

```

4.
5. intBookPages = InputBox _  

   ("Enter the number of pages in the last book you read.", _  

    "The Electronic Book Critic")
6. If intBookPages > 1000 Then  

7.   MsgBox "That book is seriously long.", vbOKOnly _  

     + vbExclamation, "The Electronic Book Critic"
8. Else  

9.   MsgBox "That book is not so long.", vbOKOnly _  

     + vbInformation, "The Electronic Book Critic"
10. End If
11.
12. End Sub

```

以下是程序清单 11.1 的说明：

- ◆ 第 1 行为程序开始。第 12 行程序结束，第 2、4、11 行为空白行。
- ◆ 第 3 行声明整型变量 intBookPages。第 5 行要求用户在信息框中输入读书的最后一页的页码，并将结果赋值给整型变量 intBookPages。
- ◆ 第 6 行检验变量 intBookPages 是否超过 1000，如果是，就执行第 7 行的语句，显示信息框以说明该书太长。
- ◆ 如果变量 intBookPages 未超过 1000 行，VBA 将转到 Else 的第 8 行语句上，显示信息框以告诉用户该书不是很长。
- ◆ 第 10 行结束 If 条件。

If…Then…Else If…Else 语句

最后要介绍的 If 语句是 If…Then…Else If…Else 语句，用来进行多条件选择。可以使用任意个 Else If，这取决于条件的复杂情况。

同样，也可以使用一行的 If…Then…Else If…Else 语句或 If…Then…Else If…Else 语句块。在所有的情况下，If…Then…Else If…Else 语句块更容易构造、阅读和调试。和其他 If 语句一样，一行 If…Then…Else If…Else 语句不需要 End If 结束语句，而 If…Then…Else If…Else 语句则需要 End If 语句。

语法

If…Then…ElseIf…Else 语句的语法如下：

If 条件 1 Then

语句 1

ElseIf 条件 2 Then

语句 2

[ElseIf 条件 3 Then]

语句 3]

[Else 条件 4 Then]

语句 4]

End If

如果条件 1 为真，VBA 执行语句 1，即第 1 个语句，然后恢复执行 End If 语句之后的

命令。如果条件 1 为假, VBA 转入第 1 个 ElseIf 语句, 检测条件 2 的内容。如果条件 2 为真, VBA 执行语句 2, 然后恢复执行 End If 语句之后的命令; 如果为假, VBA 转入第 1 个 ElseIf 语句 (如果有的话), 检测条件 (这里为条件 3)。

如果所有的 ElseIf 语句为假, VBA 转入 Else 语句 (如果有的话), 执行语句 4。End If 语句结束条件语句, 执行 End If 后面的命令。

在 If 语句块中可以有任意数量的 ElseIf 语句, 各自有各自的条件。然而, 如果需要相当数量的 ElseIf 语句 (超过 5 个或者 10 个), 就应当使用 Select Case 语句, 这将在后面介绍。

Else 语句是可选的, 然而在大多数情况下最好使用 ElseIf 语句, 因为它可以在 If 和 ElseIf 的条件都不能满足时再给出一个不同的选择。

举例

本小节给出两个 If...Then...ElseIf...Else 语句的例子:

- ◆ 简单的 If...Then...ElseIf...Else 语句, 用来执行 3 个按钮的信息框。
- ◆ 没有 Else 语句的 If...Then...ElseIf 语句。

简单的 If...Then...ElseIf...Else 语句

简单的 If...Then...ElseIf...Else 语句参见程序清单 11.2, 它可十分完美地用于处理带有 3 个按钮的信息框。

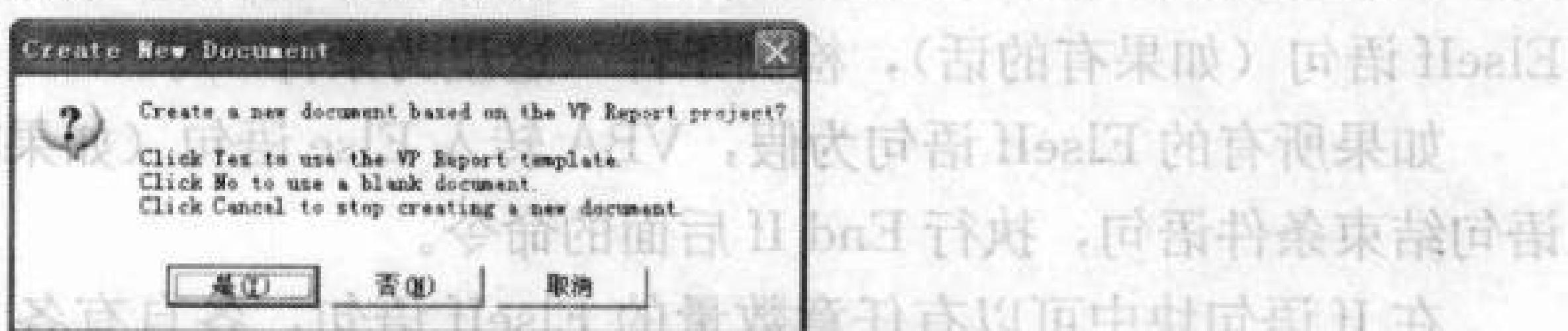
```

程序清单 11.2
1. Sub Creating_a_Document()
2.
3.     Dim lngButton As Long
4.     Dim strMessage As String
5.
6.     strMessage = "Create a new document based on the " & _
    "VP Report project?" & vbCrLf & vbCrLf & _
    "Click Yes to use the VP Report template." & vbCrLf & _
    "Click No to use a blank document." & vbCrLf & _
    "Click Cancel to stop creating a new document."
7.
8.     lngButton = MsgBox _
    (strMessage, vbYesNoCancel + vbQuestion, "Create New Document")
9.
10.    If lngButton = vbYes Then
11.        Documents.Add Template:="z:\public\template\vpreport.dot"
12.    ElseIf lngButton = vbNo Then
13.        Documents.Add
14.    Else ' lngButton is vbCancel
15.    End
16. End If
17.
18. End Sub

```

程序清单 11.2 显示一个信息框, 含有“是”、“否”、“取消”按钮, 要求用户根据 VB

报告建立一个新的文档。用户可选择“是”按钮以生成一个新文档，选择“否”按钮生成一个空文档，或者选择“取消”按钮不生成任何文档。



以下是程序的说明：

- ◆ 第 1 行开始程序，第 18 行结束程序。
- ◆ 第 2 行是空行，第 3 行声明长整型变量 lngButton。第 4 行声明字符型变量 strMessage。第 5 行为空行。
- ◆ 第 6 行对字符型变量 strMessage 赋值，包括信息框上所有的内容。第 7 行是空行。
- ◆ 第 8 行显示信息框，使用 strMessage 信息，并设定 vbYesNoCancel 常量以产生含有“是”、“否”、“取消”按钮的信息框，并且使用了合适的标题。将信息框的结果赋值给长整型变量 lngButton，第 9 行是空行。
- ◆ 第 10 行开始 If…Then…ElseIf…Else 语句，比较 IngButton 和 vbYes 的值。
- ◆ 如果第 10 行相吻合，第 11 行根据 vpreport. dot 模板生成一个新的文档，如果不吻合，使用第 12 行中 ElseIf 的条件比较 IngButton 与 vbNo 的值。

注意：如果运行该程序并且在信息框中选择“是”按钮，必须保证在目录 z:\public\templete\中有一个名为 vpreport. dot 的模板，如果没有该模板，将出现错误。如果没有这个模板，就需要改变路径和名称。

- ◆ 如果第二个条件相吻合，第 13 行使用 Add 方法产生一个空文件。如果不吻合，将执行第 14 行的 Else 语句，因为用户必须选择信息框中的“取消”按钮。第 15 行的 End 语句结束程序的执行。
- ◆ 第 16 行结束 If 语句，第 17 行为空行。

注意：本例有些特殊，在于 Else 语句受到信息框的回答限制，即“是”、“否”和“取消”，因为 If 语句检查 vbYes，Else 检查 vbNo，只有 vbCancel 可以触发 Else 语句。在其他场合，Else 语句应该包括 If 和 ElseIf 语句之外的所有内容，因此应当使 If 和 ElseIf 语句涵盖所有需要检验的条件，然后再使用 Else 语句。

没有 Else 条件的 If…Then…ElseIf 语句

可使用没有 Else 条件的 If…Then…ElseIf 语句，这样的情况是：在 If 语句中所有的条件都不为真时且又不需要采取任何措施。在前面的例子中，有三个明确的出路，用户可选择“是”按钮、“否”按钮或者“取消”按钮。因此 If 条件用来检测“是”按钮，ElseIf 条件用来检测“否”按钮，Else 条件用来针对没有选择的情况，即选择“取消”按钮。

注意：在信息框中单击“关闭”按钮等同于单击“取消”按钮。

如果所有的条件都不为真且不需要采取任何步骤，请考虑程序清单 11.3 中的 If 语句。该程序检测用户输入的密码为指定的长度。

程序清单 11.3

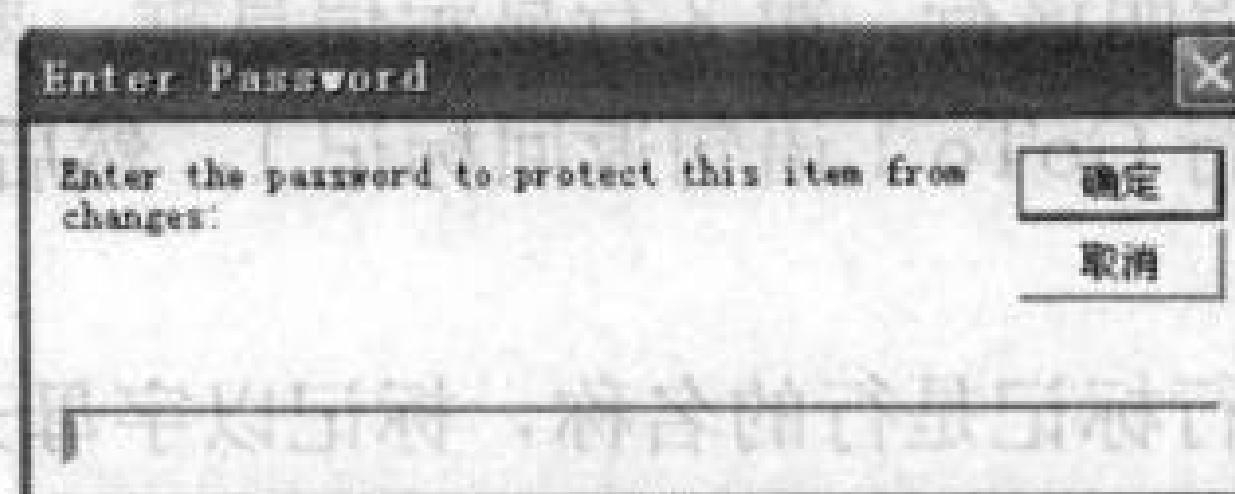
```

1. Sub Check_Password()
2.
3.     Dim strPassword As String
4.
5.     BadPassword:
6.
7.     strPassword = InputBox _
        ("Enter the password to protect this item from changes:", _
        "Enter Password")
8.
9.     If Len(strPassword) = 0 Then
10.    End
11.    ElseIf Len(strPassword) < 6 Then
12.        MsgBox "The password you chose is too short." & vbCrLf
            & vbCrLf & "Please choose a password between " &
            "6 and 15 characters in length.", _
            vbOKOnly + vbCritical, "Unsuitable Password"
13.        GoTo BadPassword
14.    ElseIf Len(strPassword) > 15 Then
15.        MsgBox "The password you chose is too long." & vbCrLf
            & vbCrLf & "Please choose a password between " &
            "6 and 15 characters in length.",
            vbOKOnly + vbCritical, "Unsuitable Password"
16.        GoTo BadPassword
17.    End If
18.
19. End Sub

```

该程序要求用户输入一个合适的密码以保护一个项目。（程序并未真实地保护某个项目。）程序的说明如下：

- ◆ 第 1 行开始程序，第 9 行程序结束。
- ◆ 第 2 行是空行，第 3 行声明字符型变量 strPassword。
- ◆ 第 4 行为空行，第 5 行为 BadPassword 标签。如果用户输入的密码不合适，VBA 将指向该语句。第 6 行又是空行。
- ◆ 第 7 行显示输入栏，要求用户输入密码，VBA 将其保存在变量 strPassword 中第 8 行为空行。



- ◆ 第 9 行检查 strPassword，判断长度是否为 0，若为 0 是空字符串。这意味着用户单击了“取消”按钮，或者未在信息框的文本框内输入内容就单击“确定”按钮。任何一种情况都将进入第 10 行，以 End 语句结束该程序。
- ◆ 如果 strPassword 的长度不为 0（用户在信息框中输入了内容，并且单击“确定”按钮），第 9 行的条件为假，VBA 进入第 11 行，检查其长度是否小于 6 个字符。

- ◆ 如果 strPassword 的长度小于 6，VBA 执行第 12 行到第 13 行的语句，第 12 行显示信息框，说明密码太短，并提供密码的标准。该信息框只有“确定”按钮，因此当用户单击时 VBA 进入第 13 行，返回第 5 行的 BadPassword 标签，程序在这里反复执行，再次出现输入栏，要求用户重新输入。
- ◆ 如果 strPassword 的长度超过 6 个字符，程序从第 11 行转入第 14 行的 ElseIf，检查 strPassword 是否超过 15 个字符。
- ◆ 如果 strPassword 超过 15 个字符，VBA 执行第 15 行和第 16 行的语句。第 15 行显示信息框，告诉用户密码太长，第 16 行回到 BadPassword 标签，又显示信息框。

在这里不需要 Else 语句，因为如果用户输入的密码不能触发 If 语句或者 ElseIf 语句，程序将执行 End If 后面的语句。

用 If 和 GoTo 产生循环

前面在执行其他任务的内容中，已经介绍过几个使用 For…Next 循环以及 For Each…Next 循环的例子（第 12 章将介绍怎样构建这一类的循环以及其他循环，例如 Do Loops。）也可以使用 If 语句和 GoTo 语句构造循环，正如上面的例子所示。

许多程序员不喜欢用 If 和 GoTo 语句构造循环，因为有更好的构建循环的方法，而且 If…GoTo 循环构建的代码太杂乱无章，不仅怪异而且无法调试。然而使用 If 和 GoTo 语句的循环，运行十分正常。即使不打算使用，也应当了解其工作方式。

语法

GoTo 语句很直接，十分有用，在前面的介绍中，已经出现过多次。其语法如下：

GoTo 行号

在这里，“行号”可以是某一行的号码或者是程序中的标记。行号就放置在一行的顶端，例如下面的例子：

```
Sub Demo_of_GoTo()
    1 If MsgBox("Go to line 1?", vbYesNo) = vbYes Then
        GoTo 1
    End If
End Sub
```

第 2 行只有行号 1，用来指明该行。第 3 行显示信息框，要求选择是否返回第一行，如果用户选择“是”，VBA 将执行 GoTo 1 语句返回标记 1，然后再次显示信息框。（如果用户选择“否”，If 语句结束。）

使用行标记比行号容易。行标记是行的名称，标记以字母开始，以冒号结束，在字母和冒号之间可以包括任意字符组合，例如本章前面使用的 BadPassword:，用来在某个条件满足时，返回程序的前半部分。也许最好的例子是 Bye: 标记，一般放在程序的末尾和 GoTo 语句一块使用。

GoTo Bye

GoTo 通常和条件一起使用。如果在 GoTo 语句之前没有返回的语句，将会出现死循

环。如果使用 GoTo 语句，不设定条件，必须保证其为最后的语句，即下面没有可执行的语句。

举例

举一个 GoTo 语句的例子，可使用 GoTo 语句和一个信息框，明确用户要执行某个程序。

```
Response = MsgBox("Do you want to create a daily report for " & _
    "the head office from the current document?", _
    vbYesNo + vbQuestion, "Create Daily Report")
If Response = vbNo Then GoTo Bye
```

如果在信息框中选择“否”按钮，VBA 将执行 GoTo Bye 语句，指向在程序末尾的 Bye: 标记。

If 语句嵌套

可使用 If 语句嵌套产生逻辑运算。每个嵌套的 If 语句必须完整。例如，如果在另一个 If 语句块中嵌套 If 语句块，而忘记嵌套语句的 End If 行，VBA 会认为外层的 End If 语句属于嵌套的 If 语句。

要想使 If 语句块容易理解，应该采用不同的缩位。嵌套 If 语句时，这一点尤其重要，这有助于搞清哪一个 If 语句对应相应的 End If 语句。要了解这样的情况，可参照下面的嵌套语句：

