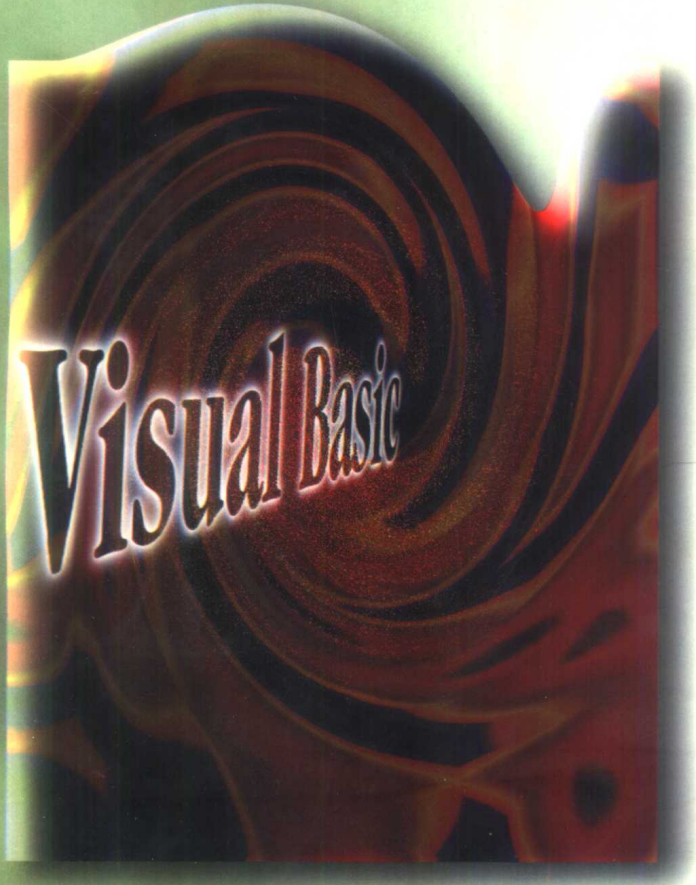
计算机语言函数应用丛书

Visual Basic

计算机语言函数应用

袁晓君　李　明　袁晓文　编著



科学出版社

计算机语言函数应用丛书

Visual Basic

计算机语言函数应用

袁晓君　李　明　袁晓文　编著

科学出版社

2000

内 容 简 介

Visual Basic 的基础是不断发展的 Basic，这一基础在近 20 多年取得很大改进。由于 Basic 是一种简单有力的开发工具，因此通常认为该语言是一种高级的程序开发环境。

现在，Visual Basic 发展到最新的 Visual Basic 6.0 版本。

本书全面介绍了 Visual Basic 6.0 语言的操作符、语句、函数、方法、对象和控件，是一本非常全面具体的 Visual Basic 6.0 参考大全。

图书在版编目（CIP）数据

Visual Basic 计算机语言函数应用/袁晓君等编．-北京：科学出版社，2000.1

ISBN 7-03-007946-9

Ⅰ.计… Ⅱ.袁… Ⅲ.BASIC 语言 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（1999）第 63470 号

科学出版社 出版

北京东黄城根北街 16 号

邮政编码：100717

北京双青印刷厂 印刷

新华书店北京发行所发行 各地新华书店经售

*

2000 年 1 月第 一 版 开本：787×1092 1/16

2000 年 1 月第一次印刷 印张：33

印数：1—5 100 字数：779 000

定价：46.00 元

（如有印装质量问题，我社负责调换〈环伟〉）

前 言

Visual Basic 是不断发展一种语言，在近 20 多年得到了很大的改进。由于 Basic 本身是一种简单且功能强大的开发工具，因此该语言通常被认为是一种高级的程序开发环境。概括来讲，Visual Basic 是 Basic 的一个结构化版本，是一种解释型的语言，不是编译型语言，即每一条 Basic 指令均以操作码形式存放。一个 op-code 是一个描述确定功能的数，它可通过解释器被传送。发送到微处理器的指令完成所要执行的任务。而一个 Basic 程序是一系列顺序执行的 op-code。

现在，Visual Basic 已经发展到最新的 Visual Basic 6.0 版本。本书全面介绍了 Visual Basic 6.0 语言的操作符、语句、函数、方法、对象和控件。可以说，本书是一本非常全面且具体的 Visual Basic 6.0 参考大全。本书是为用 Visual Basic 6.0 语言编程的程序员编写的。

