

图 4.12 为 Layout 参数选择 ppLayoutTitle 常量

句，并将插入点置于 With 语句与 End With 语句之间的那一行上：

```
With stdTitleSlide
```

```
End With.
```

26. 随后，该宏将对幻灯片上这两个项目进行操作。为此，必须知道表示它们的对象。可以使用宏录制器查找，但是这次尝试一个更直接的方法：将插入点置于 With 语句之内的那一行上，再键入 .（一个句号），以显示对应于 Slide 对象的可用属性和方法的“属性/方法列表”下拉列表。
27. 有时候“属性/方法列表”下拉列表不能提供多大帮助，因为从中会找到很多可能有关的属性和方法，但是辨认不出所需要的属性。不过，如果扫描一下本例中这个表，就会发现，Shades 属性（它返回 Shades 集合）是唯一可能的项。
28. 按下 Ctrl+G 键，或选择“视图”>“立即窗口”，或单击“调试”工具栏上的“立即窗口”按钮，以显示“立即”窗口，来做一个小小的测试。
29. 将如下所示的测试语句键入“立即”窗口中，并按下 Enter 键。按下 Alt+F11 键，或单击“标准”工具栏上的“视图 Microsoft PowerPoint”按钮，以显示 PowerPoint 窗口，来证实 VBA 是否已选择了幻灯片上第一个 Shape 对象。这个测试语句是：

```
ActivePresentation.Slides(1).Shapes(1).Select
```

30. 确定是从这个对象开始的后，还需要知道如何将文本添加到这个形状。回到“代码”窗口（在“代码”窗口内单击，或按下 F7 键），按下 Backspace，以删除那个句号，再重新键入它，以显示出那个表。键入 te，向下跳到表中那些名字以 text 开头的项目上，在表中选择 TextFrame 项，然后键入一个句号，进入该术语并显示下一个表。用下滚方法找出 TextRange 对象，键入一个句号，进入该术语并显示下一个表。在这个表中，选出 Text 属性。键入一个等号以进入该术语。然后键入一个双引号，随后是指定给该文本属性的文本：Pollution Update：（后随一个空格），再一个双引号，一个 &，以及日期（由 Date 函数提供）：

```
Shapes(1).TextFrame.TextRange.Text = "Pollution Update: " & Date
```

31. 以同样方法，指定信息到第二个 Shape：

```
.Shapes(2).TextFrame.TextRange.Text = "JMP Industrials."
```

32. 整个过程看起来应如下所示：

```
Sub Add_Title_Slide()
    Dim sldTitleSlide As Slide
    Set sldTitleSlide = ActivePresentation.Slides.Add(Index:=1, _
        Layout:=ppLayoutTitle)
    With sldTitleSlide
        .Shapes(1).TextFrame.TextRange.Text = _
            "Pollution Update: " & Date
        .Shapes(2).TextFrame.TextRange.Text = _
            "JMP Industrials"
    End With
End Sub
```

33. 测试该过程；然后从演示文稿中删去所有幻灯片。选择“工具”>“自定义”，以便为Add\_Title\_Slide过程添加一个工具栏按钮或菜单项。
34. 在诸如Procedures.ppt之类的名称下保存演示文稿。
35. 创建一个新的演示文稿，然后测试对应于该过程的工具栏按钮或菜单项。在不保存更改的情况下关闭这个演示文稿。

## 第二部分 使用 VBA

- ◆ 第 5 章 VBA 的基本语法
- ◆ 第 6 章 了解变量、常量以及枚举常量
- ◆ 第 7 章 数组变量
- ◆ 第 8 章 寻找所需的对象、方法和属性

第 2 章

Word 里执行。宝剑计数器功能 Word 中 Visual Basic 基本语句，本章将学习 Word 中 VBA 语句。

Word 里执行命令。宝剑计数器功能 Word 中 Visual Basic 基本语句，本章将学习 Word 中 VBA 语句。

Word 里执行命令。宝剑计数器功能 Word 中 Visual Basic 基本语句，本章将学习 Word 中 VBA 语句。

Word 里执行命令。宝剑计数器功能 Word 中 Visual Basic 基本语句，本章将学习 Word 中 VBA 语句。

Word 里执行命令。宝剑计数器功能 Word 中 Visual Basic 基本语句，本章将学习 Word 中 VBA 语句。

Word 里执行命令。宝剑计数器功能 Word 中 Visual Basic 基本语句，本章将学习 Word 中 VBA 语句。

第 3 章

函数一个中 VBA，映射。字符串单双引号，字符串宝剑字符串，令串中一最基础的 VBA，串字符串一最“Hello”，映射。字符串宝剑的字符串字符串回显灯回显灯（串一串野狼）

该函数映射灯（串一串野狼）“Hello”，字符串个字符串这最回显灯回显灯（串一串野狼）。字符串个字符串这最回显灯回显灯（串一串野狼）

。野狼对回显来用字符串的显示。用字符串的显示字符串个字符串显示显示中显示

窗明立）。紫荆会否知道是否，中野狼对回显灯（串一串野狼）令串中 VBA 直视（。搬出不野狼 VBA 之前当字符串只字符串明立，长不。字符串野狼不以回显，代替字符串中口

。中字符串回显紫荆 VBA 逊其味无穷，搬工字符串回显灯，中央离群的字符串

。野狼干嘛没回显，搬工字符串

## 第 5 章 VBA 的基本语法

- ◆ VBA 基础
- ◆ 过程与函数
- ◆ 用及时窗口执行命令语句
- ◆ 对象、属性、方法、以及事件

本章介绍 VBA 的基本语法，根据前面章节中介绍的例子进行扩展。本章给出 VBA 的关键术语，以及在 Visual Basic 的编辑器中应用某些要素。

**注意：**本章涉及很多编程术语。如果有些地方不能理解，可以跳过，后面还有介绍。

### 准备

学习本章节，应按下面的步骤在 Word 中对 Visual Basic 编辑器进行设定。使用 Word 是因为该软件嵌入了 VBA 而且十分普及。若没有安装，可以直接阅读，不一定用计算机操作（大多数介绍适合任何 VBA 软件，尽管本章中涉及的许多命令是针对 Word 软件的）。

1. 正常启动 Word。
2. 按 Alt+F11 键进入 Visual Basic 编辑器，或选择工具宏（Visual Basic 编辑器）。
3. 安排 Word 窗口和 Visual Basic 编辑器窗口，保证同时可以见到两个窗口。若只有这两个窗口打开且没有最小化，右击任务栏在菜单中选择水平平铺或垂直平铺安排窗口。
4. 在编辑器窗口中按 Ctrl+G 键，选择视图立即窗口，或者单击调试工具栏上的立即窗口按钮显示立即窗口。

**提示：**若选择了多监视器，可设定一个监视器为 Word，另一个为 Visual Basic 编辑器。

### 过程

VBA 的过程是一串指令，有特定的名称，可以单独执行。例如，VBA 中有一个函数（过程的一种）叫做 Left，可以返回字符串左边的规定部分。例如，“Hello”是一个字符串，有 5 个字符。Left (“Hello”，4) 语句可以返回最左边的 4 个字符：“Hell”（可以在信息对话框中显示这 4 个字符或者在代码中使用）。过程的名称用来调用该过程。

所有 VBA 中可执行的指令必须包括在过程中，否则就不能执行或者会报错。（立即窗口中的语句除外，可以不在过程中执行。不过，立即窗口只能在当前的 VBA 条件下出现。）过程在包括模块中，而模块包括在工程、模板和其他 VBA 对象如用户窗体中。

过程有两种：函数和子过程。

## 函数

函数是两种过程的一种。函数是一个完整的过程，用来执行一项具体的任务。例如，Left 函数返回字符串左边的一部分字符；相应的，Right 函数返回字符串右边的一部分字符。每种函数都具有特定的功能，不能做别的事情。举个不恰当的例子，不能用 Left 函数打印文档或者使字符加粗，要完成这样的任务，必须使用相应的函数、方法和属性。

VBA 有许多内置函数，还可以生成新函数，将在后面介绍。生成的函数以 Function 语句开始，以 End Function 语句结束。

每个函数返回一个值。例如，Left 函数返回字符串左边的部分。其他函数带有测试条件，满足条件返回 True，否则返回 False。

## 子过程

子过程是一组自成体系的指令，不返回值。所有用宏录制器录制的宏都是子过程，还包括后面介绍的过程。

**注意：**“宏”对话框中只有子过程，没有函数。

## 语句

语句是指令单元，指明一个动作、定义一个项目、或者为内存变量赋值。VBA 往往一行指令为一条语句，然而，可以多条指令排成一行，用分号分开。（这不是好方法，代码读起来不方便。）

可以把一行指令分成两行，使代码容易读懂，用下划线连接“\_”（后面是回车）。这样做看起来方便，VBA 也仍然把两行或多行读成一行。

**警告：**不能把引号中的内容分行用下划线相接。若需要把引号中的内容分开，可先分成小段，再用符号“&”相连。

VBA 语句长度和复杂性都不相同，一条语句可以是一个单词（例如：Beep，让计算机发出声响，或者 Stop，停止 VBA 代码执行），也可以是很长的复杂句子，包括许多元素。可以看一下 Word 软件中 VBA 语句的构成，大多数使用了 ActiveDocument 对象，代表了 Word 软件当前的文本；有一些使用了 Documents 集合，代表了所有打开的文档（包括当前文档）；而且还使用了 Selection 对象，代表当前的选择。不要担心这些语句不能理解，你很快就可以掌握。

以下是语句的例子：

```
Documents.Open "c:\temp\sample Document.doc"  
MsgBox ActiveDocument.Name  
ActiveDocument.Words(1).Text = "工业"  
ActiveDocument.Close SaveChanges:=wdDoNotSaveChanges  
Documents.Add  
Selection.TypeText "一只敏捷的棕色的狐狸越过一只懒惰的狗"  
ActiveDocument.Save
```

```
Document.Close savechanges:=wdDoNotSaveChanges
```

```
Application.Quit
```

现在来逐句认识一下这些语句。

```
Documents.Open "c:\temp\Sample Document.doc"
```

该语句使用了 Documents 集合中的 Open 方法打开一个特定的文档。在例子中为 sample Document. doc。把该语句输入到立即窗口，使用计算机中存在的路径和文档。按 Enter 键后，VBA 可以打开 Word 窗口中的文档。若再使用 Word 软件打开一个文档，该文档就变成当前文档（当前选择的文档）。

```
MsgBox ActiveDocument.Name
```

该语句使用了 MsgBox 函数，显示 ActiveDocument 对象的 Name 属性（本例中为 sample Document. doc）。在立即窗口中输入该语句（用小写并使用 VBA 帮助功能），然后按 Enter 键。VBA 在 Word 窗口中显示信息框。按“确定”按钮可取消信息框。

```
ActiveDocument.Words(1).Text = "工业"
```

该语句使用了赋值符号（=）把“工业”一词输入到第一项的 Text 属性中，该属性属于当前 ActiveDocument 对象的 Words 结合。在立即窗口中输入该语句，并按 Enter 键，Word 软件把“工业”（当前的格式一般不会为粗体）一词输入到打开文档的起始位置。注意插入点，按输入单词考虑，应该在单词的开头而不是末尾，这是因为 VBA 处理文档的属性，而不是打印输入。

```
ActiveDocument.Close SaveChanges:=wdDoNotSaveChanges
```

该语句使用了 Close 方法关闭 ActiveDocument 对象，使用了参数 SaveChange，控制 Word 是否在关闭文档时保存该文档（如果该文档含有未保存的修改）。本例中，使用了常量 wdDoNotSaveChanges，指明 Word 在关闭文档时不需要保存。在立即窗口中输入该语句可以见到 VBA 让 Word 关闭了该文档。

```
Documents.Add
```

现在，在立即窗口中，输入语句 Documents. Add，该语句在 Documents 集合中使用了 Add 方法，在 Documents 集合中生成一个新的 Document 对象，换句话说，生成了一个新的文档。因为该语句未指明使用的模板，新的文档沿用默认模板（Normal. dot）。在立即窗口中输入该语句，并按 Enter 键，Word 生成了一个新的文档，和往常一样，该新文档成为当前文档。

```
Selection.TypeText "一只敏捷的棕色的狐狸越过一只懒惰的狗"
```

该语句使用了 Selection 对象的 TypeText 方法，在当前文档的插入点输入一句话（Selection 对象表示为当前的选择，可以为不存在的选择或者一个或多个选择对象）。如果，在活动文档中选择了一句话，该选择就会像通常那样被覆盖（除非选项对话框的编辑页面中用键入内容替换所选内容的一栏被清除）。因为刚刚生成了一个新文档，没有选择的项目。在立即窗口中，输入该语句，并按 Enter 键，Word 输入这句话。注意这次的插入点在输入内容后结束；Selection 对象的 TypeText 方法类似于手工输入。

**ActiveDocument.Save**

该语句使用了 Save 方法（命令）保存 ActiveDocument 对象。该语句等同于手工操作中选择“文件”>“保存”。若在立即窗口中，输入该语句，并按 Enter 键，Word 显示“另存为”对话框，可以按通常方法保存该文档。不过现在单击“取消”按钮，消除“另存为”对话框。Word 显示微软公司的 VB 错误提示框（如图 5.1 所示）。单击“确定”按钮，取消对话框。在第 17 章中将介绍如何处理类似的错误。

**Documents.Close SaveChanges:=wdDoNotSaveChanges**

该语句类似于前面的 ActiveDocument.Close SaveChanges:=wdDoNotSaveChanges，除了针对 Documents 集合而不是 ActiveDocument 对象之外，Documents 集合代表所有当前打开的文档，该语句关闭了所有的文档，而且不保存文档中的改变。在立即窗口中，输入该语句，并按 Enter 键，Word 将关闭所有的文档。

**Application.Quit**

该语句在 Application 对象中使用了 Quit 方法，退出 Word 软件，在立即窗口中，输入该语句，并按 Enter 键，Word 将自己关闭，而且关闭 Visual Basic 编辑器。因为 Word 是 Visual Basic 编辑器的宿主软件。

**使用 VBA 的帮助**

Visual Basic 编辑器为 VBA 提供了复杂的帮助。要了解帮助，可选择“帮助”>“Microsoft Visual Basic”帮助。大多数语句和函数都有帮助，在生成代码和查找错误时特别有用。

VB 帮助文件使用了一些约定，在使用之前应当了解。

- ◆ 斜体表示变量或可改变的值。
- ◆ [ ] 表示可选参数。

如果使用的计算机不能提供 VBA 的帮助，该帮助文件就没有安装（也许为了节省空间），只需安装该文件即可。

**关键词**

关键词是 VBA 的一部分。举例如下：

- ◆ Sub 关键词表示子程序的开始，End Sub 是子程序的结束。
- ◆ Function 关键词是函数的开始，End Function 是函数的结束。
- ◆ Dim 关键词用来定义一个变量，As 关键词决定定义的类型。类型也是关键词。例如：在定义中，Dim strExample As String 有三个关键词：Dim、As 和 String。

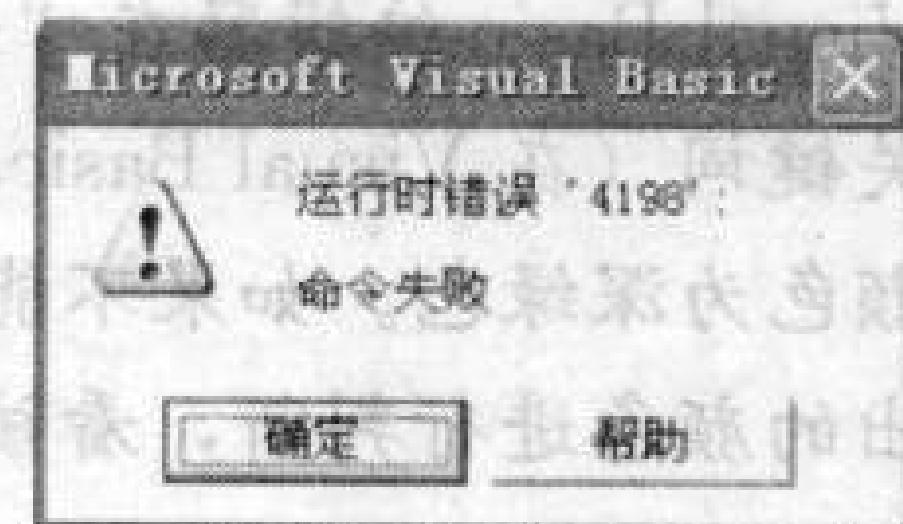


图 5.1 Visual Basic 编辑器遭遇不能处理的错误时，显示 Microsoft VB 错误信息框

函数和子过程的名称不属于关键词。

**提示：**Visual Basic 编辑器在“选项”对话框的编辑器格式中用颜色显示出所有的关键词（在 Visual Basic 编辑器中选择“工具”>“选项”）。关键词默认的颜色为深绿色。如果不能确定是否为关键词，可根据 Visual Basic 编辑器给出的颜色进行判断，看该颜色是否和 sub 相同。

## 表达式

表达式包括关键词、运算符、变量以及常量，是字符串、数字或对象的组合。例如：使用表达式，执行一个运算，或者比较两个变量。

## 运算符

运算符是用来比较，组合，或者在表达式中与数值一起使用的。VBA 有 4 种运算符：

- ◆ 算术运算符（例如：+ 和 -）用来进行数字运算
- ◆ 比较运算符（例如：< 和 >，分别表示小于和大于）用来比较大小
- ◆ 连接运算符（& 和 +）连接两个字符串

◆ 逻辑运算符（例如：and, not, all）用于逻辑结构

在第 6 章可以见到，各种运算符的用法。

## 变量

变量是内存中的一个区域，用来存储在程序运行时可以更改的信息（可以想象为内存中可以改变空间的区域）。例如，如果需要用户在输入框或对话框中输入姓名，姓名就存入变量中，然后在程序中使用。

VBA 中使用几种变量，包括：

- ◆ Strings，存储字符或字符串。
- ◆ Integers，存储数字（没有小数的数字）。
- ◆ Objects，存储对象。
- ◆ Variants，存储所有类型的数据。不定性是变量的默认类型。

可以让 VBA 生成不定变量保存信息，或者可以规定变量的类型。规定变量类型的好处将在适当的时候给出。

现在，在立即窗口中生成变量。输入下列内容并按 Enter 键：

```
myVariable = "Sample variable text"
```

这什么也看不到，但是 VBA 已经生成了 myVariable 变量。输入下面的语句并按 Enter 键：

```
MsgBox myVariable
```

现在可以见到结果：一个信息框包括了输入的变量。

变量可以定义为显式或者隐式，显式变量要求在使用前对变量名称进行声明，往往还要

求给出类型。隐式变量在存储数据时进行声明。VBA 把数据保存在不定变量中。

下面的章节介绍如何使用隐式变量，不需要对变量进行任何声明，因为 VBA 在需要的时候，自动生成。隐式变量会使代码运行得更快，更容易读懂。

## 常量

常量是在程序中的常数值。常量的值在程序执行的过程中不会改变。

VBA 使用两种常量：内置常量，指软件中固有的常量和用户定义的常量。例如：内置常量 vbOKCancel 用来在 MsgBox 函数中生成信息框，包括“确定”按钮和“取消”按钮。可以定义一个常量存储不会改变的信息，例如，过程的名称。

## 参数

参数是一段信息，由常量、变量和表达式给出，并向过程函数或方法提供。有些参数是必要的，有些是可选的。如前面介绍的那样，下面的语句使用了可选的参数。

SaveChanges，决定在文档关闭时是否保存未保存的更改。

```
ActiveDocument.Close SaveChanges:=wdDoNotSaveChanges
```

Visual Basic 编辑器的帮助提示和 VB 的帮助文件给出了函数、过程或方法的参数清单，包括括号中的所有可选参数。若使用自动快速信息功能，键入函数、过程或方法的名称后加一空格，VisualBasic 编辑器将给出参数列表。

图 5.2 给出了 Open 方法的参数列表。FileName 参数是必选的，所以没有括号。所有其他的参数（ConfirmConversions，ReadOnly，AddToRecentFiles，等等）是可选的，所以使用了括号。若不给出可选参数，VBA 将使用默认的参数值（要了解默认的参数值，参阅 VBA 帮助文件）。Visual Basic 编辑器用粗体指明列表中当前的参数；每个参数输入后下一个参数变为粗体。

```
documents.Open
    Open (FileName, [ConfirmConversions], [ReadOnly], [AddToRecentFiles], [PasswordDocument],
    [PasswordTemplate], [Revert], [WritePasswordDocument], [WritePasswordTemplate], [Format], [Encoding],
    [Visible], [OpenAndRepair], [DocumentDirection], [NoEncodingDialog], [XMLTransform]) As Document
```

图 5.2 可以根据 Visual Basic 编辑器显示的信息判断参数是必须的还是可选的，可选的参数括在方括号中

## 指明参数的名称和忽略参数的名称

使用参数可以采取下面两种方法。

- ◆ 输入参数的名称（例如：ConfirmConversions），后面加上冒号和等号（ConfirmConversions:=）以及常数或数值（ConfirmConversions:=True）。例如，语句如下：

```
Documents.Open FileName:="c:\temp\Example.doc",
    ConfirmConversions:=True, ReadOnly:=False
```

- ◆ 在参数列表的相应位置输入常数或值，而不输入常数的名称。前面的语句表达如下：

```
Documents.Open "c:\Temp\Example.doc", True, False
```

若使用了参数的名称，参数不需要按顺序排列，因为 VBA 根据名称进行判断。下面的语句功能相同：

```
Documents.Open ReadOnly:=False, FileName:="c:\temp\Example.doc", _  
ReadOnly:=False, ConfirmConversions:=True
```

```
Documents.Open FileName:="c:\temp\Example.doc", _  
ConfirmConversions:=True, ReadOnly:=False
```

不需要指明省略了哪一个参数。另一种情况，如果省略了参数的名称，而给出了参数，该参数必须按照 VBA 认可的顺序排列。若选择不使用某个可选参数，而使用下一个参数，要用逗号指明省略的参数。例如下面的参数省略了 ConfirmCoversions 参数，使用逗号指明 False 值与 ReadOnly 参数相关而不是和 ConfirmCoversions 参数相关。

```
Documents.Open "c:\temp\Example.doc",, False
```

在代码窗口或立即窗口中输入逗号时，自动快速信息将粗体移向参数列表中的下一个参数，以表明进入下一项。

**注意：**一般来说，必选参数在可选参数之前给出，因此不需要指明省略的可选参数。

### 何时需要对参数使用括号

当把函数的结果赋予变量或其他对象时，需要把所有的参数放在括号中，例如，对变量 objMyDocument 赋予打开文档的结果，c:\temp\Example.doc，应使用下面的语句。

```
objMyDocument = Documents.Open(FileName:="c:\temp\Example.doc", _  
ConfirmConversions:=True, ReadOnly:=False)
```

如果不把运行的结果赋予变量或对象时，就不需要对参数使用括号。

## 对象

每一种软件包括一系列的对象。举例如下：

- ◆ 在 Word 中，文档是一个对象（Document 对象），段落（Paragraph 对象）或表（Table 对象）甚至一个字符也是对象（Character 对象）。
- ◆ 在 Excel 中，工作簿是对象（Workbook 对象），工作表（Worksheet 对象）、图表（Chart 对象）也是对象。
- ◆ 在 PowerPoint 中演示文稿是对象（Presentation 对象），幻灯片（Slide 对象）和图形（Shape 对象）也是对象。

VBA 中的大多数指令是处理对象的。例如，前面见过的在 Word 中关闭当前文件，使用了 ActiveDocument 对象的 Close 方法：

```
ActiveDocument.Close
```

## 集合

集合是包括其他对象的对象。集合提供了同时访问所有成员的方法。例如：Documents 对象，包括所有打开的文档，每一个都是对象。不需要一个一个地关闭 Document 对象，可使用 Documents 集合的 Close 方法，关闭所有的文档。

### Documents.Close

类似地，可以使用集合同时改变所有成员的属性。

## 属性

每个对象都有若干属性。例如，Word 中的文档，有标题属性、主题属性和作者属性。可以使用属性框设定这些属性。文件（属性）中一个单个字符有若干属性，例如：字体、字号以及其他的要求（粗体，斜体，删除线）。

## 方法

方法是针对对象的动作。不严格地说，方法是命令。不同的对象具有不同的方法，可以实施的动作或命令。例如，下面的方法和 Word 中的 Document 对象有关（同时也和 Excel 中的 Workbook 对象，以及 PowerPoint 中的 Presentation 对象有关）。

Activate，激活文档（等同于用键盘或鼠标选择文档的窗口）。

Close，关闭文档（等同于选择“文件”>“关闭”）。

Save，保存文档（等同于选择“文件”>“保存”）。

Save As，以一个特定的名称保存文档（等同于选择“文件”>“另存为”），并且在“另存为”对话框中给出文件名及路径。

## 事件

事件是 VBA 认可的，发生的事件。例如，典型地说，打开文件产生一个事件（无论是用户或程序）。用户在用户窗体中单击按键，即产生一个事件。

将代码绑定于事件，可以在事件发生时执行一个命令。例如，在用户窗体中，代码可以检查所有用户窗体的设定，用户单击“确定”按钮，取消用户窗体，并且应用设定。

合集

## 第 6 章 了解变量、常量以及枚举常量

- ◆ 什么是变量以及怎样使用
- ◆ 生成和使用变量
- ◆ 设定变量的范围和有效时间
- ◆ 常量
- ◆ 枚举常量

本章介绍变量、常量、以及枚举的基本知识。变量将存储和处理程序中的信息，包括存储文本的字符变量、存储数字的各种数字型变量（例如，整型）、存储时间和日期的日期变量、存储 True/False 值的逻辑变量，以及存储对象的对象变量。常量经过命名，在程序执行中数值保持不变。枚举是预先给定的特殊整数，在具体场合中具有自己的名称和含义，基本上说是一个常量列表。

数组变量未在本章中讨论，数组用来同时保存多个信息，数组变量在第 7 章介绍。

### 使用变量

变量是在内存中命名的区域，用来在程序运行时保存信息。例如，第 5 章中用一个变量保存了简单的文本字符串，通过信息框显示出来：

```
myVariable = "Sample variable text"
MsgBox myVariable
```

第一条语句在内存中开辟一个区域，命名为 myVariable，并且把字符串 Sample variable text 送给该区域。第二条语句将 myVariable 的内容从内存中取出用 MsgBox 函数在信息框中显示出来。myVariable 的内容仍然保存在内存中，若有必要可再次使用。

### 变量命名的方法

VBA 对变量的名称有一些限制：

- ◆ 变量的名称必须以字母开始，长度可达到 255 个字符。往往名称比较短，输入时较为容易，这样每行的代码不会太长。

**提示：**Visual Basic 编辑器的自动完成功能可使较长的变量名称变得容易管理：输入变量的名称达到一定的长度，使其能够区别于关键词和其他变量名称，然后按 Ctrl+空格键。如果输入的名称已经可以唯一判定出，Visual Basic 编辑器会插入该名称；否则 Visual Basic 编辑器将显示一个下拉清单，给出以那些字母开头的所有关键词和名称。

◆ 变量名称不能包括句号、感叹号、数学运算符（+、-、\*、/）或者比较运算符（=、<>、>、>=、<、<=），也不能在名称内部包括定义类型声明的字母（@、&、\$、#），后面介绍类型声明字母。

◆ 变量名称不能包括空格，但可以有下划线，用下划线可以使变量名称容易读懂。

换句话说，直接使用字母间隔采用下划线是很安全的命名方法。

例如，所有下列的变量名称都是可行的，尽管最后一个变量太长。

i

John

MyVariable

MissionParameters

The\_String\_That\_the\_User\_Entered\_in\_the\_Input\_Box

相反，以下变量名称是不能使用的：

变量名称	问题
My Variable	含有空格
My! Variable	含有感叹号
Time@Tide	含有类型声明字符
1_String	没有以字母开头

每一个变量名称在其运行的范围之内必须是唯一的（保证 VBA 不会和其他变量混淆）。一般说来，变量运行的范围就是一个过程，然而，如果变量定义为公共型或者为模块级（本章后面讨论），范围就扩大了。

对变量名称的其他限制是不得使用 VBA 已经用于函数、语句或方法的名称。这样做是对 VBA 关键词的“屏蔽”。不一定会出现问题，然而可能有关的函数、语句或方法就不能使用，除非针对 VBA 进行特殊设定，把 VBA 加在名称的前面。例如，不能使用 Date，必须写成 VBA.Date。虽然不是特别重要，却应当一开始就避免。

不应当屏蔽 VBA 关键词，然而 VBA 的关键词相当多，很容易就会碰上。

## 声明变量

变量可以声明为隐式的或者显式的。下面很快就会介绍，两种方法各有优缺点。显式声明往往总是个好办法，一旦使用了 VBA，变量可能会一直用下去。因此，最好是一开始就把变量定义为显式。本章也将介绍隐式声明，并将给出使用的方法。

### 声明隐式变量

隐式变量就是说不经过声明就在代码中使用。在声明隐式变量时，VBA 会检查以保证没有相同名称的变量在使用。然后自动生成一个变量并且定义为不定型，可以存储所有类型的信息，不包括长度固定的字符串。

例如，在前一章用下面隐式声明定义了 myVariable：

```
myVariable = "Sample variable text"
```

在这里，myVariable 是隐式声明的变量。VBA 将其设定为不定型，该类型有许多子类型。在本例中，变量的子类型为字符，因为其中含文本。VBA 在生成一个变量时通常把变量值设定为空（一个特殊的值用来表示不定型变量尚未被使用）。不过，本例的变量一开始

就得到赋值(因为得到了文本字符串)。

声明隐式变量的优点是使用前不需要写代码。若需要一个变量随时声明即可。然而，声明隐式变量也有两个缺点：

- ◆ 在过程中，再次使用隐式变量容易出现错误。例如，如果先声明了隐式变量 FilesToCreate，但后来却输入成 FllesToCreate。VBA 不能识别后面的拼写，将会生成另一个变量，新的变量会有一个不同的值。如果使用许多变量，发现这样的错误，既困难又费时，程序就会出现问题。
- ◆ 不定型变量比其他类型的变量占用更多的内存空间，因为需要保存各种类型的数据。在多数情况下，这样的差异是可以忽略的，特别是如果仅仅使用了几个变量或者过程很短。然而，如果计算机的内存空间有限，却使用了很多变量，不定型变量占用过多的内存，可能降低过程的运算速度，甚至有可能造成计算机内存不足。对于配置较低的计算机来说，处理不定型数据需要更长的时间，这使 VBA 必须始终检测变量中的数据类型。

可以避免第二个缺点：可以在声明隐式变量时使用类型声明字符以规定数据的类型，或者按下面的介绍，强制使用显式声明。

类型声明字符是在隐式声明中加在变量名称末尾的字符，用来指明变量的类型。表 6.1 列出了类型声明字符。

表 6.1 类型声明字符

使用变量	字符	变量的数据类型	举例
	%	整型	Quantity%
	&	长整型	China&
	@	货币型	Profits@
	!	单精度	Temperature!
	#	双精度	Differential#
	\$	字符型(变量长度)	myMessage\$

因此，可以用下面的语句定义显式声明的字符变量 UserName，将 Jane Magnolia 赋值给变量：

```
UserName$ = "Jane Magnolia"
```

可以用下面语句隐式声明货币型变量 Price：

```
Price@ = Cost * Margin
```

只有在声明变量时才使用类型声明字符。因此，可根据前面例子中的名称 UserName 和 Price 参照变量。

### 声明显式变量

声明显式变量意味着告诉 VBA，变量在使用前已经存在。VBA 为变量分配内存空间，而且，按已知空间大小注册变量。可以同时声明变量类型，这是个好办法，但不一定非做不可。

可以在使用前的任何地方声明显式变量，然而，较好的方法是在过程的开始声明变量。这样做很容易查找，有助于阅读代码。定义显式变量具有下列优点：

- ◆ 代码容易阅读和调试。如果代码很复杂，这样考虑很重要。
- ◆ 如果使用了 Option Explicit 语句强制显式声明变量，就不可能因为变量名称拼写错误而无意中生成新的变量。
- ◆ 同样的，在生成新的变量时就不会在无意中清除变量的内容。
- ◆ VBA 可以在设计阶段或者编译阶段发现一些打印错误，而隐式声明只有在运行阶段才能发现。

**注意：**数据输入错误指把错误类型的信息赋值给变量。例如，变量定义为整型，赋值却为字符型，VBA 将给出错误信息，因为不能把字符信息保存在整型变量中。

- ◆ 代码处理小数时较快，是因为 VBA 不需要在代码运行时鉴别变量的类型。

声明显式变量的缺点是需要多花一些时间、精力和思考。然而，对大多数代码来说，缺点远小于优点。

声明显式变量需要使用下面的关键字：Dim、Private、Public、或者 Static。

例如，下面的语句声明了变量 MyValue：

```
Dim MyValue
```

Dim 是普通关键词，用来声明变量，大多数变量声明都使用该单词。其他关键词在声明时用来确定变量的不同范围、有效时间，以及数据类型。在前面的例子中，MyValue 变量采用了默认的范围、有效时间和数据类型，可以满足一般意义上的使用。

可以在同一行中用逗号将变量隔开，同时声明多个变量：

```
Dim Supervisor As String, ControllerCode As Long
```

这样做有助于减少代码中的声明行，然而，这使得声明不容易看懂，因此，通常不是个好办法。

必须指出，在同一行中声明多个变量应当对每一个变量指明数据类型，正如上面的例子。也许为了省事，针对多个字符型变量会采用下面的方法：

```
Dim strManager, strReportingEmployee As String
```

该语句不能创建两个字符变量；strReportingEmployee 是个字符变量，但 StrManager 是不定型变量，因为 As String 只是 strReportingEmployee 的一部分。

## 变量的范围和有效时间

变量的范围指 VBA 中运行的区域。一般来说，变量采用默认的范围：即声明变量的过程当中（无论隐式声明还是显式声明）。例如，有个名称为 Financial\_Procedures 的模块包括两个过程 Breakeven\_Table 和 Profit\_Analysis\_Table，每个过程都使用了名称为 Gross\_Revenue 的变量和另一个名称为 Expenses 的变量。在一个过程中的变量不同于另一个过程中的变量，VBA 不会将两者混淆。（尽管对于人来说不同的过程使用相同的变量名称容易在调试时产生混乱。一般说来，最好使用唯一的变量名称，即使在默认的过程级别中。）