1. If condition1 Then	'start of first If
2. If condition2 Then	'start of second If
3. If condition3 Then	'start of third If
4. statements1	
5. ElseIf condition4 Then	'ElseIf for third If
6. statements2	
7. Else	'Else for third If
8. statements3	
9. End If	'End If for third If
10. Else	'Else for second If
11. If condition5 Then	'start of fourth If
12. statements4	
13. End If	'End If for fourth If
14. End If	'End If for second If
15. Else	'Else for first If
16. statements5	
17. End If	'End If for first If

用这样的布局，可以很容易搞清程序的走向。例如，在第一行中 condition1 为假，VBA 将执行第 15 行的 Else 语句，从那里开始执行。假如第 1 行 condition1 为真，VBA 检验第 2 行的 condition2，等等。

缩位只是看起来清楚，VBA 不加区别，然而对于阅读程序的人十分有帮助。前面的嵌套 If 语句给出注释，因此可以很清楚了解，哪一个 Else、ElseIf 以及 End If 属于哪一个 If 语句。因为采取了缩位方式，注释并没有必要。另一方面，看一下下面没有缩位的嵌套语句，这样的方式很难看懂。再和别的语句混在一起就更加难以分辨。

```

1. If condition1 Then           'start of first If
2. If condition2 Then           'start of second If
3. If condition3 Then           'start of third If
4. statements1
5. ElseIf condition4 Then      'ElseIf for third If
6. statements2
7. Else                         'Else for third If
8. statements3
9. End If                        'End If for third If
10. Else                         'Else for second If
11. If condition5 Then          'start of fourth If
12. statements4
13. End If                        'End If for fourth If
14. End If                        'End If for second If
15. Else                         'Else for first If
16. statements5
17. End If                        'End If for first If

```

很少需要嵌套多层的 If 语句，通常的做法是在 If…Then…Else 语句或者在 If…Then…ElseIf…Else 语句中嵌套一个简单的 If…Then 语句，参见程序清单 11.4。

程序清单 11.4

```

1. Selection.HomeKey Unit:=wdStory
2. Selection.Find.ClearFormatting
3. Selection.Find.Style = ActiveDocument.Styles("Heading 5")
4. Selection.Find.Text = ""
5. Selection.Find.Execute
6. If Selection.Find.Found Then
7.     lngResponse = MsgBox("Make this into a special note?", _
                           vbOKCancel, "Make Special Note")
8.     If lngResponse = vbOK Then
9.         Selection.Style = "Special Note"
10.    End If
11. End If

```

程序清单 11.4 在当前文档中查找标题 5 的格式，如果查找成功，显示信息框用于特殊的注释。程序说明如下：

- ◆ 第 1 行在文档开头返回一个插入点。
- ◆ 第 2 行从 Find 命令开始清除格式（以保证不是在搜索不适当的格式）。
- ◆ 第 3 行设定标题 5 为查找命令的格式，第 4 行设定查找的字符串为空。
- ◆ 第 5 行执行查找命令。
- ◆ 第 6 行到第 11 行为外围的 If 语句。第 6 行检验第 5 行的查找命令，看是否完成查找，如果完成，VBA 将执行第 7 行到第 10 行。
- ◆ 第 7 行显示信息框，用来询问用户是否对该段落进行特殊标注。
- ◆ 第 8 行是用嵌套的 If…Then 语句来检验用户对信息框的回答。
- ◆ 如果用户的回答为 vbOK，即用户选择了“确定”按钮，VBA 执行第 9 行的语句，将特殊注释风格用于该段落（假定在该文档或模板中含有该注释）。

◆ 第 10 行为嵌套 If 语句的 End If 语句。第 11 行为外层 If 语句的 End If 语句。

提示：如果希望在一个文档中包括多个标题 5 的风格，应使用 Do While…Loop 循环进行搜索。详情可参见第 12 章的 Do While…Loop 循环。

Select Case 语句

Select Case 语句是多条件 ElseIf 语句的另一种有力的方式，决策功能相同，代码更加紧凑有效。

如果在决策时条件变量或表达式取决于 3 个或 4 个以上的值，就应当使用 Select Case 语句。条件变量或表达式称做测试条件。

Select Case 语句比 If…Then 语句容易读懂，主要是因为代码少。这一点也使得它们容易修改：如果要调整一些值，涉及的代码不多。

语法

Select Case 语句的语法如下：

Select Case 测试表达式

Case 表达式 1

语句 1

[Case 表达式 2

语句 2]

[Case Else

其他语句

End Select

语句展开如下：

◆ 以 Select Case 语句开始，End Select 语句结束。

◆ “测试表达式”用来决定执行哪一个 Case 表达式。

◆ “表达式 1”、“表达式 2”等用来和“测试表达式”相对应。

例如，可以测试用户窗体上使用了哪个按钮。“测试表达式”必须和使用的按钮绑定在一起：如果是第一个按钮，VBA 和“表达式 1”相对应，并执行下面的“语句 1”；如果是第二个按钮，VBA 和“表达式 2”相对应，并执行下面的“语句 2”，以此类推。

Case Else 语句类似于 If 语句的 Else 子句。Case Else 是可选条件（如果使用），在所有给出的条件都不能满足时执行。

举例

Select Case 语句的例子参见程序清单 11.5，该程序要求用户输入打字速度，然后给出回应。

程序清单 11.5

```
1. Sub Check_Typing_Speed()
2.
```

```

3. Dim varTypingSpeed As Variant
4. Dim strMsg As String
5.
6. varTypingSpeed = InputBox _
    ("How many words can you type per minute?", "Typing Speed")
7. Select Case varTypingSpeed
8.     Case ""
9.         End
10.    Case Is < 0, 0, 1 To 50
11.        strMsg = "Please learn to type properly before " & _
    "applying for a job."
12.    Case 50 To 60
13.        strMsg = "Your typing could do with a little brushing up."
14.    Case 60 To 75
15.        strMsg = "We are satisfied with your typing speed."
16.    Case 75 To 99
17.        strMsg = "Your typing is more than adequate."
18.    Case 100 To 200
19.        strMsg = "You wear out keyboards with your blinding speed."
20.    Case Is > 200
21.        strMsg = "I doubt that's true."
22. End Select
23.
24. MsgBox strMsg, vbOKOnly, "Typing Speed"
25.
26. End Sub

```

以下是程序清单 11.5 的解释：

- ◆ 第 1 行开始程序，第 26 行结束程序。
- ◆ 第 2 行为空行。第 3 行声明不定型变量 varTypingSpeed。第 4 行声明字符型变量 strMsg。第 5 行为空行。
- ◆ 第 6 行显示输入框，要求输入打字速度，并存入变量 varTypingSpeed。
- ◆ 第 7 行开始 Select Case 语句，用变量 varTypingSpeed 进行判断。
- ◆ 接着，VBA 轮流对比 Case 条件，直到条件符合。第 8 行的第一个条件语句将变量 varTypingSpeed 和空字符（""）相比较，了解用户是否使用了输入框上的“取消”按钮，或者未输入任何值就单击“确定”按钮。如果条件符合，VBA 执行第 9 行的 End 语句，结束程序。
- ◆ 如果条件不符合，VBA 转向第 10 行的 Case 语句，将变量 varTypingSpeed 和 3 个项目比较：小于 0、等于 0 以及每分钟 1 到 50。注意 3 点：
 1. 可包括多种条件，用逗号隔开。
 2. 使用 Is 关键词和比较运算符（本例为小于）比较两个关联的数字。
 3. To 关键词用来针对一个范围。
- ◆ 如果变量 varTypingSpeed 和第 10 行的条件相等，VBA 将字符型变量 strMsg 赋值为第 11 行的文本信息，然后执行 End Select 后面的语句。
- ◆ 如果变量 varTypingSpeed 不在此范围，VBA 将转向另一个 Case 条件，依次对比。如果发现一个条件相符就执行下面的语句，将字符型变量 strMsg 赋值为文本信息，然后执行 End Select 后面的语句。

◆除了第 8 行（结束程序）的情况外，第 24 行将显示 strMsg 语句中保存的信息。

Select Case 语句用列表框或组合框中的项目做条件十分方便，特别是当列表框或组合框中的项目很多时。例如，可以检验列表框或组合框中的值，然后采取相应的步骤。

息自动输入中选择“显示为文本”，代码部分（右侧窗格）行 8 第 1 列◆
合里如进类例当是限制，更广泛十扑杀端日页中结合且走重卖机用时否 Select Case 是时中列

第 12 章 使用循环重复执行

- ◆ 何时使用循环
- ◆ 用 For 循环于固定的循环次数
- ◆ 用 Do 循环于可变的循环次数
- ◆ 循环的嵌套
- ◆ 避免死循环

VBA 如同在生活中一样，时不时地需要重复某个动作以达到某种效果。有时候重复的次数是事先给定的：打 6 个鸡蛋做煎蛋，或者根据一个模板生成 6 个新文档。更多的时候，重复的次数由某个条件设定：每星期买两张彩票直到中 2000 美元，或者根据 Excel 工作表上某个值出现的次数每次执行一个动作。这时候，无法判断什么时候才能中彩，也不知道工作表上该值会出现几次——只要条件满足就执行。

在 VBA 中使用循环重复执行。用循环可以把一条简单的录制宏针对某个操作的对象反复执行。VBA 提供若干表达式以生成循环。本章将介绍几种不同的循环以及每一种循环的使用方法。

何时使用循环

要重复一条命令或一组命令，可以先用宏录制器（如果软件支持宏录制器）把指令录制下来或者编写一段相关的程序，然后用“复制”和“粘贴”命令重复若干次。例如，可以录制一条宏以便根据默认的模板生成新的演示文稿，然后打开 Visual Basic 编辑器，将语句复制 5 次，从而形成一段程序，可以生成 6 个新的演示文稿。不用考虑，用循环结构重复执行要好得多。

循环比单一的重复有以下优势：

- ◆ 程序较短，代码或指令较少，容易维护。
- ◆ 程序更加灵活，不需要每一次写出同样的指令，可以控制重复的次数。
- ◆ 程序容易测试和修改，特别是由别人做这项工作时。

可以这么理解，如果在一个程序里有一两条指令重复两到三次，并且程序也需要将指令重复相同的次数，那么把指令一条条写出来是没有问题的，使用也会很正常。这样做很容易，也不需要花时间考虑循环语句。程序可能很长，不容易维护，但只要工作正常就不是什么大事。

循环的基本知识

在 VBA 中，循环是一种结构，用来重复一些指令，这些指令每执行一次就返回到结构的开头。循环的每一次执行叫做重复。

循环有两种基本模式：

- ◆ 固定次数的循环重复次数固定。
- ◆ 可变次数的循环重复次数可变。

无论哪一种循环都被循环因素控制，可以是数字表达式或者逻辑表达式。固定次数的循环一般使用数字表达式，而可变次数的循环则使用逻辑表达式。例如，固定次数的循环可指定循环 5 次，可变次数的循环一直到某种状态结束才终止。

表 12.1 介绍 VBA 的循环类型。

表 12.1 VBA 的循环类型

循环	类型	说明
For… Next	固定	按给定的次数重复执行一条或一组命令
For Each… Next	固定	在 VBA 的集合中针对每个对象重复执行一条或一组命令
Do While…Loop	可变	在条件为真时重复执行一条或一组命令，直到条件为假
While…Wend	可变	在条件为真时重复执行一条或一组命令，直到条件为假
Do Until…Loop	可变	同 Do While…Loop 但几乎不使用
Do…Loop While	可变	在条件为假时重复执行一条或一组命令，直到条件为真
Do…Loop Until	可变	重复执行一条或一组命令，直到条件由真变为假

用 For 循环于固定的循环次数

For 循环可用于固定的循环次数。For…Next 循环按选择的次数重复，而 For Each…Next 循环按集合中的对象重复。

For…Next 循环

For…Next 循环按给定的次数重复，次数由计数器变量设定。计数器变量可以直接写入程序，由输入框或对话框给出，或者由程序本身的其他部分或者另一个程序产生。

语法

For…Next 循环的语法如下：

For counter = start To end [Step stepsize]

[语句]

[Exit For]

[语句]

Next [counter]

表 12.2 介绍上述语法的各部分。同样，方括号为可选项目，斜体是占位符。

表 12.2 For…Next 循环的语法组成部分

部分	说明
counter (计数)	数字变量或者产生数字的表达式。默认情况下，VBA 每次重复增加 1，但可以改变，即使用可选关键词 Step 和 stepsize 参数。counter 在 For 语句中是必选的，在 Next 语句中是可选的。但最好在 Next 语句中使用以使代码容易读。在程序中多次使 For…Next 语句或 For…Next 语句嵌套是非常重要的

(续表)

部分	说明
start (开始值)	数字变量或数字表达式, counter 的开始值
end (结束值)	数字变量或数字表达式, counter 的终止值
stepsize (步长)	数字变量或数字表达式, 指明 counter 的增减量。必须和 Step 关键词一起使用, 默认为 1, 但可以为正数, 也可以为负数
Exit For	退出 For 循环的语句
Next	循环的结束语句。重申一次, 在 Next 语句中使用 counter, 代码容易读

For…Next 循环的工作方式如下:

1. VBA 进入 For 语句时, 将 start 值赋值于 counter, 然后执行循环中的语句。执行到 Next 语句时 counter 增加 1, 或者按指定的步长, 然后又回到 For 语句。
2. VBA 将 counter 变量和 end 变量进行比较, 如果步长为正值, 如果 counter 大于 end, VBA 结束循环, 继续执行 Next 语句后面的指令 (可以是任何指令, 或者结束程序)。如果 counter 小于或等于 end, VBA 重复循环中的语句。使 counter 增加 1 或者按步长增加。然后再回到 For 语句 (如果步长为负值, 当 counter 大于或等于 end 时, 循环继续执行; 当 counter 小于或等于 end 时, 循环终止)。
3. Exit For 语句退出 For 循环, 后面将介绍如何使用 Exit For 语句, 并给出不同用法的例子。

For…Next 循环的直接运用

直接运用 For…Next 循环, 需先指定 counter 变量、开始值和终止值:

```
Dim i As Integer
```

```
For i = 1 to 200
```

在这里, i 是计数器变量, 1 是开始值, 200 是终止值。默认情况下, VBA 每次循环使计数器变量增加 1。本例计数器应该为 1、2、3、4 直到 200, 到 201 (或更大的值, 在本例中不可能大于 201, 因为步长是 1) 时结束循环。可使用 step 关键词决定不同的步长, 该步长可以为正值也可以为负值, 后面将详细说明。

注意: i 是整型计数器变量, 用于 For…Next 循环, 除了 i 外一般也使用 j、k、l、m 和 n 作为计数器变量。单字母的名称来自使用卡片的时代, 名称太长, 使用不方便。现在 VBA 使用长名称也很简单。如果希望程序紧凑, 应该使用 i 或者相应的字母。否则, 如果希望代码更容易看懂, 也可使用有含义的变量名称, 例如 LoopCounter 或者 intLoopCounter。

以上两条语句后面应当给出要执行的命令, 然后用 Next 关键词结束循环。

```
Application.StatusBar =_
"Please wait while Excel checks for nonuniform prices: " & i & "..."
```

该代码生成一个状态栏, 用于检验工作表中不合理的值。

检查收到的 Word 文档中每一个段落, 确认没有不合适的格式。用循环对当前文档中的

每一个段落 (Active Document 对象中的 Paragraphs 集合设定为 Counter 属性) 依次检查，在显示状态栏时提供引用点：

```
Dim i As Integer
For i = 1 To ActiveDocument.Paragraphs.Count
    CheckParagraphForIllegalFormatting
    Application.StatusBar =
        "Please wait while Word checks the formatting in " & i & " out of "
        & ActiveDocument.Paragraphs.Count & "..."
    Selection.MoveDown Unit:=wdParagraph, _
        Count:=1, Extend:=wdMove
Next i
```

这段代码应当在文档的起始位置开始，对当前段落执行名为 CheckParagraphForIllegalFormatting 的程序，显示状态栏的信息，指明哪一个段落数超出总段落数，然后到下一个段落。当 VBA 执行到 Next 语句时，i 增加 1（因为变量 stepsize 未指定），然后回到 For 语句，将 i 的值与 ActiveDocument. Paragraphs. Count 的值进行比较。程序继续循环，直到 i 达到 ActiveDocument. Paragraph. Count 的值，即最后一次的循环。

同样，可使用简单的 For…Next 循环在 Excel 中建立时间表。下面的语句使用 For…Next 循环在当前工作簿的工作表的当前列中，输入 1 到 24 小时。

```
Dim i As Integer
For i = 1 To 24
    ActiveCell.FormulaR1C1 = i & ":00"
    ActiveCell.Offset(RowOffset:=1, ColumnOffset:=0).Select
Next i
```