本书的结构如下：

第一章　详细阐述了 Visual Basic 的语言风格，并详细介绍了操作符和语句，同时附有大量实例。

第二章　详细说明 Visual Basic 6.0 语言的函数的使用，同时附有大量范例。

第三章　全面介绍 Visual Basic 6.0 语言涉及的方法，同时附有大量范例。

第四章　说明 Visual Basic 6.0 语言的对象的使用，同时附有大量范例。

第五章　说明 Visual Basic 6.0 语言的控件的使用，同时附有大量范例。

目　录

前　言
1　Visual Basic 语言 …… (1)
　1.1　Visual Basic 语言风格 …… (1)
　1.2　操作符 …… (3)
　1.3　语句 …… (17)
2　函数 …… (83)
3　方法 …… (158)
4　对象 …… (366)
5　控件 …… (455)

1 Visual Basic 语言

1.1 Visual Basic 语言风格

Visual Basic 是 Basic 的一个结构化版本，它缺少在 Basic 以往格式中的传统行号，且不强调 GOTO 命令。Basic 中的 GOTO 命令使程序直接跳到执行代码的另一点，从而使程序流不易跟踪和理解。所以 Basic 程序的维护非常简单，但代价很高。Basic 是一种解释型语言，不是编译型语言，它每一条 Basic 指令均以操作码形式存放。一个 op-code 是一个描述确定功能的数，它可通过解释器被传送。发送到微处理器的指令完成所要执行的任务。Basic 程序是一系列顺序执行的 op-code。

下面详细介绍 Visual Basic 语言的风格。

Visual Basic 程序含有表格和模块。其中，表格含有窗口、代码和将被显示的控制层次信息，模块仅含有不带可视部件的代码文件。对象是含有基本代码和变量的存储结构。

Visual Basic 中含有多种术语来描述子例程。在 Visual Basic 中，一个 Sub-End Sub 单元是一个可以接受值但是不能返回值的例程。

1.1.1 表格和控制

表格是窗口和它所显示的控制，以及在表格和它的控制中的事件代码的组合。一般地说，存在两类表格文件：.FRM 文件和 .FRX 文件。其中 .FRM 文件是用文本方式存放的表格文件。.FRX 是用二进制格式存放的表格文件。在控制调色板上选择一个控制，并在该表格上拖动它可以构成一个表格。如果显示 Properties 窗口，则仅出现公共域。如果改变 Properties 窗口中控制的名字，所有与该控制代码关联的代码不自动随名字的改变而改变。如果从该屏幕上删除一个控制的话，程序不自动移去与之关联的代码。

1.1.2 模块

模块是代码文件，含有程序与所有表格关联的代码和从任何其他文件访问到的函数和子例程。子例程和函数可以接收从其他模块或表格中的子例程传送给他们的数据。

模块是存放所有代码和需要共享的变量的空间。这时，变量要求关键字 GLOBAL 定义在它们的前面，使它们对于任意子例程都是可访问的，否则只能由在该模块中的子例程使用。在以往的程序设计风格中，全局变量和例程在该程序的任意地方均可访问和修改。

另外，如果很多其他子例程使用一个全局子例程，则对该例程功能的改变可能会要求其他的例程来重写。而现在将代码和变量封在一个对象中，他们可以独立地工作，而不用担心与有些微变的整个程序产生冲突。而且，由于直接和该代码关联的代码和变量保持

与该对象一致,从而使代码更清楚。

1.1.3 对象和实例

Visual Basic 中程序中的多数公共项被认为是对象。对象是内存的结构,有特定的性质,且通常与子例程关联。内存中对象的拷贝是实例。

Visual Basic 缺少真正面向对象语言的一些要求,但它继承了许多相同的规则,因此具有许多相同的优点。

1.1.4 程序文件

一旦完成 Visual Basic 程序,则创建一个.exe 文件。该文件含有组成该程序的所有 op-code 指令和另外负责装入 Visual Basic 运行时间库和负责程序开始执行的代码。