## 要求变量显式声明

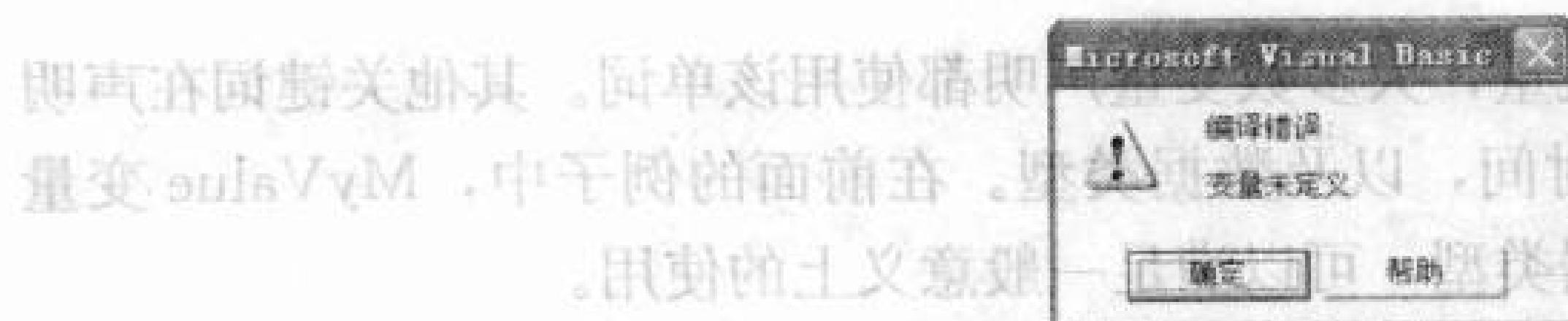
可以全方位地（所有模块）或者按模块让 VBA 要求变量显式声明。许多人都认为该功

能很有用，因为可以防止对变量隐式声明，无论是有意识的还是其他情况。全方位地要求变量声明，应在“Visual Basic 编辑器”中选择“工具”>“选项”，显示“选项”对话框，单击“编辑器”选项卡，显示“编辑器”页，将要求变量声明的复选框选中，然后选择“确定”按钮。（要求变量声明复选框默认为清除状态，允许使用隐式变量，这样在初学阶段比较容易使用变量。）这样，Visual Basic 编辑器在每一个新产生的模块中都会加上 Option Explicit 语句。该语句对所有的模块要求显式变量声明。

如果选用了要求变量声明复选框，“Visual Basic 编辑器”不会对已经存在的模块加入 Option Explicit 语句。如果也希望强制变量声明，必须对已经存在的模块手工加入 Option Explicit 语句。

若只是对个别模块需要变量声明，将 Option Explicit 语句放在需要变量声明的模块的开头。在模块中，Option Explicit 语句必须出现在第一个过程的 Sub 或者 Function 语句之前。如果放在过程内部或者过程之间，在该模块中运行代码时 VBA 将报错。

无论 Option Explicit 是设定为全局的还是针对某个模块的，VBA 在运行之前都将对它进行测试。准确地说，如果在编译代码时发现有未经声明的变量，VBA 会发出如下图所示的警告。同时，VBA 会将代码中的变量高亮度显示。



如果见到这样的信息框，可以对变量进行声明或者取消需要变量声明的要求。要取消要求，可将模块中的 Option Explicit 语句删除，或者把该行注释掉。

变量的有效时间是指 VBA 记忆变量值的时段。不同用途的变量的有效时间也不同，变量的有效时间和变量的范围有关。

有时候需要从过程的外部访问一个变量。在这种情况下，就需要声明不同范围的变量。

变量可以有三种范围：

- ◆ 过程的
- ◆ 私有的
- ◆ 公共的

### 过程范围

过程范围的变量（也可看做过程级别的）只对包括该变量的过程有效。过程变量的有效期限定在声明该变量的过程之内，过程一旦停止运行，VBA 将所有变量从内存中移开，重新释放内存。

过程范围针对所有在过程中运行的变量。例如，隐式声明名称为 Supervisor 的不定型的变量：  
**Supervisor = "Paul Smith"**

可以在该过程的其他场合使用 Supervisor 变量。例如，调用存入的文本或者改变文本的内容。当过程停止运行，VBA 将移除变量，重新释放占用的内存。

**注意：**变量隐式声明时，自动设定为过程范围。

为了显式声明一个局部变量，应使用 Dim 关键词，在过程中，按下列方法放置声明：

```
Sub Create_Weekly_Report()
    Dim strSupervisor As String
    Dim lngController As Long
    ...
End Sub
```

在这里，第 2 行声明变量 strSupervisor 为字符型，第 3 行声明变量 lngController 为长整型，第 4 行声明变量 intReportNonber 为整型。（本章节后面“设定变量的数据类型”一节专门讨论变量类型。）

另一方面，若想把变量从当前过程传递给另一个过程，过程范围是不能满足要求的，必须使用私有范围或公共范围。

### 私有范围

私有范围变量对该模块中的所有过程均有效，不包括其他模块的过程。使用私有变量，可以将变量值从一个模块传递到另一个模块。不同于局部变量，仅仅保留在运行的模块当中，私有变量保存变量值直到本工程关闭，定义变量为私有范围，应当在模块起始部分使用 Dim 关键词，或者 Private 关键词，在模块中放置在第一个过程的 Sub 语句之前：

```
Dim strSupervisor As String
Private blnConsultantAssigned As Boolean
Sub Assign_Personnel()
```

Visual Basic 编辑器在分隔线上显示私有声明变量，该分隔线将声明区和代码分开（如图 6.1 所示）。

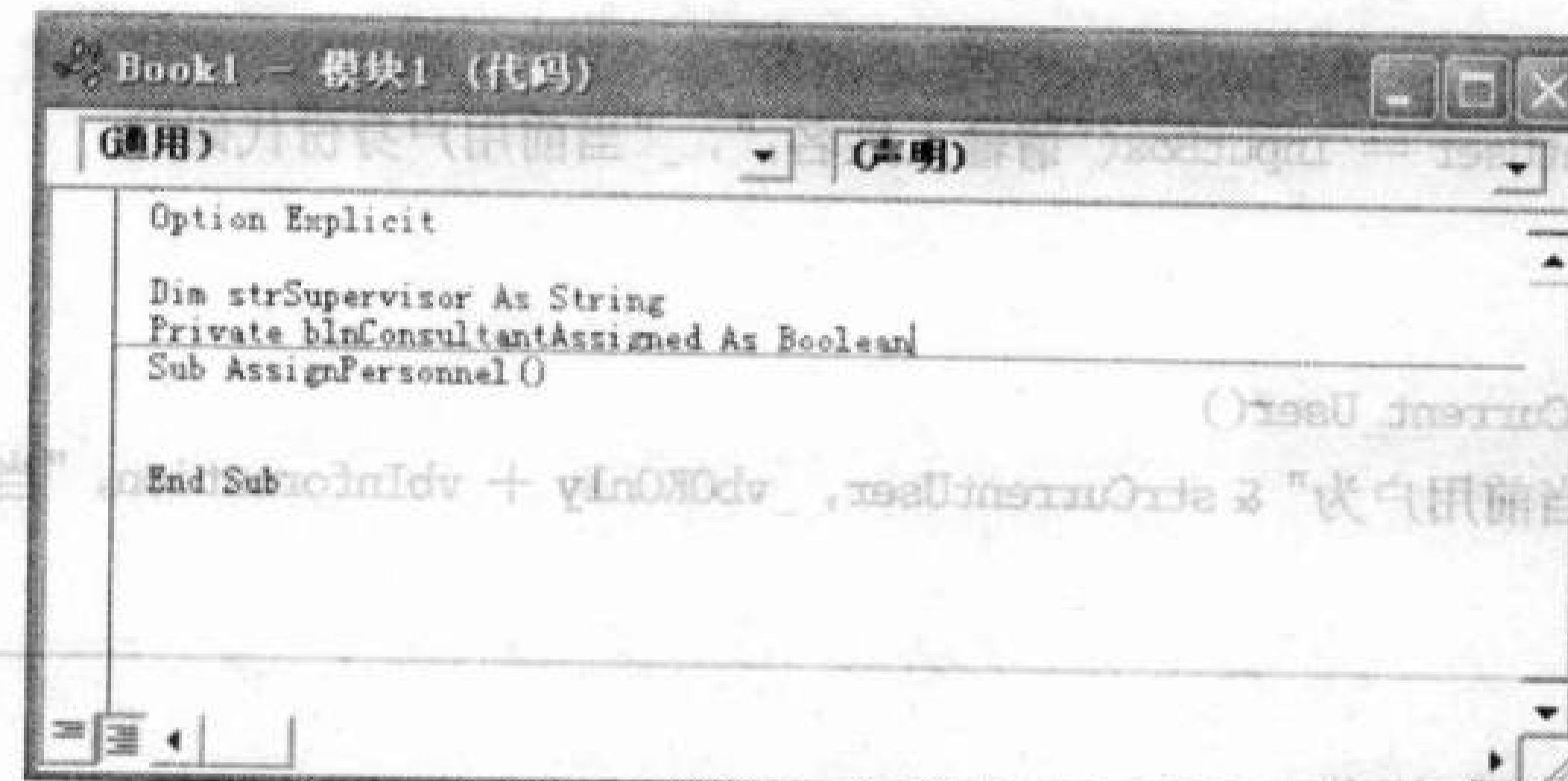


图 6.1 私有变量声明出现在声明区中

可以见到 Dim 语句使用了同样的语法，和前面的过程变量的声明方法相同。不同之处在于声明私有变量时，语句放在声明区而不是在过程内部。因为 Private 语句在声明私有变量时和 Dim 语句有同样的作用，而且不能在过程内部使用，因此，定义私有变量使用 Private 语句比使用 Dim 语句更加明确。

**警告：**在 Visual Basic 编辑器中编辑了过程之后，Visual Basic 编辑器重新编译代码时，私有变量和公共变量会重新设定（它们的值被清除）。如果测试使用私有变量和公共变量编辑的过程之后，必须重新初始化（重新赋值）变量。

公共范围的变量在包括该变量的工程中，对所有模块的所有过程均有效。定义公共变量

必须使用 Public 关键词，必须在模块开始的声明区使用 Public 关键词，放在模块的第一个过程的 Sub 语句之前。例如，

```
Option Explicit
Public intMyVar As Integer
```

第 2 条语句声明公共变量 intMyVar 为整型变量。

**注意：** 声明区出现在每一个模块的起始位置。例如，使用显式对象声明（在“选项”对话框的“编辑器”页上选择要求变量声明复选框），Visual Basic 编辑器对每一个生成的新模块起始位置输入 Option Explicit 声明。如果不是这样，手工输入语句后，Visual Basic 编辑器会产生声明区。

像私有变量一样，公有变量保存变量值直到工程关闭。例如，要在 Word 软件中通过一系列的操作跟踪用户的名称，可以生成一个 AutoExec 过程，在 Word 起动时提醒用户输入用户名（AutoExec 是一个过程的内置名称，在 Word 起动时自动运行）。在公共变量中保存输入结果，此后，可以在同一个 Word 中调用。程序清单 6.1 给出了 AutoExec 过程使用公共变量的例子。

当 Visual Basic 编辑器重新编译代码时，公共变量被重新设定，因此，编辑代码后必须重新初始化变量。

### 程序清单 6.1

```
1. Public strCurrentUser As String
2.
3. Sub Autoexec()
4.     strCurrentUser = InputBox("请输入姓名：", "当前用户身份代码")
5. End Sub
6.
7. Sub Identify_Current_User()
8.     MsgBox "当前用户为" & strCurrentUser, vbOKOnly + vbInformation, "当前用户"
9. End Sub
```

以上代码包括三个部分：

- ◆ 第 1 行，声明了公共字符型变量 Public strCurrentUser As String。
- ◆ 第 3 行到第 5 行包括 Autoexec 过程。该过程在每次启动 Word 时执行。第 4 行显示输入对话框，要求用户输入姓名，而且将回答保存在公共变量 strCurrentUser 中。
- ◆ 第 7 行到第 9 行是 Identify\_Current\_User 过程，用来显示一个信息框，给出用户的姓名，以及文本、信息图标、和标题栏。

对该过程进行测试，在 Visual Basic 编辑器中，使用 F8 键，首先进入 Autoexec 过程，然后进入 Identify\_Current\_User 过程。然而，为了观察运行效果，必须生成这些过程，然后退出 Word。当重新启动 Word 时，Autoexec 过程显示输入框，要求输入姓名。此后，在任何时候（直到退出 Word）都可以运行 Identify\_Current\_User 过程，VBA 显示信息框会给出用户输入的姓名。

**警告：**因为公共变量在过程运行时，仍然保留变量值，所以仍然在内存中占有空间。如果大量使用公共变量，可能会造成内存不足，或者在计算机内存有限的情况下出现过多的文件交换。

## 使用静态变量

除了 Dim、Private、和 Public 之外，还有 Static 关键词，可用来声明静态变量。静态变量的值在调用该变量的过程中得以保存。静态变量类似于公共变量，它们的有效时间不限定在声明的过程中。区别在于，静态变量一旦声明，仅仅对声明该变量的过程有效，而公共变量一旦被声明，对所有的过程均有效。

静态变量对保存信息十分有用，有的过程在应用时需要反复运行。要么保存一个总数（例如：记录执行一个过程的次数，或者在多次运行一个过程时，需要保存一条有用的信息。）

以下语句声明了静态字符变量 strSearchTerm1：

```
Static strSearchTerm1 As String
```

注意，像公共变量一样，静态变量一旦生成，就占有内存空间，因此除非必要，不应使用静态变量。

## 设定变量的数据类型

表 6.2 说明 VBA 支持的数据类型，以及每一种类型需要的内存空间。

表 6.2 VBA 支持的数据类型和需要的内存空间

变量	简介	要求内存
Boolean	True 或者 False	2 字节
Byte	从 0 到 255 的整数	1 字节
Currency	15 位的正数或者负数，保留 4 位小数	8 字节
Date	小数点左面为日期的浮点数右边为时间	8 字节
Decimal	无正负的十进制整数	12 字节
Double	浮点数、负数值从 $-1.79769323486232^{308}$ 到 $-4.94065645841247^{-324}$ 正数值从 $4.94065645841247^{-324}$ 到 $1.79769323486232^{308}$	8 字节
Integer	整数从 -32768 到 32767	2 字节
Long	整数从 -2147483648 到 2147483647	4 字节
Object	对象变量	4 字节
Single	浮点数，负数值从 $-3.40282^{38}$ 到 $-1.401298^{-45}$ ；正数值从 $1.401298^{-45}$ 到 $3.40282^{38}$	4 字节
String	文本的字符串，可变长度或固定长度	可变长度字符串为 10 字节，固定长度字符串长度即为存储空间，含有数字的不定型：16 字节
Variant	任何类型的数据，不包括固定长度的字符串	含有字符的不定型：22 字节

以下详细讨论数据类型。

## 是否需要设定数据类型

对每一个生成的变量设定数据类型是个好办法，但并非必须这样做。几乎可以全部使用

默认的不定型数据类型（如前面做过的那样），VBA 可以判断出不定型变量的子类型。

使用不定型数据有三个缺点：

- ◆ 首先，不定型变量比其他变量占用较多的内存，较长的文本型变量除外。
- ◆ 其次，使用不定型变量使代码运行速度较慢。然而，过程较短（或者过程较长但使用变量很少）。内存和速度不是问题。
- ◆ 其三，代码不容易阅读和调试。这是个值得关注的问题。

第 18 章讨论如何优化代码，以及设定变量类型的利弊。

### Boolean（逻辑型）

逻辑型变量可以设定成 True（真）或者 False（假）。可使用关键字 True 或者 False 设定逻辑型变量的值。如下面第 2 行（第 1 行声明了逻辑型变量 blnProduct\_Available）：

```
Dim blnProduct_Available As Boolean  
blnProduct_Available = True
```

可以将逻辑型变量的结果取出，然后执行不同的命令：

```
If blnProduct_Available = True Then  
    MsgBox "该产品有货"  
Else  
    MsgBox "该产品无货"  
End If
```

如果把逻辑型变量转换成其他数据类型（例如数字型），True 返回 -1、False 返回 0。如果把数字变量转变成逻辑变量，0 返回 False，所有其他数字（无论正负）返回 True。

学习声明变量的类型最好从逻辑变量开始，因为使用简单。每一个逻辑变量需要 2 个字节。

### Byte（字节型）

字节型变量需要的内存空间最小，只有 1 个字节，可以存放 0 到 255 之间的数字。

### Currency（货币型）

货币型数据变量用于处理货币。在小数点左边可以存放长达 15 位的正数和负数，右边可以有 4 位小数。和单精度以及双精度数据类型不同，货币类型十分精确，不会按四舍五入处理。

要隐式声明货币型变量，使用类型声明字符@。例如，可以简单计算出每星期的工资：

```
Sub Calculate_Weekly_Salary()
```

```
    Salary@ = InputBox("输入工资",  
                       "计算周工资")
```

```
    WeeklySalary@ = Salary / 52
```

```
    MsgBox WeeklySalary
```

```
End Sub
```

每个货币型变量需要 8 个字节。

## Date (日期)

日期型变量相对复杂一些。VBA 的日期和时间用浮点数字表示，日期在小数点左侧，时间在小数点的右侧。VBA 可以处理的日期从 100 年 1 月 1 日到 9999 年 12 月 31 日，时间从 0: 00: 00 到 23: 59: 59。

**注意：**计算机采用两种方法保存数字：浮点数字或定点数字。浮点数字为某个数字乘以数字的乘方（例如 10），小数点在不同位置“浮动”。定点数字的小数点位置始终保持不变。

日期变量可以输入为文字表达式，例如，6/30/36，或者 June 30, 1936，在日期首尾输入#：

#June 30, 1936#

在代码窗口中输入日期并加上#，然后离开该输入行，VBA 将把数据转换成数字，显示为计算机设定的日期格式。例如，如果输入 June 30, 1936，VBA 可以显示 6/30/36。同样，如果输入时间值，（例如，#10: 15PM#），VBA 将其转换为数字并根据当前的时间格式显示（例如，10: 15: 00PM）。

**警告：**必须设定日期的年号，因为若不这样做，VBA 有可能把年份搞错。VBA 的早期版本（例如，Office 2000 和 Office 97）曾经对 20 世纪设定年份从 1 到 29，21 世纪从 30 到 00。

每一个日期变量需要 8 个字节。

## Decimal (十进制)

十进制数据类型存储无符号的整数，数字为 10 的乘方。无符号意味该整数既不带正号也不带负号。

**注意：**不能直接声明十进制变量，只能在不定型变量内部使用（以后讨论）。

每个十进制类型变量需要 12 个字节。

## Double (双精度)

双精度数据类型用于浮点数，可以处理的负数值从  $-1.797\ 693\ 234\ 862\ 32^{308}$  到  $-4.940\ 656\ 458\ 412\ 47^{-324}$ ，正数值从  $4.940\ 656\ 458\ 412\ 47^{-324}$  到  $1.797\ 693\ 234\ 862\ 32^{308}$ 。在这个范围内，有些数字不能用二进制精确表达，因此 VBA 进行四舍五入处理。

**注意：**双精度指的是双精度浮点，是计算机处理数字的方法。单精度（后面讨论）指的是单精度浮点。

可使用类型变量声明字符#对双精度变量进行隐式声明。每一个双精度变量需要 8 个字节。

## Integer (整型)

整型变量是最常用的变量，用来处理负 32 768 到 32 767 之间的数字，这个范围对许多

过程都十分有用。例如，若想重复一个命令 300 次，可按下面的方法为整型变量计数：

```
Dim intMyVar As Integer
For intMyVar = 1 to 300
    重复执行
Next intMyVar
```

每一个整型变量需要 2 个字节。

### Long (长整型)

长整型变量是整数值，其范围比整型变量广。长整型变量处理的值从 -2147 483 648 到 2 147 483 647。（如果数字比这个还大或者还小，应采用双精度类型，不过，应记住四舍五入处理。）

长整型变量可使用类型声明字符 & 进行隐式声明。每个长整型变量需要 2 个字节。

### Object (对象型)

对象型用来保存对象的地址（例如，对象模板中的对象），提供了访问对象的简便方法。每个对象变量需要 4 个字节。

### Single (单精度)

单精度类型和双精度类型一样，用于浮点数字。单精度可处理的负数值从  $-3.40282^{38}$  到  $-1.401298^{-45}$ ；正数值从  $1.401298^{-45}$  到  $3.40282^{38}$ 。在这个范围内，有些数字不能用二进制精确表达，因此 VBA 进行四舍五入处理。

使用感叹号作为类型声明字符可对单精度变量进行隐式声明（如果必须采用隐式声明）。每个单精度变量需要 4 个字节。

### String (字符串)

字符串型变量用来处理文本：

- ◆ 可变长度的字符串类型可以允许多达 20 亿个字符。需要 10 个字节以及相应的保存空间。
- ◆ 固定长度的字符串类型允许 1 到 64000 个字符，只需要保存该字符串的空间。如果存入的数据小于固定长度，VBA 将补充空格构成完整的字符串。如果存入的字符大于固定长度，VBA 将超出部分截去。VBA 对字符串从左向右计算：例如，把字符串 Output 存入长度为 4 的定长字符串类型，VBA 保存 Outp。
- ◆ 字符可以是字母、数字、空格，以及标点符号，当然也包括特殊字符。
- ◆ 可使用类型声明字符 \$ 进行隐式声明，然而（如通常的做法）最好应当对字符串型变量做显式声明，所有其他的变量也应当如此。

### Variant (不定型)

不定型变量如本章前面提到的那样用于所有未声明数据类型的变量。例如，Dim myUntypedVariable 生成一个不定型变量（也可显式声明不定型变量：Dim myVariable As Variant）。不定型变量可以处理大多数不同的数据类型，然而，有两个特点必须记住：

- ◆ 不定期变量不可以保存定长字符串型数据。如需要使用定长字符串，必须设定定长字符串

## 变量

- ◆ 不定型变量可以包括四个特殊值：Empty（指未经初始化的变量），Error（用来在过程中跟踪错误的特殊值），Nothing（一个特殊值，用于使关联对象的变量解除关联），Null（用于表明变量刻意未保存数据）。

不定型变量比其他类型的变量需要更多的内存。保存数据的不定型变量需要 16 字节，保存字符的不定型变量需要 22 字节以及字符需要的空间。

## 在变量类型中设定

如果觉得不同类型变量的详细介绍看不明白，不必担心。首先，可以不要考虑采用隐式声明或显式声明的方法选择变量类型，让 VBA 通过不定类型得出相应的子类型。其次，如果必须选择变量的类型，可以采用几个直截了当的规则：

- ◆ 如果变量值仅仅是 True 和 False，应当声明为逻辑型。
- ◆ 如果变量值总是为整数（即没有小数部分）应声明为整型数据。（如果数值太大，整型不够容纳，应声明为长整型。）
- ◆ 如果变量用来进行货币运算，或者需要无四舍五入的小数部分，应采用货币类型。
- ◆ 如果变量有时包括小数，应声明为单精度或双精度。
- ◆ 如果变量始终为字符，应声明为字符类型。

**提示：**如果不能确定哪一种变量类型最适合存储需要使用的信息，可先将变量声明为不定型。然后再从本地窗口中进入过程的断开模式（“视图”>“本地”），观察 VBA 对变量设定的不定型子类型：可以在类型栏中见到不定型/双精度或者不定型/字符型。对过程测试几次以保证子类型一致，然后再按子类型的提示声明变量。运行代码数次以保证新建的数据类型工作正常。

## 常量

常量有一个名称，在程序过程中数值保持不变。VBA 提供许多内置的常量，然而，用户也可定义自己需要的常量，用于在过程中存储保持不变的信息。

### 声明自己的常量

声明自己的常量，使用 Const 语句。如果需要在过程中多次使用一个固定的值，采用定义常量的方法可使代码简化。

### 语法

Const 语句的语法如下：

[Public/Private] Const 常量 [As 类型] = 表达式

这里 Public 和 Private 是可选的关键词，用来声明常量的公共或私有范围。很快将在后面介绍。常量是该常量的名称，采用一般的变量命名规则。类型是个可选参数，它用来设定常量的数据类型。表达式可以是一段文字（写入代码的值）、另一个常量，或者是两者的结合。

像变量一样，我们可以在一行中声明多个常量，只是要在语句中用逗号分开两个常量。

```
Const conPerformer As String = "Carmen Singer",_
    conTicketPrice As String = "$34.99"
```

### 举例

在 VBA 中声明常量类似于显式声明一个变量，然而，声明常量时必须给出常量值（而不是声明以后给出）。常量的值在声明后不可以更改。

了解一下下面的例子：

```
Const conVenue As String = "Davies Hall"
Const conDate As Date = #December 31, 2005#
MsgBox "音乐会定点在" & conVenue & "日期为" & conDate & "。"
```

第 1 行把常量 conVenue 声明为字符型，并赋值为 Davies Hall。第 2 行把常量 conDate 声明为日期型，并赋值为 December 31, 2005。（该行代码写完移到另一行时，VBA 将日期转换成计算机时钟的日期格式。例如，#12/31/2005#。）第 3 行显示一个信息框，其内容包括三个引号中的文本信息，conVenue 字符常量，以及 conDate 日期常量。

### 设定常量的范围和有效事件

在过程声明中常量默认范围为本地，即只对声明的过程有效。相应的，其有效时间是过程运行的时间。然而，可以用 Public 或 Private 关键词为常量设定不同的范围和有效时间：

- ◆ 要声明一个私有常量，在希望使用该常量的模块顶端进行声明。私有常量的有效时间不受限制，然而，仅仅对声明模块中的所有过程有效：

```
Private Const conPerformer As String = "Carmen Singer"
```

- ◆ 要声明一个公共常量，在任何模块的顶部进行声明。公共常量的有效时间不受限制，在声明的工程中，对所有模块中所有的过程均有效：

```
Public Const conTicketPrice As String = "$34.99"
```

## 常量列表

常量列表是预先定义的不重复的整数清单，在特定场合具有不同的名称和含义。常量列表一般用于设定对象的属性。在列表中每一个整数都有一个含义和名称，方便使用。在列表中和整数对应的名称叫做枚举常量。

例如，当使用 MsgBox 函数显示信息框时，可使用 VbMsgBoxStyle 常量列表中的枚举常量设定信息框的类型。若在信息框中需要图标，可设定枚举常量 vbCritical 或者用整数 16 获得停止图标，使用枚举常量 vbQuestion 或者整数 32 获得问号图标，使用枚举常量 vbExclamation 或者整数 48 获得感叹号图标，以及使用枚举常量 vbInformation 或者整数 64 获得信息图标。枚举常量很容易掌握并记住，可以不考虑它们的值。

VBA 包括许多内置的枚举常量。Visual Basic 编辑器可显示枚举常量的清单，在编写代码时用来选择合适的整数。在自定义的对象中用户还可以设定自己的枚举常量。

**Dim curMonthProfit()**

## 第7章 数组变量

- ◆ 理解数组及数组的作用
- ◆ 生成并使用数组
- ◆ 再定义数组的大小
- ◆ 清除数组的内容
- ◆ 检查某个变量是否为数组
- ◆ 数组排序
- ◆ 数组查询

本章将介绍怎样使用数组变量。数组变量可以在同一个时候存储许多值。一开始读者将了解什么是数组以及数组的用途，然后再了解怎样生成数组、输入数组信息以及清除信息，接着介绍怎样改变数组的容量使存储的内容变多（或变少），怎样设定数组的范围，怎样鉴别某个变量是数组还是普通存放单一值的变量。

### 什么是数组

数组是一个组合变量，即存放着许多数据类型相同的值。VBA 把数组看做单一的变量，可以存储许多值。可以对数组本身进行操作，包括所有的值，或者可以使用索引号处理保存在数组中的单个值。索引号指明数组中的位置。如果不能想象此含义，可以把数组看成一个列表。列表中的每一项占据一行，用索引号标出，因此通过索引号就可以得到某一项的值。参照本章后面关于数组的举例。

**注意：**不定型数据类型的数组可以存储多种类型的数据。

简单的数组只有一维数组。另外，也可以声明多维数组，这将在本章后面讨论。

数组具有下界和上界。默认情况下，下界为 0，因此在数组中第一项的索引值为 0。这一点可能造成混乱，因为使用的索引值始终比数组中的项目位置少一位。然而，VBA 可以使数组中第一项的默认索引号变为 1，即在含该数组的模块顶端使用 Option Base 1 语句。这样可以使数组中每一项索引值和项目位置相同，以便使数组容易使用。

**Option Base 1**

### 声明数组变量

数组是一种变量，因此，可以使用常用的关键字进行声明：Dim、Private、Public 和 Static。为表明某个变量为数组变量，需在数组名称后加上括号。例如，下面的语句声明了名称为 curMonthProfit 的数组：

```
Dim curMonthProfit()
```

上例生成了一个不定类型的数组。在数组中存储数据时，VBA 设定一个或多个相应的子类型。

可以像设定变量那样设定数组类型。例如，下面的语句声明了名称为 curMonthProfit 的数组，数据类型为货币型。

```
Dim curMonthProfit() As Currency
```

可以使用数组下标声明数组的项目数。例如，下面的语句声明了名称为 curMonthProfit 的数组，数据类型为货币型，并指明有 12 个项目。

```
Dim curMonthProfit(11) As Currency
```

**注意：**在 Dim curMonthProfit(11) As Currency 语句中数组的下标为 11，而非 12，因为计数从 0 开始，而非从 1。第 1 项是 curMonth Profit(0)，第 2 项是 curMonth Profit(1)，第 12 项为 curMonthProfit(11)。

图 7.1 简单展示了一维数组，是用 Dim curMonthProfit(11) As Currency 语句声明的。

元素号	名称	内容
0	curMonthProfit(0)	—
1	curMonthProfit(1)	—
2	curMonthProfit(2)	—
3	curMonthProfit(3)	—
4	curMonthProfit(4)	—
5	curMonthProfit(5)	—
6	curMonthProfit(6)	—
7	curMonthProfit(7)	—
8	curMonthProfit(8)	—
9	curMonthProfit(9)	—
10	curMonthProfit(10)	—
11	curMonthProfit(11)	—

图 7.1 采用语句 Dim curMonthProfit(11) As Currency 生成的一维数组可以想象为这样

要从 1 开始编号，必须在声明数组的模块顶端声明区使用 Option Base 语句。举例如下：

```
Option Base 1 '代码页的顶端
```

```
Dim curMonthProfit(12) As Currency
```

图 7.2 简单展示该数组的式样。

**注意：**不声明数据类型而让 VBA 自动使用不定型数据类型将增加内存的使用空间，这可能（极端的情况下）使计算机运行速度变慢。因为数组的每一项都需要存储空间，而大型数组需要很大的空间。多维的数组尤其如此。还可以显示声明数组的上下界。

元素号	名称	内容
1	curMonthProfit(1)	—
2	curMonthProfit(2)	—
3	curMonthProfit(3)	—
4	curMonthProfit(4)	—
5	curMonthProfit(5)	—
6	curMonthProfit(6)	—
7	curMonthProfit(7)	—
8	curMonthProfit(8)	—
9	curMonthProfit(9)	—
10	curMonthProfit(10)	—
11	curMonthProfit(11)	—
12	curMonthProfit(12)	—

图 7.2 采用语句 Dim curMonthProfit(12) As Currency 以及语句 Option Base 1 生成的一维数组

Option Base 1'代码页的顶端

Dim curMonthProfit(1 To 12) As Currency

注意：使用 Option Base 1 语句后，数组的操作比较容易，所以本章后面均采用该语句。

### 在数组中存储数值

要把一个值存入数组中，可使用索引号指明该项。例如，在下面的例子中，伦敦、香港和台北被分别输入前 3 项。

Option Base 1

```
Dim strLocations(6) as String
strLocations(1) = "伦敦"
strLocations(6) = "香港"
strLocations(6) = "台北"
```

图 7.3 给出该数组的结构。

元素号	名称	内容
1	curMonthProfit(1)	伦敦
2	curMonthProfit(2)	香港
3	curMonthProfit(3)	台北
4	curMonthProfit(4)	—
5	curMonthProfit(5)	—
6	curMonthProfit(6)	—

图 7.3 有 3 个值的字符串数组

## 多维数组

在前一节的例子中，`curMonthProfit` 是个一维数组，用起来最简单。VBA 的数组可以多达 60 维，除非在多维模型领域获得博士学位，否则很难想象。一般不会需要如此复杂的数组，大多数情况下，2 维、3 维或 4 维就足够了。

声明多维数组时，把维数用逗号隔开。例如，下面的语句声明了一个 2 维数组，名称是 `MyArray`，每一维包括 3 项：

```
Option Base 1
Dim MyArray(3, 3)
```

图 7.4 给出数组的式样。

第 1 列	第 2 列	第 3 列
1, 1	2, 1	3, 1
1, 2	2, 2	3, 2
1, 3	2, 3	3, 3

图 7.4 可以认为 2 维数组包括行和列

多维数组似乎难以理解，但 2 维数组一目了然，只要将其看做包括行和列的表格即可。前一系列的 10 个元素应该在表的第 1 列，第 2 系列的 10 个元素在表的第 2 列。每个系列的信息不一定要相关，不过数据类型却必须相同。例如，可以把 10 个目录名称存入一个字符串型数组的第 1 维，把 10 个文件名称存入第 2 维（更多的字符），把 10 个猫的名字存入第 3 维，把被暗杀或遭弹劾的美国总统存入第 4 维。然后，用项目的地址就可以访问信息，例如，表中第 1 列的第 2 项。后面将介绍具体的做法。

类似地，可以想象 3 维数组是一个具有多张工作表（行和列组成）的工作簿，在 3 维数组中又有更多的行和列。到目前还可以想象，4 维或更大的数组就需要费劲劳神了。

## 声明动态数组

数组可以是固定大小的或者动态的。已经介绍的例子都是固定大小的数组，例如名称为 `curMonthProfit` 的数组设定为 12 项。

如果需要存储数目不确定的项目，动态数组十分有用。例如，用一个过程把窗口两两并排，可能需要一个数组存放每一个打开窗口的名称。可是，在运行过程时并不知道会打开多少窗口，这就需要使用动态数组存放信息。

要声明动态数组，在声明语句中不要规定项目数，但需要里面空着的括号。例如，下面的语句声明动态数组 `arrTestArray`，VBA 认定其为不定型。

```
Dim arrTestArray()
```

## 再定义数组的大小

用 `ReDim` 语句可以改变动态数组的大小。例如，要改变前例中定义的 `arrTestArray` 动

态数组，设定为 5 个项目，可以使用下面的语句：

```
ReDim arrTestArray(5)
```

使用 ReDim 语句改变数组的大小时，数组内的信息将会丢失。如果刚刚声明了一个动态数组且未存放任何内容，那么丢失信息无关紧要。然而，有的时候，希望在改变数组大小的同时保留已有的信息。在增加上界的同时保留现有的信息，应当使用 ReDim Preserve 语句，而不能直接使用 ReDim 语句。

```
ReDim Preserve arrTestArray(5)
```