在这里，ActiveCell. FormulaR1C1 语句插入自动增加的字符串，计数器 i、冒号以及两个零（时间格式）。ActiveCell. Offset (RowOffset: = 1, ColumnOffset: = 0). Select 语句选择同一列下一行的单元格。循环从 1 到 24，当 i 为 25 时停止。

有步长的 For…Next 循环

如果 counter 变量增加默认值 1 不符合要求，可使用 Step 关键词指定一个不同的增加量或者减少量。例如下面的语句将 counter 变量增加 20，因此，序列为 0、20、40、60、80、100：

```
For i = 0 to 100 Step 20
```

也可使用负值，指定减少量：

```
For i = 1000 to 0 Step -100
```

这样本语句产生的序列为 1000、900、800 等，直到 0。为了避免出现前面的超出范围的情况，可以反向执行，写成 ActiveDocument. Paragraphs. Count 到 0：

```
Dim i As Integer
For i = ActiveDocument.Paragraphs.Count To 0 Step -1
    CheckParagraphForIllegalFormatting
    Application.StatusBar =
        "Please wait while Word checks the formatting in this document: " & i
    Selection.MoveDown Unit:=wdParagraph, Count:=1, Extend:=wdMove
Next i
```

使用输入框驱动 For…Next 循环

有时候循环的次数必须直接给出，而有时候循环次数需根据另一种处理得到，例如前例中的 ActiveDocument.Paragraphs.Count 属性。然而，常常需要输入值驱动循环。最简单的办法是在输入框中输入值。

例如，程序清单 12.1 包含一个名为 CreatePresentations 的程序，显示输入框，要求生成演示文稿的数量。然后用 For…Next 循环在 PowerPoint 中生成文档。

程序清单 12.1

```

1. Sub CreatePresentations()
2.     Dim intPresentations As Integer
3.     Dim i As Integer
4.     intPresentations = InputBox_(
5.         "Enter the number of presentations to create:", -
6.         "Create Presentations")
7.     For i = 1 To intPresentations
8.         Presentations.Add
9.     Next i
10.    End Sub

```

以下是程序清单 12.1 的说明：

- ◆ 第 2 行声明整型变量 intPresentations。第 3 行声明整型变量 i。
- ◆ 第 4 行生成信息框要求给出演示文稿的数量。
- ◆ 第 5 行到第 7 行为 For…Next 循环，从 i = 1 执行到 i = intPresentations，采用了默认步长。每次循环，都将执行第 6 行的语句以便根据默认的模板生成一个新的演示文稿。

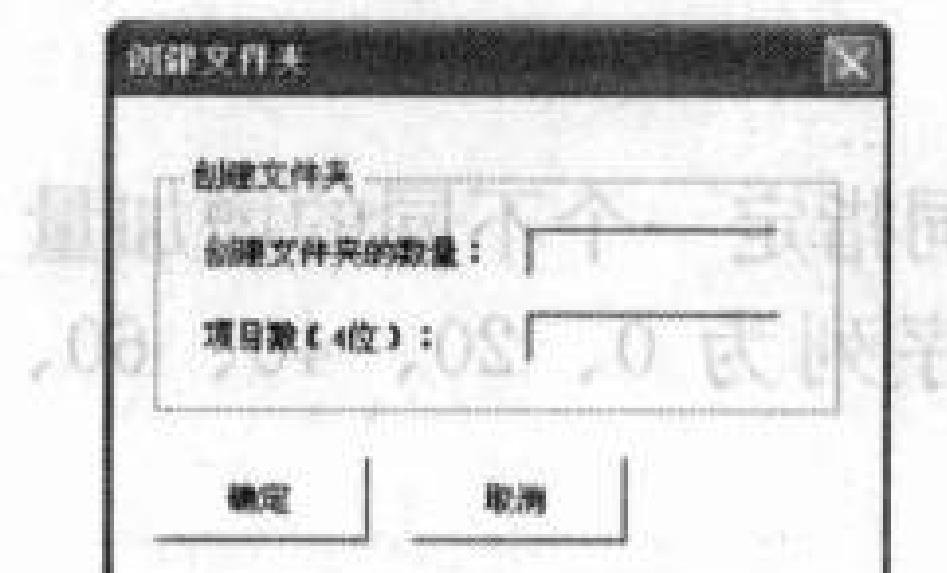


图 12.1 如果需要的信息比输入框提供的信息多，应当使用自定义的对话框

用对话框驱动 For…Next 循环

如果输入框不能满足要求，可以方便地使用对话框驱动 For…Next 循环。本书到目前为止还没有介绍如何生成对话框，但本章在 Create_Folders 程序中给出简要说明，该程序使用给定的名称，如工程的不同部分生成多个目录。

例如可使用四位数以识别工程，s 表示 section，两位数表示部分。于是目录的名称为 1234s01、1234s02、1234s03，等等。手工生成很简单，然而如果数量太大未免也很乏味。

这是一个简单的窗体对话框，要求输入创建文件夹的数量以及项目数。图 12.1 给出该对话框的式样。

使用 Show 方法显示对话框时，也许首先要用 Load 语句：

```
Load frmCreateFolders
frmCreateFolders.Show
```

本例的对话框叫做 frmCreateFolders，其实任何合法的名称都可以。第一个文本框给出

创建文件夹的数量，其名称为 txtFolders，第二个文本框的名称是 txtProjectNumber。

“取消”按钮的相关语句是 End，用于 Click 事件，因此当单击它时 VBA 结束程序。

```
Private Sub cmdCancel_Click()
    End
End Sub
```

“确定”按钮的代码在程序清单 12.2 中给出，和 Click 事件相关。

程序清单 12.2

```
1. Private Sub cmdOK_Click()
2.
3.     Dim strMsg As String
4.     Dim strFolder As String
5.     Dim i As Integer
6.
7.     frmCreateFolders.Hide
8.     Unload frmCreateFolders
9.     strMsg = "The Create_Folders procedure has created "
    & "the following folders: " & vbCrLf & vbCrLf
10.
11.    For i = 1 To txtFolders.Value
12.        strFolder = txtProjectNumber.Value & "p" & Format(i, "0#")
13.        MkDir strFolder
14.        strMsg = strMsg & "    " & strFolder & vbCrLf
15.    Next i
16.
17.    MsgBox strMsg, vbOKOnly + vbInformation, _
        "Create Folders"
18.
19. End Sub
```

程序清单 12.2 中的 cmdOK_Click 程序在单击对话框中的“确定”按钮时执行。

- ◆ 第 1 行声明 cmdOK_Click 子程序。第 19 行结束程序。第 2 行为空行。
- ◆ 第 3 行声明字符型变量 strMsg，用来保存字符串在程序末尾的信息框中显示的结果。
- ◆ 第 4 行声明字符型变量 strFolder，用来保存每次循环时创建的当前文件夹的名称。
- ◆ 第 5 行声明整型变量 i，它为 For…Next 循环的计数器变量。
- ◆ 第 6 行为空行。
- ◆ 第 7 行隐藏窗体。
- ◆ 第 8 行卸载窗体。
- ◆ 第 9 行为 strMsg 赋值，以冒号和两个 vbCrLf 回车符结束。
- ◆ 第 10 行为空行。
- ◆ 第 11 行到第 15 行为 For…Next 循环语句，用来创建文件夹。第 11 行用来指定循环从 1 到 txtFolders.Value，该值由用户在窗体中输入。第 12 行给出字符型变量 strFolder，由 txtProjectNumber 的属性值、字母 p 以及 i 的值组合而成，格式为个位数前加 0（1 写成 01，等等）。第 13 行使用 MkDir strFolder 命令创建一个文件夹。第 14 行给 strMsg 增加一些空格（用来表示退格）、strFolder 的内容以及 vbCrLf 字符。第 15 行返回 For 语句并提供一个增加值。然后 VBA 比较计数器 i 和 txtFolders.Value 的

值，按需要再次循环。

注意：本程序在当前的文件夹中产生新的文件夹，未提供选择。实际中最好不要这么做。最好将文件夹放到一个明确的地点（可以把所有的工程文件放在一起）；但是一般情况下，希望用户选择某个地点，例如显示普通的对话框，如大多数软件提供的“另存为”对话框。

For Each…Next 循环

For Each…Next 循环十分独特，基本方式和 For …Next 相同，即循环次数是已知的。在这种情况下，循环次数是集合中对象的数量，例如演示文稿中 Slides 集合或者 Word 文档中的 Documents 集合。例如，可选择对每一个 Slide 对象执行一条命令，而不需要知道集合中有多少张幻灯片，至少应该有一张（如果没有什么也不会发生）。

语法

For Each…Next 语句的语法如下：

For Each 对象 In 集合

[语句]

[Exit For]

[语句]

Next [对象]

VBA 首先检测指定集合中的对象数量，然后对第一个对象执行语句。到 Next 关键词时回到 For Each 语句，再次检测对象的数量，然后执行相应的循环。

例如，Documents 集合包括 Word 中所有打开的文档。因此，可以使用 For Each…Next 循环生成一个简单的程序来关闭所有打开的文档：

```
Dim Doc As Document
For Each Doc in Documents
    Doc.Close SaveChanges:=wdSaveChanges
Next
```

VBA 使用 Close 方法依次关闭每一个打开的文档。该语句使用了 wdSaveChanges 常量指明在文档关闭时保存所有未保存的改变。只要在 Documents 集合中有打开的文档，VBA 将重复执行，直到关闭所有的文档然后退出程序。

提示：本例只是为了说明 For Each…Next 语句的用法，最好不要实际使用。也许在 Documents 集合中使用 Close 方法关闭所有文档比较简单。然而可以用 For Each…Next 循环在文档关闭之前检查每一个文档的某些特征。

使用 Exit For 语句

在本章前面介绍 For 语句的语法时，指出如果满足某些条件，可使用 Exit For 语句退出循环，Exit For 语句是可选的，往往并不需要。如果发现程序中需要使用 Exit For 语句，可能是构造循环时有错误，就是说 Exit For 语句有用处，例如在程序出现错误时，或者程序取消时。

在这样的情况下，的确需要 Exit For 语句退出循环。一般来说，条件十分直接，例如，如果在 Word 中需要关闭打开的窗体直到留下某一个文档，就可以使用 Exit For 语句：

```

Dim Doc As Document
For Each Doc In Documents
    If Doc.Name = "Document1" Then Exit For
    Doc.Close
Next Doc

```

以上语句检查文档的 Name 属性，如果名称为 Document1，Exit For 语句退出循环。否则 VBA 关闭文档，返回循环起始点。

注意：如果需要，可使用多个 Exit For 语句。例如，在循环时需要检查两个或两个以上的条件。

用 Do 循环于可变的循环次数

Do 循环比 For 循环更灵活，可用于测试条件，指引程序的走向。VBA 包括几种 Do 循环：

- ◆ Do While … Loop
- ◆ Do … Loop While
- ◆ Do Until … Loop
- ◆ Do … Loop Until

Do 循环可分成两种类型：

- ◆ 在执行前测试条件。Do While … Loop 和 Do Until… Loop 属于这种情况。
- ◆ 在执行后测试条件。Do Loop … While 和 Do Loop … Until。

这两种循环的区别在于每个 While 循环时条件必须为真（直到条件为假），而每个 Until 循环重复执行，直到条件为真（当条件为假时）。这就意味着，从某种程度上说，可以只使用 While 循环或者 Until 循环——只需要建立某些条件。例如，可使用 Do While 循环条件为 $x < 100$ 或者 Do Until 循环条件为 $x = 100$ ，效果是一样的。

本讨论假定需要掌握所有的循环，在需要的时候可以任意使用。

Do While … Loop

在 Do While … Loop 循环中，条件必须为真，命令才可执行。如果条件不为真，命令就不执行，而且循环结束。例如，可以在一个文档中查找特定的单词或词组，每发现一次就执行一个动作，图 12.2 为 Do While … Loop 循环。

语法

Do While … Loop 循环的语法很直接，具体如下：

Do While 条件

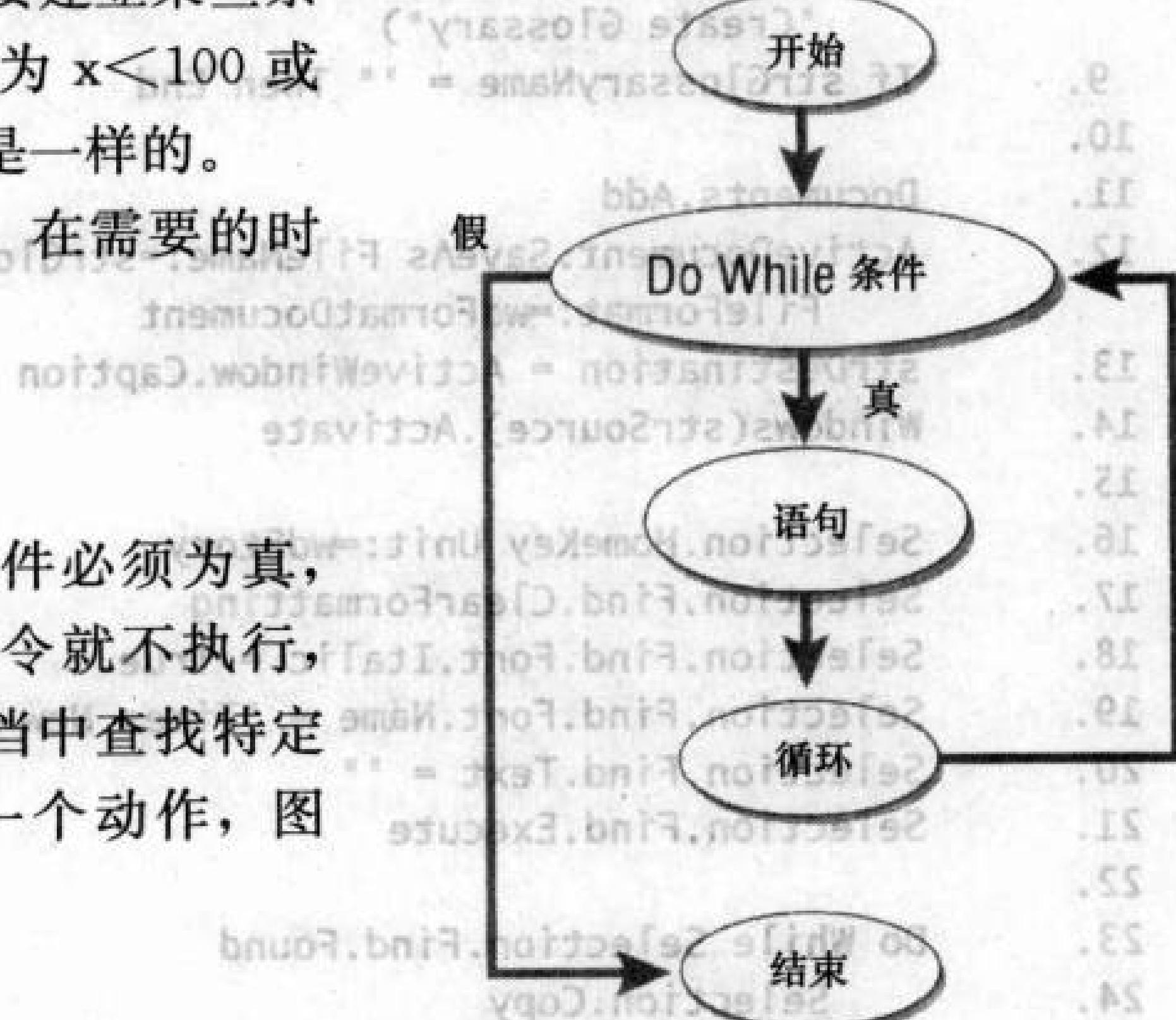


图 12.2：Do While … Loop 循环在执行命令前先测试条件，然后执行循环中的命令

[语句]

[Exit Do]

[语句]

Loop

如果“条件”满足，循环中的语句就执行。Loop 关键词将执行返回到 Do While，然后再进行检测，如果条件为真，循环继续，如果条件为假，将执行 Loop 关键词以后的命令。如果有必要，可以使用一处或多处 Exit Do 语句将循环分段。

例如，希望从较长的 Word 文档中构建一个词汇表，该文档使用斜体解释文章中的术语，并列出段落（两者都使用了 Times New Roman 字体），这时不希望考虑用于其他场合的斜体字（例如标题或者图题）。可命令 Word 在 Times New Roman 字体的文章中查找斜体属性。如果 Word 发现结果，将采取相应的动作，例如把包括术语的句子找出来，和以下的句子一起（段落中的其他内容）复制到另一个文档的末尾。然后将继续查找直到不再有斜体的 Times New Roman 字体为止。