可以说,Visual Basic 是一种非常有使用价值的语言,它可以帮助你在 Basic 中做任何想做的事,同时插入件使你节约了时间。

一般认为,在着重于用户界面的应用程序中,Visual Basic 比传统的开发环境节约 75%的时间。

事件源

Visual Basic 语言以事件驱动来完成功能。它在事件驱动方面作出了很大的贡献。在开发系统中提供完整的事件循环和处理大部分共同事件的缺省例程。

如果需要剪裁该应用程序对一个给定事件的反应,只需要简单剪裁这个特定件出现时被调用的例程。

项目

Visual Basic 开发环境是基于项目隐喻的。一个项目含有该应用程序使用的所有文件。项目隐喻提供一种实际的方法,使应用程序需要的所有文件放在一起。文件可以被网络上的项目开发组共享,但同一时刻只有一个用户可以修改一个给定文件。项目中的任何文件都可以以二进制或文本文件格式存放,缺省方式是二进制。项目应该包含以后将编译.EXE 文件所需的所有文件。

外部资源

Visual Basic 程序员的一个较大的优势是,具备 Visual Basic 开发环境以外的许多资源能完成程序设计任务的能力。

Windows API

API 是与给定应用程序进行交互的方法,Windows API 允许任何程序设计语言传递参数和执行嵌入在操作系统中的例程。

Visual Basic 通过提供较简单命令的方式管理 API 调用,这些命令与 Visual Basic 环境紧密地结合。通过简化命令集合和避免用户使用的复杂性,Visual Basic 删除了一些复杂但却有用的功能的访问。Visual Basic 可以帮助你直接调用 Windows API,给 Visual Basic 程

序员提供具有 C 程序员一样的能力，也增加了程序的设计复杂性和避免冲突的能力。多数 Visual Basic 程序员在能用传统 Visual Basic 命令的地方尽量使用他们，在 Visual Basic 没有强有力功能的地方可使用 Windows API 访问。

Windows API 函数通常放在代码模块中。

插入部件

插入部件是预先编译的代码段，通常将一个特殊的任务完成得很好。

插入部件包括协议书界面、API 界面、用户界面附属物或客户机/服务器开发辅助。DLL 是最简单类型的部件。

OCX 是 VBX 的新的全特征版本，VBX 在 Windows 95 操作系统中是突出的，OCX 标准较 VBX 提供的标准更全面。创建该标准使 OCX 可以插入到一些诸如 C/C + + 和像 MS Access 这样的应用程序的开发环境中。OCX 还可以插入到任何程序设计语言中，其中包括 Visual Basic。

帮助文件编译器

Windows API 包含一个帮助文件编译器，使你可以建立一个完整的帮助文件，并使它包含在你的程序中。

Setup Wizard

为了简化将客户应用程序分发给多个用户安装的任务，Microsoft 含有 Setup Wizard。

Setup Wizard 将一步一步带你建立安装盘。写到软盘上的实际安装程序是一个 VB 程序，可以修改该程序来做想做的事情。

Crystal 报告

Crystal 报告是含有在 Visual Basic 中的报告产生软件段。

如果你开始用 Crystal 报告 OCX 开发应用程序，则有一个强有利的升级版本可用。Crystal 报告专业版能满足几乎任何数据库报告生成的需要。

1.2 操作符

列表如下：

操作符	功　能
+	求两数之和。
-	计算两个数之间的差值，或计算一个数值表达式的负数。
*	用来使两数相乘。
/	用来使两数相除，返回一个浮点数结果。
\	用来使两数相除，返回整数结果。
&	强制地把两上表达式作为字符串连接。

续表

操作符	功　能
^	计算数值的幂值。
=	向变量或属性赋值。
AddressOf	把跟随在它后面的过程的地址传递给一个 API 过程，此过程的参数列表的相应位置需要一个函数指针。
And	执行对两个表达式的逻辑“与”。
Comparison	比较两个表达式的大小。
Eqv	执行对两个表达式的逻辑等价比较
imp	用来求两个表达式的逻辑蕴涵值。
Is	比较两个对象引用变量。
Like	比较两个字符串。
Mod	执行取模运算。
Not	对表达式执行逻辑求反。
Or	对两个表达式执行逻辑“或”。
Xor	对两个表达式执行逻辑“异或”。