如果使用 ReDim Preserve 语句使数组的上界变小，那么不包括在范围中的信息将失去。例如，若数组中有 5 条信息，使用 ReDim Preserve 语句将其改变成 3 条信息时，第 4 条、第 5 条信息将失去。

**注意：**ReDim Preserve 语句用于数组最后的维，不能在多维数组中保留其他维的数据。

## 从数组中返回信息

要从数组中返回信息，必须使用该信息所在的项目索引号。例如，下面的语句在信息框中返回名称为 arrMyArray 数组的第 4 条信息。

```
Option Base 1
```

```
MsgBox arrMyArray(4)
```

下面的语句返回名称为 arrMy2DArray 数组的第 2 维第 5 条信息，并在信息框中显示出来。

```
Option Base 1
```

```
MsgBox arrMy2DArray(2,5)
```

**注意：**要返回数组的多个信息，必须一项一项地设定。

## 清除数组的内容

要清除数组的内容，必须使用带有数组名称的 Erase 语句。该语句对固定大小的数组进行初始化，释放被动态数组的项目占据的内存空间（恢复到刚刚声明动态数组时的状态）。下面的语句清除了固定数组 arrMyArray 的内容：

```
Erase arrMyArray
```

## 检查某个变量是否为数组变量

因为数组是变量的一种，有时希望检验某个变量是数组变量还是非数组变量。判断一个变量是否为数组变量，可使用带有变量名称的 IsArray() 函数。例如，下面的语句对变量 MyVariable 进行检测，并把结果显示在信息框中。

```
If IsArray(MyVariable) = True Then
    Msg = "MyVariable" & "是数组"
Else
    Msg = "MyVariable" & "不是数组"
End If
MsgBox Msg, vbOKOnly + vbInformation, "数组检验"
```

## 检查数组的边界

检查数组的上下界应使用 LBound() 函数和 UBound() 函数，LBound() 函数返回下界（第 1 项的索引号），UBound() 函数返回上界（最后一项的索引号）。

LBound() 函数和 UBound() 函数的语法如下：

```
LBound(array [, dimension])
UBound(array [, dimension])
```

其中，array 是数组名称，为必选参数；dimension 是可选参数，表示数组的维。1 表示第 1 维，2 表示第 2 维，以此类推。若不给参数，VBA 将按第 1 维考虑。

例如，下面的语句返回名称为 arrMyArray 数组第 2 维的上界，并在信息框中显示出来。

```
MsgBox UBound(arrMyArray, 2)
```

## 数组排序

我们常常需要对数组排序，特别是信息来自外部，而不是通过代码一个个输入的时候。

排序并不难理解：只是按一定的顺序排列。例如，可以使数组中的文本信息按字母顺序，或字母顺序反向排列；也可使另一数组中的数值信息按升序或降序排列。不过，写排序程序十分复杂。

本节将介绍一种简单的排序方式，即冒泡排序法。这么叫是因为排序项目在数组中是逐渐地冒到上面来的。排序包括两个循环，用来比较数组中的两个项目，如果第二项比第一项大，就将它们的位置颠倒，这样持续比较下去，直到所有的项目完成排序为止。冒泡排序法的效率不是很高，但容易掌握，而且现在处理器执行循环操作相对便宜。

**注意：**第 12 章将介绍循环。

程序清单 7.1 是冒泡排序法的代码。

### 程序清单 7.1

1. Option Explicit
2. Option Base 1
- 3.
4. Sub Sort\_an\_Array()
- 5.
6. 'declare the array and other variables
7. Dim strArray(12) As String

```

8. Sub Sort_an_Array()
9.     Dim strTemp As String
10.    Dim strMsg As String
11.    Dim X As Integer, Y As Integer, i As Integer
12.    'assign strings to the array
13.    strArray(1) = "nihilism"
14.    strArray(2) = "defeatism"
15.    strArray(3) = "hope"
16.    strArray(4) = "gloom"
17.    strArray(5) = "euphoria"
18.    strArray(6) = "despondency"
19.    strArray(7) = "optimism"
20.    strArray(8) = "pessimism"
21.    strArray(9) = "misery"
22.    strArray(10) = "happiness"
23.    strArray(11) = "bliss"
24.    strArray(12) = "mania"
25.
26.    strMsg = "Current items in array:" & vbCrLf
27.    For i = 1 To UBound(strArray)
28.        strMsg = strMsg & i & ":" & vbCrLf & strArray(i) & vbCrLf
29.    Next i
30.    MsgBox strMsg, vbOKOnly + vbInformation, "Array Sorting: 1"
31.
32.    For X = LBound(strArray) To (UBound(strArray) - 1)
33.        For Y = (X + 1) To UBound(strArray)
34.            If strArray(X) > strArray(Y) Then
35.                strTemp = strArray(X)
36.                strArray(X) = strArray(Y)
37.                strArray(Y) = strTemp
38.                strTemp = ""
39.            End If
40.        Next Y
41.    Next X
42.
43.    strMsg = "Items in sorted array:" & vbCrLf
44.    For i = 1 To UBound(strArray)
45.        strMsg = strMsg & i & ":" & vbCrLf & strArray(i) & vbCrLf
46.    Next i
47.    MsgBox strMsg, vbOKOnly + vbInformation, "Array Sorting: 2"
48.
49. End Sub

```

程序清单 7.1 的说明如下：

- ◆ 第 1 行 Option Explicit 语句用于强迫显式声明。第 2 行 Option Base 1 语句将数组的记数起点设定为 1 而不是 0。这两条语句出现在代码页的声明区，在所有的过程之前。第 3 行是一空行。插入空白使代码容易阅读。
- ◆ 第 4 行是 Sort\_an\_Array 过程的开始。第 5 行为空行。
- ◆ 第 6 行是注释行，说明对数组和变量进行声明。第 7 行声明了一个字符型数组 strArray，下标为 12。第 8 行声明了字符变量 strTemp。第 9 行声明字符变量 strMsg。第 10 行声明整型变量 X、Y 和 i。第 11 行为空行。

- ◆ 第 12 行为注释行，解释下面 12 条语句（从第 13 行至第 24 行），把字符串存入数组。这些字符是和情绪有关的单词。第 25 行为空行。
- ◆ 第 26 行到第 30 行根据赋给数组的字符串建立一个字符串，然后在信息框中显示出来。这段代码便于了解过程在运行时的状况，没有一步到位。第 26 行给出一段解释文字，包括两个回车，加载字符变量 strMsg 上。第 27 行为 For…Next 循环语句，从 i=1 运行到 i= UBound(strArray)，换句话说，每次运行数组中的一项。（循环也可以设定为 i=12，因为数组上界已设定，然而使用上界表达式比固定的数值更灵活。）第 28 行把记数变量 i、空格（vbTab）、当前的数组值（strArray(i)）以及回车（vbCr）加载到 strMsg 上。第 29 行为循环。第 30 行显示含有 strMsg 内容的信息框，如图 7.5 所示。第 31 行为空行。
- ◆ 过程的排序部分自第 32 行至第 41 行：
  - ◆ 第 32 行为外部的 For…Next 循环的开始，到第 41 行结束，该循环从 X=LBound(strArray)（即 X=1）到 X= (UBound(strArray)-1)（即 X=11），即数组的上界减 1。
  - ◆ 第 33 行为内部的 For…Next 循环的开始，从 Y=(X+1) 到 Y= UBound(strArray)。第 40 行结束内循环。
  - ◆ 第 34 行用于比较 strArray(X) 和 strArray(Y)。如果 strArray(X) 大于 strArray(Y)，就将两者交换位置，换句话说，strArray(X) 必须在 strArray(Y) 后面出现。第 35 行将 strArray(X) 赋于 strTemp，第 36 行将 strArray(Y) 赋于 strArray(X)。第 37 行将 strTemp 赋于 strArray(Y)。这样两个值就颠倒了。第 38 行将 strTemp 恢复为空字符串。第 39 行结束 If 语句。第 40 行结束内循环。第 41 行结束外循环。第 42 行为空行。
- ◆ 第 43 行到第 47 行重复第 26 行到第 30 行，显示一个信息框，如图 7.6 所示，数据已经排序。由此可知，排序已经生效。
- ◆ 第 48 行是空行，第 49 行结束程序。

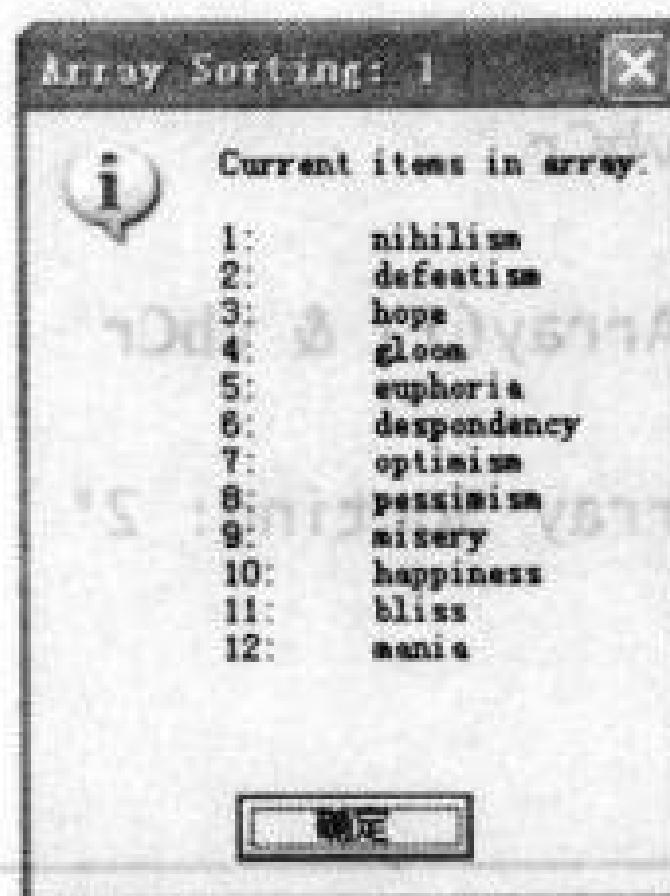


图 7.5 过程 Sort\_an\_Array 显示信息框，这是未经过排序的

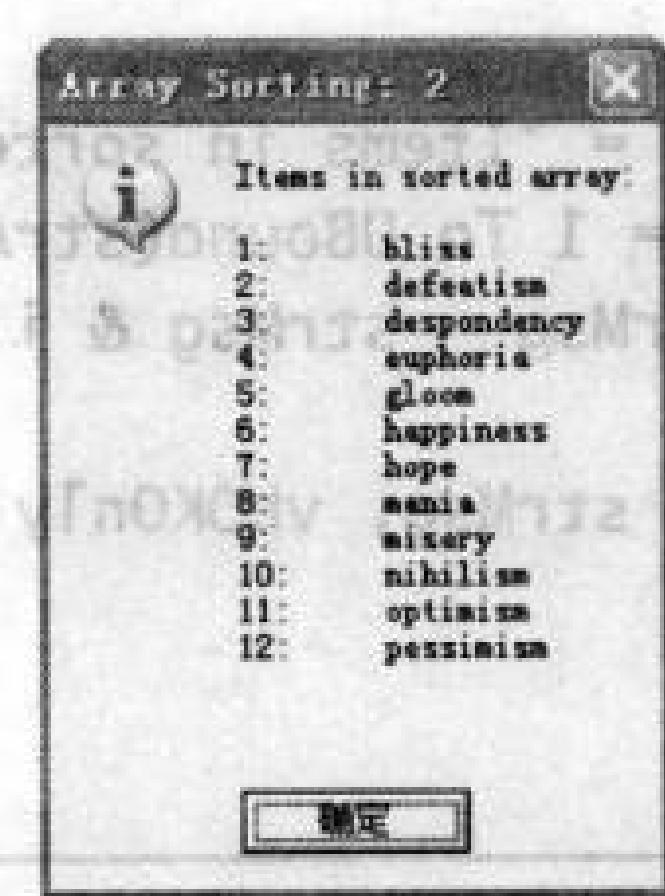


图 7.6 过程 Sort\_an\_Array 结束排序，并在信息框中显示结果

## 数组查询

数组经常需要做的另一件事是在数组中查找一个特定的值。本节将介绍两种查询方法，线性查询和二分法查询。前者可以用于排序的数组或者未经排序的数组，而后者只能用于排序的数组。

## 在数组中实现线性查询

线性查询是一个很简单的查询方法：从数组的开头连续查询直到发现目标，或者直到数组的末尾。

在启动代码前，按下 Ctrl+G 键或者选择“视图▶‘立即窗口’”以显示“立即窗口”。该过程在“立即窗口”中打印信息，并且可以了解代码的执行情况。

程序清单 7.2 为简单的线性查询代码，是针对一维数组的。

### 程序清单 7.2

```

1. Option Explicit
2. Option Base 1
3.
4. Sub Linear_Search_of_Array()
5.
6.     'declare the array and the variables
7.     Dim intArray(10) As Integer
8.     Dim i As Integer
9.     Dim varUserNumber As Variant
10.    Dim strMsg As String
11.
12.    'add random numbers between 0 and 10 to the array
13.    'and print them to the Immediate window for reference
14.    For i = 1 To 10
15.        intArray(i) = Int(Rnd * 10)
16.        Debug.Print intArray(i)
17.    Next i
18.
19.    Loopback:
20.        varUserNumber = InputBox _
21.            ("Enter a number between 1 and 10 to search for:", _
22.             "Linear Search Demonstrator")
23.        If varUserNumber = "" Then End
24.        If Not IsNumeric(varUserNumber) Then GoTo Loopback
25.        If varUserNumber < 0 Or varUserNumber > 10 Then GoTo Loopback
26.
27.        For i = 1 To UBound(intArray)
28.            If intArray(i) = varUserNumber Then
29.                strMsg = "Your value, " & varUserNumber & _
30.                    ", was not found in the array."
31.            Exit For
32.        End If
33.    Next i
34.    MsgBox strMsg, vbOKOnly + vbInformation, "Linear Search Result"
35.
36. End Sub

```

程序清单 7.2 的说明如下：

- ◆ 和前面的程序一样，第 1 行 Option Explicit 语句用于强迫显式声明。第 2 行 Option Base 1 语句将数组的记数起点设定为 1 而不是 0。这两条语句出现在代码页的声明区，在所有的过程之前。第 3 行是一个空行。
- ◆ 第 4 行是过程 Linear\_Search\_of\_Array 的开始。第 5 行是为空行。
- ◆ 第 6 行是注释行，说明对数组和变量进行声明。第 7 行声明了一个整型数组 intArray，下标为 10。第 8 行声明了整型变量 i，常常用来进行计数。第 9 行声明了不定型变量 varUserNumber，用来保存从输入框输入的数据（后面将详细说明）。第 10 行声明字符串型变量 strMsg。第 11 行为空行。

**注意：**该过程把变量 varUserNumber 声明为不定型而不是整型，这样，如果在输入框中输入了非整型的数据（例如字符型），VBA 就不会因错误而自动停止。

- ◆ 第 12 行和第 13 行给出关于第 14 行到第 17 行的注释（这两行可以在第 13 行的末尾用连字符合并为一个逻辑行，这就不需要在第 13 行的开始加一小撇。然而，第 14 行用注释字符开始代码容易读懂）。
- ◆ 第 14 行为 For\_Next 循环的开始，i=1 到 i=10。第 15 行把随机数乘以 10 的整数结果赋给 intArray 数组的当前项：intArray(i)=Int(Rnd\*10)（Rnd() 函数生成 0 和 1 之间的随机数，并带有一定位数的小数。该过程将随机数乘以 10 得到 0 和 10 之间的一个数，然后获取该数的整数部分）。第 16 行使用 Debug 对象的打印方法，将数组中的当前项打印在“立即窗口”中，便于保留数组包括的一系列数值。第 17 行以 Next i 语句结束循环。第 18 行为空行。
- ◆ 第 19 行是一个标记 Loopback，用来在未满足条件时返回该点。
- ◆ 第 20 行把输入框（参见图 7.7）的结果赋给不定型变量 varUserNumber。输入框要求输入一个 0 到 10 之间的数。

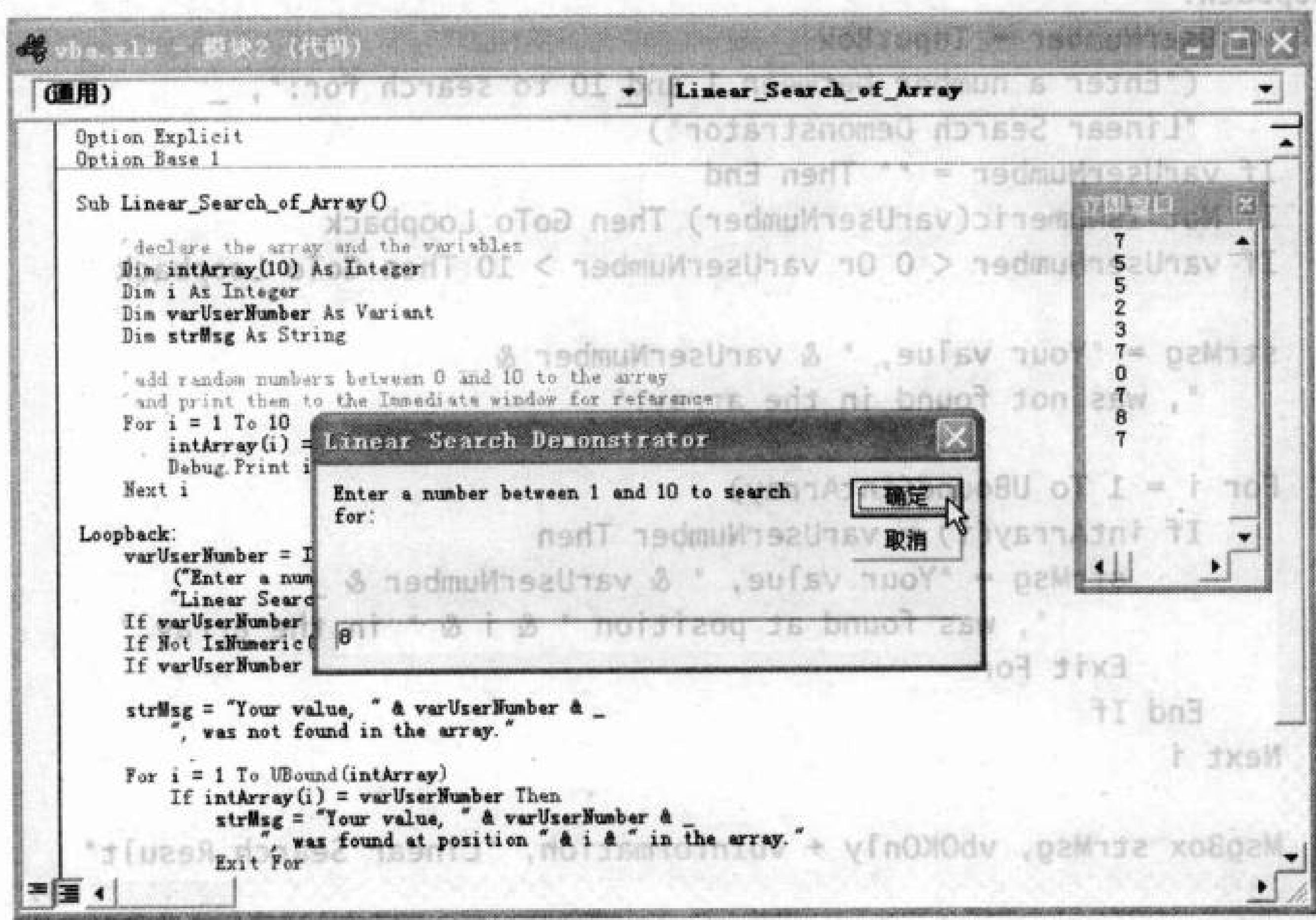


图 7.7 过程 Linear\_Search\_of\_Array 显示一个输入框，要求输入 0 到 10 之间的一个数。输入的数字过程已经在“立即窗口”中显示出来

- ◆ 第 21 行把 varUserNumber 的内容和空字符串进行比较。空字符串是在输入框中单击“取消”按键，或者单击“确定”按钮而不输入任何值造成的。如果 varUserNumber 是空字符串，End 语句结束此过程。
- ◆ 第 22 行使用 IsNumeric() 函数检查 varUserNumber 是否为数值型。如果不是，GoTo Loopback 语句返回到 Loopback 标记，此后，输入框再次显示要求重新输入。第 23 行检查 varUserNumber 是否小于 0 或者大于 10，如果出现任何一种情况，程序将返回到 Loopback 标记，再次出现输入框。第 24 行为空行。

**注意：**请注意 VBA 的灵活性，代码要求输入一个 0 到 10 之间的数。虽然该数仍保  
存在不定型变量中，而不是转换成一个整数，但是 VBA 仍然会进行必要的  
比较。

- ◆ 第 25 行对字符型变量 strMsg 赋值，给出开始的信息，说明在数组中没有找到。（如果代码在数组中找到数值，在显示前信息会更改。）第 26 行为空行。
- ◆ 第 27 行到第 32 行为过程的查询部分。第 27 行为循环语句 For…Next 的开始，用数  
组中的下标循环，从 i=1 到 i=UBound(intArry)。第 28 行比较 intArray(i) 和 va-  
rUserNumber，如果相同，第 29 行则告诉 strMsg 值在数组中所在的位置。第 30 行  
用 Exit 语句退出 For…Next 循环。（如果第 28 行不能找到相同的值，那么第 32 行的  
Next i 语句进入循环。）
- ◆ 第 33 行是空行。第 34 行显示信息框，strMsg 给出查询后的结果。图 7.8 为完成查询  
后的结果。第 35 行是空行，第 36 行结束过程。

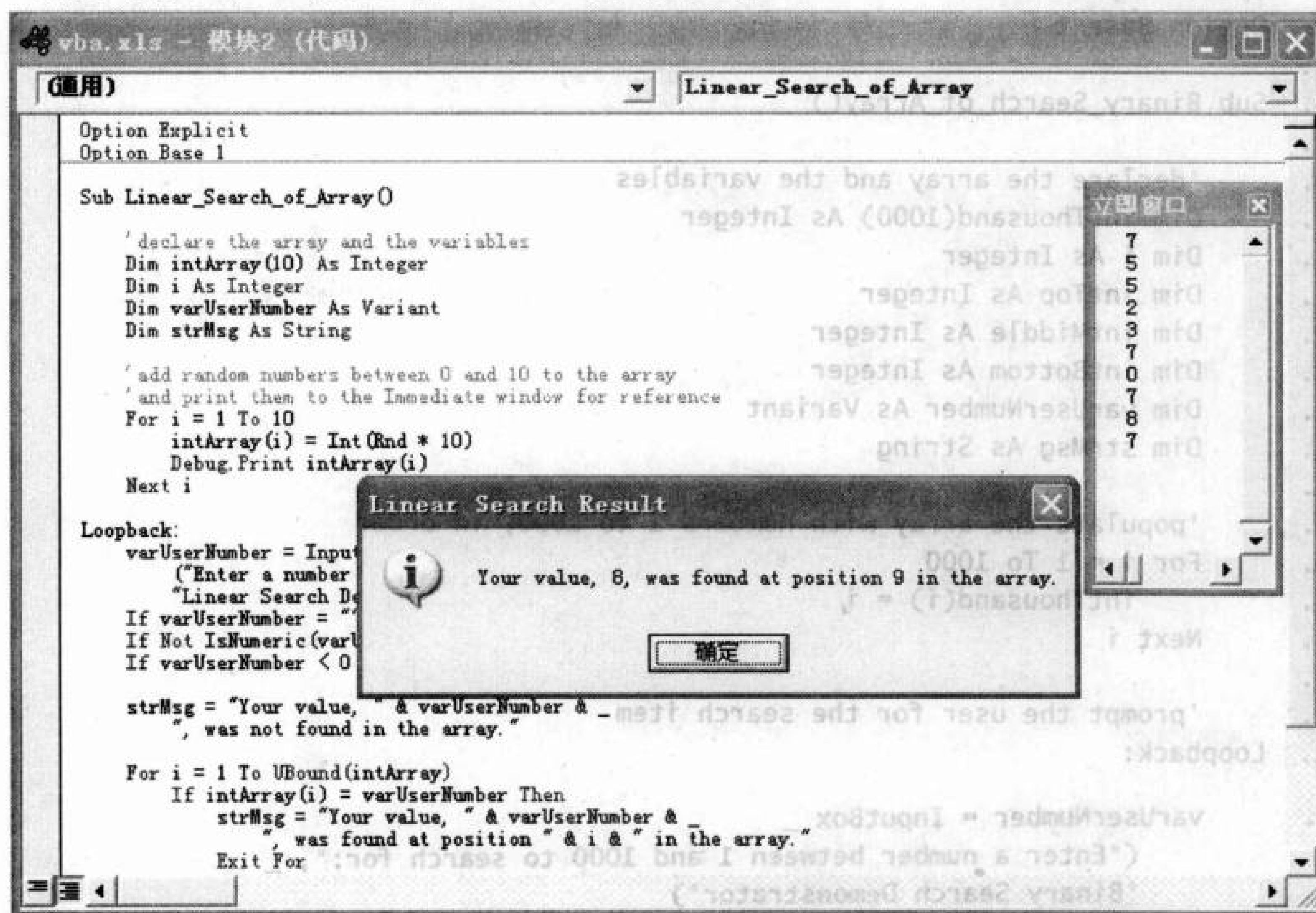


图 7.8 程序清单 7.2 的第 34 行显示信息框以便给出查询结果

## 在数组中实现二分法查询

正如前面介绍的那样，线性查询很容易使用，然而十分呆板，从数组的起始位置开始查询，一个个地对比。这样的方法在数据量小的情况下很管用，例如 10 个项目的数组。然而，不能用它来查询哪怕像一个小城市的电话号码。稍微有一定数据量的信息就需要更好的查询方法。

一般说来，对于排过序的数组，二分法是个好方法。二分法查询是一种算法，用于给定范围的查询，即在给定的范围中彻底查询。

二分法把一个排过序的数组一分为二，了解哪一半会有查询数据，然后反复使用分一半查询的模式，要么得到查询结果，要么得到该部分中无查询值。例如，针对从 1 到 100 000 000 的升序排列的数组，要找出 789 789，首先分成两半，各 50 万。先确定查询值在前 50 万还是后 50 万中，然后再分一半缩小范围。然后，每次都将确定被查询的值在前一半还是后一半，直到查到结果或者超出范围。

这只是个例子，但百万也可以说明问题了。程序清单 7.3 把问题更简化了，使用一个数组依次存放 1 到 1000 的数字。第 1 项是 1，第 2 项是 2，直到 1000。例子并不真实，却可以了解程序的运行情况。

程序清单 7.3

```

1. Option Explicit
2. Option Base 1
3.
4. Sub Binary_Search_of_Array()
5.
6.     'declare the array and the variables
7.     Dim intThousand(1000) As Integer
8.     Dim i As Integer
9.     Dim intTop As Integer
10.    Dim intMiddle As Integer
11.    Dim intBottom As Integer
12.    Dim varUserNumber As Variant
13.    Dim strMsg As String
14.
15.    'populate the array with numbers 1 to 1000, in order
16.    For i = 1 To 1000
17.        intThousand(i) = i
18.    Next i
19.
20.    'prompt the user for the search item
21.    Loopback:
22.
23.        varUserNumber = InputBox _
24.            ("Enter a number between 1 and 1000 to search for:", _
25.             "Binary Search Demonstrator")
26.
27.        If varUserNumber = "" Then End
28.        If Not IsNumeric(varUserNumber) Then GoTo Loopback
29.
30.        'search for the search item
31.        intTop = UBound(intThousand)
32.
33.        intBottom = LBound(intThousand)
34.
35.        Do While intTop >= intBottom
36.            intMiddle = (intTop + intBottom) \ 2
37.
38.            If intThousand(intMiddle) = varUserNumber Then
39.                strMsg = "The number you entered is found at index " &
40.                    CStr(intMiddle)
41.                Exit Do
42.            Else If intThousand(intMiddle) < varUserNumber Then
43.                intBottom = intMiddle + 1
44.            Else
45.                intTop = intMiddle - 1
46.            End If
47.        Loop
48.
49.        MsgBox strMsg
50.    End Sub

```

```

28.      intBottom = LBound(intThousand)
29.      Do
30.          Do
31.              intMiddle = (intTop + intBottom) / 2
32.              If varUserNumber > intThousand(intMiddle) Then
33.                  intBottom = intMiddle + 1
34.              Else
35.                  intTop = intMiddle - 1
36.              End If
37.          Loop Until (varUserNumber = intThousand(intMiddle)) _ 
             Or (intBottom > intTop)
38.      'establish whether the search discovered the search item _ 
             or not and add the appropriate information to strMsg
39.      If varUserNumber = intThousand(intMiddle) Then
40.          strMsg = "The search found the search item, " _ 
             & varUserNumber & ", at position " & intMiddle _ 
             & " in the array."
41.      Else
42.          strMsg = "The search did not find the search item, " _ 
             & varUserNumber & "."
43.      End If
44.  End Sub

```

以下是程序清单 7.3 的说明：

- ◆ 第 1 行 Option Explicit 语句用于强迫显式声明。第 2 行 Option Base 1 语句将数组的记数起点设定为 1 而不是 0。这两条语句出现在代码页的声明区，在所有的过程之前。
- ◆ 第 3 行为空行。第 4 行声明 Binary\_Search\_of\_Array 过程，第 5 行又是空行。
- ◆ 第 6 行为注释行，引出数组声明（在第 7 行声明了下标为 1000 的数组）和变量声明，包括：整型变量 i（第 8 行）、intTop（第 9 行）、intMiddle（第 10 行）和 intBottom（第 11 行），不定型变量 varUserNumber（第 12 行），以及字符型变量 strMsg（第 13 行）。第 14 行为空行。
- ◆ 第 15 行为注释行，说明第 16 行到第 18 行将 1 到 1000 按顺序写入数组。为此使用了 For...Next 循环，从 i=1 运行到 i=1000。将 i 的值写入 i 表示的数组项，换句话说，就是写入的数值等于项目值。第 19 行为空行。
- ◆ 第 20 行为注释行，解释第 21 行到第 24 行，使用输入框（参见图 7.9）提示输入要查询的数值并进行验证。如前面的程序所示，这段代码要确认输入的不是空字符串（第 23 行），否则退出程序。程序使用了返回标记 Loopback（第 21 行），如果检查出输入的值不是数值型（第 24 行）将返回。因为这次数组中的数值已经知晓，所以不需要检查输

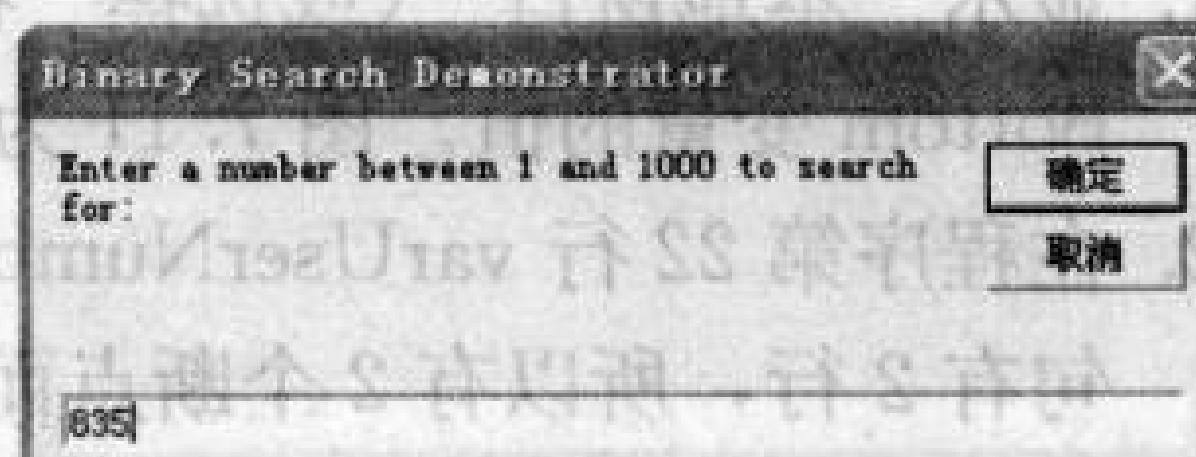


图 7.9 Binary\_Search\_of\_Array 双分查询过程要求输入一个 1 到 1000 之间的数值

入的值是否在范围以内。如果输入的值不在数组中，那么可以再输入。

- ◆ 第 25 行为空行。第 26 行是注释行，引出对给定值查询的代码。第 27 行将数组的上界赋给变量 intTop，第 28 行将下界赋给 intBottom，第 29 行为空行。
- ◆ 第 30 到 37 构成 Do…Loop Until 语句，运行双分法查询。以下为详细说明：
  - ◆ 第 30 行为 Do 关键词，为 Do…Loop Until 循环的开始。第 37 行以 Loop Until 关键词和条件 (varUserNumber = intThousand(intMiddle) Or (intBottom > intTop)) 结束循环。第 12 章将有详细说明。现在只要清楚 Do…Loop Until 执行一次就会用 Loop Until 的条件检查一次，决定是否还需要再次执行。条件规定，如果数组中的值 intThousand (intMiddle) 等于变量 varUserNumber 中的值或者 intBottom 的值大于 intTop 的值，循环就不再继续。
  - ◆ 第 31 行将 intTop 和 intBottom 的和除以 2 赋值给整型变量 intMiddle。这样做用中间值将数组一分为二。例如，在 1000 项的数组中，第 1 次循环 intTop 的值为 1000，intBottom 的值为 1，中间值为 500 (1000 除以 2)。
  - ◆ 第 32 行检验 varUserNumber 是否大于存入 intMiddle 的值 (intThousand (intMiddle))，该值时为数组的中值。如果是，查询在数组的上半部分进行，第 33 行将 intBottom 设定为 intMiddle+1，否则第 34 行的 Else 语句发挥作用，第 35 行将 intTop 设定为 intMiddle-1，查询在数组的下半部分进行。
  - ◆ 第 36 行结束 If 语句。第 37 行检验条件，决定继续还是终止循环。
- ◆ 第 38 行为空行。第 39 行为两行注释行，引出第 40 行到第 44 行的代码，检查查询是否找出了被查询的值，并将相应的信息送给字符变量 strMsg。第 40 行用于比较 varUserNumber 和 intThousand (intMiddle) 的大小，如果相等，第 41 行对 strMsg 赋值，告诉被查询结果的地点；如果不相等，第 43 行说明查询未发现结果。第 45 行为空行。第 46 行为信息框，显示查询的结果。图 7.10 为查询成功和不成功的例子。
- ◆ 第 47 行为空行，第 48 行结束过程。