程序清单 12.3 将介绍如何使用 Do While…Loop 循环构建这样的过程。该清单包括一些尚未使用过的命令，但循环的工作方法并不难理解。

程序清单 12.3

```

1. Sub GenerateGlossary()
2.
3.     Dim strSource As String
4.     Dim strDestination As String
5.     Dim strGlossaryName As String
6.
7.     strSource = ActiveWindow.Caption
8.     strGlossaryName = InputBox _
        ("Enter the name for the glossary document.", _
        "Create Glossary")
9.     If strGlossaryName = "" Then End
10.
11.    Documents.Add
12.    ActiveDocument.SaveAs FileName:=strGlossaryName, _
        FileFormat:=wdFormatDocument
13.    strDestination = ActiveWindow.Caption
14.    Windows(strSource).Activate
15.
16.    Selection.HomeKey Unit:=wdStory
17.    Selection.Find.ClearFormatting
18.    Selection.Find.Font.Italic = True
19.    Selection.Find.Font.Name = "Times New Roman"
20.    Selection.Find.Text = ""
21.    Selection.Find.Execute
22.
23.    Do While Selection.Find.Found
24.        Selection.Copy
25.        Selection.MoveRight Unit:=wdCharacter, _
        Count:=1, Extend:=wdMove
26.        Windows(strDestination).Activate
27.        Selection.EndKey Unit:=wdStory

```

```
28. Selection.Paste
29. Selection.TypeParagraph
30. Windows(strSource).Activate
31. Selection.Find.Execute
32. Loop
33.
34. Windows(strDestination).Activate
35. ActiveDocument.Save
36. ActiveDocument.Close
37.
38. End Sub
```

程序清单 12.3 中的 GenerateGlossary 程序在当前的文档中根据 Times New Roman 字体查找斜体字，并且将它们保存在生成的新文档中。执行情况如下：

- ◆ 第 1 行开始程序。第 2 行为空行。
- ◆ 第 3、4 和 5 行声明字符型变量 strSource、strDestination 以及 strGlossaryName。第 6 行为空行。
- ◆ 第 7 行将字符型变量 strSource 赋值为当前窗口的 Caption 属性。程序使用该变量激活需要操作的文档。
- ◆ 第 8 行显示信息框，要求输入文档的名称，用来保存从当前文档中取出的词汇。输入的字符串保存在字符型变量 strGlossaryName 中。
- ◆ 第 9 行将 strGlossaryName 和空字符串进行比较，以保证未使用“取消”按钮取消程序，或者未输入名称就单击了“确定”按钮。如果 strGlossaryName 是空字符串，第 9 行用 End 语句终止程序执行。
- ◆ 如果第 9 行未终止程序，程序继续向前。第 10 行为空行。第 11 行生成一个空文档（该文档根据 Normal.dot 通用模板生成，因为模板参数未特别指出）。该文档将作为词汇表文档。
- ◆ 第 12 行根据输入框中输入的名称保存文档。
- ◆ 第 13 行将该文档的 Caption 属性保存在 strDestination 变量中，用来在整个程序中需要的地方激活该文档。现在已经有原文档 strSource 变量和目标文档 strDestination 变量。
- ◆ 第 14 行使用 Activate 方法激活 strSource 窗口。第 15 行为空行。
- ◆ 第 16 行使用 Selection 对象的 HomeKey 方法的 wdStory 单元，将插入点移到文档的开头，程序从这里开始查找所有的 Times New Roman 字体的斜体字。
- ◆ 第 17 行到第 20 行为程序中的查询处理：第 17 行将当前发现的内容格式去除，第 18 行用来查找斜体格式，第 19 行定义查找的字体为 Times New Roman，第 20 行定义查找字符串，此时为空字符串，用来查找指定的格式。
- ◆ 第 21 行用 Execute 方法执行查找命令。第 22 行为空行。
- ◆ 第 23 行到第 32 行执行 Do While … Loop 循环。第 23 行给出条件，即找出前面定义的 Times New Roman 的斜体字。如果该条件满足，循环中的命令将执行。
- ◆ 第 24 行复制选择（查找到的结果）。
- ◆ 第 25 行将插入点右移一个字符，用来有效地取消选择以便用来准备下一个查询。必须使插入点离开选择，这样下一个查找才不会找出同样的结果。（如果程序查找整个

文档, 而不是向下查找, 就需要将插入点向左离开选择, 使用 Selection. MoveLeft 语句。)

- ◆ 第 26 行激活 strDestination 窗体, 使 Word 获得焦点。
- ◆ 第 27 行将插入点移到词汇文档的末尾, 第 28 行在插入点的位置粘贴复制的内容。移到文档的末尾, 并非完全必要, 如果模板未含有任何内容, 即 Normal. dot 是空的, 那么在第 11 行生成的新文档也应当是空的, 文档的开头与末尾应当在同样的位置。每一次粘贴之后, Word 停在粘贴内容的末尾。然而如果 Normal. dot 模板有内容, 这一步是必需的。
- ◆ 第 29 行使用 TypeParagraph 方法在每次插入的粘贴后进入一个段落。
- ◆ 第 30 行再次激活 strSource 文档, 第 31 行重复查找处理。
- ◆ 第 32 行的 Loop 语句使循环回到第 23 行, Do While 对条件进行再次认定, 比较最后查找的结果是否满足条件。
- ◆ 如果满足条件, 循环继续执行。如果不满足, 将执行第 34 行的语句, 再次激活词汇文档。第 35 行保存当前的文档(即词汇文档, 因为刚刚激活), 第 36 行关闭文档。
- ◆ 第 37 行是空行。第 38 行结束程序。

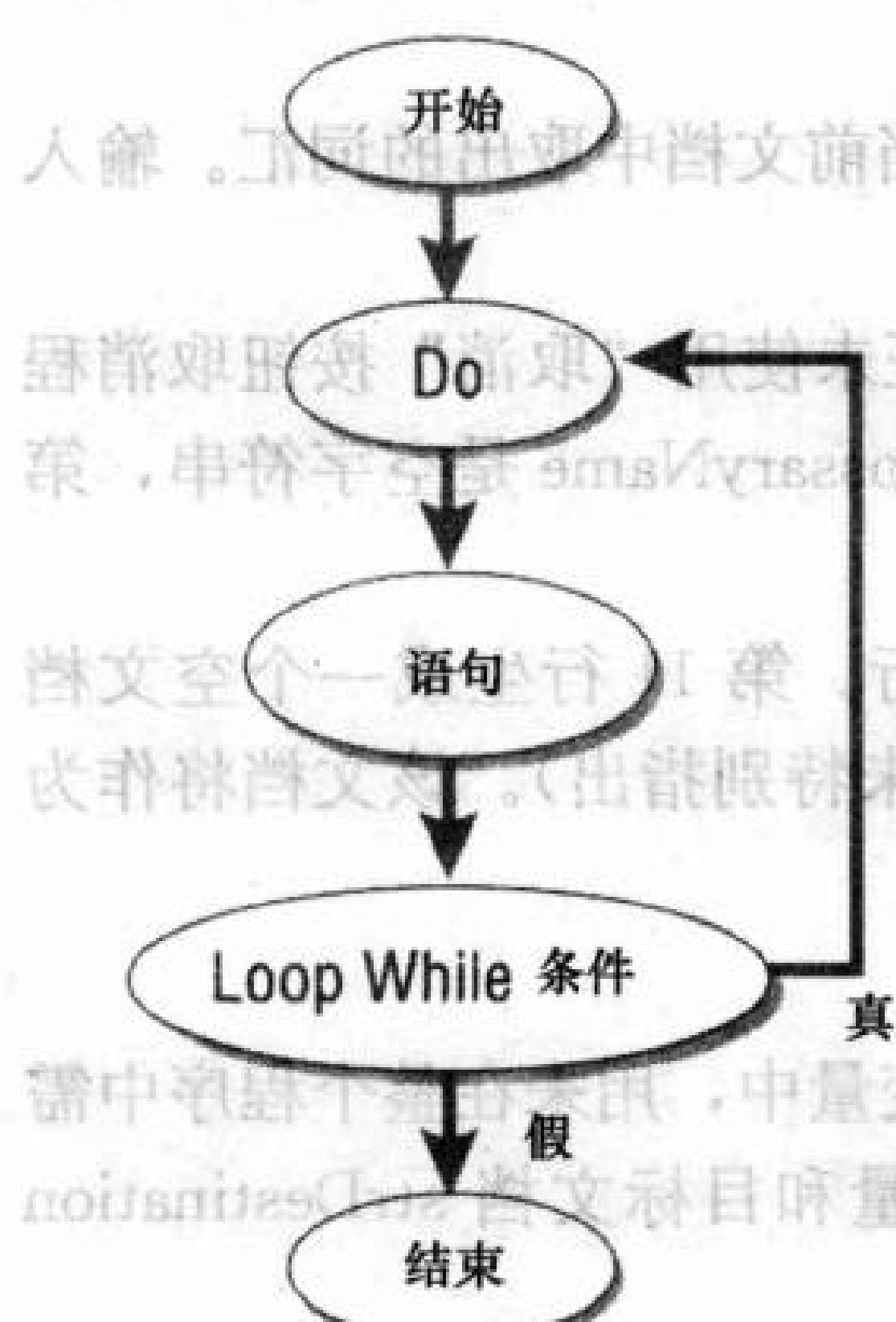


图 12.3 在 Do ... Loop While 循环中, 在条件测试之前先执行一次语句

Do ... Loop While 循环

Do ... Loop While 循环类似于 Do ... While Loop 循环, 区别在于 Do ... Loop While 循环中的命令至少执行一次不论条件是否满足, 如果条件为真, 循环继续直到条件为假。图 12.3 为 Do ... Loop While 循环的执行情况。

如果 Do ... While Loop 循环已经掌握, Do ... Loop While 循环似乎有些奇怪, 难道检验条件之前就执行命令? 然而 Do ... Loop While 循环十分有用, 但它与 Do ... While Loop 循环的条件的使用场合不同。

考虑一下本章开始的彩票例子。在这种情况下, 检验条件之前就需要执行。首先购买彩票, 然后检测是否中彩。如果未中彩或者中彩的数目不大, 就返回去在下次继续购买。(实际上, 逻辑上这是 Do ... Loop Until 循环而不是 Do ... Loop While 循环, 因为条件为假时才继续循环。如果中彩的数目较大, 就满足了条件。) 同样地, 在程序中, 希望执行一个命令并检查是否需要重复。例如, 要对一个段落进行格式处理, 然后检查其他段落是否需要同样的处理。

语法

Do ... Loop While 的语法如下:

Do

[语句]

[Exit Do]

(果语句则省略) 然后是 Do ... Loop While 语句。

语句部分可以是一个或多个语句, 语句之间用分号隔开。

[语句]

Loop While 条件

VBA 执行循环中的语句，然后检测条件。如果为真，VBA 返回到 Do，再次执行。如果为假，将执行 Loop While 后面的语句。

举一个 Do … Loop While 的例子。检验输入的密码，以防止不能提供正确密码的使用程序：

```
Dim varPassword As Variant
Do
    varPassword = InputBox _
        ("Enter the password to start the procedure:", _
        "Check Password 1.0")
Loop While varPassword <> "CorrectPassword"
```

在这里，Do … Loop While 循环给出一个信息框要求输入密码。Loop While 将信息框的值存在 varPassword 变量中以便和正确的密码进行比较。如果两者不相同，循环继续，再次显示输入框。

该循环只是一个例子，不可在实际情况中使用。原因是选择了信息框的“取消”按钮将导致返回一个空字符串，同样不能符合正确的字符串，就会造成再次循环。安全没有问题，但终止循环的唯一方法是输入正确的密码。如果这一点做不到，将始终出现输入框。如果需要构建一个检查密码的程序，就必须考虑一些可能出现的不正确的密码（也许需要 3 种情况）。程序可以自行终止，或者可使用 End 语句终止程序，或者在输入了空字符串时使用 End 语句结束程序。

```
Do
    varPassword = InputBox _
        ("Enter the password to start the procedure:", _
        "Check Password 1.0")
    If varPassword = "" Then End
Loop While varPassword <> "CorrectPassword"
```

Do Until … Loop 循环

Do Until … Loop 循环类似于 Do … While Loop 循环，除了 Do Until … Loop 循环在条件为假时执行，直到条件为真。图 12.4 给出 Do Until … Loop 循环的示意图。

注意：Do Until … Loop 循环十分有用，在条件为假时执行，一直到条件为真。否则，可使用 Do While… Loop 循环将有关的条件反过来使用获得同样的效果。

语法

Do Until … Loop 循环的语法如下：

Do Until 条件

[语句]

[Exit Do]

[语句]

Loop

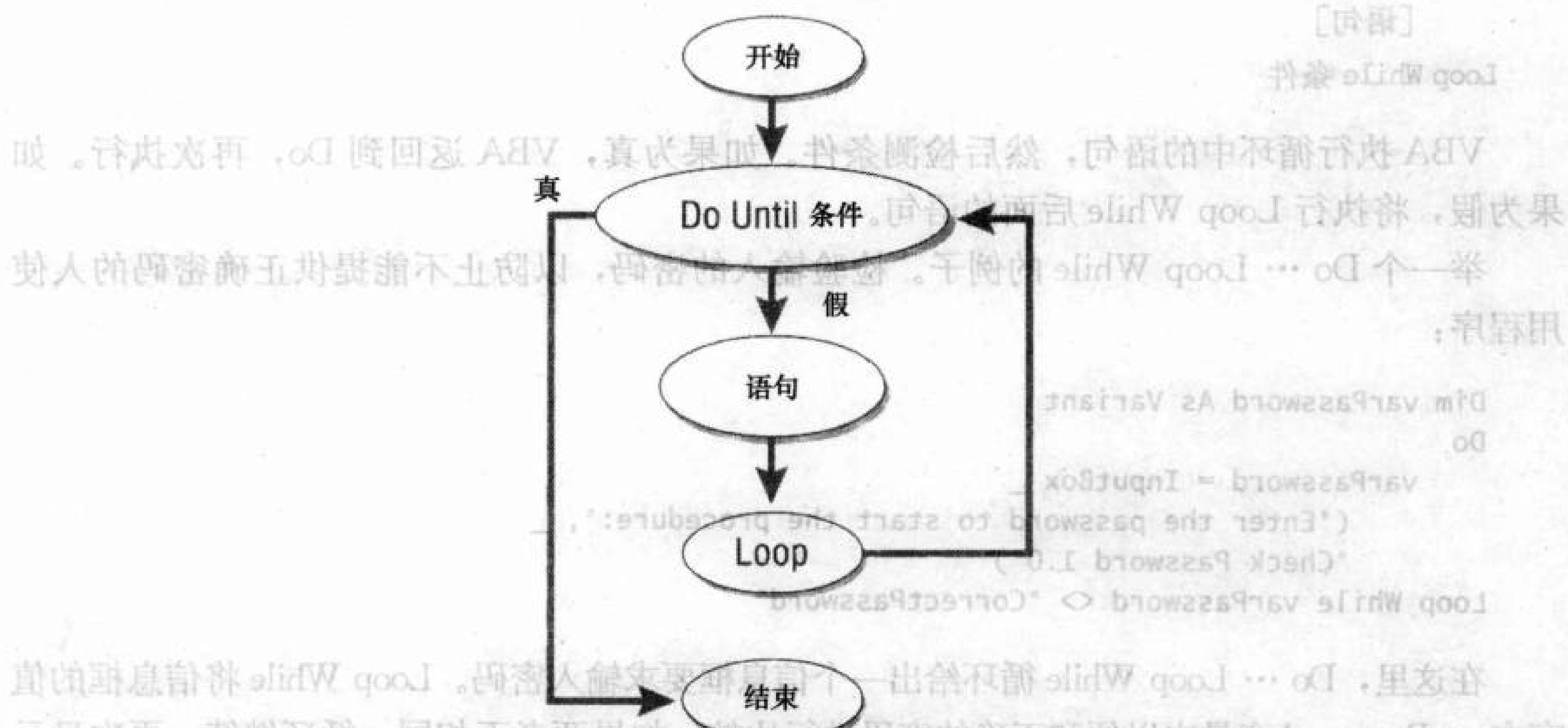


图 12.4 Do Until … Loop 循环在条件为假时执行，直到条件为真

当 VBA 进入循环时，检验条件。如果条件为假，VBA 执行循环中的语句，遭遇 Loop 关键词，回到循环的起始，再根据条件检验。如果条件为真，终止循环，并执行循环后面的语句。例如，考虑一下程序清单 12.4 的彩票案例。

程序清单 12.4

```

1. Sub Lottery_1()
2.     Dim sngWin As Single
3.     Do Until sngWin > 2000
4.         sngWin = Rnd * 2100
5.         MsgBox sngWin, , "Lottery"
6.     Loop
7. End Sub
  
