date)，包含具有具体时、分、秒的时间。

语法

TimeSerial(*hour*, *minute*, *second*)

TimeSerial 函数语法有下列的命名参数：

部分	描述
<i>hour</i>	必要；Variant (Integer)。其值从 0 (12:00 A.M.) 到 23 (11:00 P.M.)，或一数值表达式。
<i>minute</i>	必要；Variant (Integer)。任何数值表达式。
<i>second</i>	必要；Variant (Integer)。任何数值表达式。

说明

为了指定一个时刻，如 11:59:59，**TimeSerial** 的参数取值应在正常范围内；也就是说，钟点应介于 0-23 之间，而分钟与秒应介于 0-59 之间。但是，当一个数值表达式表示某时刻之前或其后时的、分钟或秒数时，也可以为每个使用这个数值表达式的参数指定相对时间。以下示例中使用了表达式代替绝对时间数。**TimeSerial** 函数返回中午之前六小时 (12 - 6) 又十五分钟 (-15) 的时间，即 5:45:00 A.M.

TimeSerial(12 - 6, -15, 0)

当任何一个参数的取值超出正常范围时，它会适时进位到下一个较大的时间单位。例如，如果指定了 75 (75 分钟)，则这个时间被解释成一小时又十五分。如果一个参数值超出 -32, 768 到 32, 767 的范围，就会导致错误发生。如果三个参数指定的时间会使日期超出可接受的日期范围，则亦会导致错误发生。

(一百一十九) TimeValue 函数

返回一个包含时间的 Variant (Date)。

语法

TimeValue(*time*)

必要的 *time* 参数，通常是一个字符串表达式，表示 0:00:00 (12:00:00 A.M.) 到 23:59:59 (11:59:59 P.M.) 之间的时刻。但是，*time* 也可以是表示在同一时间范围取值的任何其它表达式。如果 *time* 包含 Null，则返回 Null。

说明

可以使用 12 小时制或 24 小时制的时间格式。例如，" 2:24PM" 和 "14:24" 均是有效的 *time* 表达式。

如果 *time* 参数包含日期信息，**TimeValue** 将不会返回它。但是，若 *time* 包含无效的日期信息，则会导致错误发生。

(一百二十) TypeName 函数

返回一个 **String**，提供有关变量的信息。

语法

TypeName (*varname*)

必要的 *varname* 参数是一个 **Variant**，它包含用户定义类型变量之外的任何变量。

说明

TypeName 所返回的字符串可以是下面列举的任何一个字符串：

返回字符串	变量
对象类型	类型为 <i>objecttype</i> 的对象
Byte	位值
Integer	整数
Long	长整数
Single	单精度浮点数
Double	双精度浮点数
Currency	货币
Decimal	十进制值
Date	日期
String	字符串
Boolean	Boolean 值
Error	错误值
Empty	未初始化
Null	无效数据
Object	对象
Unknown	类型未知的对象
Nothing	不再引用对象的对象变量

如果 *varname* 是一个数组，则返回的字符串可以是任何一个后面添加了空括号的可能的返回字符串（或 **Variant**）。例如，如果 *varname* 是一个整数数组，则 **TypeName** 返回 "Integer()"。

(一百二十一) UBound 函数

返回一个 **Long** 型数据，其值为指定的数组维可用的最大下标。

语法

UBound (*arrayname* [, *dimension*])

UBound 函数的语法包含下面部分：

部分	描述
<i>arrayname</i>	必需的。数组变量的名称，遵循标准变量命名约定。
<i>dimension</i>	可选的； Variant (Long) 。指定返回哪一维的上界。1 表示第一维，2 表示

第二维，如此等等。如果省略 *dimension*，就认为是 1。

说明

UBound 函数与 **LBound** 函数一起使用，用来确定一个数组的大小。**LBound** 用来确定数组某一维的上界。

对具有下述维数的数组而言，**UBound** 的返回值见下表：

```
Dim A(1 To 100, 0 To 3, -3 To 4)
```

语句	返回值
UBound(A, 1)	100
UBound(A, 2)	3
UBound(A, 3)	4

(一百二十二) UCase 函数

返回 **Variant (String)**，其中包含转成大写的字符串。

语法

UCase(*string*)

必要的 *string* 参数为任何有效的字符串表达式。如果 *string* 包含 **Null**，将返回 **Null**。

说明

只有小写的字母会转成大写；原本大写或非字母之字符保持不变。

(一百二十三) Val 函数

返回包含于字符串内的数字，字符串中是一个适当类型的数值。

语法

Val(*string*)