+

语法：result = expression1 + expression2

功能：求两数之和。

参数：expression1 必备，任何表达式。

expression2 必备，任何表达式。

result 必备，任何数值型变量。

说明：当使用“+”操作符时，不需要判断将要进行的是求和操作还是进行字符串合并操作。根据表达式的不同，有以下几种情况：

若两个表达式都是变体，则使用以下规则：

若两个变体都是数值型，则执行求和运算；若两个变体表达式都是字符串，则执行字符串连接运算；若一个变体表达式是数值型的，另一个是字符串，则执行求和运算。对于只有数值型数据参与的简单算术求和运算，结果的数据类型一般与最精确的表达式相同。由低到高的精确级顺序为：Byte 型、Integer 型、Long 型、Single 型、Double 型、Currency 型和 Decimal 型。以下是这个顺序的例外情况：

若一个 Single 型数据和一个 Long 型数据相加，则结果为 Double 型；若运算结果是溢出合法范围的 Long 型、Single 型和 Date 型变体，则将其转换成 Double 型变体；若运算结果是溢出合法范围的 Byte 型变体，则将其转换成 Integer 型变体；若运算结果是溢出合法范围的 Integer 型变体，则将其转换成 Long 型变体；若是一个 Date 型数据与其他任何数据类型相加，则结果为 Date 型。若一个表达式是 Null 表达式，或者两个表达式都是 Null 表达式，结果为 Null。

若两个表达式都是空(Empty),结果是一个 Integer 型数据。但是,若只有一个表达式为空(Empty),则另一个表达式不能改变,作为结果返回。

若至少有一个表达式不是变体,使用以下规则:

若两个表达式都是字符串,则执行字符串连接运算;若两个表达式都是数值型的数据(Byte 型、Boolean 型、Integer 型、logn 型、Single 型、Double 型、Date 型、Currency 型或 Decimal 型),则执行求和运算;若一个表达式是字符串,另一个表达式是任何除 Null 之外的变体,则执行字符串连接运算;若一个表达式是数值型数据,另一个表达式是任何除 Null 之外的变体,则执行求和运算。若一个字符串是空(Empty)变体,则把另一个表达式不加改变,作为结果返回;若一个表达数值型数据,另一个是字符串,则产生类型不匹配错误;任何一个表达式为 Null,则结果为 Null。

注意:加、减法运算的精确度等级与乘法运算的精确度等级不同。

范例:下例中实现两整数相加。

```
int myNumA = 4, myNumB = 6, myNumC;
myNumC = myNumA + myNumB;
```

-

语法:result = number1-number2

功能:计算两个数之间的差值,或计算一个数值表达式的负数。

参数:number　必备,任何数值型表达式。

number1　必备,任何数值型表达式。

number2　必备,任何数值型表达式。

result　必备,可以是任何数值型变量。

说明:分两种情况:在第一种语法中,操作符"-"是一个算术运算符,用来计算两个数之间的差值;在第二种语法中,操作符"-"是一个负号操作符,用来表示表达式的负值。另外,结果的数据类型一般与最精确的表达式的数据类型相同。精确度的顺序,自低到高依次为:Byte 型、Integer 型、Long 型、Single 型、Doulbe 型、Currency 型和 Decimal 型。这个顺序的例外情况如下所列:若减法中包括了一个 Single 型和一个 Long 型数据,则把结果转换为 Double 型;若结果是溢出合法边界的 Long 型、Single 型或者 Date 型变体,则把结果转换成包含 Double 型的变体;若结果是溢出边界的 Byte 型变体,则把结果转换成 Integer 型变体;若结果是溢出边界的 Integer 型变体,则把结果转换成 Long 型变体;若在减法中包括了两个 Date 型表达式,则结果为 Double 型;若有一个表达式是 Null,或者两个表达式都是 Null,则结果为 Null;若有一个表达式是 Empty,则把该表达式作为 0 对待。

注意:加、减法所使用的精确度等级与乘法所使用的精确度等级不同。

范例:下例中实现两整数相减。

```
int myNumA = 6, myNumB = 4, myNumC;
myNumC = myNumA - myNumB;
```