```

程序清单 12.4 的说明如下：

- ◆ 第 2 行声明单精度型变量 sng Win，第 3 行开始 Do Until 循环，其条件为 sngWin>2000，该变量的值必须大于 2000 才可终止循环，否则循环将始终执行。
- ◆ 第 4 行将 sngWin 赋值为 2100 乘以 Rnd 函数产生的随机数的结果，该随机数为 0 到 1 之间的数。（这就意味着，循环需要得到一个随机数，略大于 0.95 才可退出循环，略小于 1/20 的概率或者比所有的彩票几率要高。）
- ◆ 第 5 行显示简单的信息框，其中含有变量 Win 的当前值，从中可以了解它是否为幸运值。
- ◆ 第 6 行含有 Loop 关键词，为循环的结束语句。
- ◆ 第 7 行终止循环。

程序清单 12.5 给出了一个有用的例子，其中在 Word 中使用了 Do Until … Loop 循环。

程序清单 12.5

```

1. Sub FindNextHeading()
2.     Do Until Left(Selection.Paragraphs(1).Style, 7) = "Heading"
3.         Selection.MoveDown Unit:=wdParagraph, _
        Count:=1, Extend:=wdMove
4.     Loop
5. End Sub

```

程序清单 12.5 是一个很短的过程，将插入点移到当前文档的另一个标题处。工作方式如下：

- ◆ 第 2 行开始 Do Until … Loop 循环，第 4 行以 Loop 关键词结束循环。循环的条件是，最左边 7 个字符的名称必须为 Heading。这符合任何标题式样（内置的标题从 1 到 9，或者用户自定义的名称，以 Heading 开始）。
- ◆ 直到条件满足，VBA 将执行第 3 行的语句，使选择内容下移一个段落。

Do … Loop Until 循环

Do … Loop Until 循环类似于 Do Until … Loop 循环，只是 Do … Loop Until 循环中的命令至少执行一次，而不论条件为真还是为假。如果条件为假，循环将继续，直到条件为真。图 12.5 给出 Do … Loop Until 循环的示意图。

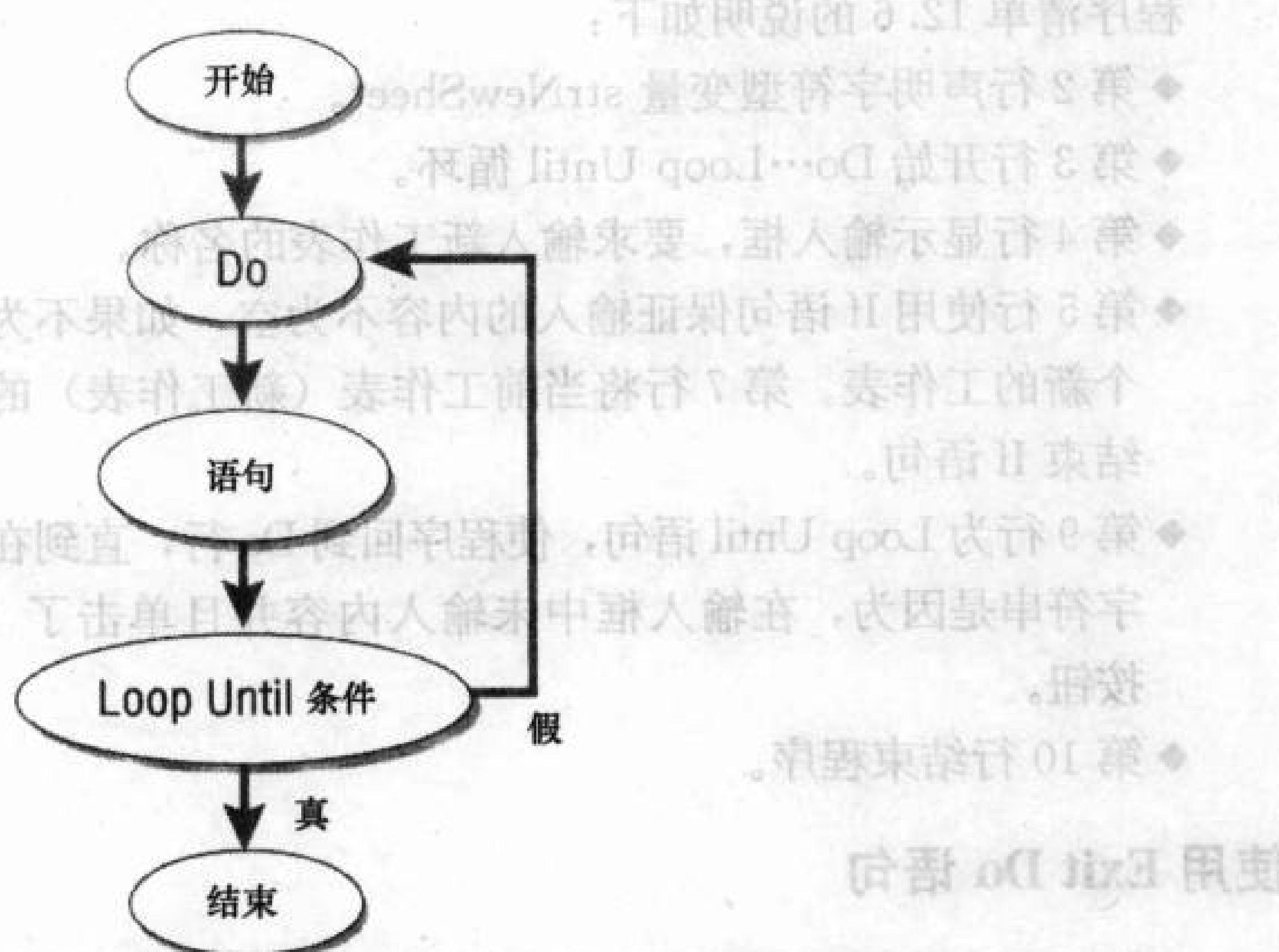


图 12.5 Do…Loop Until 循环中的命令在条件检测之前至少执行一次

语法

Do … Loop Until 的语法如下：

Do

[语句]

[Exit Do]

[语句]

Loop Until 条件

2.5 单击按钮

VBA 进入 Do 行的循环，执行循环中的语句。当遇到 Loop Until 时检测条件，如果条件为假，VBA 返回到 Do 行再次执行语句。如果条件为真，VBA 终止循环，并执行循环后面的语句。

举一个例子，通过输入框在工作簿中快速增加一个新的工作表，直到使用“取消”按钮，或者输入一个空字符串。代码参见程序清单 12.6。

程序清单 12.6

```

1. Sub Create_Worksheets()
2.     Dim strNewSheet As String
3.     Do
4.         strNewSheet = InputBox _
            ("Enter the name for the new worksheet " _ 
             & "(31 characters max.):", "Add Worksheets")
5.         If strNewSheet <> "" Then
6.             ActiveWorkbook.Worksheets.Add
7.             ActiveSheet.Name = strNewSheet
8.         End If
9.     Loop Until strNewSheet = ""
10.    End Sub

```

程序清单 12.6 的说明如下：

- ◆ 第 2 行声明字符型变量 strNewSheet。
- ◆ 第 3 行开始 Do…Loop Until 循环。
- ◆ 第 4 行显示输入框，要求输入新工作表的名称。
- ◆ 第 5 行使用 If 语句保证输入的内容不为空，如果不为空，第 6 行对当前工作簿增加一个新的工作表。第 7 行将当前工作表（新工作表）的名称赋给 strNewSheet。第 8 行结束 If 语句。
- ◆ 第 9 行为 Loop Until 语句，使程序回到 Do 行，直到在信息框中输入空字符串。出现空字符串是因为，在输入框中未输入内容并且单击了“确定”按钮，或者单击“取消”按钮。
- ◆ 第 10 行结束程序。

使用 Exit Do 语句

正如在 For 循环中使用 Exit For 语句一样，可使用 Exit Do 语句退出 Do 循环而不执行其余的语句。Exit Do 语句是可选的，并且如果循环设计正确的话，应该在循环中少用 Exit Do 语句。

如果确实需要使用 Exit Do 语句，一般需要有条件。程序清单 12.7 的彩票的例子更加有趣，增加了有 If 条件的 Exit Do 语句，在中彩金额小于 500 美元时发挥作用。

程序清单 12.7

```

1. Sub Lottery_2()
2.     Dim sngWin As Single
3.     Do Until sngWin > 2000
4.         sngWin = Rnd * 2100
5.         If sngWin < 500 Then
6.             MsgBox "Tough luck. You have been disqualified.", _
vbOKOnly + vbCritical, "Lottery"
7.         Exit Do
8.     End If
9.     MsgBox sngWin, , "Lottery"
10.    Loop
11. End Sub

```

程序清单 12.7 的工作方法和程序清单 12.4 相同，区别在于第 5 行增加了新的 If 条件。如果变量 sngWin 小于 500，第 6 行和第 7 行的语句将起作用。第 6 行显示信息框，指明彩票玩家不符合条件，第 7 行退出循环。

使用 Exit Do 是否很拙劣

有些程序员认为，使用 Exit Do 语句退出 Do 循环是最后的手段，或者至少很笨拙。另一些人则不同意。许多人认为这是可以接受的，用于反馈一个错误，或者用于取消过程。

VBA 执行 Exit Do 语句毫无问题，所以使用起来不会有害，不过不使用 Exit Do 语句也可获得同样的效果。例如，在循环中检测是否退出循环的条件，可以构建到循环的主要条件中，这时可以使用运算符 End、Or 或者 Not。

如果代码简单，可以使用运算符。然而如果生成和维护很困难，不应该强迫使用。这时候，Exit Do 语句可以很好地起作用。

While…Wend 循环

除了 For…Next 循环、For Each…Next 循环以及介绍过的 4 个 Do 循环外，VBA 还支持 While…Wend 循环。While…Wend 属于 VBA 的循环结构，用于早期的编程语言。例如，WordBasic 编程语言，用于 Word 直到 Word 95 版本。VBA 包括 While…Wend，主要用于和早期的版本兼容，而不作为推荐工具。但如果需要，也可以使用。Do 循环超越 While…Wend 但 While…Wend 仍然可以使用。

While…Wend 循环的语法如下：

While 条件

[语句]

Wend

如果条件为真，VBA 执行循环中的语句。当遇到 Wend 关键词时（为 While End 的缩写），返回到 While 语句，再次检验其条件。如果条件为假，循环中的语句不再执行，将执行 Wend 后面的语句。

下面的语句为简单的 While ... Wend 循环：

```
While Documents.Count < 10
    Documents.Add
Wend
```

如果 Documents 集合中的文档数（这里通过 Documents 集合中的 Count 属性来测量）小于 10，循环执行。每一次将根据通用模板通过 Documents All 语句生成一个新的文档（因为没有指定文档）。新的文档生成后，第 3 行的 Wend 语句将返回到第 1 行，再次检测 While 的条件。

警告：使用 While ... Wend 循环时，必须明确进入循环的唯一方法是 While 条件。

如果指向 While ... Wend 循环中间（例如使用标记和 GoTo 语句），将导致错误。

循环嵌套

在一个循环中可使用一个或多个循环以构建重复执行的结构。可以在 For 循环中嵌套另一个 For 循环，在 Do 循环中嵌套 For 循环，在 For 循环中嵌套 Do 循环，在 Do 循环中嵌套 Do 循环。

注意：VBA 允许循环嵌套 16 层，不过达到该数目的一半就很难理解了。如果代码变得这样复杂，应考虑是否应当采用简单的方法解决问题。

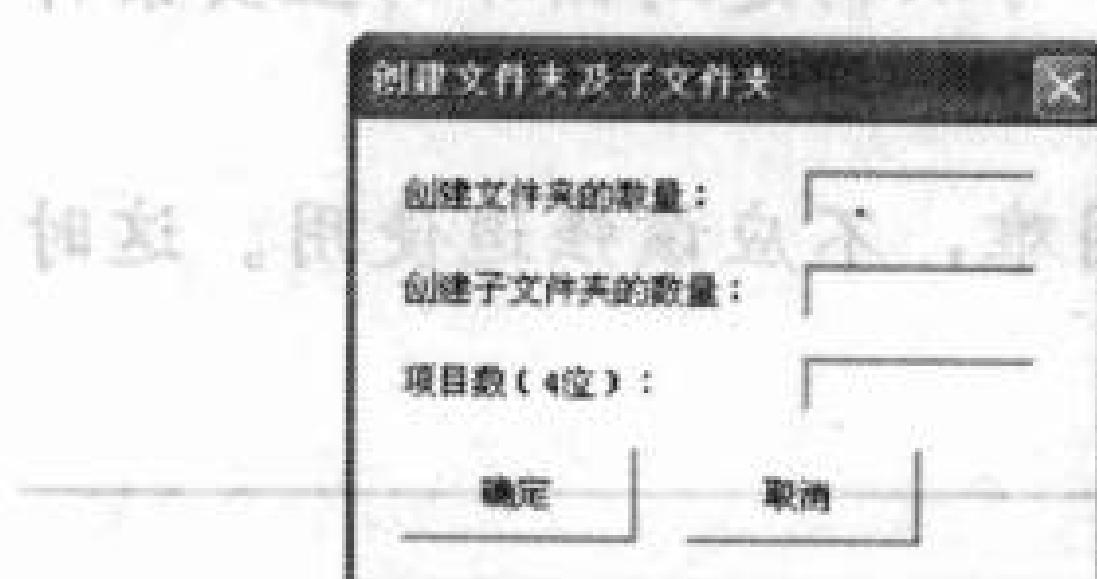


图 12.6 “创建文件夹及子文件夹”对话框

例如，需要生成多个目录，而每个目录中包含子目录，可以把前面介绍过的 Create_Folders 程序改变一下。

该程序的对话框需要另一个文本框，即创建子文件夹的数量。新的对话框名为 frmCreateFoldersAndSubFolders，创建子文件夹的数量的文本框名为 txtHowManySubFolder。图 12.6 为对话框的式样。

程序清单 12.8 为窗体上“确定”按钮的单击事件。

程序清单 12.8

```
1. Private Sub cmdOK_Click()
2.
3.     Dim strStartingFolder As String
4.     Dim strFolderName As String
5.     Dim strSubfolderName As String
6.     Dim intSubfolder As Integer
7.     Dim intLoopCounter As Integer
8.
9.     frmCreateFoldersAndSubfolders.Hide
10.    Unload frmCreateFoldersAndSubfolders
11.
12.    strStartingFolder = CurDir
```

```
13.  
14.    For intLoopCounter = 1 To txtHowManyFolders.Value  
15.        strFolderName = txtProjectNumber.Value & "s" &  
16.            Format(intLoopCounter, "0#")  
17.        MkDir strFolderName  
18.        ChDir strFolderName  
19.        For intSubfolder = 1 To txtHowManySubfolders.Value  
20.            strSubfolderName = "Subsection" & intSubfolder  
21.            MkDir strSubfolderName  
22.        Next intSubfolder  
23.        ChDir strStartingFolder  
24.    Next intLoopCounter  
25. End Sub
```

程序清单 12.8 的解释如下：

- ◆ 第 1 行开始程序，第 25 行结束程序，第 2 行为空行。
- ◆ 第 3 行到第 5 行声明 3 个字符型变量。
- ◆ 第 6 行声明整型变量，第 7 行声明另一个整型变量，第 8 行为空行。
- ◆ 第 9 行隐藏用户窗体，第 10 行卸载窗体，第 11 行为空行。
- ◆ 第 12 行把当前文件夹的名称保存在字符型变量 strStartingFolder 中，该变量用来保证后面的事情在适当的文件夹中发生。第 13 行是空行。
- ◆ 第 14 行到第 16 行以及第 23 行和前面的程序基本相同。它们用来创建文件夹，名称使用 txtProjectNumber 文本框中的 Value 属性、字母 s、两位数的数字以及 i 变量。然后使用 MkDir 语句创建一个新的文件夹。
- ◆ 第 17 行使用 ChDir 语句改变文件夹为刚刚生成的文件夹 strFolderName。
- ◆ 第 18 行是嵌套 For…Next 循环的开始，该循环将条件变量 intSubFolder 从 1 变到 txtHowManySubfolders，该值是对话框中输入的子文件夹的数量。
- ◆ 第 19 行根据单词 SubSection 和计数器变量 intSubFolder 的值生成字符型变量 strSubFolderName。在本过程中，可假定每个部分的子部分不超过 10，因此单位数字就可以满足条件。
- ◆ 第 20 行使用 MkDir 语句和字符变量 strSubFolderName 生成子文件夹。
- ◆ 第 21 行使用 Next intSubFolder 语句回到嵌入 For 语句的开始。VBA 开始测试条件，并在必要的条件下重复循环。
- ◆ 第 22 行改变文件夹回到 strStartingFolder，执行外层的循环（否则下一个文件夹将在当前文件夹 strFolderName 中生成）。
- ◆ 第 23 行回到外层循环的开始。