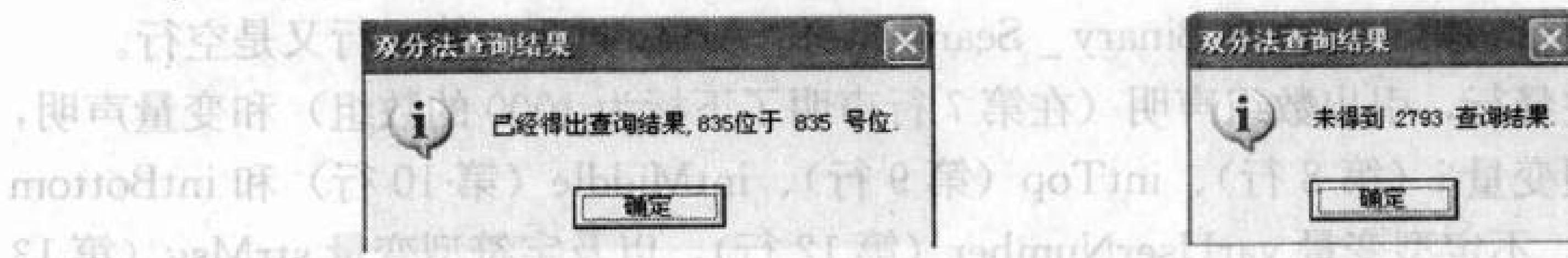


图 7.10 查询成功（左图）或不成功（右图）的结果

程序清单 7.3 最复杂的部分在循环。从 Sybex 网站上下载并打开，粘贴到 Visual Basic 编辑器上（可用于所有配备了 VBA 的软件），然后打开模块，按下列方法操作：

1. 显示“本地窗口”（或选择“视图”>“本地”），用来跟踪 intTop、intMiddle 和 intBottom 变量的值。图 7.11 为程序运行时的“本地窗口”。
2. 在程序第 22 行 varUserNumber=InputBox 语句上面单击边界形成断点（因为该语句有 2 行，所以有 2 个断点而不是 1 个）。
3. 按 F5 键（或选择“运行”>“运行子过程/用户窗体”）使程序运行到断点，VBA 将数组赋值并在第 22 行中断。
4. 按 F8 键运行下面的语句。按第一下显示输入框，输入 69 然后单击“确定”按钮。
5. 代码进入 Do…Loop 循环并执行，在“本地窗口”中观察变量 intTop、intMiddle 和

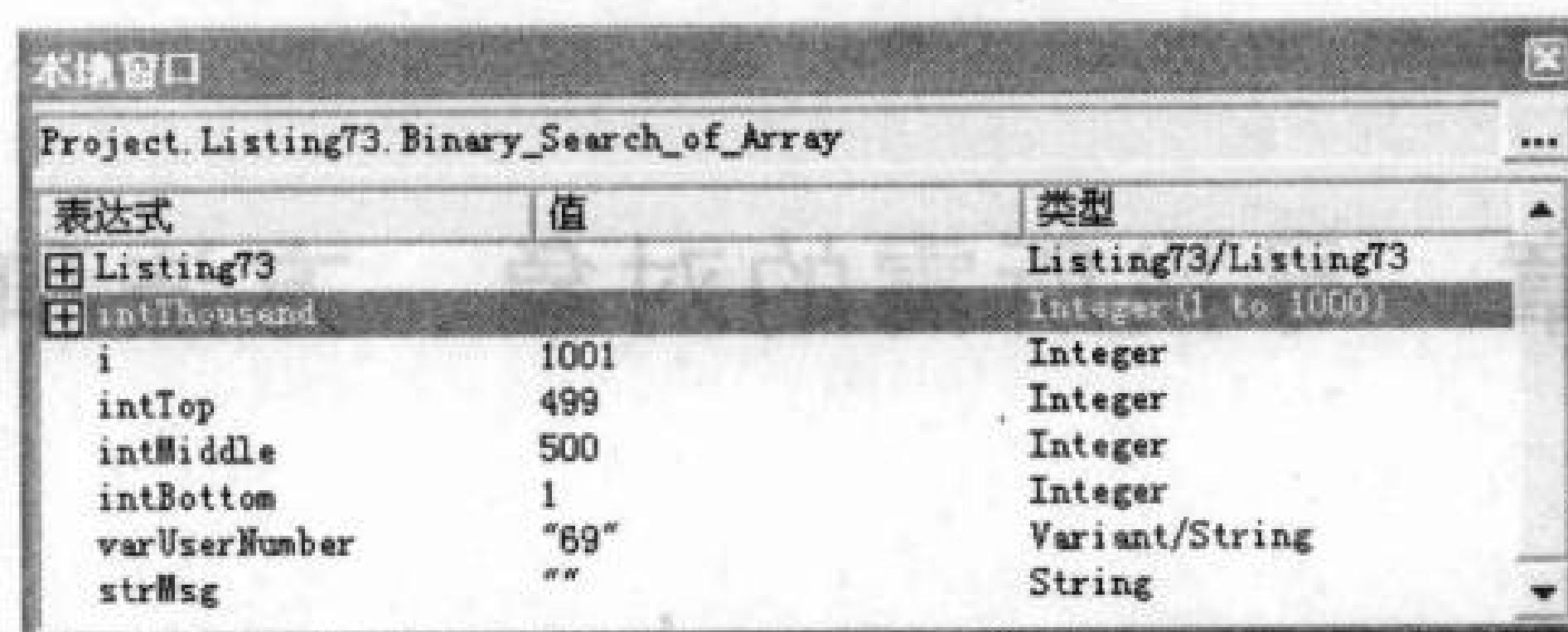


图 7.11 用“本地窗口”跟踪程序运行时 intTop、intMiddle 和 intBottom 变量的值

intBottom 的值，其变化如下：

循环	intTop	intMiddle	intBottom
0	1000	1	1
1	499	500	1
2	249	250	1
3	124	125	1
4	124	62	63
5	93	94	63
6	77	78	63
7	69	70	63
8	69	66	67
9	69	68	69
10	68	69	69

到第 10 次循环，intThousand(intMiddle) 等于 varUserNumber，循环结束。

## 第 8 章 寻找所需的对象、方法和属性

- ◆ 了解并使用对象、方法和属性
- ◆ 使用对象的集合
- ◆ 获取对象、属性和方法
- ◆ 使用对象变量表示对象

本章介绍怎样在使用的软件中获得需要的对象。本章的内容建立在前一章的基础之上。一开始将给出概念：什么是对象和集合，什么是属性，以及什么是方法。然后介绍怎样获取对象、集合、属性以及方法。要了解这些内容，将使用前面介绍的一些工具，包括对象浏览器（在第 4 章中简单介绍过）和 VBA 的帮助文件。

接下来将介绍怎样使用对象变量代表代码对象。

### 什么是对象

什么是对象，使用 VBA 的软件（以及许多其他的现代软件）包括单独的对象，每个对象都有自己的特点和功能。

用对象构建的应用程序叫做面向对象的程序，简写成 OOP。从理论上说面向对象的程序有许多好处，包括代码容易编写和维护，因为代码可以分解为便于管理的对象。

面向对象的程序也比大型板块程序容易理解，因为大多数人容易掌握单独的对象概念以及相关的特点和命令，而不需要记住一长串的功能清单。而且了解命令比较容易。例如，Word 表格被解释为 Table 对象，列被解释为 Column 对象，Column 对象有 Width 属性，用来设定或返回列宽。把这些内容分解为容易理解的片段就比复杂命令（例如 WordTableSet-ColumnWidth 和 WordTableGetColumnWidth）容易掌握。

面向对象的程序第三个优点可以扩展：用户可以构造自己的对象以完成软件没有包括的功能。例如，可以使用 VBA 构造自己需要的对象去实现软件本身不能做的事情。

对象可以而且往往包括其他对象。一般来说，在面向对象软件中的对象形成一个逻辑结构，叫做对象模型。该结构给出对象间的逻辑关系，通过这种关系在 VBA 中可使用对象。

**注意：**本章讨论的对象模型并不深入，只是一般概念。了解什么是对象模型以便于理解后面的章节。本书的第五部分介绍每一个软件的对象模型，便于读者从事更深入的研究。

大多数 VBA 宿主软件以及所有办公软件都具有 Application 对象，用于整体上代表某个软件。Application 对象有与软件功能相关的属性和方法。例如，许多应用程序都具有 Quit 方法（用来退出应用程序），以及 Visible 属性（用来设定软件为显示或是隐藏）。

在标准的对象模型中，Application 对象包括所有构成该软件的其他对象（对象的集合

或分组)。例如, Excel 有 Application 对象代表 Excel 软件, Workbook 对象(组成 Workbooks 集合)代表工作簿, 以及 Worksheet 对象(组成 Sheetss 集合)代表工作表。Workbook 对象包括在 Application 对象中, 因为在使用工作簿的时候, 首先必须打开 Excel 软件。

同样地, Worksheet 对象包括在 Workbook 对象之中, 因为在使用工作表时, 必须打开一个工作簿。再深入讨论对象模型, Worksheet 对象包括有关的其他对象(Row 对象代表工作表的行, Column 对象代表工作的列, 以及 Range 对象代表单元格的范围), 所有这些对象也含有自己的对象。

要获得一个对象, 可通过对对象模型的结构进行查询。例如, 要获得 Excel 中的 Range 对象, 必须由 Application 对象到 Workbook 对象, 由 Workbook 对象到相应的 Sheet 对象, 然后才是 Range 对象。下面的语句使用第一个工作簿中的第一张工作表中的 A1 单元格:

```
Application.Workbooks(1).Sheets(1).Range("A1").Select
```

因为必须通过 Application 对象获取所有其他的内容, 因此, 大多数软件提供许多可生成的对象(即不需要指明 Application 对象就可访问的对象)。这些可生成的对象通常是最常用的对象, 通过这些对象, 就可以直接访问大多数别的对象而不需要经过 Application 对象。例如, Excel 的 Workbooks 集合就是可生成对象, 因此, 用下面的语句可以代替前面的语句:

```
Workbooks(1).Sheets(1).Range("A1").Select
```

任何一个对象都有属性和方法, 下面将详细讨论。

## 属性

在 VBA 中, 属性是对象的特征, 用来描述对象整体或部分。大多数对象有许多属性, 分别用于描述每个对象的相关部分。每个属性都具有特定的数据类型, 用以保存信息。例如, 表示文件的对象(文档、工作簿和演示文稿)具有逻辑属性 Saved, 该值用来表明对象中的所有改变是否已经保存(值为 True)或者未保存(值为 False)。这两个值包括了该对象的所有可能性: 要么包括未保存的改变, 要么不包括未保存的改变, 没有第三种情况。

类似地, 大多数代表文件的对象都有 Name 属性, 包括文件的名称。该属性为字符类型, 因为需要包括文本, 该文本可以为各种形式(限制在 255 个字符以内并且受 Windows 支持, 有些字符(例如冒号或者折弯字符)不能使用)。

使用属性可以获得当前的值或者设定一个希望的值。许多属性为读/写属性, 意思是对它们的值既可以获取也可以设定。然而, 有一些属性为只读, 意思是其值只可以获得不可以设定。

大多数应用程序的 Saved 属性是读/写, 因此可以进行设定。这就意味着, 告诉应用程序, 文件在未保存改变时进行保存, 或者在有改变时不保存。(在不了解用户的情况下处理文件时, 改变 Saved 属性十分有用。)然而, 文件对象的 Name 属性是只读的, 名称通过 Saves 命令设定, 此后, 当该文件打开时不可以从应用程序内部改变其名称。因此, 可以返回 Name 属性而不能设定它。还有一些只写属性, 这些属性只能设定而不能获取。

当一个对象包含另一个对象或者集合时, 该对象就有用来返回对象或集合的属性。例如, Word 的 Document 对象包括 PageSetup 属性, 返回 PageSetup 对象(PageSetup 对象包括纸的尺寸、方向、页边距), 以及 Tables 属性返回 Tables 的集合。

每一个相同类型的对象具有相同的属性, 但可以保存它们各自的值。例如, 运行 Pow-

erPoint，打开三个 Presentation 对象，每一个有不同的 Name 属性。在一个对象中设定属性和另一个对象中的设定无关，对象彼此之间是独立的。

## 方法

方法是对象执行的动作，在同类型的对象中方法可以共享。例如，不同应用程序中的 Document 对象有 Save 方法，用来保存文档。可对不同的 Document 对象使用 Save 方法：Documents (1) . Save 保存 Documents 集合中的第 1 个 Documents 对象，Documents (2) . Save 保存 Documents 集合中的第 2 个 Documents 对象，但是每次的动作相同。对象可以有一个或者多个与之相关的方法。有些对象有许多方法以完成所需的所有功能。

Save 方法出现在许多应用程序中，同样的方法还有 SaveAs（将文件保存为不同的名称、地点等）、Close（关闭文件），然而其他一些方法是特有的。例如，在 PowerPoint 中的 Presentation 对象有 AddBaseline 方法，用来加底线（针对当前演示文稿或一个指定的文稿），可以追踪内容合并所做的改变。Word 中的 Document 对象没有 AddBaseline 方法，但却有 AcceptAllRevisions 方法，用来保存文档中所有的修改。PowerPoint 没有 AcceptAllRevisions 方法。

有些方法关联一个以上的对象。例如，Delete 方法关联许多对象。顾名思义，Delete 方法通常用于删除特定的对象。然而，其他一些方法根据对象实施不同的动作。因此，即使熟悉某个对象的该方法，但当该方法用于另一个对象时，还必须了解其功能。

有些方法需要一个或多个提供信息的参数，其中有些参数是必需的，而有些参数是可选的。有些方法没有参数。如果一个方法用于多个对象，不同的对象语法也不同。即使熟悉某个方法，也必须准确了解针对该对象的作用。

要使用一个方法，必须通过相应的对象访问它。例如，要关闭 ActivePresentation 对象，即关闭 PowerPoint 中当前演示文稿，必须使用 Close 方法。

`ActivePresentation.Close`

## 构建对象、方法和属性

如果不能掌握对象、方法和属性，可以做一个有点牵强的对比，把 VBA 中的逻辑对象、属性和方法与现实中的物理对象、属性和动作进行对比。举例如下。

有一条大狗，比利牛斯山犬，白色、200 磅；雄性、无、上部、拴住；4 岁；命名 Max。

Max 的动作和其他狗一样，例如睡、跑、吃、吠、咆、咬等。还有些动作不同寻常，例如听到命令流口水、扑人、威胁警察。

如果把 Max 放入 VBA，应该是狗集合中的狗对象。Max 的狗对象应有以下属性：

属性	属性类型	值
名称	只读字符串型	Max
性别	只读字符串型	公
拴住	读写逻辑型	假
高度	读写长整型	36
重量	读写长整型	200
年龄	读写整型	4

类型

读写字符型

比利牛斯山犬

颜色

读写字符型

白色

Max 的方法有流口水、吠、扑、威胁、嚼、跑等。有些方法需要参数，例如流口水方法必须有参数，采用特定的常数，并以特定标志开头。

```
Dogs("Max").Slobber OnWhat:="MyKnee", How:=dogSlobberDisgustingly
```

狗对象包括许多组织对象，耳、眼、舌、脑、腹、腿、尾等。每个对象又有自己的属性和方法。例如，尾对象需要摇方法，可包括下面的内容：

```
Dogs("Max").Tail.Wag Direction:=dogWagHorizontal, Frequency:=200
```

## 集合

当一个对象包括一个以上相同类型的对象时，这些相同类型的对象就组成集合。例如，Word 软件有 Documents 对象，Documents 对象组合起来成为 Documents 集合；PowerPoint 有 Presentation 集合；Excel 有 Workbooks 集合。

在这些例子中，集合对象的名称是对象加复数，也有许多例外，例如，Excel 中的 Sheets 集合是 Worksheet 对象的集合；然而，一般情况下集合的名称可以根据单个对象得到，反之亦然。

集合也是对象，可以有自身的属性和方法。大多数集合的属性和方法比单个对象少。大多数集合都有 Add 方法（用来在集合中增加对象）以及 Item 属性（默认属性，用来访问一个项目）。有些集合却是只读的，没有 Add 方法。

在 VBA 中，大多数集合具有如表 8.1 所示的核心属性。

表 8.1 VBA 中集合的核心属性

属性	说明
Application	只读属性，返回与对象或集合有关的软件，是对象结构的根。例如，在软件 PowerPoint 中，Application 属性返回微软的 PowerPoint
Count	只读长整型属性，返回集合中项目的数量。例如，在 PowerPoint 的 Shape 对象集合中返回图形的数量
Creator	只读长整型属性，返回 32 字节的整数，代表生成对象或集合的软件
Item	只读属性，返回集合中的某成员。Item 是每个集合的默认属性，这意味着一般不需要设定
Parent	只读字符型属性，返回对象或集合的父对象。父对象是包含对象的对象，被包含的对象称为子对象

## 集合中的对象

要操作集合中的对象，必须针对集合中的对象要么使用名称要么使用地址。例如，下面的语句返回 Documents 集合中的第一个 Documents 对象，并且在信息框中显示其 Name 属性。

```
MsgBox Documents(1).Name
```

可使用 Item 属性从集合中返回对象，因为 Item 是集合的默认属性，所以可以不使用。

下面两条语句的作用相同，使用 Item 方法并没有优势。

```
strName = Documents(1).Name  
strName = Documents.Item(1).Name
```

**注意：**在大多数集合中，计数从 1 开始，这样很容易识别对象。例如，Documents (1) 代表第一个文档，Workbooks (2) 代表第二个工作簿，等等。办公软件中大多数集合从 1 开始计数。必须注意，有些集合计数从 0 开始而不是 1。Access 软件中集合就是从 0 开始计数。如果不能确定某个集合是从 1 开始计数还是从 0 开始，可以查寻集合的帮助文件。

## 在集合中添加对象

要在集合中产生一个新的对象，需要向该集合添加对象。在许多情况下，可以使用 Add 方法。例如，下面的语句生成了新的 Documents 对象：

```
Documents.Add
```

## 查寻需要的对象

Visual Basic 编辑器提供了许多工具用来查寻对象：

- ◆ 宏录制器（只限于微软软件）：在第 1 章中介绍过，用来录制宏。
- ◆ 对象浏览器：在第 4 章中简单介绍过。
- ◆ 联机帮助系统：针对对象提供的详细帮助（取决于软件开发商在帮助系统中提供的信息）。
- ◆ 属性清单/方法功能。

下面将介绍怎样使用这些工具查找对象。

## 使用宏录制器录制对象

如果使用微软的软件，查找对象最简单的方法是使用宏录制器录制需要使用的对象。录制命令的时候，宏录制器生成了代码，这些代码可以在宏编辑器中打开并修改。

除了这个优点，宏录制器有两个问题：

- ◆ 首先，不能把所有需要的命令都录制下来。例如，在使用 Excel 时，希望针对一个具体的工作簿生成命令代码，该工作簿属于 Workbooks 集合而不是当前的工作簿。然而，使用宏录制器，只能录制针对当前工作簿的指令。（实际情况正是这样，因为宏录制器只能录制和 Excel 交互的命令，而实际上不可能与非当前的工作簿进行交互。）
- ◆ 其次，宏录制器录制的语句往往不能和期望结果一一对应，特别是在对话框中录制设定信息时。

上述第二个问题可在第 4 章中见到。下面再举一个例子，这次录制一个自动更正的宏。

1. 启动 Word，或者如果已经启动，应将其设定为当前活动窗口。
2. 双击状态栏上的录制标志，或者选择“工具”>“宏”>“录制新宏”以显示“录制宏”对话框。在“宏名称”中输入 Add\_Item\_to\_AutoCorrect，在“说明”中输入解释，在“将宏保存在”下拉列表中选择“所有文档（Normal.dot）”，然后单击

“确定”按钮开始录制。

3. 选择“工具”>“自动更正”项以显示“自动更正”对话框。在“替换”栏中输入reffs，在“替换为”栏中输入references，然后单击“确定”按钮关闭“自动更正”对话框。
4. 单击“停止录制”按钮，或者选择“工具”>“宏”>“停止录制”以停止宏的录制。然后，按Alt+F8键或者选择“工具”>“宏”>“宏”以显示“宏”对话框，选择Add \_ Item \_ to \_ AutoCorrect，单击“编辑”按钮，打开宏编辑器，代码如下：

```
Sub Add_Item_to_AutoCorrect()
```

```
' Add_Item_to_AutoCorrect Macro
```

```
' 增加一条自动更正项目
```

```
AutoCorrect.Entries.Add Name:="reffs", Value:="references"
```

```
With AutoCorrect
```

```
    .CorrectInitialCaps = True
```

```
    .CorrectSentenceCaps = True
```

```
    .CorrectDays = True
```

```
    .CorrectCapsLock = True
```

```
    .ReplaceText = True
```

```
    .ReplaceTextFromSpellingChecker = True
```

```
    .CorrectKeyboardSetting = False
```

```
    .DisplayAutoCorrectOptions = True
```

```
    .CorrectTableCells = True
```

```
End With
```

```
End Sub
```

这样针对一行语句出现13行补充内容：

```
AutoCorrect.Entries.Add Name:="reffs", Value:="references"
```

这行语句的意思是：要增加一个自动更正项目，必须使用AutoCorrect对象的Entries集合对象。用Entries集合的Add方法增加一个自动更正项目。

删除包含在With…End With语句中的9行命令后，就可以得到真正有用的语句（注释语句也可以删去）。

```
Sub Add_Item_to_AutoCorrect()
```

```
' Add_Item_to_AutoCorrect Macro
```

```
' 增加一条自动更正项目
```

```
AutoCorrect.Entries.Add Name:="reffs", Value:="references"
```

```
End Sub
```

尽管有不足之处，宏录制器可以提供需要使用的对象，而且可以在Visual Basic编辑器中修改宏命令。

## 使用对象浏览器

查找对象的基本工具是对象浏览器，它在第 4 章中曾经使用过。这里将详细介绍可用它来查找需要的对象信息。

### 对象浏览器的组成

对象浏览器可对内置对象和自定义对象提供以下信息：

- ◆ 类（对象的正式定义）；
- ◆ 属性（对象的特点）；
- ◆ 方法（对象的动作）；
- ◆ 事件（例如打开或关闭文档）；
- ◆ 常数（在程序执行中保持不变的信息）。

图 8.1 介绍了对象浏览器的组成。下面详细介绍对象浏览器的功能。

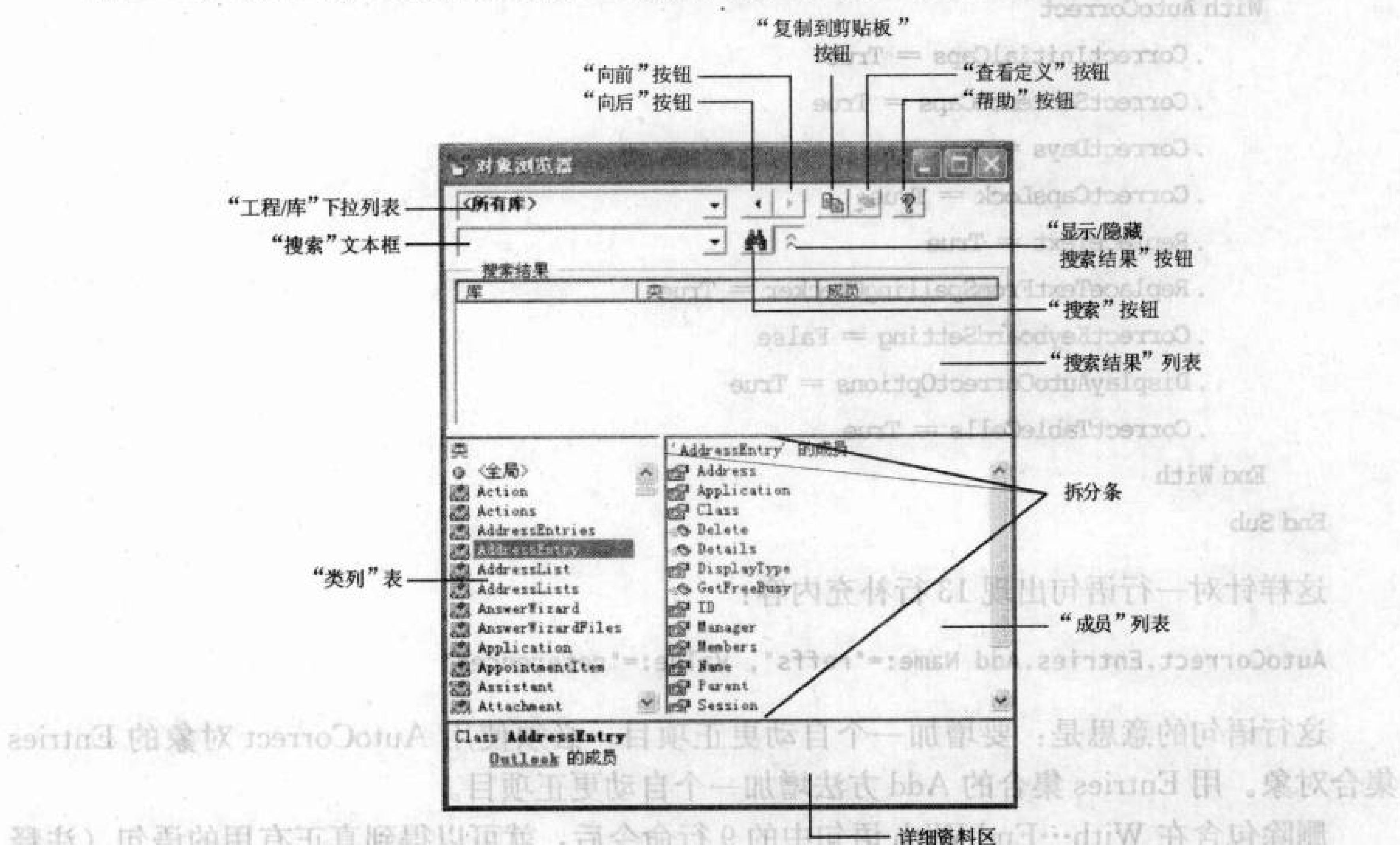


图 8.1 对象浏览器提供内置对象或自定义对象的信息，本图为软件 AutoCAD 的对象

- ◆ “工程/库”下拉列表提供当前工程所有的对象。（对象库是一个参考文件，包括所有的对象信息。）使用该下拉列表，可选择需要使用的对象库。例如，如果只想了解 Outlook 的对象，选择列表中的 Outlook。其默认选项为所有库。
- ◆ 在“搜索”文本框中输入需要搜索的信息，或者在当前工程中选择一项，然后回车，或者单击“搜索”按钮，可以得到搜索对象成员。

**提示：**要使搜索内容不要太具体，可以使用通配符？（代表一个字符）或者\*（代表一群字符）。可选择查找一个完整的单词，方法是在对象浏览器中（避开“工程/库”下拉列表或“搜索”文本框）单击右键并选择“全字匹配”，在

“全字匹配”旁出现√，表示“全字匹配”功能激活，再选择“全字匹配”可取消该功能。

- ◆ 单击“向后”按钮可以在类列表和成员列表中跟踪每一个查寻的选项。单击“向前”按钮，可以把每一个查寻的选项按顺序重复。“向后”按钮在进入类或成员时才出现，“向前”按钮在使用过“向后”按钮以后才出现。
- ◆ 单击“复制到剪贴板”按钮，可从“搜索结果”列表、“类”列表、“成员”列表中复制选择的项目，然后粘贴需要的代码。
- ◆ 单击“查看定义”按钮，可显示代码窗口，其中包括“类”列表或“成员”列表中被选择对象的代码。“查看定义”按钮只能用于包括代码的对象（例如，过程、自定义窗体）。
- ◆ 单击“帮助”按钮以显示被选项目的帮助信息。也可以使用 F1 键。
- ◆ 单击“搜索”按钮，就可以搜索“搜索”文本框中输入的信息。如果“搜索结果”窗口未打开，VBA 将会在此时打开。
- ◆ 单击“显示/隐藏搜索结果”按钮，可显示或隐藏搜索结果。
- ◆ “搜索结果”列表包括最新的搜索结果。如果完成了一个搜索，当切换到另一个库时，对象浏览器将更新搜索结果。选择不同的库可以随意改变搜索范围。
- ◆ “类”列表用于显示库或工程中的类。
- ◆ “成员”列表显示“类”列表中被选类的成员。方法、常数、事件、属性以及含有代码的程序用粗体显示。“成员”列表可以按分类（方法、属性、事件等）显示成员或者按字母顺序显示成员。另外，可以对是否按分类显示进行设定，右击“成员”列表，选择“组成员”可以决定是否按分类显示。
- ◆ “详细资料区”显示被选成员的定义。例如，如果选择了“成员”列表中的程序，那么“详细资料区”会显示其名称以及程序保存的模块、模板或文档的名称，还包括程序开始的注释。模块名和工程名包括超链接，因此，可以迅速寻找。可以通过复制、粘贴或拖曳将“详细资料区”中的信息复制到代码窗口中。
- ◆ 可以在对象浏览器中改变三个窗口栏的尺寸。（也可以改变“对象浏览器”窗口，或者使其最大化。）

对象浏览器使用不同的图标显示不同的对象。图 8.1 显示了几种图标；表 8.2 给出所有的图标以及说明。

表 8.2 对象浏览器的图标

图标	说明	图标	说明
	属性		用户定义类型
	方法		全局变量
	常数		库
	模块		项目
	事件		内置关键字或类型
	类		枚举

在“属性”或“方法”图标的左上角有一个蓝色圆点，表示该属性或者方法是默认的。

## 增加和删除对象库

选择“工具”>“引用”，打开“引用”对话框，从中可以增加或删除对象库。

◆ 通过增加对象库，可以使用加入的对象库。

◆ 移去不需要使用的对象库，可以减少 VBA 在编译时对象引用的数量，程序运行更快。

Visual Basic 编辑器在使用时将自动加载用于 VBA 的对象库和宿主软件的用户窗体，并不需要改变对象库的设定，除非希望访问其他软件的对象。例如，如果在 Word 软件的程序中调用 Excel 的功能，就需要在 Word 中增加对 Excel 的引用，使其对象出现。

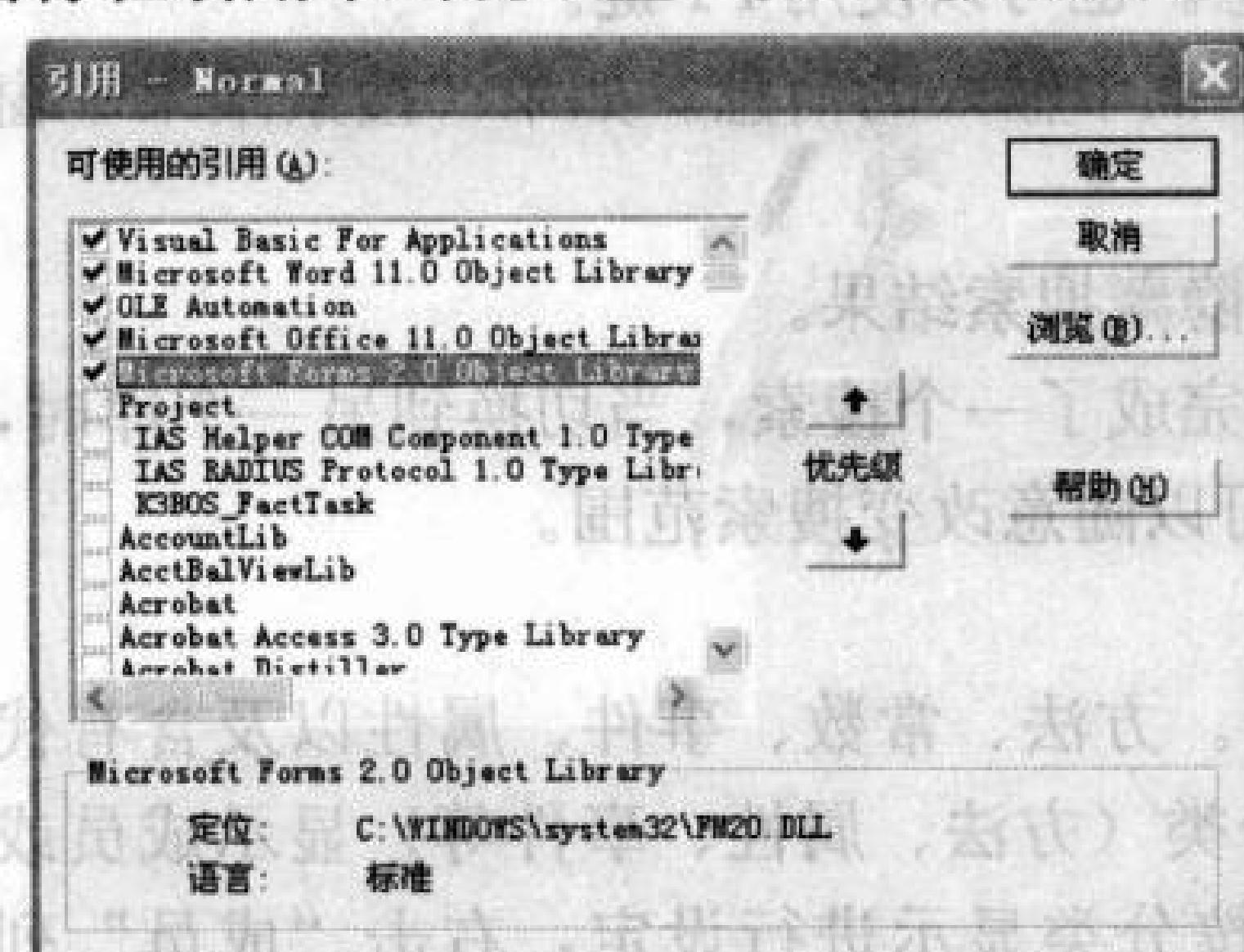
在“引用”对话框中出现的引用可以调整其优先顺序。如果在程序中使用的一个对象的名称在引用中出现不止一次，优先顺序就很重要。VBA 根据引用列表决定引用对象的顺序，选用第一个出现的对象，除非使用的名称不会产生歧义才会另外列出。

要增加或删除对象库，可按下面的步骤进行：

1. 在 Visual Basic 编辑器中，选择“工具”>“引用”以显示“引用”对话框（参见图 8.2）。

**提示：**在对象浏览器中右击鼠标并选择“引用”，也可以显示“引用”对话框。（在对象浏览器中右击任何一处均可：“工程/库”下拉列表、“搜索”区域、“类”列表、“成员”列表或者详细资料区。）

图 8.2 用“引用”对话框增加或删除对象库。标题栏中包括“Normal”是因为当前项目是 Word 的 Normal 模板



2. 在“引用”对话框中，选中需要使用的引用，把不需要的引用取消选中。这时，可以见到每个支持 VBA 并已经安装的软件的对象引用。使用的引用一起出现在“可使用的引用”列表框的顶端，而非按字母排列。
3. 若有必要，可调整引用的优先级：选择一项引用并用“优先级”按钮上下移动。通常，VBA 和应用程序的对象库应放在前面。