*

语法:result = number1 * number2

功能:使两个数相乘。

参数:number1 必备,任何数值型变量。

number2 必备,任何数值型变量。

result 必备,任何数值型变量。

说明:结果的数据类型总是与最精确的表达式的数据类型相同。数据类型的精确度等级,从低到高依次为:Byte,Integer,Long,Single,Currency,Double 和 Decimal。在下面列举了这个顺序的例外情况:若操作的结果涉及到一个 Single 型和一个 Long 型变量,则把结果转换为 Double 型;若结果的数据类型是超出了合法范围的 Long 型、Single 型或者 Date 型变体,则把结果转换为包含 Double 型的变体;若结果的数据类型是超出合法范围的 Byte 型变体,则把结果转换为 Integer 型变体;若结果的数据类型是超出合法范围的 Integer 型变体,则把结果转换为 Long 型变体;若两个表达式中有一个是 Null 表达式,或者两个表达式都是 Null 表达结果为 Null;若一个表达式为 Empty,则把这个表达式作为 0 对待。

注意:乘法运算所使用的精确度等级与加法运算和减法运算所使用的精确度等级不同。

范例:下例中实现两整数相乘。

```
Dim myProd
myProd = 2 * 4 (返回 8)
myProd = 100.3 *0.1. (返回 10.03)
```

/

语法:result = number1/number2

功能:使两个数相除,返回一个浮点数结果。

参数:number1 必备,任何数值型表达式。

number2 必备,任何数值型表达式。

result 必备,任何数值型变量。

说明:结果的数据类型一般是 Double 型或 Double 型变体。其例外情况如下:

在两个表达式都是 Byte 型、Integer 型或者 Single 型表达式时,若计算结果没有溢出,则结果为 Single 型;若溢出,则产生错误。当两个表达式都是 Byte 型、Integer 型或者 single 型变体时,若计算结果没有溢出,则结果为 Single 型变体;若溢出,则结果为包含 Double 型的为体;若在除法中包括一个 Decimal 型和另一个其他任何数据类型,则结果 Decimal 型数据;若有一个表达式是 Null,或者两个表达式都是 Null,结果为 Null;任何为 Empty 的表达式都作为 0 对待。

范例:下例使用该操作符完成浮点除。

```
Dim MyValue
MyValue = 10 / 4
返回 2.5
```

MyValue = 10 / 3
返回 3.333333

\

语法:result = number1 \ number2
功能:用来使两数相除,返回整数结果。
参数:number1 必备,任何数值型表达式。
number2 必备,任何数值型表达式。
result 必备,任何数值型变量。
说明:在执行除法前,参与运算的数值型表达式都被取整为 Byte 型、Integer 型或 Long 型表达式。
一般而言,无论结果是不是纯粹的数,结果的数据类型都是 Byte 型、Byte 型变体、Integer 型、Integer 型变体、Long 型或 Long 型变体。但是,若有一个表达式为 Null,结果就是 Null。任何为 Empty 的表达式作为 0 对待。
范例:下例使两整数相除。

```
Dim myQuot
myQuot = 11 \ 2 (返回 5)
myQuot = 12 \ 3 (返回 4)
```

&

语法:result = expression1&expression2
功能:强制地把两个表达式作为字符串连接。
参数:expression1 必备,可以是任何表达式。
expression2 必备,可以是任何表达式。
result 必备,可以是任何字符型变量或变体型变量。
说明:若两个表达式都是字符型表达式,则结果的数据类型是字符型;若一个表达式不是字符串,则将把它转换成字符型变量;若两个表达式都为 Null,则结果为 Null。但是,若只有一个表达式为 Null,则在把这个表达式连接到另一表达式时,把它作为一个零长度的字符串("")对待。任何为 Empty 的表达式也被作为零长度字符串对待。
范例:下例演示如何使用此操作符。

```
Dim myStr
myStr = "Hello" & " World"
myStr = "Check " & 123 & " Check"
```

^

语法:result = number1^exponent
功能:计算数值的幂值。
参数:number1 必备,任何数值型表达式。
number2 必备,任何数值型表达式。

result 必备，任何数值型变量。

说明：一般情况下，结果的数据类型是 Double 型或包含 Double 型的变体。但是，若底数或指数是 Null 表达式，结果是 Null。只有当指数是整数时，底数才可以是负数。若在同一个表达式中执行多个乘幂，则按照自左到右的顺序逐个执行^操作。