提示： 嵌套 For 循环时，应当使用计数器参数，定义循环的终止。使用该语句使程序更容易阅读，以防止 VBA 出现不希望出现的结果。嵌套的循环必须完全符合顺序，并且计数器需要匹配。

防止死循环

如果产生死循环，将会永远执行，或者直到计算机崩溃。例如，有一种尚未介绍过的循环是 Do…Loop。参见程序清单 12.9，没有条件制约，该结构将产生死循环。

程序清单 12.9

```

1. Sub InfiniteLoop()
2.     Dim x
3.     x = 1
4.     Do
5.         Application.StatusBar =
6.             "Your computer is stuck in an endless loop: " & x
7.         x = x + 1
8.     Loop
9. End Sub

```

在程序清单 12.9 中，第 2 行定义了变量 x，第 3 行将 x 赋值为 1。第 4 行开始 Do 循环，显示状态栏信息，每次增加 1。该循环的结果是显示信息，每次针对状态栏增加数字，直到按下 Ctrl+Break 组合键结束程序，或者溢出变量。这样做毫无意义（除非也许是为了烧坏一台新计算机），基于这样的情况，最好不使用 Do…Loop 结构，至少在没有条件制约的情况下不要使用。

不管使用什么样的循环，为了避免死循环，都必须保证终止循环的条件在某种状态下能够满足。例如，对于正在编写或清除的程序，常常需要执行一个命令直到文档的末尾，然后停止。通常，需要包括一些计数方式，以保证 Do 循环不会超过某个循环次数。

第四部分 使用消息框、输入框和对话框

- ◆ 第 13 章 使用消息框和输入框以获得用户输入
- ◆ 第 14 章 生成简单的自定义对话框
- ◆ 第 15 章 生成复杂对话框

· 单工音振以野以一某天时

· 設置音讯工被端其辭音步內口窗“剪刀”而器辭以Basic Alphab. 五式發曰人師
· 乳膠甲迎斯德自式。剪刀聲陰中辭甲迎一某次方五果吸。I
· 衣露甲迎主从。“器辭諭”<“本”<“工具”等類處，HFT/A 不述。S
· 器辭辭以自。Basic Alphab.
· 斜音音頻視音，“器辭音賓露工”銀剪。辭內口窗“剪刀”宜剪以勝其个一天时。3

第 13 章 使用消息框和输入框以获得用户输入

- ◆ 在状态栏上显示信息

- ◆ 显示消息框

- ◆ 显示输入框

- ◆ 了解消息框和输入框的局限性

本章说明如何开始向已录制或编写好的代码添加一个用户界面，以便增加代码的效率和功能。本章将学习三种与代码用户沟通的最容易的方式，其中两种是用户能够在过程中做出决定的最容易的方式和请求用户输入的最容易的方式。在学习中，将会学到如何确定在各种给定的环境下，哪种方式是与用户沟通的最好方式。这将为本书后面开始考查借助于自定义对话框来实现更复杂的与用户的互动设置舞台。

在大多数应用程序中，VBA 可提供多至五种与过程用户进行沟通的方式：

- ◆ 在窗口底部的状态栏上显示信息（如果该应用程序提供了状态栏的话）。这种方式有点受到限制，但应该算是一种与用户沟通的有效方式。
- ◆ 显示一个消息框（通常是在屏幕中部）。无论是与用户沟通，还是让用户在已提供的信息的基础上做出单项选择，消息框都是很有用的手段。本章很大的篇幅是研究消息框。
- ◆ 显示一个输入框（通常也是在屏幕中部）。可以使用输入框来与用户沟通，但是它的首要用途是请求一个信息项。输入框也向用户提供了手段，使用户可以在控制某一过程的流程时做出单项选择，尽管提供这一选择的机制比起在消息框中提供选择，受到更多的限制。在本章末尾将讲述输入框。
- ◆ 显示一个对话框（通常也是在屏幕中部）。既可以使用对话框来与用户沟通，也可以让用户做出很多选择。最好把对话框留到其他沟通方式都不够用时才使用；换句话说，当简单消息框或输入框还管用时，不使用对话框。在本书后面，就能看到如何使用 VBA 用户窗体来创建自定义对话框。
- ◆ 通过文档或应用程序的界面进行直接沟通。例如，用户可以填一张表格，单击屏幕上某一个按钮，然后等待反应。同样，也可以使用工具栏或菜单来提供互动。

打开某一过程以进行工作

确认已经在 Visual Basic 编辑器的“代码”窗口内进行编辑做好了所有设置：

1. 如果正在为某一应用程序创建代码，先启动该应用程序。
2. 按下 Alt+F11，或选择“工具”>“宏”>“Visual Basic 编辑器”，从主应用程序启动 Visual Basic 编辑器。
3. 打开一个过程以便在“代码”窗口内编辑：使用“工程资源管理器”，导引到包含该

过程的模块，然后在“代码”窗口内滚动到该过程，或从“代码”窗口内的“过程”下拉列表中选择该过程。

提示：也可以选择“工具”>“宏”>“宏”以显示“宏”对话框，在“宏名”列表框内选中已创建的某个过程，然后单击“编辑”按钮，以显示出“Visual Basic 编辑器”，且“代码”窗口内有已打开的该过程。

如果想针对一个新建过程而不是已有过程进行工作的话——这可能是个好主意，因为它有助于避免损坏已有过程——可以创建一个新过程：在某一模块的空白行上输入 Sub 关键字和过程名，然后按下 Enter 键。VBA 添加上括号和 End Sub 语句。例如，可键入下述内容并按下 Enter 键：

```
Sub Experimentation_Zone
```

VBA 添加括号和 End Sub 语句，同时添加一个分隔行，以便将这个过程与“代码”窗口内其他相邻过程隔开。

```
Sub Experimentation_Zone()  
End Sub
```

如果已经打开了一个过程，为了要测试它，可以使用 F8 键在“中断”模式中单步执行它；或使用“运行子过程/用户窗体”按钮来运行它，这时不会有各语句依次增亮的情况出现。（也可以将过程名称键入到“立即”窗口，并按下 Enter 键来执行这个过程。）

在 Word 和 Excel 中显示状态栏信息

在 Word 和 Excel 中显示状态栏信息，就有了一个便利方式，可以在不停止代码执行的情况下告诉用户，过程中有何事发生。通过在过程工作时在状态栏上显示状态信息，不仅可以告诉用户该过程正在做什么，还能指出它确实是正在运行。

注意：有时会碰到这种问题：用户认为过程崩溃了或工作失败了，因为在屏幕上看不到任何变化，但实际上，过程正在后台正常工作。在这种情况下，状态栏上的更新信息会让用户看到过程正在工作。

在状态栏上显示信息的主要缺点是用户可能会错过这些信息，因为用户可能没加注意，或者并未想到在那里能看到信息。如果应用程序能以令人注目的方式在状态栏上向用户给出信息（Word 和 Excel 就是这样做的），这不应该是问题。但是，如果拿不准的话，就要告知用户，将有信息在状态栏上显示。例如，可以在过程起始处显示一个消息框，告诉用户注意观看状态栏上的更新信息。

为了在 Word 和 Excel 中的状态栏上显示信息，可以将 Application 对象的 StatusBar 属性设定为一个适当的文本字符串。下面为显示如图 13.1 所示的状态栏信息的代码：

```
Application.StatusBar = "Word 正在格式化报表,请稍等 . . . "
```

在状态栏上显示的信息，在更改它之前，或应用程序自身要在那里显示信息之前，其显示保持不变。例如，如果在状态栏上显示了信息，然后又在 Excel 中调用“复制”命令，

Excel 就会在状态栏上显示它的正常的有关“复制”的信息，“选定目标区域，然后按 Enter 键或者选择粘贴”，将用户创建的信息擦除掉。应用程序信息优先于用户创建的信息。

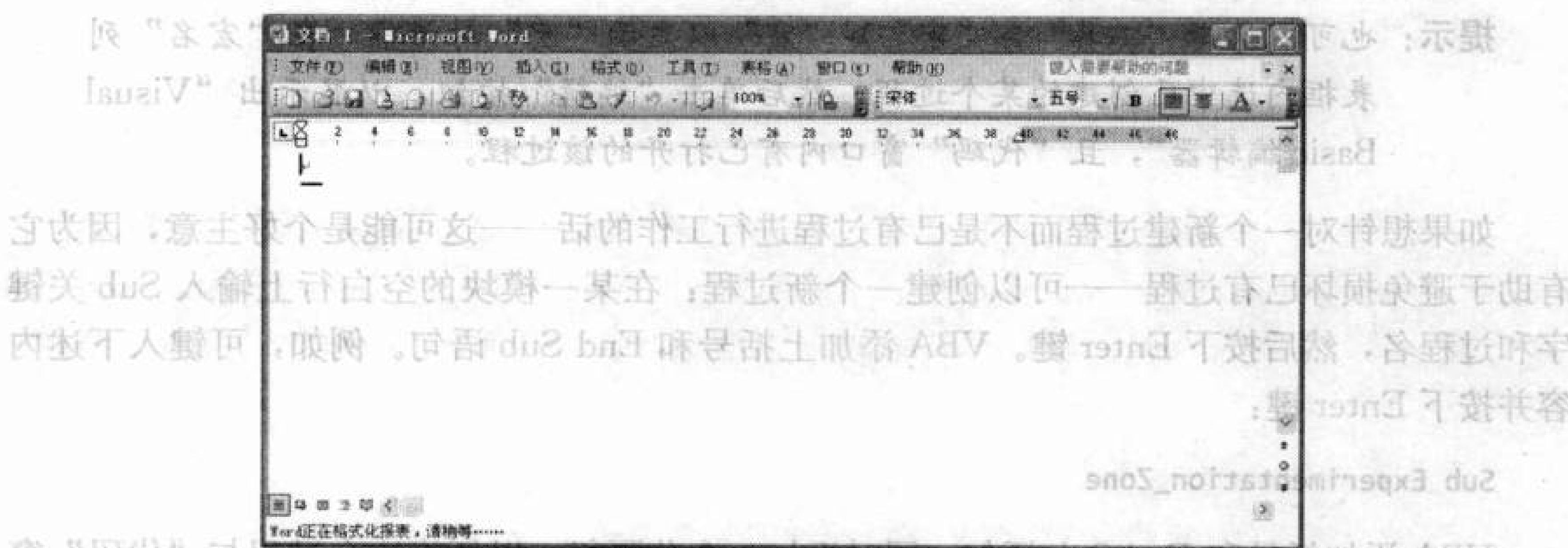


图 13.1 在某些应用程序中，可以在状态栏上显示信息

如果要在过程的进程中在状态栏上显示信息，通常必须在过程中使其更新，以避免在该过程已完成了运行之后，还有误导信息留在状态栏上。例如，可以显示另一条信息，说明过程已经结束，或者清除状态栏，代之以显示空字符串。

为了清除状态栏，可使用如下语句，向它指定空字符串：

```
Application.StatusBar = ""
```

为了看看这一语句的效果，从带有 Word 或 Excel 可视窗口（至少要使状态栏能看到）的 Visual Basic 编辑器上运行它。要想看到更佳效果，可以先运行一个在状态栏上显示信息（如：Application.StatusBar = "Hello, World!"）的语句，再用对应于 Application.StatusBar = "" 语句的信息来清除它：

```
Application.StatusBar = "Hello, World!"  
Application.StatusBar = ""
```

提示：在较长过程中在状态栏上显示进展指示信息是特别有帮助的。这样，用户可

以得知过程仍在运行并取得进展。例如，可以显示这样的进展信息：“Excel

正在工作在 150 张工作表的第 9 张上。”还有更简单的，在状态信息结束处添

加若干句号，就给出了进展指示，虽然它不能告诉用户过程已经走了多远或

还有多长的路要走。

消息框

第二个可向用户提供信息的工具是消息框，在用过的几乎每一种 Windows 应用程序中，都能看到其实例。消息框虽然既简单又有限，但它们几乎能在任何过程或模块中起重要的作用。

消息框的典型使用包括：

- ◆ 告诉用户，这个过程打算做什么（如果该过程不是用户想要的，应该使用户有机会去取消过程的运行）。

- ◆ 向用户说明，该过程下一步将要做什么，并要求用户做出简单的决定（通常是指让它继续下去，还是转至另一进程）。
- ◆ 警告用户，该过程遇到一个错误，并允许用户就此采取行动。
- ◆ 告知用户，该过程已经成功地运行过了并已经结束。对于那种关闭了屏幕更新，或其他方法向用户隐瞒自己的作为的过程来说，这一信息是特别有用的。因为这种过程会使用户无法确认，过程是仍在进行还是已经结束。你也可以使用消息框来报告过程已经做过了什么——例如，它已经更改了一些特定项，或是发现了文档中的问题，这些都需要引起注意。

本章说明如何创建满足上述各项任务要求的消息框。在后面两章里，将学习如何创建专门的消息框来增进各种过程。

使用消息框的优点和缺点

使用消息框的优点是：

- ◆ 用户不会错过观看消息框。（如果愿意的话，通过按下 Alt+Tab 键“冷切换”到另一个应用程序，会显示出用户不会漏看的消息。这在本章后面要讨论。）
- ◆ 可以让用户在两个或三个选项中做简单的选择。

使用消息框的缺点是：

- ◆ 一个消息框仅能提供一个、两个或三个按钮，这意味着它只能向用户提供很有限的选项组。
- ◆ 消息框内的各按钮是按组来预定义的——不能将自定义按钮放入消息框。（为此，必须使用对话框。）
- ◆ 不能在消息框中使用文本框、组合框或列表框等特性。

消息框的语法

消息框的基本语法表示如下：

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

语法中各成分的含义为：

MsgBox 它是一个函数，VBA 用它来显示一个消息框。使用它时，在它后面的括号中要带上一些参数。

prompt 它是 MsgBox 函数必需的一个参数，它控制显示在消息框中的文本。prompt 是一个字符串参数，指明需要在选择的文本中进行输入；它最多可以有 1023 个字符的长度，但是写得简练一些当然更好。（当 prompt 长于 1023 个字符时，只留下 1023 个字符，其余则舍去，且不给出警告。）

buttons 它是一个可选参数。它说明消息框包含哪些按钮，从而控制 VBA 显示的消息框的形式。例如，同在很多页面里看到的那样，可以显示仅带一个“确定”按钮的消息框；仅带“确定”和“取消”两个按钮的消息框；带有“终止”、“重试”和“忽略”三个按钮的消息框，等等。也可以向 buttons 添加参数以控制消息框中的图标和消息框的模式。本章后面将看到这些选项。

title 它是一个可选参数，用于控制消息框的标题栏。它也是一个字符串参数。如果不

指定 title，VBA 就使用应用程序的标题——对 Word 使用 Microsoft Word，对 Excel 使用 Microsoft Excel，对 PowerPoint 使用 Microsoft PowerPoint，等等。通常，最好还是要指定标题，因为使用应用程序的名称的好处不大（除非用户搞不清是哪种应用程序在运行过程）。

helpfile 它是一个可选参数，用于控制当用户在消息框内按下 F1 键（或在含有“帮助”按钮的消息框中单击“帮助”按钮）以寻求帮助时，VBA 显示哪个“帮助”文件。

context 它是一个可选参数，用于控制 VBA 跳到“帮助”文件中的哪一个主题上。如果指定了 helpfile 参数，就必须要指定 context 参数。

在下面各节中，首先讨论如何构建最简单的消息框，再讨论如何向它添加参数，让消息框更复杂一些。

显示简单的消息框

可以通过把 prompt 指定为包含在双引号内的文本字符串来显示一个简单的消息框：



图 13.2 当仅仅使用 prompt 参数来显示一个简单消息框时，VBA 使用其应用程序的名称作为其标题

MsgBox "这是一个简单的消息框。"

从 Excel 上运行它，这个语句便产生出如图 13.2 所示的简单消息框。因为 prompt 是提供的唯一参数，所以，VBA 产生的这个消息框只有一个“确定”按钮，而标题栏则是应用程序的名称。除了显示信息外，这个消息框不做其他事。

将这个 MsgBox 语句输入到某一过程的任何空白行上。键入 MsgBox 关键字之后，VBA 的“自动列出成员”特性会显示出该函数的语法，如图 13.3 所示。