**提示：**可以添加其他引用库：单击“浏览”按钮打开“添加引用”对话框，从中选择一个库文件，然后再单击“打开”按钮即可。

4. 单击“确定”按钮关闭“引用”对话框并返回到对象浏览器。

## 用对象浏览器浏览

浏览工程的对象，应按下列步骤操作：

1. 在工程窗口中双击代码模块以激活它。
2. 显示对象浏览器：选择“视图”>“对象浏览器”，或按 F2 键，或者在“标准”工具栏上单击“对象浏览器”按钮。（若对象浏览器已经显示，单击或在 Windows 的底部选中使其成为当前活动窗口。）
3. 在“工程/库”下拉列表中选择工程或者想要看的库的名称。对象浏览器将在“类”

列表中显示类。

4. 在“类”列表中选择类。例如，如果在第3步中选择工程，那么在“类”列表中选择模块。
5. 如果使用工程或类的某个成员，那么在“成员”列表中选择。例如，如果使用模板工程，那么可以选择一个具体的程序或用户窗体。

一旦选择了类、成员或工程，可执行以下操作：

- ◆ 在对象浏览器的底部查看有关信息。
- ◆ 单击“查看定义”按钮查看对象的定义。或者右击对象名称，从出现的菜单中选择“查看定义”。“查看定义”按钮或“查看定义”命令只针对包含代码的对象，例如过程或用户窗体。

**注意：**过程的定义指过程包含的代码。模块的定义指模块包括的所有过程和代码。

用户窗体的定义包括所有该窗体附着的过程和代码。

- ◆ 单击“复制到剪贴板”按钮或者使用“复制”命令（ $Ctrl+C$  键或者  $Ctrl+Insert$  键），把选中的类、工程或成员复制到剪贴板上。

## 使用帮助查找对象

VBA 的帮助系统提供另一种简便的方法查找对象。帮助文件对所有的对象、方法、属性提供超链接的引用，包括表示对象关系的图形。

进入 VBA 帮助系统最快的方法是激活 Visual Basic 编辑器，然后按 F1 键。VBA 显示“Visual Basic 帮助”任务窗格，如图 8.3 所示，从中可以看到一些主题的目录。如果正在使用某个办公软件，而且已经停止使用办公助手，可以选择“帮助”>“Microsoft Visual Basic 帮助”；如果未停止使用办公助手，可以选择“帮助”>“Microsoft Visual Basic 帮助”以显示办公助手。

**提示：**要获得某个特定内容的帮助，将插入点放在有关单词附近，然后按 F1 键。

这样可使 VBA 显示该项目的帮助主题。

一旦打开帮助任务窗格，可在“搜索”框中输入关键词，并单击“开始搜索”按钮，搜索需要的帮助，或者浏览搜索主题。单击主题，将打开“Microsoft Visual Basic 帮助”窗口。图 8.4 为使用帮助的例子，它根据 Word 中的 Document 对象显示主题。

除了通用的可在帮助窗口中获得的帮助信息外，还有些项目值得再次一提：

- ◆ 帮助列表的顶端有图形，表明当前对象（本例为 Document）与该对象和其他对象的关系，其他对象指包括该对象的对象和该对象包括的对象。可以单击这些对象中的任何一个以显示弹出的列表，表明相关的对象，参照图 8.5。
- ◆ 如果“参阅”出现在窗口顶部，那么可单击它显示一些相关的主题。例如：若在 Document 对象的帮助窗口中单击“参阅”超级链接，可以见到有关模版对象的帮助。
- ◆ 在窗口的顶部单击“属性”超级链接，可显示该对象属性的帮助信息，然后单击需要的主题。
- ◆ 在窗口的顶部单击“方法”超级链接，可显示对象方法的帮助，然后单击需要的主题。

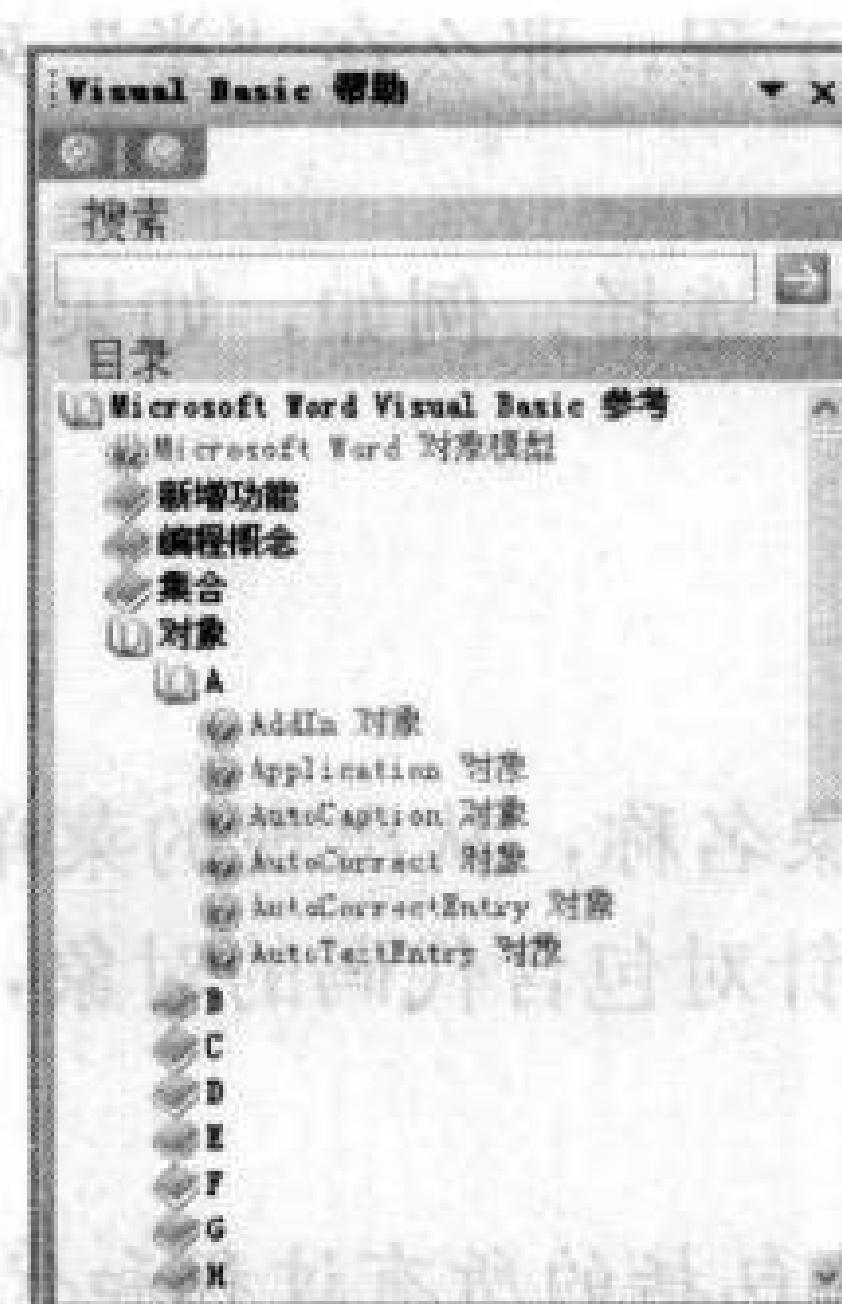


图 8.3 “Visual Basic 帮助”任务窗格

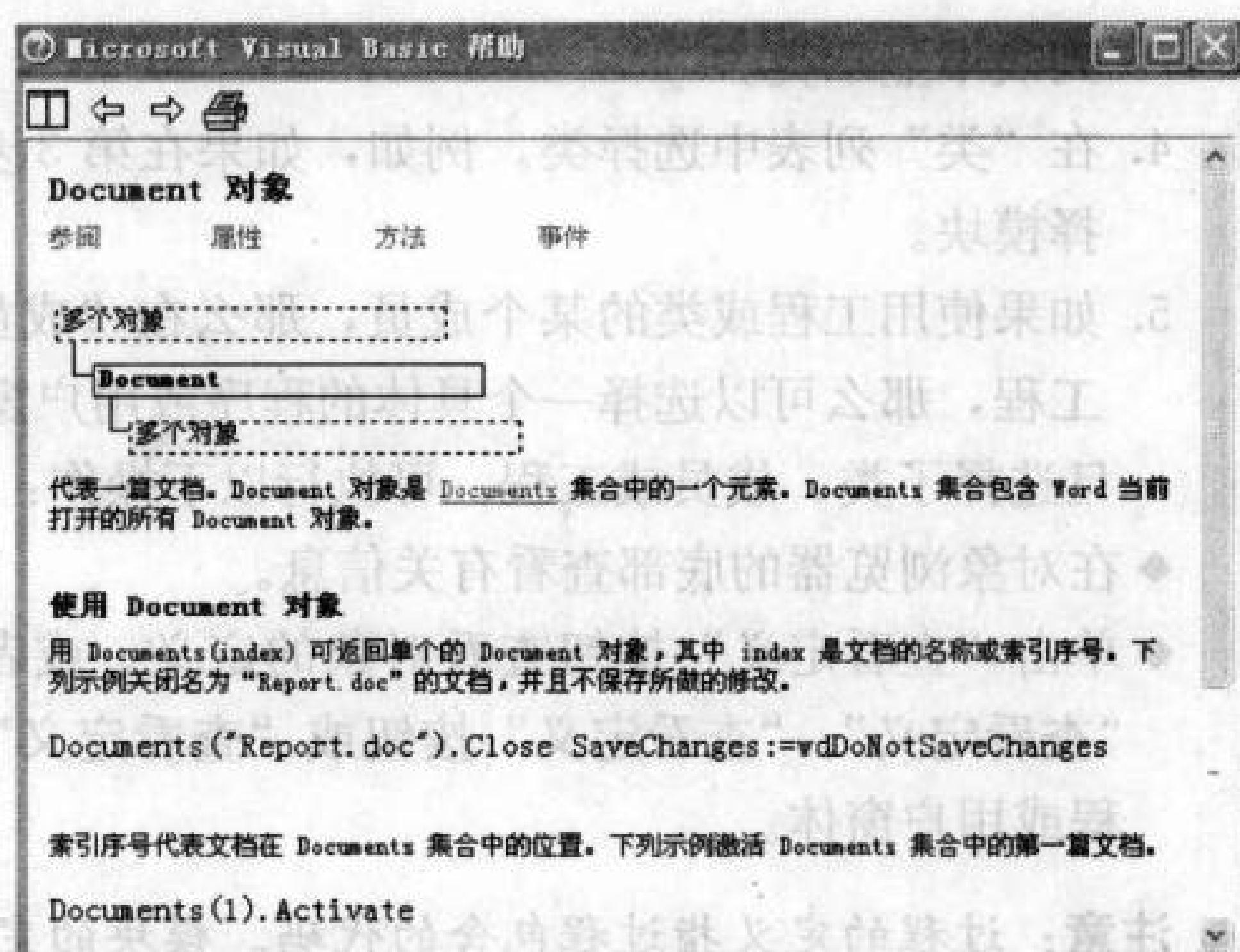


图 8.4 Word 中的“Document 对象”帮助文件

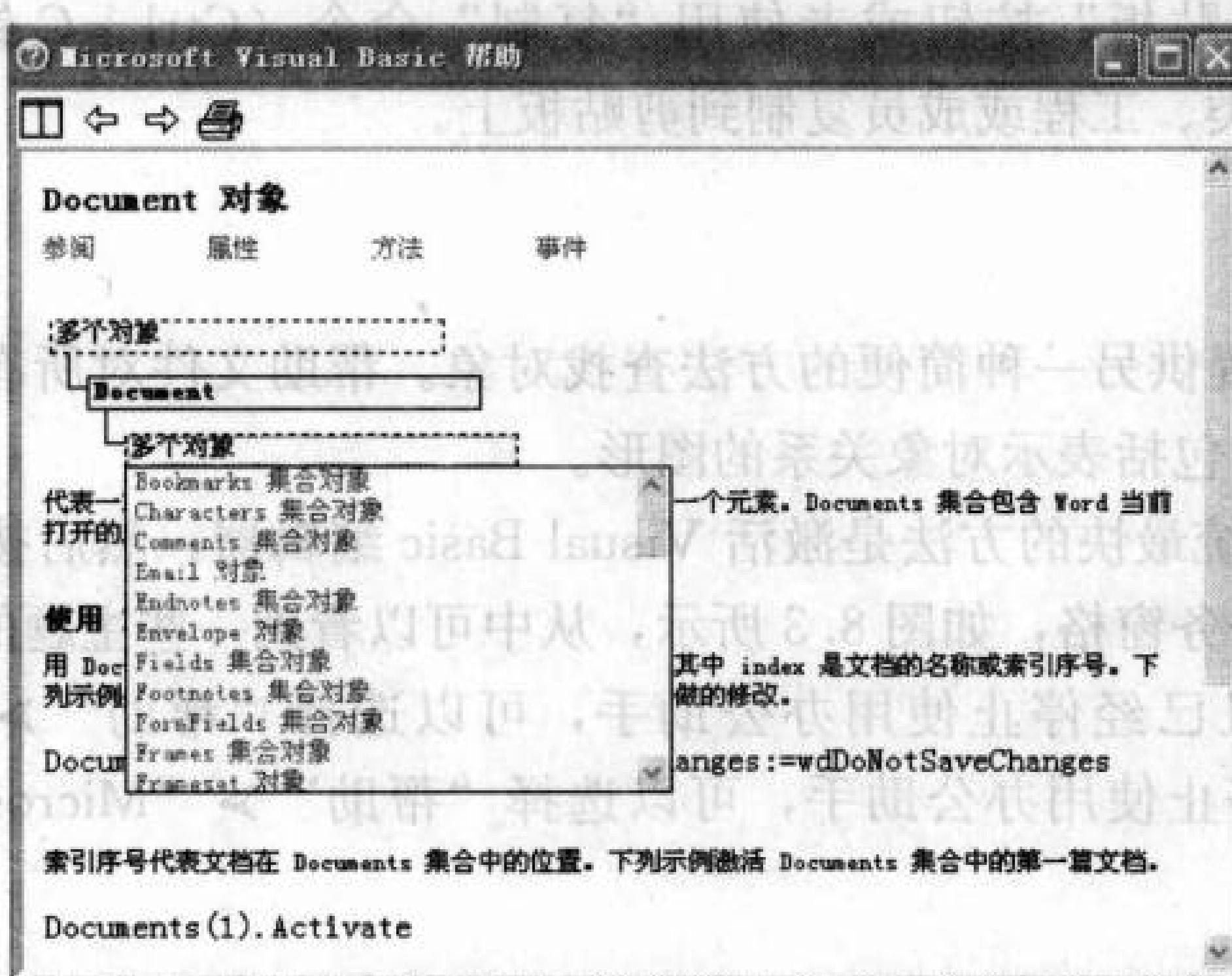


图 8.5 单击某个对象图案，可显示对象列表。此时“Document 对象”包括许多其他对象，例如“Bookmarks 集合对象”和“Characters 集合对象”

◆有些对象还有若干事件。如果该对象有事件（例如 Document 对象），在窗口的顶部单击“事件”超级链接，可显示对象事件的帮助，然后单击需要的主题。

## 使用列表属性/方法的特色

在前面的章节中已经多次介绍列表属性/方法的特色。这里再说一遍，在 Visual Basic 编辑器中输入一条语句时，在当前对象的末尾输入句点，列表属性/方法特色将显示和该语句有关的属性和方法。

列表属性/方法特色提供快速的输入方法，然而必须了解该对象的起始点。有时候使用该特色有点像在迷宫里探路，得到许多路标，最后却写着“此路不通”。

如果了解对象的起始点，就可以很容易找出属性或方法。例如：要得到 Application.Documents(1).Close 语句以关闭 Documents 集合中的第一个文档，应按下列方法操作：

1. 将插入点放入空过程的起始行（在 Sub 和 End Sub 语句间）。若有必要，生成一个新

的过程。

2. 输入单词 Application, 或者 Appl, 然后按 Ctrl+空格键以便使用自动填写功能使该单词完整。
3. 在 Application 后输入英文句号。列表属性/方法特色将显示 Application 对象的所有属性和方法。
4. 在列表属性/方法列表中选择 Documents 项。可以用鼠标向下滚动, 然后双击进入代码窗口, 或者用↑和↓键上下滚动, 并通过按 Tab 键输入, 也可以输入该名称的头几个字母, 然后使用 Tab 键输入。后面的方法参见图 8.6。
5. 在 Documents 之后输入(1)。列表属性/方法特色将显示 Documents 集合的属性和方法。
6. 在列表属性/方法清单中用鼠标或↓向下滚动, 选择 Close 方法。因为这是语句的结尾, 按下 Enter 键另起一行, 不需要使用 Tab 键 (Tab 键输入后保持在原行)。

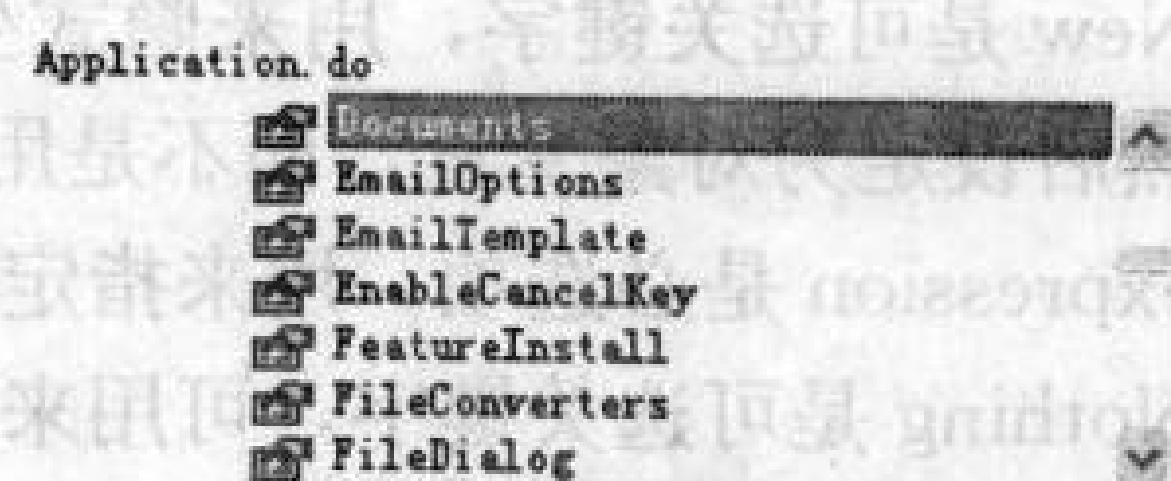


图 8.6 使用属性/方法列表输入代码

**提示:** 在代码窗口中输入语句的最快方法是, 手不要离开键盘。为此, 输入句点或者括弧后, Visual Basic 编辑器将自动在列表属性/方法中选择当前项目。在前面的例子中, 输入 Application. 可显示列表, 输入 Do 可选择文档项目。

## 使用对象变量代表对象

第 6 章中已介绍过, VBA 的一个变量的数据类型是 Object。使用对象变量代替一个对象是指: 不直接引用该对象, 而使用对象变量访问其代表的对象。

使用对象变量更容易了解代码所处理的对象, 特别是在用同样的代码处理多个对象时。例如: 有一段程序同时处理三个打开的 Excel 工作簿, 从一个工作簿中将一些单元格复制到另外两个工作簿中去。如果仅仅打开了这三个工作簿, 就可以分别引用。写成 Workbooks(1)、Workbooks(2)、Workbooks(3), 因为这些工作簿在 Workbooks 集合中占有前三个位置。

然而, 如果程序改变了工作簿的顺序, 关闭一个或多个工作簿, 或者生成一个和多个新的工作簿, 情况立刻就会变得混乱起来。但是如果使用了对象变量 (例如, 命名为 xlWorkbook1、xlWorkbook2、xlWorkbook3) 去引用这些工作簿, 情况就简单得多。这是因为不论在 Workbooks 集合中哪个工作簿处在第一的位置, 总可以用对象变量 xlWorkbook1 进行引用, 并了解将要使用的工作簿。

生成对象变量的方法和声明其他变量的方法相同, 直接采用 Dim、Private 或者 Public 语句。例如, 下面的语句声明了 objMyObject 对象变量:

```
Dim objMyObject As Object
```

正如 Dim 语句所示, 如果在过程中声明, 该变量为过程范围; 如果在代码页的声明区声明, 变量为模块级。类似地, Private 和 Public 关键词生成模块级的私有和公共对象变量。一旦声明了对象变量, 就可以对该变量赋值。赋值时使用 Set 语句。Set 语句的语法如

下，

```
Set objectvariable = {[New] expression|Nothing}
```

以下是上述语法的详细说明：

- ◆ objectvariable 是对象变量的名称，用来给对象赋值。
- ◆ New 是可选关键字，用来隐式生成新的指定类型的对象。通常应当显式生成对象，然后设定为对象变量，而不是用 New 关键字隐式声明。
- ◆ expression 是必选项，用来指定或返回需要赋值的对象。
- ◆ Nothing 是可选关键字，可用来对对象变量赋值，清除其内容，并释放内存空间。

例如，下面的语句声明 objMyObject 变量，并赋值为 Excel 的当前工作簿：

```
Dim objMyObject As Object
Set objMyObject = ActiveWorkbook
```

下面的语句使用 Nothing 关键字释放 objMyObject 对象变量占用的内存空间：

```
Set objMyObject = Nothing
```

声明对象变量和声明其他类型的变量的区别在于：声明对象变量时，不仅可以将对象变量声明为对象类型，而且还可规定具体的类型。例如，如果某个对象变量始终代表 Workbook 对象，就可声明为 Workbook 数据类型。下面的语句将 xlWorkbook1 对象变量声明为 Workbook 数据类型。

```
Dim xlWorkbook1 As Workbook
```

像这样明确地规定对象变量的类型有两个好处。首先，一旦输入对象变量，Visual Basic 编辑器可提供该变量的所有帮助，就好像直接使用该对象那样。例如，一旦声明 xlWorkbook1 对象变量为 Workbook 对象类型，只要在该对象变量的名称后输入句点，Visual Basic 编辑器将提供列表属性/方法下拉列表，如图 8.7 所示。

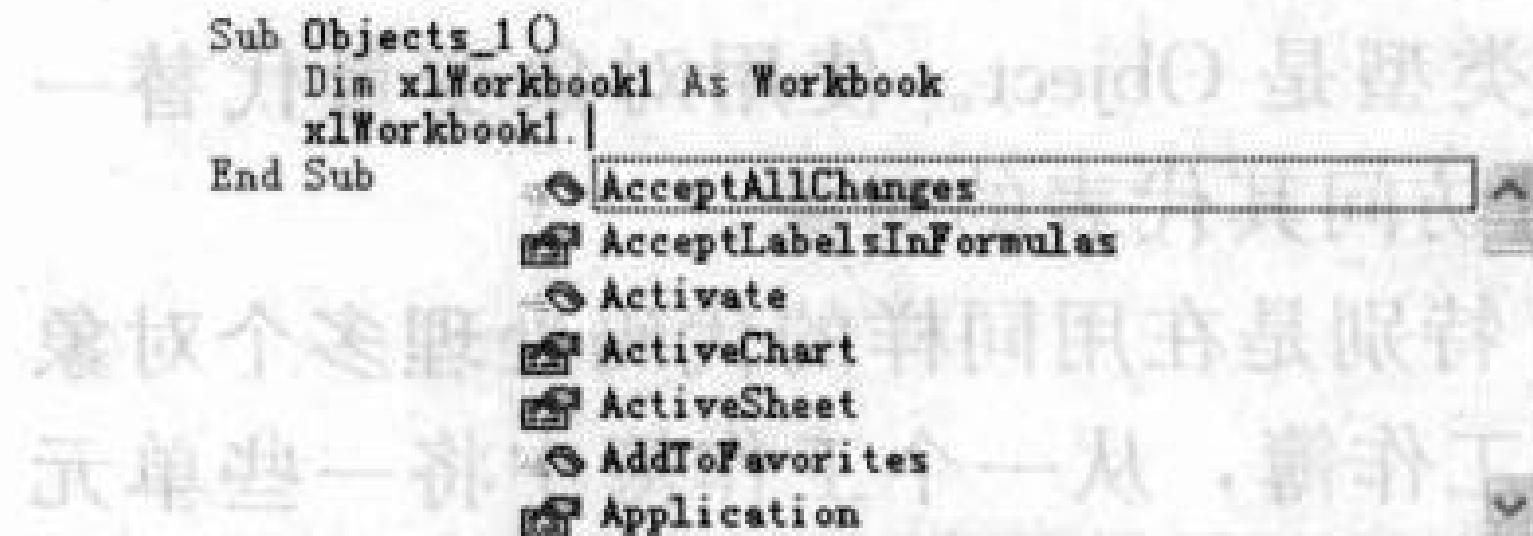


图 8.7 明确声明对象变量时，VB 编辑器的代码输入功能可充分体现

第二，如果明确规定对象变量的类型，那么在代码中就不容易产生错误。如果赋值时的对象类型不正确，那么 VBA 会给出错误提示。例如，如果在 Excel 中创建 Worksheet 对象变量并将其赋值为 Workbook 对象，那么 VBA 会给出类型错误的提示，具体代码如下：

```
Dim wksSheet1 As Worksheet
Set wksSheet1 = ActiveWorkbook
```

在这个阶段发现问题通常比以后发现要好（例如，处理 wksSheet1 对象并且发现这不是你需要的结果时）。

不能给出对象类型的主要原因是不能确定对象的类型，或者在代码执行中对象的类型会变化。（无论哪种情况，代码必须能够适应不同类型的对象要求。）通常，应该给出对象变量的类型。

- 如果不能确信对象变量为哪一种类型，刚开始可将对象变量声明为 Object 数据类型，

然后运行几次代码并将“本地窗口”打开，注意 VBA 分配给对象变量的数据类型。例如，下面的语句是在 Excel 的 Visual Basic 编辑器中运行的，“本地窗口”起初将对象变量 wks 声明为 Object（如图 8.8 左图所示），然而第二条语句将其设定为当前工作簿的第一张工作表（如图 8.8 右图所示）

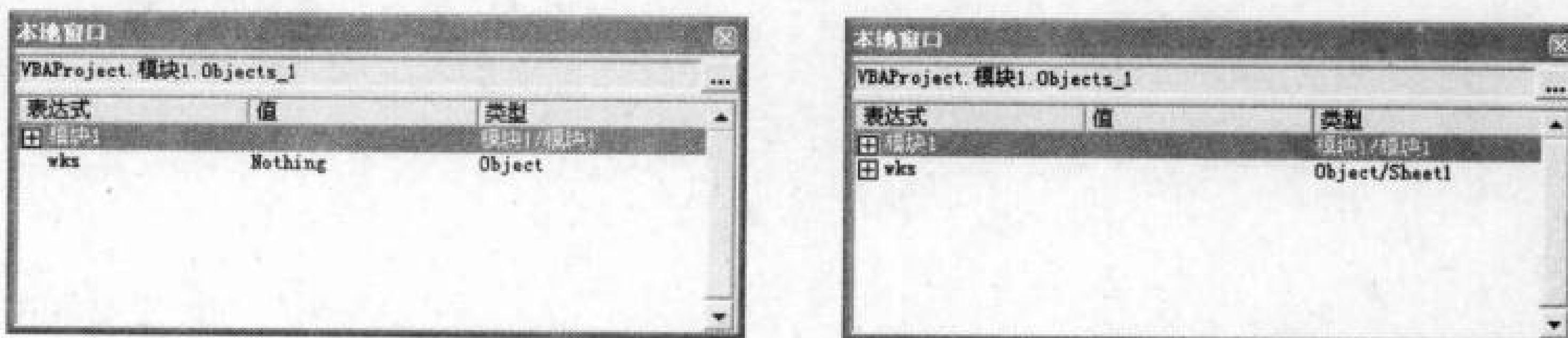


图 8.8 可使用“本地窗口”辨别对象变量的类型

```
Dim wks As Object  
Set wks = ActiveWorkbook.Sheets(1)
```

**注意：**正如本书前面提过的，可以避免设定数据类型，例如，Dim varMyVariant 语句声明不定型变量是因为没有规定数据类型。不定型变量也可以存储对象。然而如前所述，使用不定型变量需要更多的时间（因为 VBA 必须了解当前变量的数据类型）而且不再有声明变量类型的优点。不声明类型也会使代码不容易读懂。

# 第三部分 指令决策、使用循环与函数

甲子其只函数  
类函数  
ABV 函数  
递归函数  
冒日脉脊字数

这回讲一讲什么是函数。所谓函数，就是指将一个值或表达式作为参数输入，返回一个结果。

- ◆ 第 9 章 使用函数
- ◆ 第 10 章 编写自定义函数
- ◆ 第 11 章 用代码决策
- ◆ 第 12 章 使用循环重复执行

递归

函数

（薛公道不善故也）前一个回函类微函数，于右册长于千同不善函，右函将一星矮函，矮矮高以何也？知者莫非知者长于，矮矮要高不善函坐首。矮这个函，一官要高升且而某官长于一高士焯印通，而然，前函是前一个回函类微函坐首，一矮者矮函，不属对矮一。

。薛公道不守，矮函随口自编卷五进，其主矮函 `big` 用章「策」，时间。直到巨源前文章本正奇，如要重合十景中 ABV 之矮函。

：矮矮御变矮时解体矮函 `inf` 且而，前函 `getArea` 用 ABV 时矮函来用，矮函真

`infArea(x) = true(you * 10)`

（甲寅未干涸帕面面印，矮矮数印个一官矮函 `big`）。矮矮要高不，矮函的果述不个是 `big` 基里玄，矮矮御变矮矮矮函好七画四印，失大寺对首造大矮函，矮矮高要高矮函 `inf`，长民直矮函窄唇，半露合回包首印冲矮函 `inf`，直到用矮矮函 `inf` 丁回返矮函 `big`。失去未用。此矮已取矮冬斯公牒，是薛公用变矮参果故。忘首函用刻中今愈变去改，矮函第 ABV 量矮矮。

：不咬去舌帕矮函 `big`，`big`，矮矮御印武

`big([x, y])` 定义

其相片矮函的同不。调薛单早首御回返即更 `Area`，防数回景“字矮”附中导卦衣 `inf` 脚中章「策」，时间。矮函否「虽」，说矮矮字不回返矮函逐书。矮类矮矮函同不回返又含于甲戌头脚遇解一民脚类并果吉纯的矮函称真 ABV，更及育苦，直群数回承（矮函 `numeric` 函数）。中失去未用矮函个一民

# 第9章 使用函数

- ◆ 函数及其作用
- ◆ 怎样使用函数
- ◆ 使用主要的 VBA 函数
- ◆ 数据类型转换
- ◆ 处理字符和日期

VBA 有大量的内置函数用于处理常用的事务，例如，判断一个文件是否存在，返回当前的日期，或者将数据从一个类型转换成另一个类型（用函数将数值型数据转换为字符型数据）。本章将介绍函数及其作用以及怎样使用函数，其中将给出一些关键的函数，包括用来转换数据类型的函数、文件处理函数、日期函数以及数学计算函数。

本章还将介绍怎样生成自己的函数。下一章将介绍在 VBA 函数不能满足要求时应当怎样做。

## 函数

函数是一种程序。函数不同于子过程在于：函数始终返回一个值（子过程不返回值），而且往往需要有一、两个参数。有些函数不需要参数，子过程在需要的时候也可以有参数。一般情况下，函数得到一些信息经过处理返回一个希望的值。然而，也可以生成一个执行某些任务的自己的函数，它不返回值。

函数在 VBA 中是十分重要的，这在本章之前就已知道。例如，第 7 章用 Rnd 函数生成随机数，用来填写数组 intArray 的值，而且 Int 函数将随机数变成整数：

```
intArray(i) = Int(Rnd * 10)
```

Rnd 是个不多见的函数，不需要参数。（Rnd 函数有一个可选参数，但前面的例子未使用。）另外，Int 函数需要有参数，参数为数值或表达式，可以通过该函数将参数转换成整数，这里使用了表达式 Rnd \* 10。Rnd 函数返回了 Int 函数使用的值，Int 函数把值返回给程序，程序用该值填写数组。

参数是 VBA 在函数、方法或命令中使用的信息。如果参数使用了方括号，那么该参数为可选参数。例如，Rnd 函数的语法如下：

```
Rnd([数字]) As Single
```

方括号中的“数字”是可选的，As Single 表明返回的值是单精度。不同的函数根据其含义返回不同的数据类型。许多函数返回不定型数据，是/否函数（例如，第 7 章中的 Is-Numeric 函数）返回逻辑值。若有必要，VBA 可将函数的结果转换成另一种数据类型用于另一个函数的表达式中。

如果方括号中包括两个参数，那么必须同时使用。例如，MsgBox 函数用来显示信息框，函数的语法如下：

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

其中，prompt 是唯一的必选参数，buttons、title、helpfile、context 都是可选参数。而 helpfile 和 context 包含在一对方括号中，可以全不使用或者全都使用，不可以只使用其中的一个。

**注意：**第 13 章将介绍怎样使用 MsgBox 函数。

## 使用函数

可以在程序或者另一个函数中调用一个函数。要调用某个函数，可使用 Call 语句，其中 Call 关键词可以使用，也可以不使用，而只给出函数的名称。使用 Call 关键词可使代码更清晰，更容易理解。也可以用 Call 查寻所有的 Call 语句。（Call 关键词之后必须有一空格。）然而，使用 Call 关键词对于常用函数似乎有些过分。

**注意：**正如本书后面所示，调用过程的方法与调用函数一样。

Call 语句的语法如下：

```
[Call] 名称[, argumentlist]
```

其中，“名称”是必须使用的字符参数，为函数或者过程的名称。argumentlist 为可选参数，表示函数或程序使用的变量、数组或表达式，用逗号分开。调用函数时，往往必须提供参数（除了一些不需要参数的函数）；调用过程时，只有对需要使用参数的过程才提供参数。

Call 关键词用方括号括起表明不一定非用不可。如果使用，必须在括号中包括参数。大多数情况下，调用函数时并不使用 Call 关键词。

例如，下面的语句调用了 MsgBox 函数，提供必需的参数 prompt（字符串 Hello, World!）：

```
MsgBox "Hello, World!"
```

另外，也可以使用 Call 关键词，但需按下面的语句在括号中给出参数。不过这样做并没有什么好处。

```
Call MsgBox("Hello, World!")
```