范例：下例演示如何使用此操作符。

```
Dim myPow
myPow = 2 ^ 2
myPow = (-5) ^ 3
```

=

语法：variable = value

功能：向变量或属性赋值。

参数：Value　任何数值型字符串、常数或表达式。

variable　任何变量或任何可读写属性。

说明：等号左边的属性必须是运行时可读写的属性。等号左边的名称可以是单个变量名或数组元素名。

AddressOf

功能：把跟随在它后面的过程的地址传递给一个 API 过程，此过程的参数列表的相应位置需要一个函数指针。

语法：AddressOf procedurename

参数：procedurename　必备，它指定被传递地址的过程，必须是作出取地址调用的工程中某一个标准模块中的一个。

说明：虽然可以使用 AddressOf 操作符通过 Basic 过程传递过程指针，但是不能经由来自 Basic 内部的这种指针调用函数。这就意味着，编写在 Basic 代码中的类不能使用这样的指针对控制它的类作出回调。若使用 AddressOf 在 Basic 过程中传递过程指针，则被调用的过程的参数的类型必须为 Long 型。当过程名出现在参数列表中时，一般都执行该过程，并且传递该过程返回值的地址。操作符 AddressOf 允许把过程的地址而不是过程返回值的地址传递给动态链接库(DLL)中的一个 WindowsAF 函数。然后这个 API 函数可以使用传递过来的函数调用 Basic 过程，这种处理方式被称“回调”。AddressOf 操作符只出现在对 API 过程的调用中。

注意：若没有彻底理解函数回调的概念，使用 AddressOf 可能导致不可预期的后果。必须首先理解回调处理中 Basic 部分的工作原理，弄清楚将要接收函数地址的 DLL 中的代码。因为程序在开发环境中有相同的运行过程，调试这种交互是十分困难的。在某些情况下，系统调试也是不可能的。另外，可以创建自己的回调函数模型，并把这些模型放在由 Microsoft Visual C++（或相似工具）编译的 DLL 中。为了使用 AddressOf 操作符，模型必须使用 _stdcall 调用协定。默认的调用协定(cdecl)不能同 AddressOf 一起使用。

参照：Declare 语句，Function 语句，Property Get 语句，Property Let 语句，Property Set 语句，Sub

语句。

范例:下例演示如何使用此操作符。

```
Public Const LF_FACESIZE = 32
Public Const LF_FULLFACESIZE = 64
Type LOGFONT
        lfHeight As Long
        lfWidth As Long
        lfEscapement As Long
        lfOrientation As Long
        lfWeight As Long
        lfItalic As Byte
        lfUnderline As Byte
        lfStrikeOut As Byte
        lfCharSet As Byte
        lfOutPrecision As Byte
        lfClipPrecision As Byte
        lfQuality As Byte
        lfPitchAndFamily As Byte
        lfFaceName(LF_FACESIZE) As Byte
End Type

Type NEWTEXTMETRIC
        tmHeight As Long
        tmAscent As Long
        tmDescent As Long
        tmInternalLeading As Long
        tmExternalLeading As Long
        tmAveCharWidth As Long
        tmMaxCharWidth As Long
        tmWeight As Long
        tmOverhang As Long
        tmDigitizedAspectX As Long
        tmDigitizedAspectY As Long
        tmFirstChar As Byte
        tmLastChar As Byte
        tmDefaultChar As Byte
        tmBreakChar As Byte
        tmItalic As Byte
        tmUnderlined As Byte
```

```
        tmStruckOut As Byte
        tmPitchAndFamily As Byte
        tmCharSet As Byte
        ntmFlags As Long
        ntmSizeEM As Long
        ntmCellHeight As Long
        ntmAveWidth As Long
End Type

Public Const NTM_REGULAR = &H40&
Public Const NTM_BOLD = &H20&
Public Const NTM_ITALIC = &H1&
Public Const TMPF_FIXED_PITCH = &H1
Public Const TMPF_VECTOR = &H2
Public Const TMPF_DEVICE = &H8
Public Const TMPF_TRUETYPE = &H4
Public Const ELF_VERSION = 0
Public Const ELF_CULTURE_LATIN = 0

Public Const RASTER_FONTTYPE = &H1
Public Const DEVICE_FONTTYPE = &H2
Public Const TRUETYPE_FONTTYPE = &H4

Declare Function EnumFontFamilies Lib "gdi32" Alias _
    "EnumFontFamiliesA"_
    (ByVal hDC As Long, ByVal lpszFamily As String,_
    ByVal lpEnumFontFamProc As Long, LParam As Any) As Long
Declare Function GetDC Lib "user32" (ByVal hWnd As Long) As Long
Declare Function ReleaseDC Lib "user32" (ByVal hWnd As Long,_
    ByVal hDC As Long) As Long

Function EnumFontFamProc(lpNLF As LOGFONT, lpNTM As NEWTEXTMETRIC,_
    ByVal FontType As Long, LParam As ListBox) As Long
Dim FaceName As String
Dim FullName As String
    FaceName = StrConv(lpNLF.lfFaceName, vbUnicode)
    LParam.AddItem Left$(FaceName, InStr(FaceName, vbNullChar) - 1)
    EnumFontFamProc = 1
End Function
```

```
Sub FillListWithFonts(LB As ListBox)
Dim hDC As Long
    LB.Clear
    hDC = GetDC(LB.hWnd)
    EnumFontFamilies hDC, vbNullString, AddressOf EnumFontFamProc, LB
    ReleaseDC LB.hWnd, hDC
End Sub
```