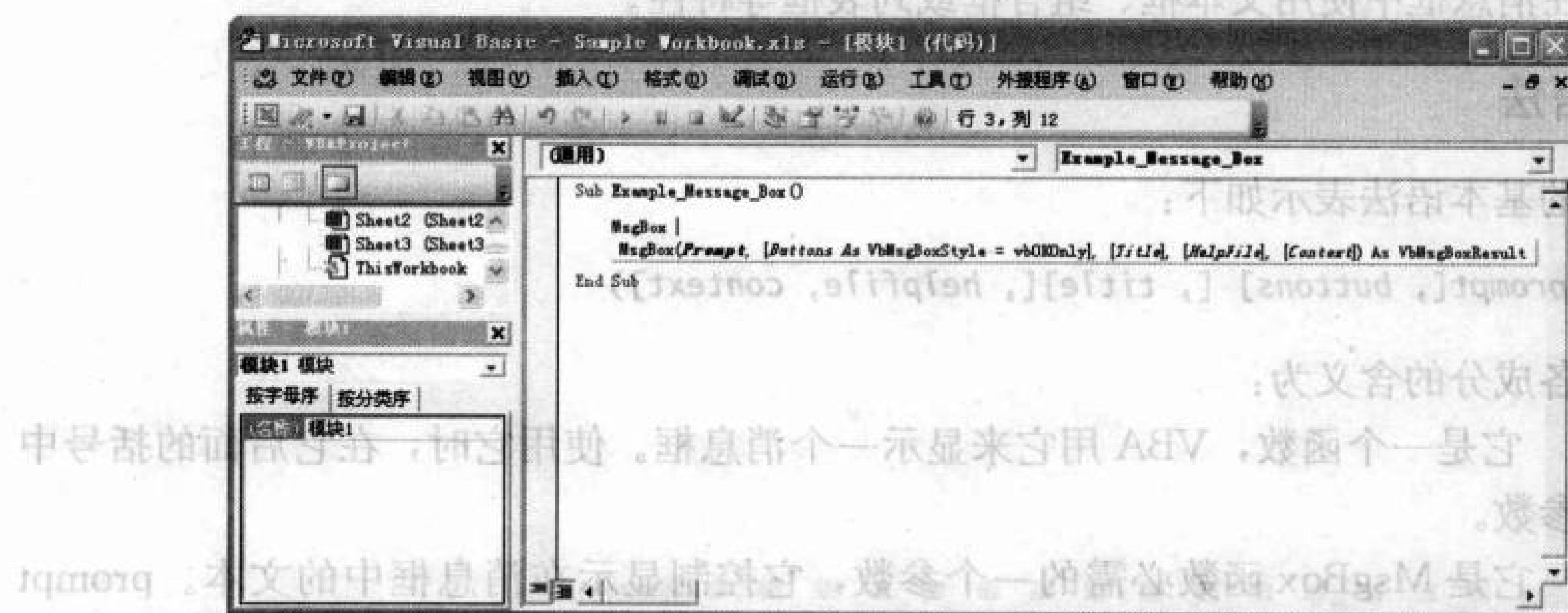


图 13.3 VBA 的“自动列出成员”特性显示该消息框的语法

注意：“自动列出成员”屏幕提示中出现的语法，与在 MsgBox 函数的“帮助”列表中看到的内容略有不同，在“帮助”列表中，helpfile 和 context 是在一个方括号之内——`[, helpfile, context]`——不是各有各的括号。这是因为，可以同时使用这两个参数，或者都不使用，但不能只使用其中一个。屏幕提示未能表达这一差别。由于这类互相依存的参数比较少，屏幕提示的这一局限性不至于会引起太多问题；但是，如果发现 VBA 在某一语句处发生障碍，而该语句明显带有屏幕提示中显示的那种参数，则应在“帮助”文件中查看。

一旦输入了 MsgBox 语句及其必需的参数 (prompt)，就可以通过单步执行代码的方式（按下 F8 键，或单击“调试”工具栏上的“逐语句”按钮）或运行过程的方式（单击“运行子过程/用户窗体”按钮，或选择“运行”>“运行子过程/用户窗体”，或按下 F5 键）来显示该消息框。

也可以使用 String（字符串）变量来代替输入对应于 prompt 参数的文本字符串。在下例中使用了名为 strMsg 的 String 变量：

```
Dim strMsg As String
strMsg = "This is a simple message box."
MsgBox strMsg
```

当碰到长字符串，或者需要显示以前在过程中定义过的字符串，或者显示由过程动态地生成字符串时，这一方法是很有用的。

显示多行消息框

按照默认方式，VBA 可在消息框里显示单行的短消息字符串，并在必要时把较长的字符串变成两行或多行，一直达到一个字符串最多为 1024 个字符的限度。

也可以有意识地把一个字符串拆成多行，即在字符串内插入换行字符和回车字符：

- ◆ Chr (13) 或 vbCr 表示回车。
- ◆ Chr (10) 或 vbLf 表示换行。
- ◆ Chr (10) + Chr (13) 或 vbCrLf 表示换行+回车组合。

在消息框中，这三种字符的作用相同。如果使用常量 (vbCr、vbLf 或 vbCrLf) 而不是相应的 Chr () 结构，代码会更容易阅读，也能更快地输入。通常，使用 vbCr 常量最为清楚。

使用 Chr (9) 或 vbTab，可以向字符串添加一个制表符。同样，vbTab 更易于阅读和输入。

例如，下面的代码会显示出如图 13.4 所示的 Word 消息框。注意：文本字符串的每一部分都包含在双引号之内（以便告诉 VBA，它们是字符串的一部分）。Chr (149) 是一个项目符号，在项目符号后面的文本都以若干空格开头，以便给项目符号留出位置：

```
Dim strMsg As String
strMsg = "Word has finished formatting the report you requested." &
    vbCr & vbCr & "You can now run the following procedures:" & vbCr &
    vbCr & Chr(149) & " Distribute_Report will e-mail the report to " &
    "the head office." & vbCr & vbCr & Chr(149) & _
    " Store_Report will copy the report to the holding directory." &
    vbCr & vbCr & Chr(149) & " Backup_Report will create a backup " &
    "of the report on the file server."
MsgBox strMsg
```

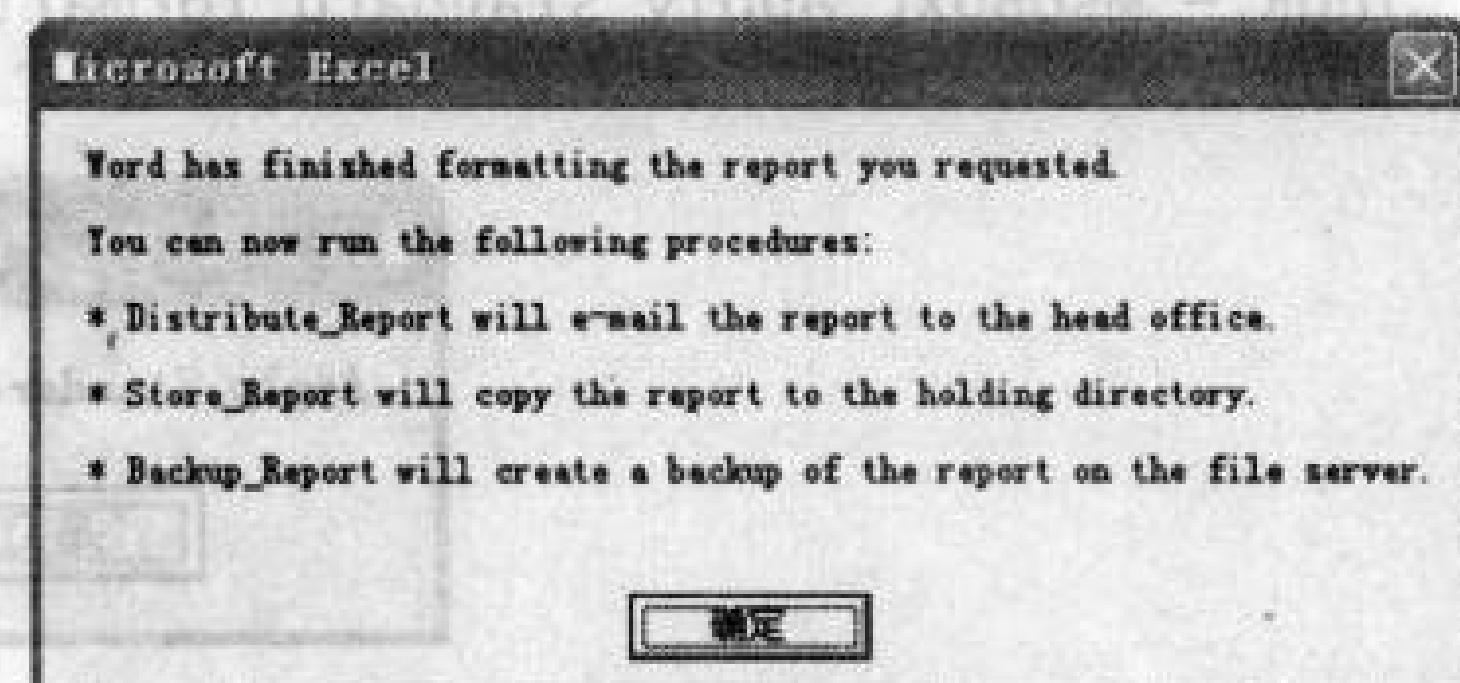


图 13.4 在 prompt 字符串内使用换行和回车字符，可以显示多行消息框

提示：在这个例子中，要注意 & 号和=号前后都有一个空格。可以自己输入这些空格，也可以让 VBA 在插入点移到另一行时输入这些空格。（将插入点移到另一行，会使 VBA 对刚刚在上面工作过的那行进行检查。）

为消息框选择 buttons

参数 buttons 用来控制消息框应该包含哪些按钮。VBA 提供了如表 13.1 所示的消息框样式，该样式由 buttons 参数进行控制。

表 13.1 消息框样式由 buttons 参数控制

数值	常量	按钮
0	vbOKOnly	OK (确定)
1	vbOKCancel	OK, Cancel (确定, 取消)
2	vbAbortRetryIgnore	Abort, Retry, Ignore (终止, 重试, 忽略)
3	vbYesNoCancel	Yes, No, Cancel (是, 否, 取消)
4	vbYesNo	Yes, No (是, 否)
5	vbRetryCancel	Retry, Cancel (重试, 取消)

可以使用数值或者常量来引用这些消息框样式。例如，可以指定 1，也可以指定 vbOKCancel 来生成带有“确定”按钮和“取消”按钮的消息框。数值比较便于输入，而常量便于阅读。从 PowerPoint 上运行时，两个 lngR 语句的任一个都可以用来生成如图 13.5 所示的消息框：

```
Dim lngR As Long
lngR = MsgBox("Apply standard formatting to the slide?", vbYesNo)
lngR = MsgBox("Apply standard formatting to the slide?", 4)
```

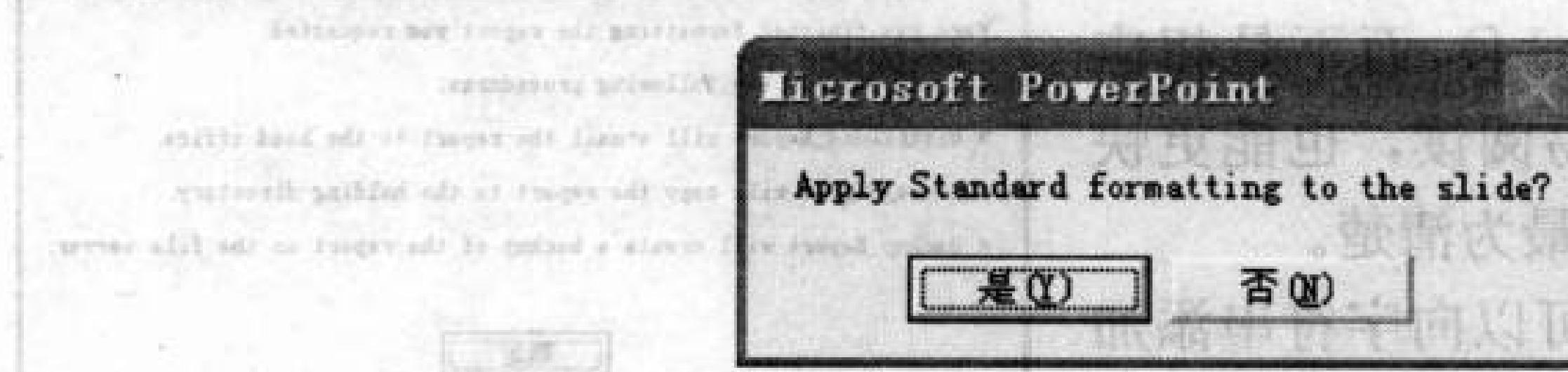


图 13.5 vbYesNo 常量产生出带“是”和“否”按钮的消息框

从 VBA 的观点来看，在生成对应于过程的消息框时，使用数值或常量都是可以的。从人的观点来看，更喜欢使用常量，因为它会使代码好读得多。

为消息框选择图标

通过使用表 13.2 所示选项中的适当数值或常量，也可以将图标添加到消息框去。

同样，既可以使用数值，也可以使用常量来引用这些图标：48 或 vbExclamation 都能生成一个感叹号图标。同样，也是常量更易读。

表 13.2 用于消息框图标的参数

数值	常量	显示
16	vbCritical	停止图标 (X)
32	vbQuestion	问号图标 (?)
48	vbExclamation	感叹号图标 (!)
64	vbInformation	信息图标 (i)

可以使用加号 (+) 把对应于消息框的数值或常量与对应于图标的数值或常量连接起来。例如，输入 vbYesNo + vbQuestion (或 4 + 32，或 vbYesNo + 32，或 4 + vbQuestion)，就能生成包含“是”和“否”按钮并有问号图标的对话框 (见图 13.6)：

```
lngR = MsgBox("Apply standard formatting to the slide?", -  
    vbYesNo + vbQuestion)
```

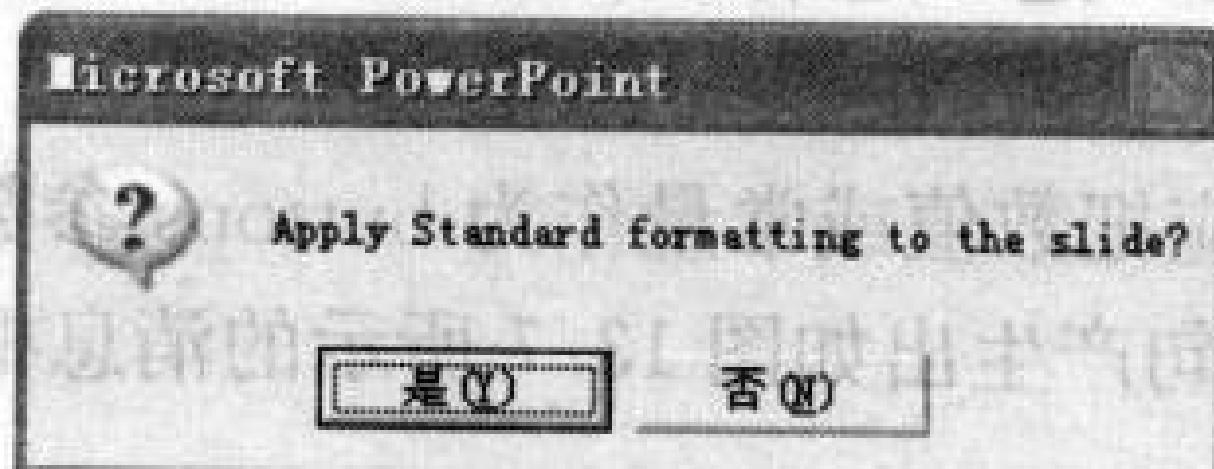


图 13.6 增加图标使消息框产生更大的视觉效果

为消息框设置默认按钮

通过在 MsgBox 语句中指定一个按钮，可以把这个按钮设置成消息框的默认按钮。在对能够采取剧烈行动的过程进行分配时，指定默认按钮会是一个明智之举。

例如，假设有这样一个过程，它在文档未关闭时就删除当前文档，然后转到文件管理程序（如 Windows Explorer），或者在某一个常用的对话框内（如“打开”对话框或“保存”对话框）造成麻烦。如果有用户无意中运行了这个过程，它就会破坏用户的工作，所以，希望能在“确认”消息框里设置默认的“否”或“取消”按钮，这样用户可以主动地去选择只运行该过程的其余部分。

如同在 Windows 界面里常见的那样，消息框内的默认按钮是可以看出来的，它的外边缘四周是黑色边界线，而它的文本区四周是虚线。使用 Tab、Shift + Tab、→、←、↑或↓键，可以把选择移到另一个按钮。