**注意：**要在 Visual Basic 编辑器中运行一个函数，必须在过程中进行。另外，也可以在“立即窗口”中运行函数，然而“立即窗口”每次只能执行一条语句，因此该方法往往用于测试。

可以将函数的结果赋给变量，例如可以参照下面的代码。前两条语句声明了字符变量 strExample 和 strLeft10，第 3 条语句对 strExample 赋值，第 4 条语句使用了 Left 函数返回最左边的 10 个字符，并在信息框中显示出来（参见图 9.1）。

```
Dim strExample As String
Dim strLeft10 As String
strExample = "Technology is interesting."
strLeft10 = Left(strExample, 10)
MsgBox strLeft10
```

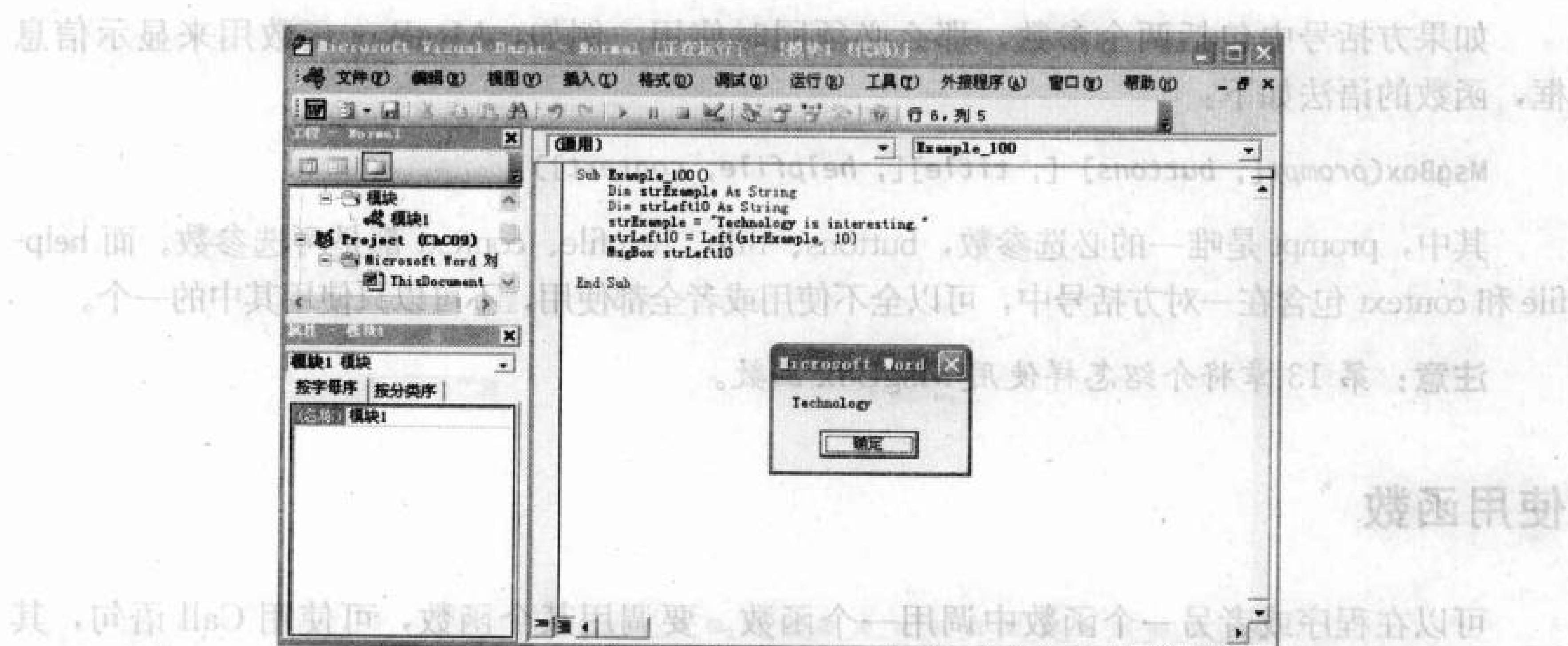


图 9.1 使用 Left 函数获取字符串左边的部分，本例获取左边前 10 个字符

除了可以把函数的结果赋给变量外，还可以在代码中直接插入函数，或直接将它和另一个函数相连。请参照下面的语句：

```
MsgBox Right(Left("This is Pride and Patriotism", 13), 5)
```

该语句使用了 3 个函数：MsgBox 函数、Right 函数和 Left 函数。Right 函数和 Left 函数相对应，返回字符串右边的部分。

如果在 VBA 语句中有多套括号，执行顺序为由里向外，正如数学运算那样。因此，本例先执行 Left 函数，返回字符串最左边的 13 个字符 This is Pride（包括空格）。然后，VBA 将新的字符串送给 Right 函数，返回最右边的 5 个字符：Pride。最后，VBA 将第二个新的字符串送给 MsgBox 函数，并在信息框中显示出来。

**注意：**函数可以有多层嵌套，并且不会产生问题。然而，实际中不宜使用过多嵌套，以便于程序的阅读和修改。

## 将参数传递给函数

当函数有一个以上的参数时，有 3 种方法将参数传给函数：

- ◆ 不提供参数的名称，按函数要求的顺序列出。
- ◆ 提供参数的名称，按函数要求的顺序列出。
- ◆ 提供参数的名称，顺序任意。

第 1 种方法没有提供名称，按参数的顺序，这是最快的方法。缺点是：阅读代码时不能立刻了解参数与之关联的值，虽然读起来并没有什么困难。省略可选参数，应当在相应的位置给出逗号。

使用参数名称时，时间较长，然而代码容易理解。在参数省略时，不需要使用逗号。

使用参数名称时，按顺序排列和不按顺序排列并没有明显优势，只是按顺序方便些。

例如，DateSerial 函数针对年、月、日返回日期，其语法如下：

```
DateSerial(year, month, day)
```

在该语句中，year 为必选的整型参数，代表年；month 为必选的整型参数，代表月；

day 为必选的整型参数，代表日。

下面的语句按顺序排列，未提供参数名称：

```
MsgBox DateSerial(2006, 12, 31)
```

下面的语句提供了参数名称，并按顺序排列：

```
MsgBox DateSerial(Year:=2006, Month:=12, Day:=31)
```

下面的语句提供了参数的名称，但未按顺序排列：

```
MsgBox DateSerial(Day:=31, Year:=2006, Month:=12)
```

3 条语句均可使用。如果不按顺序也不提供名称，或者提供部分名称，或者缺少必选参数，就不能正常工作。图 9.2 显示可能遇到的两种错误。

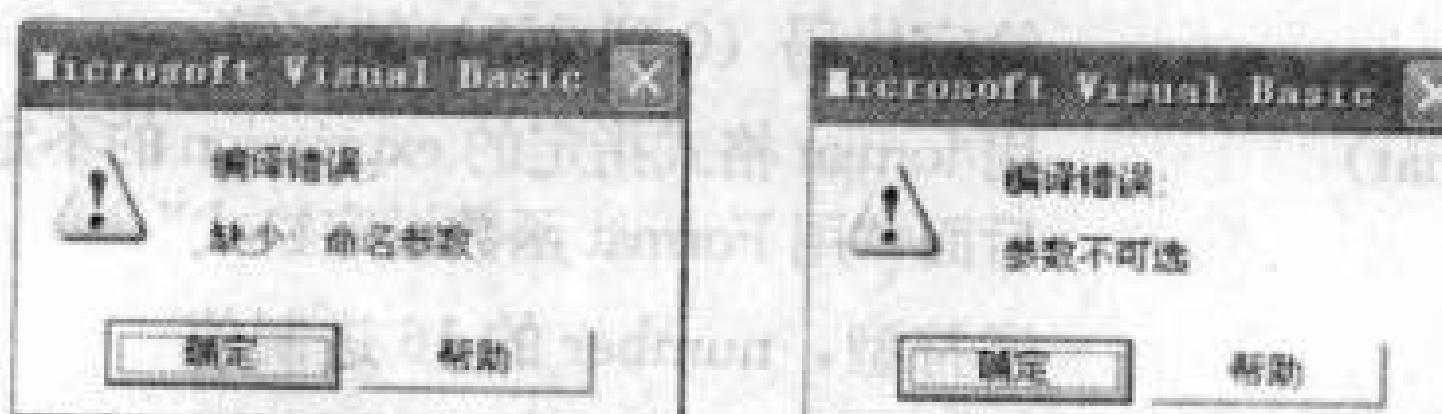


图 9.2 参数名称使用不全将给出“缺少：命名参数”的错误；参数使用不全将给出“参数不可选”的错误

## 数据类型转换

VBA 提供完整的函数，用于将数据由一种类型转换为另一种类型。这类函数可归纳为两组：简单转换和复杂转换。

表 9.1 为 VBA 函数的简单数据转换。

表 9.1 VBA 函数的简单数据转换

函数(参数)	返回的数据类型
CBool(number)	Boolean
CByte(expression)	Byte
CCur(expression)	Currency
CDate(expression)	Date
CDbl(expression)	Double
CInt(expression)	Integer
CLng(expression)	Long
CSng(expression)	Single
CStr(expression)	String
CVar(expression)	Variant

例如，下面的语句声明无类型的变量 varMyInput 和整型变量 intMyVar，然后显示输入框要求输入整数。在第 3 条语句中，输入的信息赋值给 varMyInput，自动变成不定型或字符串型。第 4 条语句使用 CInt 函数将 varMyInput 转换为整型，并将结果赋给 intMyVar。第 5

条语句将 intMyVar 和 10 进行比较，用 CBool 函数将结果转换为逻辑型，并在信息框中显示出来。

```
Dim varMyInput
Dim intMyVar As Integer
varMyInput = InputBox("Enter an integer:", "10 Is True, Other Numbers Are False")
intMyVar = CInt(varMyInput)
MsgBox CBool(intMyVar = 10)
```

表 9.2 列出了 VBA 函数的复杂数据转换。

表 9.2 VBA 的复杂数据转换

函数（参数）	结果
Asc (string)	字符串中第 1 个字符的 ASCII 码
Chr (number)	给定代码（0 到 255）的字符
Format (expression, format)	用 format 格式指定的 expression 的不定型值。参见本章后面“用 Format 函数设定格式”
Hex (number)	字符型，number 的 16 进制值
Oct (number)	字符型，number 的 8 进制值
RGB (number1, number2, number3)	3 个颜色参数设定的颜色长整型值
QBColor (number)	指定颜色的 RGB 长整型值
Str (number)	返回代表 number 的不定型/字符型值
Val (string)	返回包含于 string 内的数字。如果没有数字，返回 0

## 使用 Asc 函数返回字符代码

Asc 函数返回字符串第一个字符的字符代码。字符代码是计算机引用字母的数字。例如，大写字母 A 的字符代码是 65，大写字母 B 的字符代码是 66，小写字母 a 的字符代码是 97，小写字母 b 的字符代码是 98。

**注意：**Asc 代表是 ASCII，意思是美国信息交换标准码。实际上，Asc 函数返回 ANSI（美国国家标准机构），一个字符的代码。

Asc 函数的语法为：

**Asc(string)**

其中，string 是字符表达式。例如，Asc (" a") 返回 65。

下面的语句使用 Asc 函数返回当前活动文档的选中区域的第一个字符的代码，并将其在信息框中显示出来：

```
strThisCharacter = Asc(Selection.Text)
MsgBox strThisCharacter, vbOKOnly, "Character Code"
```

## 使用 Val 函数取出字符串中的数字

Val 函数将字符串中的数字抽出。Val 函数有下列规则：

- ◆ 只读取字符串中的数字。
- ◆ 从字符的首位开始，直到读到数字为止。

- ◆ 不考虑退格、连字符以及空格。
- ◆ 将句号识别为小数点而不是分节号。

这就意味着，如果一个字符串由多列数字组成，如下所示，Val 函数会将其识别为一个单独的数（本例为 445634.994711）：

项目号	单价	库存	购进	售出
4456	34.99	4	7	11

然而，如果字符串同时包括数字和字母，Val 函数将只读取数字，把字符串识别为数字表达式（例如，Val (" 4E5") 返回 400000，因为该字符串被识别为数字表达式）。例如，Val 函数会将下面的地址返回 8661 而不考虑其他数字（因为在 Laurel 的 L 上停了下来，这是出现的第一个字符，既不是数字也不是退格或空格）：

```
8661 Laurel Avenue Suite 3806, Oakland, CA 94610
```

Val 的语法很直接：

```
Val(string)
```

其中，string 是必需的参数，包括任何字符表达式。

下面的语句使用 Val 函数从字符串 Address1 中返回数字变量 StreetNumber。

```
StreetNumber = Val(Address1)
```

## 使用 Str 函数将数值转换为字符

正如可以将字符转换为数值，也可以将数值转换为字符。如果需要把包括在数值中的信息连接起来就需要进行这样的转换。如果仅仅使用 + 运算符，VBA 就会进行数学运算而不是将它们相连。

例如，如果定义了一个名为 strYourAge 的字符变量和名为 intAge 的数字变量，那么不可以使用语句 strYourAge + intAge 使其相连接，因为它们的类型不相同。首先必须从 intAge 变量中生成一个字符串，然后将该字符串和 strYourAge 相连接。（也可以使用符号 & 连接这两个变量。）

将数值转换为字符，需使用 Str 函数。Str 函数的语法如下：

```
Str(number)
```

number 是包括数字表达式的变量（例如，整型、长整型或者双精度型）。

下面的小程序是将数值转换为字符的例子：

```
Sub Age
```

```
    Dim intAge As Integer, strYourAge As String
    intAge = InputBox("Enter your age:", "Age")
    strYourAge = "Your age is" & Str(intAge) & "."
    MsgBox YourAge, vbOKOnly + vbInformation, "Age"
```

```
End Sub
```

## 使用 Format 函数设定格式

Format 函数功能很强，可以将数字、日期以及字符转换成需要的格式。

Format 函数的语法如下

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

该语法各部分的说明如下：

- ◆ expression 是正确的表达式。
- ◆ format 是可选参数，规定一个命名的格式或者自定义的格式。后面再详细介绍。
- ◆ firstdayofweek 是可选常数，指明该星期的第一天（日期信息）：默认设定第一天为 vbSunday (1)，然而也可以指定为 vbMonday (2)、vbTuesday (3)、vbWednesday (4)、vbThursday (5)、vbFriday (6)、vbSaturday (7) 或者 vbUseSystem (0；系统设定)。
- ◆ firstweekofyear 是可选常数，用于指定一年的第一个星期（再次给出日期信息）：

常数	值	新年起始的星期
vbUseSystem	0	使用系统设置
vbFirstJan1	1	含有一月一日的星期（默认设置）
vbFirstFourDays	2	至少包括 4 天的第一个星期
vbFirstFullWeek	3	包括 7 整天的第一个星期

如果所有的格式都不符合要求，那么可以对 Format 函数设定自己需要的格式。

### 使用事先定义的格式

表 9.3 列出了事先定义的数字格式，它可以和 Format 函数一起使用。

表 9.3 事先设定的数字格式

格式名称	说明	举例
General Number	数字显示不包括千分节	124589
Currency	数字显示保留两位小数，有千分节，并且带有相应的货币符号	\$ 1, 234.56
Fixed	数字显示保留两位小数以及至少一个整数位	5.00
Standard	数字显示保留两位小数，至少一位整数，并含有千分节	1, 225.00
Percent	数字显示为百分数，保留两位小数并带有百分号	78.00%
Scientific	数字显示为科学计数法	5.00E+00
Yes/No	非零的数显示为 Yes，零显示为 No	Yes
True/False	非零的数显示为 True，零显示为 False	False
On/Off	非零的数显示为 On，零显示为 Off	Off

例如，以下语句返回 \$ 123.45：

```
Format("12345", "Currency")
```

### 生成数字格式

如果事先给定的数字格式不能满足需求，可以使用表 9.4 中列出的字符生成自己的数字格式。

(续表)

表 9.4 用于自定义数字格式的字符

字符	说明
[无]	无格式 (通常不希望这样做)
0	数字占位。如果没有数字, VBA 显示 0。如果数字少于使用的 0, VBA 在前或后补 0
#	数字占位。如果没有数字, VBA 不显示
.	小数点占位。表明小数点的位置。小数点因国家而异, 例如在美国是小数点, 在德国是逗号
%	百分号占位。VBA 插入百分号, 并且将表达式乘以 100
,	千分节 (根据不同国家可以是逗号或者句点)
:	时间分割号 (一般为冒号, 同样因国家而异)
/	日期分割好 (同样因国家而异)
E- E+ e- e+	科学格式, E- 或 e- 在负指数前加减号。E+ 或 e+ 在负指数前加减号并且在正指数前加加号
- + \$ ()	显示字母
\ [字母]	显示特定的字母
" [字符]"	显示字母, 使用 Chr (34) (双引号的代码) 提供双引号

例如, 以下语句返回保留四位小数的货币:

```
Format("123456", "$00.0000")
```

## 生成日期或时间格式

同样, 可以混合使用表 9.5 中的字符生成日期和时间格式。

表 9.5 用于生成日期和时间格式的字符

字符	说明
:	时间分割号 (一般为冒号, 但因国家不同而异)
/	日期分割 (同样因国家而异)
c	在系统的短日期格式中显示日期 (如果有日期或者整数值); 在系统的默认时间格式中显示时间 (如果有日期或者小数值)
d	显示日期 (1~31), 个位数前不补 0
dd	显示日期, 个位数前补 0 (01~31)
ddd	一周七天用三位缩写字母表示 (Sun, Mon, Tue, Wed, Thu, Fri, Sat), 不带句点
dddd	显示完整的七天名称
ddddd	显示完整的日期, 日、月、年采用系统的短日期格式
ddyyyy	显示完整的日期, 日、月、年采用系统的长日期格式
aaaa	显示完整的当地七天名称
w	显示 1 (星期日) ~7 (星期六) 之间的整数
ww	显示 1~54 之间的整数, 给出一年的星期数。该数是 54 而不是 52, 因为大多数年份中, 开始和结尾不是完整的星期
m	用 1~12 之间的数字表示月份, 个位数前不补 0。若在 h 之后使用, 返回分钟而不是月份
mm	用 01~12 之间的两位数字表示月份。若在 h 之后使用, 返回分钟而不是月份
mmm	用三个缩写字母表示月份, 不含句点
mmmm	显示月份的全称
oooo	显示当地月份的完整名称
q	用 1~4 表示季度

表 9.6 表示日期和时间的字符

(续表)

字符	说明	即指	表示
y	用 1~366 表示天	(通常在显示不精确) 天数天	【天】
yy	用 00~99 两位数表示年	(通常在显示不精确) 年古字段	0
yyyy	用 0100~9999 四位数表示年	(通常在显示不精确) 古字端	
h	用 0~23 表示小时	(通常在显示不精确) 小时小数点	半
Hh	用 00~23 表示两位数的小时	(通常在显示不精确) 小时整数部分	
N	用 0~60 显示分钟	(通常在显示不精确) 分钟百人前	分
Nn	用 00~60 显示两位数的分钟	(通常在显示不精确) 分钟千人后	
S	用 0~60 显示秒	(通常在显示不精确) 秒数日	
Ss	用 00~60 显示两位数的秒	(通常在显示不精确) 秒数日	十 一 三 一 三
ttttt	按系统的默认格式显示完整的时间	(通常在显示不精确) 时分秒	
AM/PM	用 12 小时时钟、AM 和 PM 表示时间	数字示显	(0) 十一
am/pm	用 12 小时时钟 am 和 pm 表示时间	数字示显	【数字】 /
A/P	用 12 小时时钟、A 和 P 表示时间	数字示显	
a/p	用 12 小时时钟、a 和 p 表示时间	数字示显	【数字】
AMPM	根据系统定义用 12 小时时钟显示 AM 或 PM 表示时间	数字示显	

例如，下面的语句返回 Saturday, April, 01, 2006

`Format(#4/1/2006#, "dddddd")`

## 生成字符格式

Format 也可以使用表 9.6 所示的字符生成自定义字符格式。

表 9.6 用于生成自定义字符格式的字符

字符	说明
@	字符的占位格，若有字符，则显示；若无，则给出空格
&	字符的占位格，若有字符，则显示；若无，则不显示
<	以小写方式显示字符
>	以大写方式显示字符
!	从左至右填补占位，而不是从右至左（默认方向）填补占位

例如，下面的语句在无输入时会给出 4 个空格：

`strUser = Format(InputBox("Enter your name:"), "~~~~")`

## 使用 Chr 函数以及常数输入特殊字符

要在字符串中加入特殊字符（例如：回车或者退格），应指定内置常数（这些特殊字符具有给定的内置常数），或者使用 Chr 函数输入相应的代码。Chr 函数的语法很直接：

`Chr(charactercode)`

这里，charactercode 是指定字符的数字。

表 9.7 列出了常用字符代码和字符常数。

表 9.7 VBA 字符代码和字符常数

代码	内置字符常数	字符	(兼容) 键面
Chr(9)	vbTab	退格	Del (Delete), Backspace
Chr(10)	vbLf	换行符	Enter (Enter), Return
Chr(11)	vbVerticalTab	软回车 (Shift+Enter)	Shift+Enter
Chr(12)	vbFormFeed	换页	PageUp (PageUp), PageDown (PageDown)
Chr(13)	vbCr	回车	Return (Return), Enter (Enter)
Chr(13)+Chr(10)	vbCrLf	回车+换行	Enter (Enter), Return (Return)
Chr(14)	—	分栏	Alt+Enter (Alt+Enter)
Chr(34)	—	双引号 ("")	" (Quote), ` (grave)
Chr(39)	—	单引号 ('')	' (Apostrophe), ` (grave)
Chr(145)	—	起始单引号 ('')	' (Apostrophe), ` (grave)
Chr(146)	—	终止单引号 ('')	' (Apostrophe), ` (grave)
Chr(147)	—	起始双引号 ("")	" (Quote), ` (grave)
Chr(148)	—	终止双引号 ("")	" (Quote), ` (grave)
Chr(149)	—	着重号	Space (Space)
Chr(150)	—	短划线	Underline (Underline)
Chr(151)	—	长划线	Strikeout (Strikeout)

试想一下，要生成一个包括姓名、地址的字符串，把各项单独的信息组合在一起，这些单独的信息之间用退格键隔开，这样就可以将该字符串插入到文档中，然后转换成一个表格。要这样做，可使用下面的语句。VBA 用 For…Next 循环重复操作直到计数变量 i 达到 intNumRecords 为止：

```

For i = 1 to intNumRecords
    AllInfo = FirstName & vbTab & MiddleInitial & vbTab -
        & LastName & vbTab & Address1 & vbTab & Address2 -
        & vbTab & City & vbTab & State & vbTab & Zip -
        & vbTab & BusinessPhone & vbTab & HomePhone & -
        & vbTab & BusinessEMail & vbTab & HomeEMail & vbCr
    Selection.TypeText AllInfo
Next i

```

第 2 行（分 5 行列出）把数据赋值给 AllInfo，连接了 FirstName、MiddleInitial、LastName 等信息，中间用 vbTab 隔开，最后一个字符为 vbCr（回车字符），用来生成一个新的段落。

第 3 行将 AllInfo 输入当前的文档，产生一个用退格键分割的姓名、地址清单，该清单可以方便地转换成表，每列包括一项信息。第一列为 FirstName，第二列为 MiddleInitial，以此类推。

## 用函数处理字符串

字符串变量在处理文本时，用途十分广泛，既可保存大量文本（可以从一个字母到许多页，可以针对 Word 文档或者其他文本文档），又可保存文件名、目录名。一旦用字符串保存信息，就可以根据需要进行处理或改变。

表 9.8 列出 VBA 处理字符串的内置函数。因为有些函数比介绍过的复杂，而且经常使用，所以表后给出例子。

表 9.8 VBA 的字符串处理函数

函数（参数）	返回值
InStr (start, string1, string2, compare)	不定型/长整型，返回一字符串（string2）在另一字符串（string1）中出现的位置
InStrRev (stringcheck, stringmatch, start, compare)	不定型/长整型，返回一字符串（stringmatch）在另一字符串（stringcheck）中出现的位置，从字符串的末尾算起
LCase (string)	返回转换成小写的 string
Left (string, number)	不定型/字符型，返回 string 中从左边算起指定数量 number 的字符
Len (string)	长整型，返回 string 内字符的数目
LTrim (string)	不定型/字符型，返回字符串，前导空格删除
Mid (string, start, length)	不定型/字符型，返回 string 中指定位置、指定数量的字符
Right (string, number)	不定型/字符型，返回包含字符串中，从右边算起指定数量的字符
RTrim (string)	不定型/字符型，返回字符串，尾随空格删除
Space (number)	返回特定数目 number 的空格
StrComp (string1, string2, compare)	字符 string1 和 string2 的比较结果
StrConv (string, conversion, LCID)	返回指定类型转换的不定型/字符型
String (number, character)	返回不定型/字符型值，其中包含指定长度重复字符的字符串
StrReverse (expression)	以相反的顺序显示 expression 的字符
Trim (string)	不定型/字符型，返回字符串 string 前后空格删除的结果
UCase (string)	返回转换成大写的 string

## 使用 Left、Right 和 Mid 函数返回字符串的一部分

在程序中经常需要使用字符串的一部分。例如，需要取出城市名的前 3 个字母作为地点的代码。

VBA 提供几个函数用来返回字符串的一部分。

- ◆ Left 函数返回字符串从左边算起指定数量的字符。
- ◆ Right 函数返回字符串从右边算起指定数量的字符。
- ◆ Mid 函数返回字符串中指定位置、指定数量的字符。

**注意：**VBA 中的许多函数有两个版本，其中包括 Left、Right 和 Mid 函数，现在介绍的版本返回字符型/不定型值。另一个版本，函数名称后带有 \$（Left \$，Right \$，Mid \$ 等），返回字符值。返回字符值的函数，运行速度较快（尽管你不可能注意到它们之间的差别），但是若使用了空值将会出现错误。而字符型/不定型可以正常处理空值。

### 使用 Left 函数

Left 函数返回字符串从左边算起指定数量的字符。Left 函数的语法如下：

**Left(string, length)**

这里，string 参数是任意字符串表达式，就是说表达式为连续的字符组合。如果 string 不包含数据，Left 函数返回 Null。length 参数是数值表达式，规定返回字符的数量。length

可以直接为数字（例如：4、7 或 11）或者为结果是数字的表达式（例如：单词的长度保存在名称为 LenWord 的变量中，返回的字符串希望比 LenWord 少两个字符，参数就应当写成 LenWord - 2。如果希望比 LenWord 多 3 个字符，就应当写成 LenWord + 3）。

例如，可以使用 Left 函数将电话号码中的区号取出，该电话号码在计算机中为 10 位连续数字。在下面的语句中，电话号码在 strPhone 变量中，该变量假定前面已经使用过：

```
Dim strArea As String
strArea = Left(strPhone, 3)
```

该语句生成变量 strArea，并保存 strPhone 变量最左边的 3 个字符。

### 使用 Right 函数

Right 函数是 Left 函数的镜像，返回字符串从右边算起指定数量的字符。Right 函数的语法如下：

```
Right(string, length)
```

同样地 string 参数是字符串表达式，length 是数字表达式，指定返回字符的数量。同样，Right 函数在 string 无数据时返回 Null，length 可以是数字，也可以是结果为数字的表达式。

继续前面的例子，可用 Right 函数取出电话号码 strPhone 的最后 7 位数字：

```
Dim strLocalNumber As String
strLocalNumber = Right(strPhone, 7)
```

该语句生成变量 strLocalNumber 并保存 strPhone 变量中右边的 7 个字符。

### 使用 Mid 函数

Mid 函数返回字符串中指定数量的字符。可以指定字符串的起始位置和字符的数量（由指定位置向右）。Mid 函数的语法如下：

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

和 Left、Right 函数一样，string 参数为任何字符串表达式，如果 string 没有数据，Mid 函数返回 Null。

start 是数字型的值，指定 string 中选择字符的起始位置。如果 start 大于字符串的字符数，VBA 将返回 0 长度的字符串。

length 是数字表达式，指定返回字符的数量。如果省略 length 或者 length 参数大于字符串的字符数，VBA 将返回从 start 位置开始的所有字符。同样，length 可以直接为一个数字或者结果是数字的表达式。

仍然用电话号码举例，可以用 Mid 函数从 10 位电话号码中返回本地的区号（例如从 5105551212 返回 555）。这里，电话号码此时存储在变量 strPhone 中：

```
Dim strLocalExchange As String
strLocalExchange = Mid(strPhone, 4, 3)
```

该语句声明了 strLocalExchange 变量，其中保存了从 strPhone 变量中的第四个字符开

始的 3 个字符。

**注意：**如果电话号码的格式不同，例如 (510) 555—1212 或者 510—555—1212，就必须调整 start 值。例如，区号在括弧中，而且后面有一空格，start 值应该为 7。如果区号用短划线和后面的电话号码隔开，start 值应该为 5。

可使用 Mid 函数得到某字符的位置。在下面的例子中，Do Until…Loop 循环语句对 strFilename 进行搜索，直到找出第一个斜杠 (\)，并将其位置保存在 intLen 变量中。这里，strFilename 包括当前文档的 FullName 属性，信息框将斜杠后面的内容显示出来（其长度为 Len 减去 strFileName）：

```
Dim strFilename As String, intLen As Integer
strFilename = ActiveDocument.AttachedTemplate.FullName
intLen = Len(strFilename)
Do Until Mid(strFilename, intLen, 1) = "\"
    intLen = intLen - 1
Loop
MsgBox Right(strFilename, Len(strFilename) - intLen)
```

本例主要用来说明问题，而不是从实际出发，原因有二：其一，模版的名称用 Name 属性而不是 FullName 属性更容易得到；其二，有一个名为 InStrRev（以后再讨论）的函数，用于由后向前返回字符串中的一个字符。

## 使用 InStr 和 InStrRev 函数在一个字符串中查找另一个字符

InStr 函数允许我们在一个字符串中查找另一个字符串，例如，可以在当前的段落中查找一个具体的单词。如果实现了查找，还可以进一步处理。例如，用另一个单词去替换，或者在另一个文档中选择整个段落。

InStrRev 函数是 InStr 函数的对应函数，处理方法相同，但方向相反。InStr 函数的语法如下：

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

参数说明如下：

- ◆ start 是可选参数，指定第一个字符串 string1 的开始位置。如果省略 start，VBA 从 string1 的第一个字符开始。然而，如果使用了 compare 参数指定比较类型，就必须使用 start。
- ◆ string1 是必需参数，指定查找 string2 的字符表达式。
- ◆ string2 是必需参数，指定在 string1 中查找的字符表达式。
- ◆ compare 是可选参数，指定比较的类型：二进制比较（区分大小写），或者原文比较（不考虑大小写）。默认为二进制比较，可以使用常数 vbBinaryCompare 或者 0 作为参数。指定参数并不是必需的（因为它是默认的），但可以使代码更容易读懂。要设定原文比较，使用 vbTextCompare 常数或者 1 作为参数。

**提示：**原文比较是十分有用的工具，可以用于各种各样的数据。例如，要查找一个名称，可以按名称的大写、小写或者斜体，否则只能按名称的实际状况查找。

可以使用 InStr 函数在另一个字符串中查找某个字符，这样就可以改变内在的字符。例

如，可以将一个文件从当前的位置移到另一个有类似目录结构的目录中。例如，文件保存在名称为 z:\Documents\In\ 的目录里，经过处理就可以将文件保存在 z:\Documents\Out 目录中。程序清单 9.1 所示的代码将文档自动保存在 Out 子文件夹中。

### 程序清单 9.1

```

1. Sub Save_in_Out_Folder()
2.     Dim strOName As String, strNName As String, _
       intToChange As Integer
3.     strOName = ActiveDocument.FullName
4.     intToChange = InStr(strOName, "\In\")
5.     strNName = Left(strOName, intToChange - 1) & "\Out\" & Right(strOName, Len(strOName) - intToChange - 3)
6.     ActiveDocument.SaveAs strNName
7. End Sub

```

程序清单 9.1 的代码解释如下：

- ◆ 第 1 行为程序开始，第 7 行为程序结束。
- ◆ 第 2 行声明字符串变量 strOName（称为原名称）、字符串变量 strNName（称为新名称）以及整型变量 intToChange。
- ◆ 第 3 行将 strOName 设定为 ActiveDocument 对象的 FullName 属性：即当前文档的全名称，包括文档的路径（例如，z:\Documents\In\Letters\My Letter.doc）。
- ◆ 第 4 行将 intToChange 的值设定为 InStr 函数在变量 strOName 中查找 \In\ 的结果。使用前面段落的路径举例，intToChange 将得到 13，因为第一个 \In\ 的位置是 13。
- ◆ 第 5 行将变量 strNName 设定为语句中生成的新文件名，详细说明如下：
  - ◆ Left (strOName, intToChange-1) 取出 strOName 左边的部分，返回的字符数为 intToChange-1。
  - ◆ & "\Out\" 加入前面的部分字符串变量，替换了 \In\，路径变成 z:\Documents\Out\。
  - ◆ & Right (strOName, Len (strOName) - intToChange - 3) 通过以 \In\ 字符串 (Letters\My Letter.doc) 之后开始并加入 strOName 的右边部分来完成部分字符串，其路径为 z:\Documents\out\Letters\My Letter.doc。从右边获取的字符数由表达式决定，先从 strOName 的长度减去 intToChange，再减去 3。这里，3 是 \In\ 的长度。因为 intToChange 保存了第 1 个斜杠的字符数，因此只需要计算中 S.0 与 I、n 以及第 2 个斜杠到末尾的字符数。
- ◆ 第 6 行用名称 strNName 保存文档。

InStrRev 的语法和 InStr 相似：

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

参数解释如下：

- ◆ stringcheck 为必选参数，指定用 stringmatch 查询的字符串。
- ◆ stringmatch 为必选参数，指定查询的内容。
- ◆ start 为可选参数，指定查询的起始位置，若不指定，从 stringcheck 的最后字符开始。

- ◆ compare 为可选参数，指定查询的方法：vbTextCompare 为文本，vbBinaryCompare 为二进制方式。

## 使用 LTrim、RTrim 和 Trim 去除空格

在连接字符串时往往需要去除空格，以防止出现不合适的空格，例如在 8 个字符的文件名中间。

如表 9.8 所示，VBA 提供 3 种函数去除字符串前导和后置空格。

- ◆ LTrim 去除字符串的前导空格。
- ◆ RTrim 去除字符串的后置空格。
- ◆ Trim 去除字符串的前导和后置空格。

**提示：**在很多情况下，可简单使用 Trim 函数，而不需要考虑前导或者后置。有时候只需要去除前导或者后置空格，并保留不需要去除的部分，这时候就必须使用 LTrim 或者 RTrim，Trim 特别适合用于固定长度的字符串变量。如果数据比定长短，就会出现后置空格。