And

语法：result = expression1 And expression2

功能：执行对两个表达式的逻辑“与”。

参数：expression1　必备，任何表达式。

expression2　必备，任何表达式。

result　必备，任何数值变量。

说明：若两个表达式的值都为真，结果为真；只要一个表达式的值为假，结果为假。结果的确定方法如下：

expression1	expression2	result
True	True	True
True	False	False
True	Null	Null
True	Null	Null
False	True	False
Null	True	Null
Null	False	False
Null	Null	Null
False	False	False

范例：下例演示如何使用此操作符。

```
Dim A, B, C, D, myCheck
A = 10: B = 8: C = 6: D = Null
myCheck = A > B And B > C    (真)
myCheck = B > A And B > C    (假)
myCheck = A > B And B > D    (空)
```

Comparison

语法：result = expression1 comparisonoperator expression2

或：

result = object1 Is object

或：

result = string Like pattern

功能：比较表达式的大小。

参数：Comparisonoperator 必备，任何比较操作符。

expression 必备，任何表达式。

Object 必备，任何对象名。

Pattern 必备，任何字符表达式或者字符范围。

result 必备，任何数值变量。

String 必备，任何字符表达式。

说明：若 expression1 和 expression2 都是变体表达式，则它们底层的数据类型决定了它们的比较方式。在 Single 型数据与 Double 型数据比较时，Double 型数据被取值到 Single 型数据的精度，若 Curency 型数据同 single 型数据或者 Double 型数据比较，则 single 型数据或者 Double 型数据被转换为 Currency 型。同样，若 Decimal 型数据同 Single 型数据或者 Double 型数据比较，则 Single 型数据或者 Double 型数据被转换为 Decimal 型。对于 Currency 型数据，小于 00001 的那部分数据丢失；对于 Decimal 型数据，则丢失小于 1-E-28 的那部分数据，否则产生溢出错误，这样的数据丢失可能导致两个数据在其不相等时得到“相等”的比较结果。

参照：option compare 语句，IS 操作符，Like 操作符，；= 操作符。

Eqv

语法：result = expression1 Eqv expression2

功能：执行对两个表达式的逻辑等价比较。

参数：expression1 必备，任何表达式。

expression2 必备，任何表达式。

result 必备，任何数据型变量。

说明：若有一个表达式为 Null，结果为 Null；若两个表达式都不为 Null，比较结果依据下表确定：

expression1, expression2, result	
True True True	False True False
True False False	False False True

范例：下例演示如何使用此操作符。

```
Dim A, B, C, D, myCheck
A = 10: B = 8: C = 6: D = Null
myCheck = A > B Eqv B > C
myCheck = B > A Eqv B > C
myCheck = A > B Eqv B > D
```

Imp

语法：result = expression1 imp expression2

功能：用来求两个表达式的逻辑蕴涵值。

参数：expression1 必备，任何表达式。

expression2 必备，任何表达式

result 必备，任何表达式。

说明：如何判断结果将依据下表。