必要的 *string* 参数可以是任何有效的字符串表达式。

说明

Val 函数，在它不能识别为数字的第一个字符上，停止读入字符串。那些被认为是数值的一部分的符号和字符，例如美圆号与逗号，都不能被识别。但是函数可以识别进位制符号 **&O**（八进制）和 **&H**（十六进制）。空白、制表符和换行符都从参数中被去掉。

下面的返回值为 1615198：

```
Val(" 1615 198th Street N.E.")
```

在下面的代码中，**Val** 为所示的十六进制数值返回十进制数值 -1。

```
Val("&HFFFF")
```

注意 **Val** 函数只会将句点 (.) 当成一个可用的小数点分隔符。当使用不同的小数点分隔符时，如在国际版应用程序中，代之以 **Cdbl** 来把字符串转换为数字。

(一百二十四) VarType 函数

返回一个 **Integer**，指出变量的子类型。

语法

VarType (varname)

必要的 *varname* 参数是一个 Variant，包含用户定义类型变量之外的任何变量。

返回值

常数	值	描述
vbEmpty	0	Empty (未初始化)
vbNull	1	Null (无有效数据)
vbInteger	2	整数
vbLong	3	长整数
vbSingle	4	单精度浮点数
vbDouble	5	双精度浮点数
vbCurrency	6	货币值
vbDate	7	日期
vbString	8	字符串
vbObject	9	对象
vbError	10	错误值
vbBoolean	11	Boolean 值
vbVariant	12	Variant (只与变体中的数组一起使用)
vbDataObject	13	数据访问对象
vbDecimal	14	十进制值
vbByte	17	位值
vbUserDefinedType	36	包含用户定义类型的变量
vbArray	8192	数组

注意 这些常数是由 Visual Basic 为应用程序指定的。这些名称可以在程序代码中到处使用，以代替实际值。

说明

VarType 函数自身从不对 **vbArray** 返回值。**VarType** 总是要加上一些其他值来指出一个具体类型的数组。常数 **vbVariant** 只与 **vbArray** 一起返回，以表明 **VarType** 函数的参数是一个 **Variant** 类型的数组。例如，对一个整数数组的返回值是 **vbInteger + vbArray**，或 8194。如果一个对象有缺省属性，则 **VarType (object)** 返回对象缺省属性的类型。

(一百二十五) Weekday 函数

返回一个 **Variant (Integer)**，包含一个整数，代表某个日期是星期几。

语法

Weekday(*date*, [*firstdayofweek*])

Weekday 函数语法有下列的命名参数：

部分	描述
<i>date</i>	必要。能够表示日期的 Variant、数值表达式、字符串表达式或它们的组合。如果 <i>date</i> 包含 Null，则返回 Null。
<i>Firstdayofweek</i>	可选。指定一星期第一天的常数。如果未予指定，则以 vbSunday 为缺省值。

设置

firstdayofweek 参数有以下设定值:

常数	值	描述
vbUseSystem	0	使用 NLS API 设置。
VbSunday	1	星期日 (缺省值)
vbMonday	2	星期一
vbTuesday	3	星期二
vbWednesday	4	星期三
vbThursday	5	星期四
vbFriday	6	星期五
vbSaturday	7	星期六

返回值

Weekday 函数可以返回以下诸值:

常数	值	描述
vbSunday	1	星期日
vbMonday	2	星期一
vbTuesday	3	星期二
vbWednesday	4	星期三
vbThursday	5	星期四
vbFriday	6	星期五
vbSaturday	7	星期六

(一百二十六) WeekdayName 函数

描述

返回一个字符串, 表示一星期中的某天。

语法

WeekdayName(*weekday*, *abbreviate*, *firstdayofweek*)

WeekdayName 函数语法有如下几部分:

部分	描述
<i>weekday</i>	必需的。数字值, 表示一星期中的某天。该数字值要依赖于 <i>firstdayofweek</i> 设置中的设置值来决定。
<i>abbreviate</i>	可选的。 Boolean 值, 表示星期的名称是否被缩写。如果忽略该值, 缺省值为 False , 表明星期的名称不能被缩写。
<i>firstdayofweek</i>	可选的。数字值, 表示一星期中第一天。关于其值, 请参阅“设置值”部分。

设置值

firstdayofweek 参数值如下:

常数	值	描述
vbUseSystem	0	使用本国语言支持 (NLS) API 设置值。
vbSunday	1	星期日 (缺省)。
vbMonday	2	星期一
vbTuesday	3	星期二
vbWednesday	4	星期三
vbThursday	5	星期四
vbFriday	6	星期五
vbSaturday	7	星期六

(一百二十七) Year 函数

返回 Variant (Integer)，包含表示年份的整数。

语法

Year(*date*)

必要的 *date* 参数，可以是任何能够表示日期的 Variant、数值表达式、字符串表达式或它们的组合。如果 *date* 包含 Null，则返回 Null。