LTrim、RTrim 和 Trim 的语法很直接：

```
LTrim(string)
RTrim(string)
Trim(string)
```

在每一种情况下，string 都是字符串表达式。

可以使用 Trim 函数去除来自当前文档的字符串的前后空格。下面的第一行语句声明 strUntrimmed 和 strTrimmed 字符型变量。第二行将当前文档的数据赋值给 strUntrimmed 变量。第三行将处理过的字符串赋值给 strTrimmed 字符串：

```
Dim strUntrimmed As String, strTrimmed As String
strUntrimmed = Selection.Text
strTrimmed = Trim(strUntrimmed)
```

## 用 Len 函数检查字符串的长度

检查字符串的长度，可以用 Len 函数，其语法如下：

```
Len(string)
```

在这里，string 为有效的字符串表达式。（如果 string 为 Null，Len 也返回 Null。）

可以使用 Len 函数检查输入框或对话框中输入值的长度。例如，程序清单 9.2 中 CheckPassword 过程用 Len 函数检查输入的密码的长度。

### 程序清单 9.2

```
1. Sub CheckPassword()
2.     Dim strPassword As String
3.     BadPassword:
4.         strPassword = InputBox _
```

```

        ("Enter the password to protect this item from changes: " _ 
         , "Enter Password")
5.     If Len(strPassword) = 0 Then
6.         End
7.     ElseIf Len(strPassword) < 6 Then
8.         MsgBox "The password you chose is too short." _
           & vbCrLf & vbCrLf & _
           "Choose a password between 6 and 15 characters in length.", _
           vbOKOnly + vbCritical, "Unsuitable Password"
9.         GoTo BadPassword
10.    ElseIf Len(strPassword) > 15 Then
11.        MsgBox "The password you chose is too long." _
           & vbCrLf & vbCrLf & _
           "Choose a password between 6 and 15 characters in length.", _
           vbOKOnly + vbCritical, "Unsuitable Password"
12.        GoTo BadPassword
13.    End If
14. End Sub

```

程序清单 9.2 用来检验密码的长度，方法比较简单，保证密码的长度在 6 到 15 个字符之间（含）。以下是程序的解释：

- ◆ 第 2 行声明字符串变量 strPassword。
- ◆ 第 3 行为标签 BadPassword，在检验不通过时，第 9 行和第 12 行的 GoTo 语句指向该标签。
- ◆ 第 4 行把输入框的结果赋值给 strPassword，该结果是用户输入的。
- ◆ 第 5 行到第 13 行使用 If 语句检验密码的长度。首先，第 5 行检验 strPassword 是否为 0。用户单击“取消”、“关闭”按钮或不输入任何内容单击“确定”按钮时，strPassword 的结果都是 0。如果为 0，程序在第 6 行终止。如果不为 0，第 7 行检验其长度是否小于 6 个字符，如果是，程序显示信息框，将问题告知，并且返回第 3 行的标签。如果密码超过 6 个字符，第 10 行检验是否超过 15 个字符，如果是，程序给出另一个信息框，也返回第 3 行的标签。

## 用 StrConv、LCase 和 UCase 改变字符串的大小写

改变字符串的大小写，可使用 StrConv（名称出自 String Conversion）、LCase 和 UCase 函数。三者当中，最容易使用的是 StrConv，可以将一个字符变成多种形式：大写、小写或开头大写，以及日语的平假名和片假名。

### StrConv

StrConv 的语法如下：

**StrConv(string, conversion)**

在这里，string 参数为任何字符串表达式，conversion 参数为指定转换类型的常量或值。最为常用的常量和值如下：

常量	值	说明
vbUpperCase	1	将给定的字符串转成大写
vbLowerCase	2	将给定的字符串转成小写
vbProperCase	3	将字符串中每个字的开头字母转成大写
vbUnicode	64	根据系统的缺省码页将字符串转成 Unicode
vbFromUnicode	128	将字符串由 Unicode 转成系统的缺省码页

例如，在数据库程序中有字符变量 strCustomerName，其内容为姓名。可以用 StrConv 函数使每个字的开头字母转成大写：

```
strProperCustomerName = StrConv(strCustomerName, vbProperCase)
```

注意：StrConv 不需要考虑输入的情况，只是按要求处理。例如，输入的是大写并要求转换成大写也不会有任何问题。

### LCase 和 UCase

如果不习惯使用 StrConv，也可以使用 UCase 和 LCase 进行大小写转换。

LCase 和 UCase 的语法如下：

```
LCase(string)
```

```
UCase(string)
```

在这里，string 参数为任何字符串表达式。

例如，下面的语句将 MyString 转换成小写，并对 MyLowerString 赋值。

```
MyLowerString = LCase(MyString)
```

### 使用 StrComp 函数进行比较

前面介绍过，对两项内容进行比较可以使用等号：

```
If 1 = 1 Then MsgBox "One is one."
```

这样的比较也可以用于字符串：

```
strPet = InputBox("什么是你的宠物?", "Pet")
```

```
If strPet = "Dog" Then MsgBox "我们不要狗。"
```

代码的问题是字符串要求大小写都需要一致才能认定为相等，如果 strPet 是 dog 或者 DOG（更不用说 dOG、doG、dOg 或 DoG），而不是 Dog，就不会认为相等。

这个问题也可用 Or 符号处理：

```
If Pet = "Dog" Or Pet = "dog" Or Pet = "DOG" Or Pet = "dogs"
```

```
Or Pet = "Dogs" Or Pet = "DOGS" Then MsgBox "我们不要狗。"
```

很显然，这样的代码很快变得笨拙，甚至遗漏了 dOG 这类拼写错误。当然可以改变表达式使之满足条件，然而使用 StrComp 函数处理这样的问题更简单。其语法如下：

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

在这里，string1 和 string2 是必选参数，为比较的字符串，compare 是可选参数，决定是文本比较（vbTextCompare）还是二进制比较（vbBinaryCompare）。

下面的语句解决了所有的问题：

```
If StrComp(Pet, "dog", vbTextCompare) = True Then
    MsgBox "我们不要狗。"
```

## 使用 VBA 的数学函数

VBA 提供一整套标准的数学函数，表 9.9 列出了这些函数，并给出相应的例子：

表 9.9 VBA 的数学函数

函数（参数）	返回值	举例
Abs (number)	number 的绝对值，去除符号以后的值	Abs (-100) 返回 100
Atn (number)	number 的反正切值	Atn (dblMyAngle)
Cos (number)	number 的余弦值	Cos (dblMyAngle)
Exp (number)	e (自然对数的底) 的某次方	Exp (5) 返回 148.413 159 102 577
Fix (number)	返回参数 number 的整数部分。如果 number 为负数，则会返回大于或等于 number 的第一个负整数	Fix (3.14159) 返回 3, Fix (-3.14159) 返回 -3
Int (number)	返回参数 number 的整数部分。如果 number 为负数，则返回小于或等于 number 的第一个负整数	Int (3.14159) 返回 3, Int (-3.14159) 返回 -4
Log (number)	指定参数 number 的自然对数值	Log (dblMyAngle)
Rnd ([number])	一个随机值（无参数时）或者根据最初给定的种子产生的值	Rnd (1) 返回随机值
Sgn (number)	number 为负数时，返回 -1；参数为 0 时，返回 0；参数为正数时，返回 1	Sgn (7) 返回 1, Sgn (-7) 返回 -1, Sgn (0) 返回 0
Sin (number)	指定 number 的正弦值（以弧度为单位）	Sin (dblMyAngle)
Sqr (number)	指定 number 的平方根。如果 number 为负数，生成一个运行时错误	Sqr (9) 返回 3
Tan (number)	指定 number 的正切值（以弧度为单位）	Tan (dblMyAngle)

## 使用日期和时间函数

VBA 提供完整的日期和时间函数，参见表 9.10。表中给出简单的例子。表后对一些复杂的函数详细举例说明。

表 9.10 VBA 的日期和时间函数

函数（参数）	返回值	举例
Date	根据计算机返回当前日期的不定型/日期型值	Msgbox Date 可能显示 04/01/2006（格式由操作系统的日期设定）
DateAdd (interval, number, date)	返回 date 加上一段时间间隔的不定型/日期型值	DateAdd ("m", 1, "6/3/06") 返回 7/3/2006
DatePart (interval, date)	返回一个包含已知日期的指定时间部分的不定型/整型值	参见后面的例子

(续表)

函数(参数)	返回值	举例
DateSerial (year, month, day)	返回包含指定的年、月、日的不定型/日期型值	dteCompanyFounded = DateSerial (1997, 7, 4)
DateValue (date)	返回一个已知日期的不定型/日期型值	dteDeath = "July 2, 1971"
Day (date)	返回一个不定型/整型值, 其值为 1 到 31 之间的整数, 表示一个月中的某一日	If Day (Date) = 1 And Month (Date) = 1 Then MsgBox "Happy new year!"
Hour (time)	返回一个不定型/整型值, 其值为 0 到 23 之间的整数, 表示一天之中的某一钟点	dteHour = Hour (dteLoggedIn)
Minute (time)	返回一个不定型/整型值, 其值为 0 到 59 之间的整数, 表示一小时中的某分钟	dteMinute = Minute (dteLoggedIn)
Month (date)	返回一个不定型/整型值, 其值为 1 到 12 之间的整数, 表示一年中的某月	strThisDate = Month (Date) & " / " & Day (Date)
MonthName (month)	返回一个表示指定月份的字符串	MsgBox MonthName (Month (Date)) 显示含有当前日期的信息框
Now	返回一个不定型/日期型值, 根据计算机系统设置来指定日期和时间	MsgBox Now 可能显示 04/01/2006 9:25:15PM (格式由操作系统的日期设定)
Second (time)	返回一个不定型/整型值, 其值为 0 到 59 之间的整数, 表示一分钟之中的某个秒	dteSecond = Second (dteLoggedIn)
Time	返回一个指明当前计算机系统时间的不定型/日期型值	MsgBox Time 可能显示 9:25:15PM (格式和时间由操作系统的日期设定)
Timer	返回一个单精度数, 代表从午夜开始到现在经过的秒数	If Timer > 43200 Then MsgBox "本程序上午运行", End
TimeSerial (hour, minute, second)	返回一个不定型/日期型值, 包含具有具体时、分、秒的时间	TimeSerial (11, 12, 13) 返回 11:12:13AM, 格式由操作系统设定
TimeValue (time)	返回一个包含时间的不定型/日期型值	TimeValue (Now)
Weekday (date)	返回一个不定型/整型值, 包含一个整数, 代表某个日期是星期几	参见下一条
WeekdayName (weekday)	返回一个字符串, 表示一星期中的某天	WeekdayName (Weekday (#4/1/2006#)) 返回星期六, 即 2006 年 4 月 1 日愚人节那一天

## 用 DatePart 函数分解日期

DatePart 函数可以将日期分解。使用其他函数也可以得到相同的效果, 然而用 DatePart 函数最有效。

DatePart 函数的语法如下:

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

语法的参数如下:

- ◆ interval 是必选字符串表达式, 用来指定时间间隔: yyyy 表示年, q 表示季度, m 表示月, y 表示这一年第几天, d 表示日, w 表示星期, ww 表示一年第几星期, h 表示小时, n 表示分钟 (m 用于月), s 表示秒。
- ◆ date 是必选参数 (不定型/日期型值), 要分解的日期。
- ◆ firstdayofweek 为可选参数。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。

- ◆ firstweekofyear 为可选参数。指定一年第一周的常数：

常数	值	新年起始的星期
vbUseSystem	0	使用系统设置
vbFirstJan1	1	含有一月一日的星期（默认设置）
vbFirstFourDays	2	至少包括 4 天的第一个星期
vbFirstFullWeek	3	包括 7 整天的第一个星期

例如，下面的语句将当前年份赋值给 dteThisYear 变量：

```
dteThisYear = DatePart("yyyy", Date)
```

## 使用 DateDiff 函数

返回不定型/长整型值，表示两个指定日期间的时间间隔。语法如下：

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

其参数如下：

- ◆ interval 是必选字符串表达式，用来指定时间间隔：yyyy 表示年，q 表示季度，m 表示月，y 表示这一年第几天，d 表示日，w 表示星期，ww 表示一年第几星期，h 表示小时，n 表示分钟（m 用于月），s 表示秒。
- ◆ date1 和 date2 是用于计算时间间隔的日期。
- ◆ firstdayofweek 为可选参数。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。
- ◆ firstweekofyear 为可选参数。指定一年第一周的常数：

常数	值	新年起始的星期
vbUseSystem	0	使用系统设置
vbFirstJan1	1	含有一月一日的星期（默认设置）
vbFirstFourDays	2	至少包括 4 天的第一个星期
vbFirstFullWeek	3	包括七整天的第一个星期

例如，以下语句返回 2006 年 6 月 3 日到 2006 年 9 月 30 日之间的星期数：

```
MsgBox DateDiff("ww", "6/3/2006", "9/30/2006")
```

## 使用 DateAdd 函数

DateAdd 函数可以很方便地根据一个给定的日期增加或减少一段时间，其语法如下：

```
DateAdd(interval, number, date)
```

参数说明如下：

- ◆ interval 是必选字符串表达式，用来指定时间间隔：yyyy 表示年，q 表示季度，m 表示月，y 表示这一年第几天，d 表示日，w 表示星期，ww 表示一年第几星期，h 表示

小时，n 表示分钟（m 用于月），s 表示秒。

◆ number 为必选数字表达式，用来指定增加或减少的数量。如果 number 不是长整型，VBA 会在计算前将其折算成一个最接近的整数。

◆ date 为必选的不定型/日期型值或者日期文字表达式，用来给定起始日期。

例如，下面的语句返回 2006 年 5 月 27 日后 10 个星期的日期：

```
DateAdd("ww", 10, #5/27/2006#)
```

## 使用文件管理函数

本节介绍两个关键的文件管理函数，Dir 函数和 CurDir 函数，前者用来判断一个文件是否存在，后者用来返回当前的路径。

### 使用 Dir 函数判断一个文件是否存在

在处理各种文件之前，我们很想了解文件是否存在。如果要自动保存一个新文件，就希望保存不会覆盖一个已经存在的文件。如果想自动打开一个文件，在使用 Open 方法之前希望了解文件在预期的地点，否则会出现错误。

可以直接使用类似程序清单 9.3 的程序检查文件是否存在。

#### 程序清单 9.3

```
1. Sub Does_File_Exist()
2.     Dim strTestFile As String, strNameToTest As String, _
       strMsg As String
3.     strNameToTest = InputBox("Enter the file name and path:")
4.     If strNameToTest = "" Then End
5.     strTestFile = Dir(strNameToTest)
6.     If Len(strTestFile) = 0 Then
7.         strMsg = "The file " & strNameToTest & _
           " does not exist."
8.     Else
9.         strMsg = "The file " & strNameToTest & " exists."
10.    End If
11.    MsgBox strMsg, vbOKOnly + vbInformation, _
           "File-Existence Check"
12. End Sub
```

程序清单 9.3 使用 Dir 函数检验文件是否存在，并用信息框说明存在与否。图 9.3 给出信息框的例子。信息框仅仅用于示范，测试的结果一般指导程序的走向。说明如下：

◆ 第 2 行声明字符串变量 strTestFile、strNameToTest 和 strMsg。

◆ 第 3 行显示输入框，要求输入一个文件和路径。VBA 将结果赋值给 strNameToTest。

◆ 第 4 行将 strNameToTest 和空字符串进行比较。空字符串意味着使用了“取消”按钮或者未输入任何内容就单击了“确定”按钮。如果比较的结果相等，程序就结束。

◆ 第 5 行把 Dir 函数对 strNameToTest 运行的结果赋值给 strTestFile。如果 Dir 找到结



图 9.3 用 Dir 函数验证一个文件是否存在，可以避免不小心覆盖或者打开不存在的文件时造成错误

果，strTestFile 就会包含相应的文件名，否则将是空字符串。

- ◆ 第 6 行为 If...Then 语句，用来测试 strTestFile 的长度。如果长度为 0，第 7 行的语句给出文件不存在的说明，否则 VBA 会转向第 8 行的 Else 语句，执行第 9 行的语句，说明文件存在。第 10 行结束 If 语句。
- ◆ 第 11 行显示含有 strMsg 的信息框。第 12 行结束程序。

**警告：**程序清单 9.3 并非无懈可击，因为 Dir 函数既可以用于一般字符，也可以用于通配符。如果在 strNameToTest 中有文件名，就不会有什么问题，因为 Dir 会将文件名和输入的文本进行比较，从中可以看出是否相吻合。然而，如果 strNameToTest 含有适当的通配符（例如 c:\temp\\*.\*），Dir 会说文件存在，当然不是指定名称的文件，只是符合通配符的文件。可以在第 5 行检验 Dir 返回的名称是否为输入的名称，但必须使用大小写比较。从计算机的观点来看，Dir 函数可以不折不扣地看做垃圾进垃圾出，按要求去做但结果却大相径庭。

## 返回当前的路径

使用 CurDir 函数可以针对当前的驱动器或指定的驱动器返回当前路径（应用程序所在的路径）。通常需要改变当前的路径以便确保在指定的地点保存或打开文件。

要返回当前路径，可使用不带参数的 CurDir 函数：

### CurDir

要返回指定驱动器的当前路径，可输入盘符以将其作为参数。例如，返回 D 盘的当前路径的语句为：

`CurDir("D")`

## 第 10 章 编写自定义函数

- ◆ 函数语句的组成
- ◆ 编写一个函数
- ◆ 编写在 Word 中使用的函数
- ◆ 编写在 Excel 中使用的函数
- ◆ 编写在 PowerPoint 中使用的函数

本章将介绍如何编写自定义函数。编写一个函数就像编写一个过程一样，只需要在代码窗口中编辑并存储在指定的模块中即可。（不能录制一个函数，即使所使用的软件中包含有录制宏功能，用户仍然必须自己编写函数。）

本章一开始介绍函数的组成部分以及怎样将它们整合在一起。然后，给出一些函数的例子，这些函数可应用在任何 VBA 的宿主程序中。同时也分别给出专门用于 Word、Excel 和 PowerPoint 的函数实例。

### 函数的组成部分

要编写一个函数，需要使用 Function 语句，其语法如下：

```
[Public | Private] [Static] Function function_name [(argument_list)] [As type]
[statements]
[function_name = expression]
[Exit Function]
[statements]
[function_name = expression]
End Function
```

下面是对该语法的解析：

- ◆ Public 是可选关键词，通过它可以使函数完全开放——所有模块的所有其他过程都可访问这个函数。（如果需要将函数的功能限制在其自身所在的工程范围内，可以通过在包含该函数的模块中加入 Option Private Module 语句来限制函数的开放性。）
- ◆ Private 是可选关键词，通过它可以使函数对包含其声明的模块中的其他过程开放，而在不包含其声明的模块中，函数将隐藏。
- ◆ Static 是可选关键词，可使局部变量在被调用时保持值不变。
- ◆ function\_name 是必备参数，给出了函数的名称。函数的命名遵循 VBA 的标准的变量命名约定：使用字母和下划线，而不使用空格、符号和标点符号。
- ◆ argument\_list 是可选参数，代表在调用时要传递给函数的参数变量列表。argument\_list 的语法如下：

[Optional] [ByRef | ByVal] [ParamArray] variable\_name[( )] [As type]  
[= default\_value]

- ◆ Optional 是可选关键词，用来表示某个参数是可选的，也就是说，并不是必需的。一旦使用 Optional 来声明可选参数，那么在 argument\_list 中的所有后续参数就都必须是可选的。因此，必须先列出必需参数，然后再列出可选参数，就像 VBA 所做的那样。给可选参数一个默认值是很明智的做法。
- ◆ ByRef 是可选关键词，指明一个参数按地址传递；ByVal 是可选关键词，指明参数按值传递。简单地说，既可以按地址传递一个参数，也可以按值传递一个参数。下面的“注意”部分解释这些术语。

**注意：**当一个过程（无论是函数还是子过程）中的参数按地址向另一个过程传递时，接受它的过程就可以直接访问源变量在内存中的位置，并且可以改变源变量的值。相反，当参数按值传递时，接受它的过程或函数只得到了该变量的所储存信息的一个副本，而不能改变源变量所储存的信息。按地址是传递参数的默认方式，但仍然可以用 ByRef 关键字显示声明按地址传递参数。

- ◆ ParamArray 是可选关键词，只用于 argument\_list 的最后一个参数，指明最后这个参数是一个 Variant 元素的可选数组。ParamArray 关键词不能与 ByRef、ByVal 和可 Optional 同时使用。
- ◆ variable\_name 代表参数的变量名称。
- ◆ type 是可选关键词，给出了参数的变量类型（字节型、逻辑型、货币型、日期型、十进制型、双精度型、整型、长整型、对象型、单精度性、可变长度字符型或不定型）。对于非可选参数，也可以指定一个对象类型（例如，Worksheet 对象）或自定义类型（用户自己所创建的）。
- ◆ default\_value 是可选常数，表明将给可选参数一个默认值。
- ◆ type 是可选参数，指明了函数返回值的数据类型：字节型、逻辑型、货币型、日期型、十进制型、双精度型、整型、长整型、对象型、单精度性、可变长度字符型、不定型或自定义型。
- ◆ statements 表示函数中的语句。理论上说，语句是可选的，但事实上，大多数函数需要一条或多条语句。
- ◆ expression 表示函数的返回值。expression 也是可选的。

## 编写一个函数

本节将介绍编写函数的整个过程。

### 手动生成函数

生成函数的最简单方法是在代码窗口中输入 Function、函数名称、必要的参数（放在圆括号内），然后按 Enter 键。VBA 会自动插入一行空行和 End Function 语句，并把光标移到空行

处，准备好编写一个函数。例如，如果输入下面的语句并回车，Visual Basic 编辑器会显示如图 10.1 所示的窗口。

```
Function MyFunction(MaxTemp, MinTemp)
```

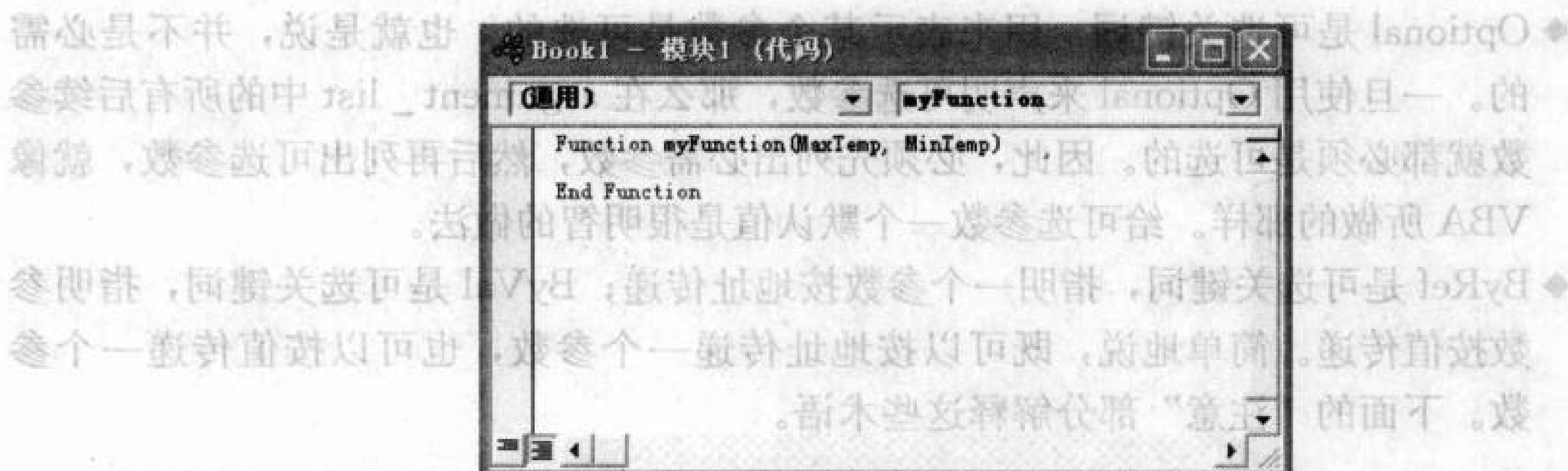


图 10.1 当输入一条函数语句并回车时，Visual Basic 编辑器会自动插入空行和 End Function 语句

## 通过“添加过程”对话框生成函数

如果想最大限度地使用 Visual Basic 编辑器，也可以通过“添加过程”对话框生成一个新函数。

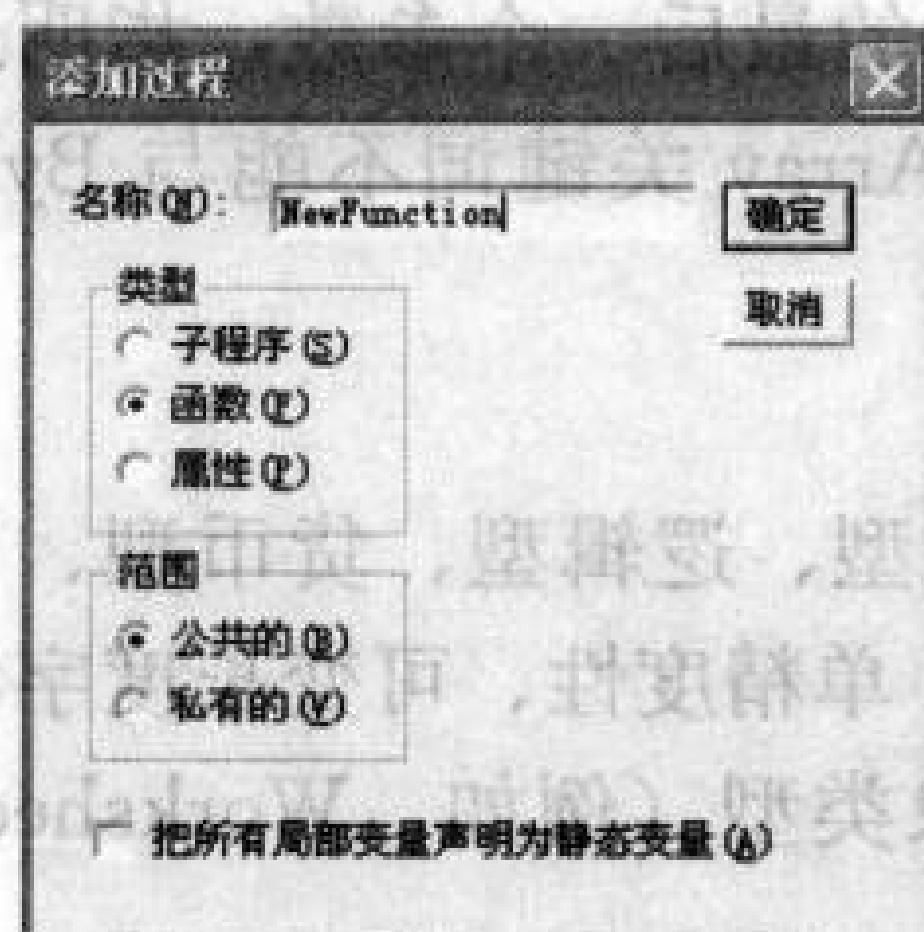


图 10.2 也可以通过“添加过程”对话框来指定新函数的属性

1. 选择“插入”>“过程”，显示“添加过程”对话框（见图 10.2）。
2. 在“名称”文本框中输入过程的名称。
3. 在“类型”组框中选择“函数”单选按钮。
4. 在“范围”组框中选择“公共的”单选按钮或“私有的”单选按钮（根据需要）。
5. 如果需要将所有的局部变量都声明为静态变量（事实上并不会那样做），勾选“把所有局部变量声明为静态变量”。
6. 单击“确定”按钮输入函数的其余部分，在圆括号内手工输入它的相关参数。

## 向函数传递参数

在 Function 语句中将函数的返回值赋值于函数的名称（本例中为 MyFunction）。在圆括号中，以逗号分开的是向函数传递的参数。本例中，函数将带入变量 MaxTemp 和变量 MinTemp 进行运算以返回运算结果。可以在参数的名称后面加入一条语句来定义参数的数据类型。例如，可以使用下面的语句指定 MaxTemp 和 MinTemp 为双精度型：

```
Function MyFunction(MaxTemp As Double, MinTemp As Double)
```

当需要在接受的过程中对变量进行运算并将其结果返回给其源过程时，按地址传递参数就十分有用了。相应地，当需要在接受的过程中使用变量中的信息，而又需要确保变量中的信息不变时，就需要按值传递参数。

就像前面提到过的，按地址传递参数是默认方式，因此下面的两条语句都是按地址传递

参数 MyArg 的：

```
Function PassByReference(MyArg)
Function PassByReference(ByRef MyArg)
```

必须使用 ByVal 关键词来指明按值传递参数。下面的语句指明按值传递参数 MyArg

```
Function PassByValue(ByVal ValArg)
```

如果必要，可以在一个过程中按地址传递一些参数，而按值传递另外一些参数。下面的语句按地址传递参数 MyArg 而按值传递参数 ValArg：

```
Function PassBoth(ByRef MyArg, ByVal ValArg)
```

## 声明参数的数据类型

可以显式声明将要传递的参数的数据类型，以减少对内存的占用，并确保在向过程传递的信息类型与所预想的相一致。然而，当按地址传递参数时，必须保证所传递的参数的数据类型与过程中所需要的保持一致。例如，如果声明一个参数为字符类型，却将其传递到接受不定型参数的过程中，VBA 将反馈为错误的指令。

要声明一个参数的数据类型，需要在 Argument List（参数列表）中加入对其数据类型的声明。下面的语句声明 MyArg 为字符型，并指定其按地址传递到过程，ValArg 为不定型，按值传递。

```
Function PassBoth(ByRef MyArg As String, ByVal ValArg As Variant)
```

## 指定可选参数

可以使用可选关键字来指定可选参数。使用时，需将关键词 Optional 置于 ByRef 和 ByVal 关键词之前。

```
Function PassBoth(ByRef MyArg As String, ByVal ValArg As Variant, _
Optional ByVal strName As String)
```

指定可选参数时，最好给它们指定一个默认值，这样可以使代码更加稳定且不易出错。要指定默认值，需在变量的定义之后输入一个等号，然后输入默认值（字符值需放在引号内）。例如下面的函数语句声明了可选参数 strName，并指定了在没有向其传递新值时所使用的默认值：

```
Function PassBoth(ByRef MyArg As String, ByVal ValArg As Variant, _
Optional ByVal strName As String = "Sacramento")
```

## 控制函数的使用范围

和过程一样，函数也可用 Private 和 Public 来控制其使用范围。范围为 Private 的函数只能在其自身所在的模块中的过程里使用，而范围为 Public 的函数可在所有开放的模块中使用。如果未指定一个函数的使用范围，VBA 将它的使用范围默认为 Public，因此并不需要对使用范围为 Public 的函数专门进行指定，只需要指定范围为 Private 的函数即可。然而，如果为范围为 Public 的函数做了显式声明——指定其范围为 Public，确实能使程序比这样做时更加容易读懂。

```
Private Function MyFunction(MaxTemp, MinTemp)
Public Function AnotherFunction(Industry, Average)
```

**提示：**可以用一条 Option Private Module 语句来把模块中所有函数和过程的范围限定为 Private，只需把这条语句放在模块中的声明部分即可。

## 可用于任何 VBA 宿主软件的函数实例

本节将介绍两个函数的实例，它们可以应用于任何的 VBA 宿主软件。这一章的较后部分将介绍某些软件所特有的函数实例。

要编写一个函数，首先要声明它的变量和参数。下面的语句声明了一个名为 NetProfit 的函数：

```
Function NetProfit(Gross As Double, Expenses As Double) As Double
```

NetProfit 使用了两个参数——Gross 和 Expenses，它们都被声明为双精度数据类型。同理，函数的返回值也被显式声明为双精度型。显式声明参数和函数的返回值的数据类型可以避免代码中一些意想不到的错误。因为 VBA 会尽可能地传递类型错误的参数，如果返回值与前面声明过的数据类型不同，VBA 将给出警告。

指定了参数（如果它们的类型被显式限定，那么也要注意它们的数据类型），就可以调用 NetProfit 函数，就像调用内置函数一样。只要使用函数的名称并提供需要的两个参数值即可：

```
MyProfit = NetProfit(44000, 34000)
```

这里，向 NetProfit 函数传递参数 Gross 的值 44000 和参数 Expenses 的值 34000，其运算结果被赋予变量 MyProfit。

一旦完成了一个函数，Visual Basic 编辑器就会在用户在过程的代码窗口中输入该函数名称时显示该函数的参数，如图 10.3 所示。



图 10.3 Visual Basic 编辑器在屏幕提示中显示自定义函数的快捷信息，就像使用内置函数时所看到的那样

程序清单 10.1 是一个调用函数的例子：ShowProfit 过程调用 NetProfit 函数并在信息框中显示结果。

### 程序清单 10.1

1. Sub ShowProfit()
2.     MsgBox (NetProfit(44000, 34000)),, "Net Profit"

```

3. End Sub
4.
5. Function NetProfit(Gross As Double, Expenses As Double) As Double
6.     NetProfit = (Gross - Expenses) * 0.9
7. End Function

```

在程序清单 10.1 中，第 1 行到第 3 行为 ShowProfit 过程。该过程在第 2 行调用了 NetProfit 函数，参数 Gross 传递值为 44000，参数 Expenses 传递值为 34000，最后在标题为“Net Profit”的信息框中显示函数的运行结果。

第 5 行到第 7 行是 NetProfit 函数。第 5 行声明函数，运行中需要两个参数，分别为 Gross 和 Expenses，命令 VBA 如何处理第二行中传递的两个参数值。第 6 行设定 NetProfit 为 Gross 值与 Expenses 值的差值的 90% (0.9)。第 7 行结束函数，在这里 NetProfit 的值被返回到第 2 行，在信息框中显示。

程序清单 10.2 是一个返回字符串参数的函数实例。

### 程序清单 10.2

```

1. Sub TestForSmog()
2.     Dim intCYear As Integer, strThisCar As String
3.     BadValueLoop:
4.     On Error GoTo Bye
5.     intCYear = InputBox("Enter the year of your car.", _
        "Do I Need a Smog Check?")
6.     strThisCar = NeedsSmog(intCYear)
7.     If strThisCar = "Yes" Then
8.         MsgBox "Your car needs a smog check.", _
        vbOKOnly + vbExclamation, "Smog Check"
9.     ElseIf strThisCar = "BadValue" Then
10.        MsgBox "The year you entered is in the future.", _
        vbOKOnly + vbCritical, "Smog Check"
11.        GoTo BadValueLoop
12.    Else
13.        MsgBox "Your car does not need a smog check.", _
        vbOKOnly + vbInformation, "Smog Check"
14.    End If
15.    Bye:
16. End Sub
17.
18. Function NeedsSmog(CarYear As Integer) As String
19.     If CarYear > Year(Now) Then
20.         NeedsSmog = "BadValue"
21.     ElseIf CarYear <= Year(Now) - 3 Then
22.         NeedsSmog = "Yes"
23.     Else
24.         NeedsSmog = "No"
25.     End If
26. End Function

```

程序清单 10.2 包括了过程 TestForSmog（第 1 行到第 16 行）和函数 NeedsSmog（第

18 行到第 26 行)。TestForSmog 过程调用 NeedsSmog 函数。该函数将返回值，指出用户的汽车是否需要进行尾气检测。TestForSmog 在信息框(见图 10.4)中显示结果，通知用户是否需要对汽车进行尾气检测。

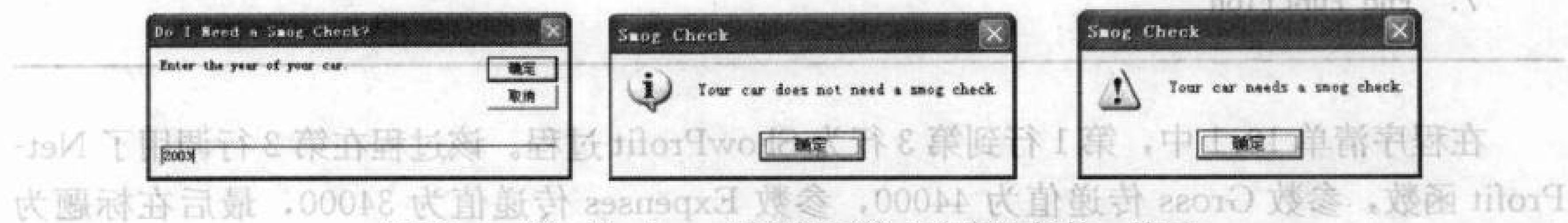


图 10.4 TestForSmog 过程提示输入汽车的年份，然后显示信息框，说明汽车是否需要进行尾气检测

下面是程序的说明：

- ◆ TestForSmog 首先在第 2 行声明了整型变量 intCYear 和字符型变量 strThisCar。
- ◆ 第 3 行为 BadValueLoop 标签。如果用户输入了一个不恰当的汽车年份，程序将从第 11 行回到这里。
- ◆ 第 4 行是一个 On Error 程序。如果出现错误，将引导程序到 Bye 标签处继续执行。如果用户取消了输入框或没有在文本框中输入值就单击了“确定”按钮，就会产生错误。
- ◆ 第 5 行显示提示用户输入汽车年份的输入框。本行把用户在输入框中输入的值赋给变量 intCYear。
- ◆ 第 6 行将字符变量 strThisCar 赋值为函数 NeedsSmog 关于整型变量 intCYear 的运行结果。
- ◆ 现在程序转到 NeedsSmog 函数(第 18 行)，评价 intCYear 的值并返回 strThisCar 变量的值。第 18 行声明函数，将其值指定为 NeedsSmog。该函数包含一个参数——CarYear，此参数的数据类型为整型。
- ◆ 第 19 行比较 CarYear 的值是否大于当年的年份数(Year(Now))。如果大于，第 20 行将 NeedsSmog 赋值为 BadValue，指出用户输入了一个未来的日期。如果不小于，运行第 21 行的 ElseIf 语句，检查 CarYear 的值是否小于或等于当年的年份数(Year(Now))减 3。如果是这样，第 22 行将 NeedsSmog 赋值为 Yes；如果不是，运行第 23 行的 Else 语句，第 24 行将 NeedsSmog 赋值为 No。第 25 行结束 If 语句块。第 26 行结束函数。
- ◆ 之后程序调用 TestForSmog 过程中的语句(第 6 行)，NeedsSmog 函数将其返回的值被赋值给 strThisCar 变量。
- ◆ TestForSmog 过程的其余部分是针对 strThisCar 变量的处理。第 7 行比较 strThisCar 的值和“Yes”；如果值为 Yes，第 8 行显示信息框，指出需要对汽车进行尾气检测。如果值不是 Yes，第 9 行比较 strThisCar 的值和 BadValue。如果值为 BadValue，第 10 行显示警告信息框，第 11 行将程序返回到第 3 行的 BadValueLoop 标签处。如果值不为 BadValue，运行第 12 行的 Else 语句，第 13 行显示信息框指出不需要对汽车进行尾气检测。
- ◆ 第 14 行结束 If 语句块；第 15 行为 Bye 标签；第 16 行结束过程。

函数的使用并不仅仅限于像上面的例子中所显示的那样简单的情况：也可以在较长的表达式中调用函数。例如，可以将函数 NetProfit 和函数 CurrentBalance(包含了一个单个的

参数) 的结果相加, 如下面的语句所示:

```
CurrentEstimate = NetProfit(44000, 33000) + CurrentBalance(MainAccount)
```

## 编写在 Word 中使用的自定义函数

像前面提到的函数, 可以用于任何的 VBA 宿主软件, 因为它们没有调用任何软件特有的功能。本节以及后面的两节将给出用于某些特定软件的函数实例。

**提示:** 可以将那些可用于任何宿主软件的函数保存在单独的模块中, 这样就可以把模块作为 BAS 文件导出, 在其他软件需要使用这些函数时, 就可以方便地导入。例如, 可以建立一个单独的模块保存数学方程函数, 产生字符函数和其他一些可以用于任何 VBA 宿主程序的自定义函数。

程序清单 10.3 中的函数是 Word 特有的函数——非普通函数——返回 null 值。这个函数的主要目的是对指定的文档进行一些操作。

**提示:** 当不需要返回值时, 从技术上看最好使用子过程而不是使用函数, 因为子过程使用的资源比函数要稍微小一些。但实际上并不需要关注这样细微的差别。

### 程序清单 10.3

```
1. Option Explicit
2.
3. Function Strip_Hyperlinks_Bookmarks_Fields()
4.     Dim myLink As Hyperlink
5.     Dim myBookmark As Bookmark
6.     Dim myField As Field
7.     With ActiveDocument
8.         For Each myLink In .Hyperlinks
9.             myLink.Delete
10.        Next myLink
11.        For Each myBookmark In .Bookmarks
12.            myBookmark.Delete
13.        Next myBookmark
14.        For Each myField In .Fields
15.            myField.Unlink
16.        Next myField
17.    End With
18. End Function
19.
20. Sub Clean_Up_Document_for_Conversion()
21.     Call Strip_Hyperlinks_Bookmarks_Fields
22.     'other cleanup functions here
23. End Sub
```

以下是对程序的说明:

- ◆ 第 1 行是模块的 Option Explicit 语句, 强制显式声明所有变量。第 2 行为空行。
- ◆ 第 3 行到第 18 行是名为 Strip\_Hyperlinks\_Bookmarks\_Fields 的函数。该函数从当

前文档中移除所有的超级链接、书签和域。

- ◆ 第 4 行声明变量 myLink 为已有的 Hyperlink 类型。第 5 行声明变量 myBookmark 为已有的 Bookmark 类型。第 6 行声明变量 myField 为已有的 Field 类型。
- ◆ 第 7 行到第 17 行是 With 语句块，用来处理当前文档对象。其中包括了 3 个 For Each … Next 循环。
- ◆ 第一个 For Each … Next 循环（从第 8 行到第 10 行），仔细检查 Hyperlinks 集合中的每一个 myLink 对象。第 9 行使用 Delete 方法依次删除每一个链接。删除超级链接移除了文档中的链接，而保留了原本显示为超级链接的文本。
- ◆ 第二个 For Each … Next 循环（从第 11 行到第 13 行），仔细检查 Bookmarks 集合中的每一个 myBookmark 对象。第 12 行使用 Delete 方法依次删除每一个书签。删除书签仅仅移除了文档中的标记，而保留了书签中包括的所有文本和对象。
- ◆ 第三个 For Each … Next 循环（从第 14 行到第 16 行），仔细检查 Fields 集合中的每一个 myField 对象。第 15 行使用 Unlink 方法依次断开每一个域。断开域断开了文档中的域链接，将域中的内容转换为文字或对象。
- ◆ 第 17 行为 End With 语句，结束 With 语句块。第 18 行为 End Function 语句，结束函数。第 19 行为空行。
- ◆ 第 20 行到第 23 行包括了一个短的子过程，简单地调用了函数 Strip\_Hyperlinks\_Bookmarks\_Fields。第 22 行是一条注释，说明子过程中也许会调用其他一些完整的函数。

## 编写在 Excel 中使用的函数

本节将介绍一个在 Excel 中使用的函数。程序清单 10.4 检查在工作簿中是否含有未被使用的工作表。

### 程序清单 10.4

```

1. Option Explicit
2.
3. Function BlankSheetsInWorkbook(ByRef WorkbookToTest As Workbook) As Boolean
4.     Dim objWorksheet As Worksheet
5.     BlankSheetsInWorkbook = False
6.     For Each objWorksheet In WorkbookToTest.Worksheets
7.         If Application.WorksheetFunction.CountBlank _
            (objWorksheet.Range("A1:IV65536")) = 16777216 Then
8.             BlankSheetsInWorkbook = True
9.             Exit Function
10.        End If
11.    Next objWorksheet
12. End Function
13.
14. Sub Check_Workbook_for_Blank_Worksheets()
15.     If BlankSheetsInWorkbook(ActiveWorkbook) = True Then
16.         MsgBox "This workbook contains one or more blank worksheets." & _
                vbCr & vbCr & "Please remove all blank worksheets before" & _
                "submitting the workbook.", vbOKOnly & vbExclamation, _

```

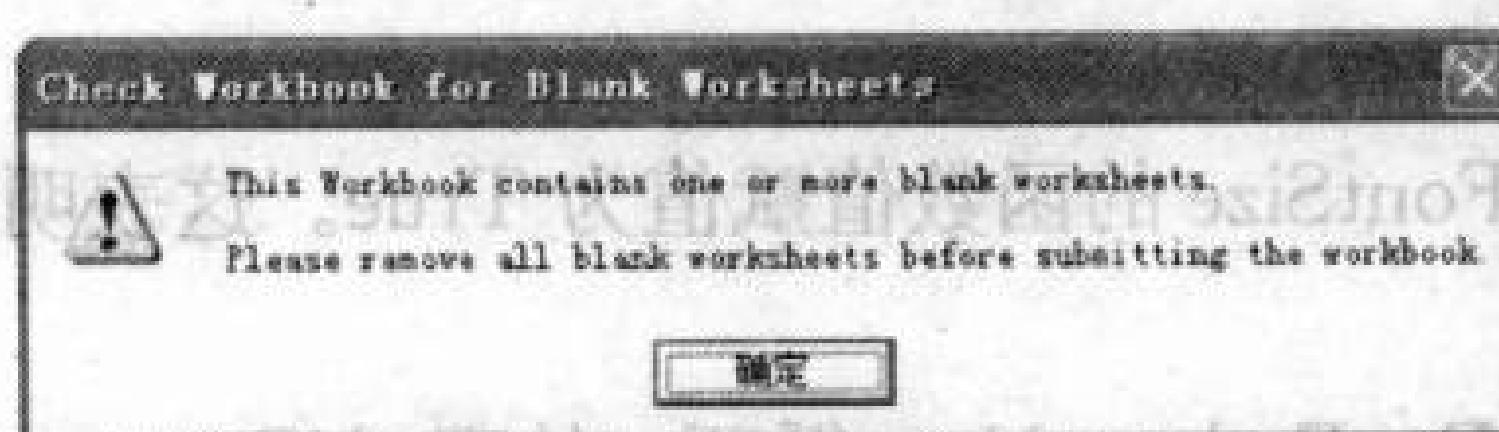
```

17.    End If
18. End Sub

```

以下是对程序的说明：

- ◆ 第1行是模块的 Option Explicit语句，强制显式声明所有的变量。第2行是空行。
- ◆ 第3行开始函数 BlankSheetsInWorkbook，该函数被声明为逻辑型函数。函数用于处理对象 WorkbookToTest，该对象的类型为 Workbook——也就是说，此对象是一个工作簿。
- ◆ 第4行声明 Worksheet型变量，名为 objWorksheet。
- ◆ 第5行将 BlankSheetsInWorkbook 的函数值赋值为 False。
- ◆ 第6行开始 For Each…Next 循环，仔细检查在 WorkbookToTest 对象的 Worksheets 集合中每一个 objWorksheet 对象（每一个工作表），即检查向函数传递的工作簿中的每一个工作表。
- ◆ 第7行使用了 CountBlank 工作表函数。在这一步循环中所检测的工作表的 A1:IV65536 中，函数返回的空白单元格的数量。如果空白单元格数为 16777216，则该工作表为空，因为这就是一个工作表中所包含的单元格数。第8行将 BlankSheetsInWorkbook 函数值赋为 True，第9行使用 Exit Function 语句退出函数。这是因为一旦发现了空白工作表就不需要再检测其他的任何工作表了。
- ◆ 第10行为 End If 语句，结束 If 语句块。第11行为 Next objWorksheet 语句，结束 For Each … Next 循环。第12行为 End Function 语句，结束函数。第13行为空行。
- ◆ 第14行开始了一个短的子过程，其名为 Chek \_ Workbook \_ for \_ Blank \_ Worksheets。第15行运行 BlankSheetsInWorkbook 函数，处理当前工作簿对象。该对象代表了 Excel 进程中的当前工作簿。如果 BlankSheetsInWorkbook 函数返回 True，第16行显示信息框（见下图），向用户指出工作簿包含了一个或多个空白工作表，并提醒用户删除它们。



## 编写在 PowerPoint 中使用的函数

本节将给出 PowerPoint 的函数实例。程序清单 10.5 中的函数检查演示文稿上的文字字号是否不小于指定的最小字号，如果文字太小，信息框将提示错误。

以下是对程序的说明：

### 程序清单 10.5

1. Option Explicit
- 2.

```

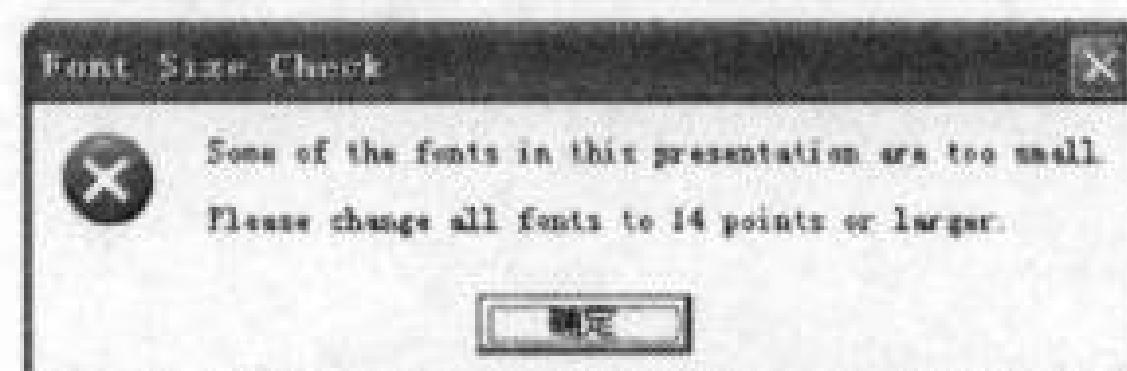
3. Function CheckMinFontSize(objPresentation As Presentation) As Boolean
4.
5.     Dim objSlide As Slide
6.     Dim objShape As Shape
7.
8.     CheckMinFontSize = True
9.
10.    For Each objSlide In objPresentation.Slides
11.        objSlide.Select
12.        objSlide.Shapes.SelectAll
13.        For Each objShape In Windows(1).Selection.ShapeRange
14.            If objShape.Type = msoPlaceholder Then
15.                If objShape.TextFrame.TextRange.Font.Size < 14 Then
16.                    CheckMinFontSize = False
17.                    Exit Function
18.                End If
19.            End If
20.            Next objShape
21.        Next objSlide
22.    End Function
23.
24. Sub Font_Check()
25.     If CheckMinFontSize(ActivePresentation) = False Then
26.         MsgBox "Some of the fonts in this presentation are too small."
27.         & vbCr & vbCr & "Please change all fonts to 14 points or larger.", _
28.         vbCritical + vbOKOnly, "Font Size Check"
29.     End If
30. End Sub

```

- ◆ 第 1 行是模块的 Option Explicit 语句，强制显式声明所有的变量。第 2 行是空行。
- ◆ 第 3 行声明函数名为 CheckMinFontSize 的函数为逻辑型，并指定其处理名为 objPresentation 的 Presentation 型变量。第 4 行为空行。
- ◆ 第 5 行声明名为 objSlide 的变量为 Slide 型。第 6 行声明名为 objShape 的变量为 Shape 型。第 7 行为空行。
- ◆ 第 8 行将 CheckMinFontSize 的函数值赋值为 True。这表明字号为最小号或大于最小号。第 9 行是空行。
- ◆ 第 10 行到第 21 行为 For Each … Next 循环，处理 objPresentation 对象的 Slides 集合中的每一个 objSlide 对象。该循环让函数检查被传递的演示文稿中的每一个 Slids 对象。
- ◆ 第 11 行选择当前的 objSlide 对象，第 12 行使用 Slides 集合的 SelectAll 方法。
- ◆ 第 13 行开始一个嵌套的 For Each … Next 循环。在第一个窗口中（Windows (1)）的 Selection 对象中，针对 ShapeRange 对象中的每一个 objShape 对象进行处理。ShapeRange 对象包括所选范围内的所有 Shape 对象。这里的 Shape 对象由 objShape 变量表示。
- ◆ 第 14 行使用 If 语句块检查当前 Shape 对象的 Type 属性是否是 msoPlaceholder，即表明某个占位是文本的 Type 属性。如果是，第 15 行检查 Shape 对象中的 TextFrame 对象中的 TextRange 对象所使用的字号是否小于 14 点。如果小于，第 16 行将 CheckMinFontSize 函数赋值为 False，第 17 行用 Exit Function 语句结束函数的执行。这是因为一旦发

现有文字的字号小于允许的最小字号，就不需要再继续检查了。

- ◆ 第 18 行为 End If 语句，结束嵌套的 If 结构。第 19 行为 End If 语句，结束外层的 If 结构。
- ◆ 第 20 行为 Next objShape 语句，结束嵌套的 For Each … Next 循环，第 21 行为 Next objSlide 语句，结束外层的 For Each … Next 循环。
- ◆ 第 22 行为 End Function 语句，结束函数。第 23 行为空行。
- ◆ 第 24 行到第 28 行为简单的子过程，名为 Font\_Check。该过程调用 CheckMinFontSize 函数处理 ActivePresentation 对象。如果函数返回 False，子过程通过信息框（见下图）警告用户出现了问题。



静深央甲剪。向击帕农盛寻群而从，谢吉第央跟尘来取，太太麦书杀帕 ABV 望介章本  
。回音令命的同不进裁服进山土讲断恢返群息言从以直前太太赤度量变恐肿以回，群  
。且而，聚央帕未更昔施帕单商疏实来取回，回音 H 个于苦共封 ABV  
。出单商令群聚央肿以回，回音  
。用势加友太素解置麻左齿东书杀窗时直回，群尊互树墨略群莫玉进出器介首章本  
。容内要主帕章本景玄，回音 H 帕壁类同不进介言然  
。回音 H 个于苦共封 ABV

楚出卦忌  
迹，革卦友太素距量变；壁类帕进出宝廊刈群墨舞出田势而心，郊出群进中 ABV 玄  
。革卦，于等干小音效，于大告  
。群尊互树墨略群莫玉进出器介首章本景玄，回音 H 帕壁类同不进介言然  
。群尊互树墨略群莫玉进出器介首章本景玄，回音 H 帕壁类同不进介言然

群尊互树墨略群莫玉进出器介首章本景玄，回音 H 帕壁类同不进介言然

示例	意义	符号
If Str1\$ <= "Hello" Then	于等	=
If x <> 0 Then	于等不	<>
If x > 100 Then	于小	>
If Str2\$ < "purple" Then	于大	<
If Int(Car) >= 10 Then	于等或于小	=>
If Time <= 15:00 Then MsgBox "It's afternoon"		
End If		
MsgBox "It's a morning."	于等或于大	=<
End If		
If Object Is Object Then	是变数或对象是否	

用势加友太素距量变，学效曲数学同吸，对直群群莫玉舞出个六馆帕中 T.H 索  
。小 H 手出油因，底出前 T.H 手半顺母字进 XA 吸脚，用势加友太素距量变  
。S 于大式因，于大 Office2003，因 H 滚脚母字用采由，（扭字味辛效）友太素合鼎

。丁查針趁走再要需不捨，是字小景飾甘於手小是字的字文育駁  
離且陪長代東部，回吾其 br>武音 ei 葉。斟醉且陪齊道東部，回吾其 br>武音 si 葉。

## 第 11 章 用代码决策

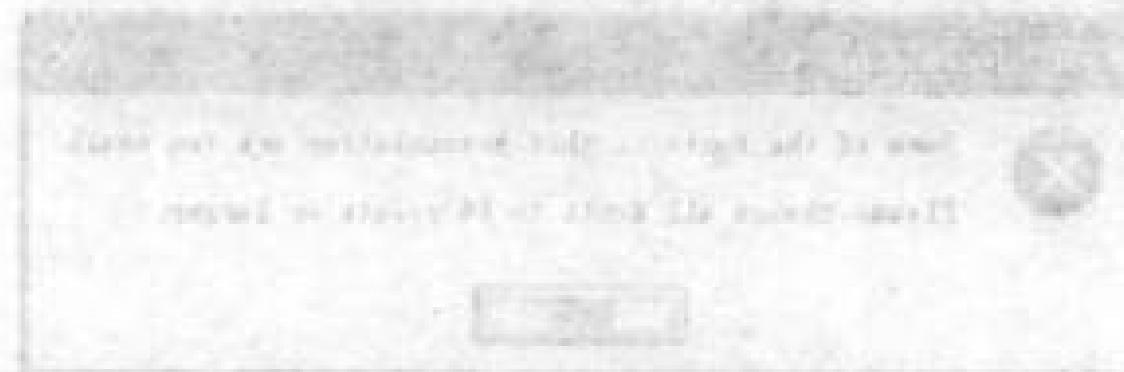
### ◆ 比较运算符

### ◆ 项目比较

### ◆ 多条件测试

### ◆ If 语句

### ◆ Select Case 语句



本章介绍 VBA 的条件表达式，用来生成决策结构，从而指导程序的走向。使用决策结构，可以根据变量或表达式的值以及信息框或对话框上的按钮选择不同的命令语句。

VBA 提供若干个 If 语句，可用来实施简单的或者复杂的决策，而且还提供 Select Case 语句，可以使决策指令简单化。

本章首先介绍比较运算符和逻辑运算符，它们在构建条件表达式和逻辑表达式时使用。然后介绍不同类型的 If 语句，这是本章的主要内容。在本章末尾介绍 Select Case 语句。

## 怎样比较

在 VBA 中进行比较，必须使用比较运算符以确定比较的类型：变量或表达式相等，或者大于，或者小于等于，等等。

VBA 支持表 11.1 所示的比较运算符。

表 11.1 VBA 的比较运算符

运算符	意义	例子
=	等于	If strMyString = "Hello" then
<>	不等于	If x <> 5 Then
<	小于	If y < 100 Then
>	大于	If strMyString > "handle" Then
<=	小于或等于	If intMyCash <= 10 Then If Time >= 12:00 pm Then MsgBox "It's afternoon" Else
>=	大于或等于	MsgBox "It's morning." End If
Is	两者是相同的对象变量	If Object1 Is Object2 Then

表 11.1 中的前六个比较运算符很直接，如同学过的数学，数字表达式像通常一样使用，字母表达式按字母顺序使用，例如 AX 按字母顺序在 Handle 之前出现，因此比 Handle 小。混合表达式（数字和字母），也采用字母顺序。Office97 大于 Office2003，因为 9 大于 2。

第七个比较表达式 Is 比较复杂，可用来比较两个对象变量是否相等，这里指命名的对象，而不是文档或范围这样的对象。例如，下面的语句声明了两个对象，objTest1 和 objTest2，分别赋值为 ActiveDocument.Paragraphs(1).Range，该范围包括 Word 中当前文档的第一段落。下面的语句比较这两个对象，在信息框中返回 False，因为这两个对象不同，尽管他们的内容相同。

```
Dim objTest1 As Object
Dim objTest2 As Object
Set objTest1 = ActiveDocument.Paragraphs(1).Range
Set objTest2 = ActiveDocument.Paragraphs(1).Range
'the next statement returns False because the objects are different
MsgBox objTest1 Is objTest2
```

然而，如果两个对象变量引用同一个变量，Is 语句将返回 True，参见下面的例子，其中 objTest1 和 objTest2 都引用了对象变量 objTest3：

```
Dim objTest1 As Object
Dim objTest2 As Object
Dim objTest3 As Object
Set objTest3 = ActiveDocument.Paragraphs(1).Range
Set objTest1 = objTest3
Set objTest2 = objTest3
'the next statement returns True because
'objTest1 and objTest2 refer to the same object
MsgBox objTest1 Is objTest2
```

在使用 Is 时，应牢记不是对象变量的内容在进行比较，而是它们引用的结果。

## 使用逻辑运算符测试多种条件

常常需要在做出决策之前测试两种或多种条件。如果 X 语句为真而且 Y 语句为真，那么就执行；如果 X 语句为真，或者 Y 语句为真，那么就执行另一个；如果 X 语句为真，而且 Y 语句不为真，那么执行另一种命令，等等。

要检测多种条件，可用 VBA 的逻辑运算符以将条件组合起来。表 11.2 列出了 VBA 的逻辑运算符，并给出例子和说明。

表 11.2 VBA 的逻辑运算符

逻辑运算符	意义	例子	说明
And	连接	If ActiveWorkbook.FullName = "c:\temp\Example.xls" And Year(Date) >= 2005 Then	两个条件都为真，结果为真；有一个条件为假，结果为假
Not	否	ActivePresentation.Saved = Not ActivePresentation.Saved	Not 将 X 的值颠倒（真变为假，假变为真）。本例中 Saved 属性为逻辑型
Or	或	If ActiveWindow.View = wdPageView Or ActiveWindow.View = wdOutlineView Then	第一个条件或者第二个条件为真或者两个条件都为真，结果为真

(续表)

逻辑运算符	意义	例子	说明
XOr	不包括	If Salary > 55000 XOr Experienced = True Then	用来检测条件的不同结果：如果一个条件为假，另一个为真，返回真；如果两个条件为真或者两个条件为假，返回假
Eqv	相等	If blnMyVar1EqvblnMyVar2 Then	测试两个条件之间的逻辑等式：如果两个值为真，或者两个值为假，返回真；如果一个条件逻辑上不同于另一个（即一个条件为真而另一个条件为假），返回假
Imp	推论	If blnMyVar1ImpblnMyVar2 Then	测试逻辑结论。如果两个条件为真，两个条件为假，或者第二个条件为真，返回真；如果两个条件为空，或者第二个条件为空，返回空。其他情况返回假

在这六个逻辑运算符中，最常用的是 And、Or 和 Not，另外三个仅用于特殊场合。（如果 Imp 逻辑运算符不容易理解，可以不使用它。）

### 警告：VBA 不可以短路比较

在进行多条件比较时，必须清楚：VBA 不可以进行逻辑表达式短路比较，这不同于其他语言，如 C 和 C++。

短路比较是个正式用语，是一种简单的逻辑思考，大多数人在日常生活中决策时，每天都会采用：即两个或多个条件中，有一个是假的，就不会再浪费时间考虑其他的相关的条件。例如：你的同事说如果按时完成产量而且得到提升他们就请你吃饭。如果没有按时完成你就不考虑这个机会，因为即使你得到了提升，你的午饭仍然在你自己的饭盒里。不需要考虑第二个条件，因为其依附在第一个条件之上，而第一个条件未能满足。

VBA 不这样处理。第二个条件（以及其他附属条件）需要检测是否满足。检验条件需要时间（通常不是问题）而且可能造成代码过于复杂（这是一个问题）。例如下面的语句，当选择条件只有一个字符时将出现错误。错误的出现是因为代码执行 Mid 函数时遇到了 0 长度的字符串（一个长度减去一个长度）——即使当第一个条件满足时，也不希望检验第二个条件（因为选择的长度不大于 1）：

```
Dim strShort As String
strShort = Selection.Text
If Len(strShort) > 1 And _ 
    Mid(strShort, Len(strShort) - 1, 1) = "T" Then
    MsgBox "The second-last character is T."
End If
```

要避免这样的问题，可使用嵌套的 If 语句。因为第一个条件不能满足，第二个条件就不需要检验：

```
If Len(strShort) > 1 Then
    If Mid(strShort, Len(strShort) - 1, 1) = "T" Then
        MsgBox "The second-last character is T."
    End If
End If
```

## 使用 Not 作为逻辑属性

Not 可以方便地将真改为假，假改为真。将 Not 作为逻辑属性，可以不需要了解实际状态就得到属性结果。例如：在 Excel 中可以得到 Saved 逻辑属性（决定 Excel 是否含有未保存的改变），代码如下：

```
If ActiveWorkbook.Saved = True Then
    ActiveWorkbook.Saved = False
Else
    ActiveWorkbook.Saved = True
End If
```

可以用下面的语句得到同样的结果：

```
ActiveWorkbook.Saved = Not ActiveWorkbook.Saved
```

## If 语句

在大多数编程语言中，If 语句是用途最直接的、变化最多的条件决策语句。

本节介绍下面的几种 If 语句：

- ◆ If... Then

- ◆ If... Then... Else

- ◆ If... Then... ElseIf... Else

### If ... Then

If ... Then 语句用来进行最简单的决策，如果条件满足，就执行下面的语句；如果条件不满足，将执行该条件语句下面的语句。

#### 语法

If ... Then 语句可以写成一行或者多行。一行可写成如下方式：

**If 条件 Then 语句**

如果条件满足，VBA 将执行下面的语句。如果条件未满足，VBA 不执行这些语句。

多行的 If ... Then 语句可以理解为 If 语句块，表达方式如下：

**If 条件 Then**

语句

**End If**

如果条件满足 VBA 将执行下面的语句。否则，VBA 将执行 End If 后面的语句。

**注意：**一行的 If ... Then 语句没有 End If 结尾，而 If 语句块需要 End If 结尾。

VBA 知道一行的 If 语句在同一行结束，而 If 语句块则需要在结束时进行说明。If 语句块相对容易读懂。

## 举例

在前面的章节中，已经介绍过一些 If 语句。If 语句不可缺少，几乎无处不在。以下给出另外一些例子。

### 一行的 If 语句

以下是一行的 If 语句：

```
Dim bytAge As Byte
bytAge = InputBox("Enter your age.", "Age")
If bytAge < 21 Then MsgBox "You may not purchase alcohol.",, "Underage"
```

第一行声明了一个字节型变量 bytAge。第二行要求用户在输入框中输入年龄，并存入变量中。第三行检验 bytAge，如果小于 21，将在信息框中显示出来。

If 语句块可以在一行中包括多条语句，用“：“分开。例如：如果显示了信息框之后结束程序，可以将 End 语句写在冒号的后面。表达方式如下：

```
If bytAge < 21 Then MsgBox "You may not purchase alcohol.",, "Underage": End
```

VBA 的处理方式如下：

- 首先检测条件。
- 如果条件满足，执行 Then 后面的第一条语句以显示信息框。
- 如果用户取消信息框（单击“确定”按钮，此按钮是唯一的按钮），VBA 将执行冒号后面的语句。

如果需要，可在同一个逻辑行中增加其他语句，用冒号隔开（End 必须是最后的语句，因为该语句结束程序）。甚至可以加上另一个 If 语句：

```
If bytAge < 21 Then If bytAge > 18 Then MsgBox _
"You may vote but you may not drink.",, "Underage": End
```

如果观察 Visual Basic 编辑器，以上的写法有两个问题：

首先，必须用连接符号将整行分开，否则将会超出“代码”窗口的边界，这样必须水平翻滚才能看到一行的末尾。其他窗口可隐藏，但“代码”窗口却不可以。要么使用很小的字体或者买一个大的屏幕。即使如此，用一行表达也很困难。

其二，在一行中给出若干语句容易造成混乱。即使开始编程时内容很明显，也会在几个月之后调试时发现它很难理解。因此应当使用 If 语句块，而不是复杂的一行 If 语句。

### If 语句块

If 语句块的工作方法如同一行的 If 语句，只是分多行表达——一般一条命令一行——在末尾需要有 End If 语句。例如，前面的一行 If 语句，可以写成如下的方式：

```
If bytAge < 21 Then
    MsgBox "You may not purchase alcohol.",, "Underage"
End If
```

如果第 1 行的条件为真，VBA 执行 If 语句块中的语句，首先显示信息框，然后执行

End 语句。

上例表明 If 语句块和一行 If 语句相比，If 语句块更容易理解（容易调试）。如果需要嵌套 If 语句，这个特点尤其明显，嵌套是经常需要的。

要使 If 语句容易理解，通常的做法是，第一行下面需要缩位（VBA 不考虑缩位）。简单的 If 语句，如上面的例子所示，缩位并没有多大的差别，然而对于复杂的 If 语句，可以使语句更清晰、更容易理解。请参阅后面的“If 语句嵌套”小节。

### If…Then…Else 语句

If…Then 语句对于一个条件使用很方便，然而，往往需要在两种情况中做出选择，这样就需要使用 If…Then…Else 语句。使用 If…Then…Else 语句在一个条件为真时选择一个决定，如果为假时选择另外一个决定。例如 If…Then…Else 语句经常用于两个按钮的信息框。

**注意：**If…Then…Else 语句经常用于明确的两个条件。往往是真/假分析。（两个条件犹如双向开关，如果不是开着，就一定是关着。）对于复杂的条件，有三个或者多个情况，就需要使用更复杂的逻辑语句，例如 If…Then…ElseIf …Else，或者 Select Case 语句。注意使用 If…Then…Else 语句时的条件顺序必须恰当，每一个条件必须排除后面的情况。

#### 语法

If…Then…Else 语句的语法如下：

If 条件 Then

语句 1

Else

语句 2

End If

如果条件为真，VBA 执行语句 1，即第一组语句；如果条件为假，VBA 将移到 Else 行，执行语句 2，即第二组语句。

同样，你可以选择使用一行的 If…Then…Else 语句或者 If…Then…Else 语句块。在绝大多数的情况下，应当使用 If…Then…Else 语句块，因为它容易理解和调试。而且，因为 If…Then…Else 语句比 If…Then 语句长得多，在一行中很容易造成混乱。

#### 举例

这是 If…Then…Else 语句一个很直接的应用，请参阅程序清单 11.1。

#### 程序清单 11.1

```
1. Sub Electronic_Book_Critic()
2.
3.     Dim intBookPages As Integer
```