注意：由于用户只要简单地按下 Enter 键就选择了默认按钮，所以，在消息框或对话框内设有默认按钮，有助于用户更快地处理消息框或对话框。VBA 自动地将消息框中第一个按钮设置成默认按钮，所以，只有当需要把其他按钮而不是第一个按钮作为默认按钮时，才进行默认按钮指定。

表 13.3 列出了与默认按钮对应的参数。

表 13.3 对应于默认的消息框按钮的参数

数值	常量	按钮
0	vbDefaultButton1	第一个按钮为默认按钮
256	vbDefaultButton2	第二个按钮为默认按钮
512	vbDefaultButton3	第三个按钮为默认按钮
768	vbDefaultButton4	第四个按钮为默认按钮

上面讲过的所有消息框都只有一个、两个或三个按钮，但是，可以给这些消息框再加一个“帮助”按钮，这样，在已有三个按钮（如，vbYesNoCancel）的消息框内就有了第四个按钮。在本章后面的“向消息框添加‘帮助’按钮”一节中，将会介绍如何去添加“帮助”按钮。

在 VBA 中，除非另加指定，每个消息框的第一个按钮被自动设置成默认按钮，例如，vbOKCancel 消息框中的“确定”按钮，vbAbortRetryIgnore 消息框中的“终止”按钮，vbYesNoCancel 消息框中的“是”按钮，vbYesNo 消息框中的“是”按钮，vbRetryCancel 消息框中的“重试”按钮，均是如此。VBA 是按照各按钮在对应于消息框样式的常量中出现的顺序来给按钮编号的（即按它们在屏幕上消息框中出现的从左到右的顺序来编号）。所以在 vbYesNoCancel 消息框中，“是”是第一个按钮，“否”是第二个按钮，“取消”是第三个按钮。

为了另行设置默认按钮，应把数值或常量作为 buttons 参数的一部分来加以指定。当在 PowerPoint 中运行时，下述语句产生出如图 13.7 所示的消息框：

```
Dim lngQuery As Long
lngQuery = MsgBox("Do you want to delete this presentation?", _
vbYesNo + vbCritical + vbDefaultButton2)
```

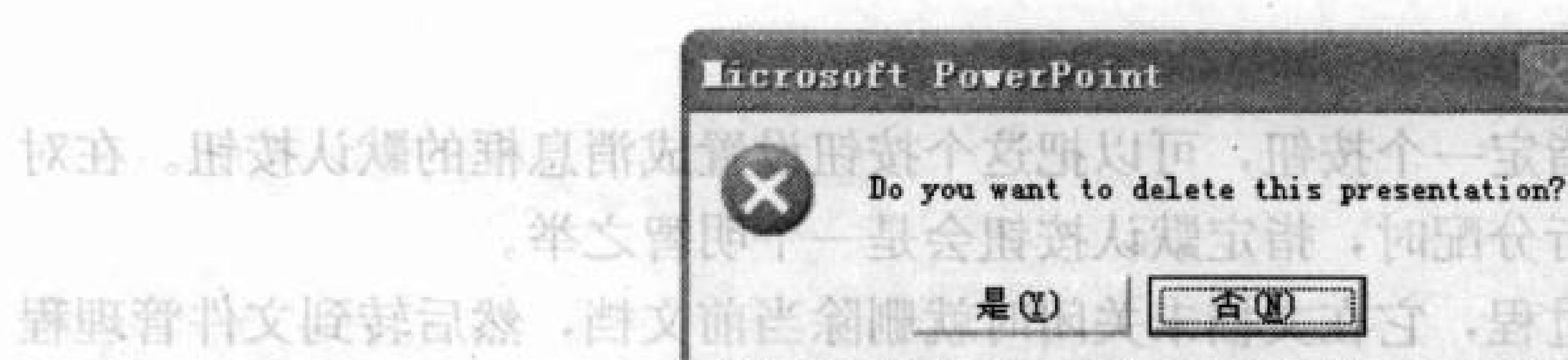


图 13.7 指定默认按钮以使用户格外注意消息框中这个特定按钮

控制消息框的模式

VBA 能够显示（至少是在理论上）应用模式消息框和系统模式消息框。用户只有对应用模式消息框做出响应并使其退出时，才能继续当前的应用，在此之前，不能在当前应用上做任何事情；而用户在对系统模式消息框做出响应并使其退出之前，不能使用计算机做任何事情。

大多数消息框是应用模式消息框，它们允许用户通过按下 Alt+Tab 键，以“冷切换”的方式（或通过任务栏切换的方式），在退出消息框之前，切换到另一应用程序并在其上工作。这给用户提供了相当程度的灵活性。相反，某些安装消息框则是系统模式消息框，坚持要用户把注意力集中在它们身上，而且只能集中在它们身上。Windows 上危险的系统错误和“你必须立即重启你的计算机”之类的消息框都是系统模式，用户对它们是逃避不了的。

用户从自己的经历中可能知道，系统模式消息框会有多恼人。所以，在设计程序时，只有在绝对必要时——例如，在某一行动可能导致数据丢失或系统失去稳定时，才使用系统模式消息框。在大多数正常情况下，应用模式消息框将做到用户需要它们做的一切——不会让过程的用户为难或烦恼。

在理论上，用户可以使用表 13.4 所示的两个 buttons 参数来控制消息框的模式。

表 13.4 对应于消息框模式的参数

数值	常量	结果
0	vbApplicationModal	消息框是应用模式
4096	vbSystemModal	消息框是系统模式

在实践中，即使使用了 vbSystemModal，用户仍可以切换到另一个应用程序中去（假

设原来已运行在一个应用程序中), 并继续工作。但是, 消息框仍然滞留“在顶部”, 保持其显示——这对用户是个干扰, 不过不能阻止用户去访问另一个应用程序。

默认情况下, 消息框是应用模式, 所以用户只有在需要系统模式消息框时才需去指定模式, 而这种情况是很少的。如果要这么做的话, 需向 buttons 参数添加常量 vbSystemModal 或数值 4096:

```
Response = MsgBox("Do you want to delete this document?", _  
    vbYesNo + vbCritical + vbDefaultButton2 + vbSystemModal)
```

注意: 系统模式消息框的外形和应用模式消息框相同。

为消息框指定标题

消息框的下一个组成部分是它的标题栏, 它是由可选的 title 参数来控制的。如果略去 title, VBA 将提供应用程序的名称作为标题, 不过, 过程用户总是希望能得到一个较有帮助的标题, 以从中受益。

title 是一个字符串表达式, 从理论上说, 其长度最多可达 1024 个字符 (更长的字符会被截去, 且不显示警告或错误消息), 但实际上, 超过 75 个字符的标题都会被截去, 而代之以省略号。如果确实希望别人去读消息框的标题栏, 25 个字符左右是比较合理的最大长度。

提示: 标题栏通常是用户关注的消息框的首要部分, 所以, 给标题栏取名应该尽量对人有帮助, 一般情况下是把过程名放在消息框的标题栏, 然后使用提示来解释消息框里的按钮将完成何种选择。此外, 如果过程会更新的话, 在标题栏中把版本号包括进去是有好处的, 这样, 用户很容易查看到, 正在使用的是哪一版本的过程 (并尽量使用最新的版本)。例如, 版本号 12.39 的删除工作簿过程在消息框中可显示为如图 13.8 所示的样子。

在 buttons 参数之后指定 title 参数, 现示例如下:

```
lngQ = MsgBox("Do you want to delete this workbook?", vbYesNo _  
    + vbCritical + vbDefaultButton2, "Delete Workbook 12.39")
```

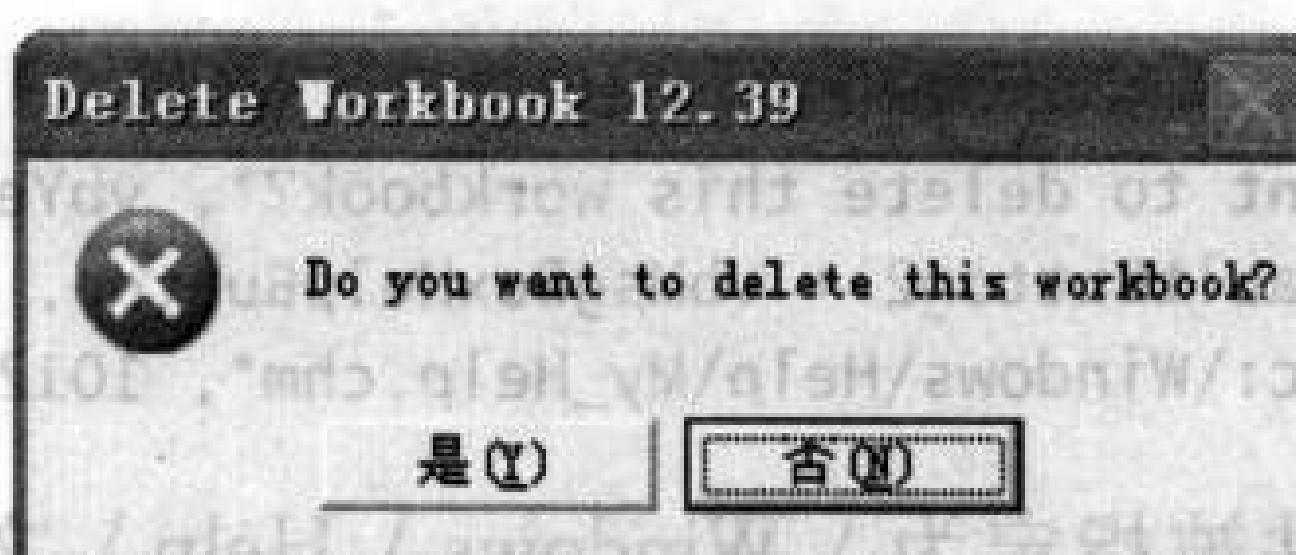


图 13.8 通常希望为消息框指定 title 参数, 也希望把代码的版本号包括进去

和 prompt 参数一样, 可以把字符串变量用做 title 参数。例如, 可以先声明一个字符串变量, 然后用过程来调用它, 把它提供给消息框做标题。也可以在消息框的标题中, 显示由过程创建的或保存在过程中的字符串。

注意: 不要把换行、回车或制表符放入 title 参数之中。VBA 能够接受它们, 但是将标题栏显示为方框, 而不是生成两行的标题栏。如果包含一个 vbCr 字符, VBA 会把标题栏内的文本上移一点; 如果包含两个 vbCr 字符, 文本就会从标题栏顶端消失。

向消息框添加“帮助”按钮

为了向消息框添加“帮助”按钮，可使用 `vbMsgBoxHelpButton` 常量。可以把它加到为消息框指定的任意 `buttons` 中去：

```
lngQ = MsgBox("Do you want to delete this workbook?", vbYesNo _  
+ vbCritical + vbDefaultButton2 + vbMsgBoxHelpButton,  
"Delete Workbook")
```

添加 `vbMsgBoxHelpButton` 是简单地把“帮助”按钮放入消息框——在指明应该使用哪一个“帮助”文件和主题之前，它不会使“帮助”按钮显示“帮助”文件（详情见下节）。图 13.9 显示出由上述语句生成的消息框。

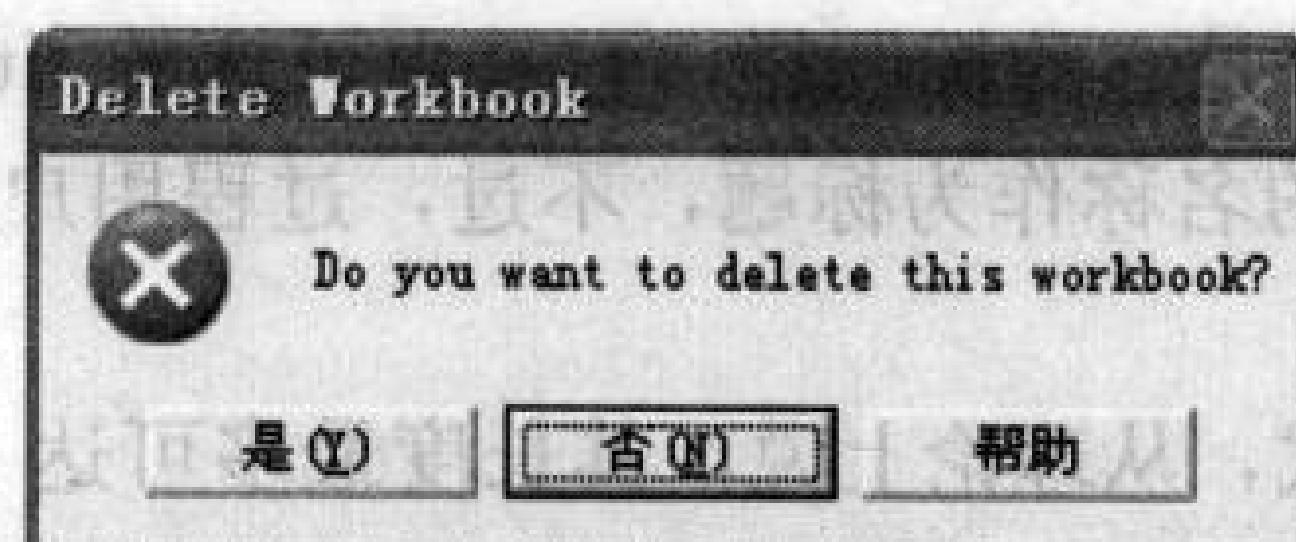


图 13.9 使用 `vbMsgBoxHelpButton` 将“帮助”按钮添加到消息框

为消息框指定帮助文件

可以用于消息框的最后两个参数是 `helpfile` 和 `context`：

- ◆ 参数 `helpfile` 是一个字符串参数，用于指明“帮助”文件的名称和存储位置，当用户通过消息框请求帮助时，VBA 显示出“帮助”文件。
- ◆ 参数 `context` 是“帮助”文件中的“帮助”上下文编号。“帮助”上下文编号控制哪一个“帮助”文件主题要被显示。

如果自己来编写“帮助”文件，`helpfile` 和 `context` 参数是非常有用的，不然的话，很难找到“帮助”上下文编号，因为它隐藏在“帮助”文件之中。如果编写自己的“帮助”文件，那么指定 `helpfile` 和 `context` 参数的语法是很简单的：

```
Dim lngQ As Long  
lngQ = MsgBox("Do you want to delete this workbook?", vbYesNo _  
+ vbCritical + vbDefaultButton2 + vbMsgBoxHelpButton, _  
"Delete Workbook", "c:\Windows\Help\My_Help.chm", 1012)
```

在本例中，“帮助”文件被指定为 \ Windows \ Help \ 文件夹中的 `My_Help.chm`。VBA 显示编号为 1012 的帮助主题。

当用户单击消息框内的“帮助”按钮时，VBA 显示“帮助”文件中预先指定的主题。消息框仍留在屏幕上，这样，当用户完成对“帮助”文件的咨询后，可以在消息框中做出自己的选择。

提示：与在屏幕上打开“帮助”文件相对应的“帮助”上下文编号是 0。当需要显示“帮助”文件，但又不知道“帮助”上下文编号时，就使用编号 0。然后，用户必须从中寻找自己需要的信息。

三种不常用但具有特殊作用的常量

VBA 为消息框提供了三种特殊的常量。你可能不会经常去用它们，但在需要时它们会很方便。在 buttons 参数中将它们指定为：

vbMsgBoxSetForeground 它告诉 VBA，要使消息框成为前景窗口。这个常量不常使用，因为按照默认方式，消息框是作为前景窗口显示的（所以用户能看到）。

vbMsgBoxRight 它告诉 VBA，消息框中的文本应向右对齐。

vbMsgBoxRtlReading 它告诉 VBA，在希伯来和阿拉伯系统中文本按从右到左的方向显示。在非 BiDi（非双向）系统中，它不起作用。

只使用部分参数

显示消息框时，某些可选参数可能做了指定，也可能省略了指定。如果想为某个函数指定后面的参数，而该参数前面的参数省略了指定的话，那么使用逗号来指明每个未被使用的可选参数即可。例如，如果想要显示一个消息框，并且没有指定 buttons 和 title 参数，可以使用下述语句：

```
Response = MsgBox("Do you want to format the report?", , ,  
"c:\Windows\Help\Procedure Help.chm", 1012)
```

这里，三个逗号表示省略了 buttons 和 title 参数（这使得 VBA 显示一个以应用程序名称为标题的 vbOKOnly 消息框），使 VBA 避免了将 helpfile 参数与 buttons 参数相混。也可以使用带名称的参数，它会使得代码不太精练但更便于阅读：

```
Response = MsgBox("Do you want to format the report?",  
HelpFile:="c:\Windows\Help\Procedure Help.chm", Context:=1012)
```

从消息框取回一个值

如果显示了一个 vbOKOnly 消息框，那么用户将很清楚要单击哪一个按钮，因为这个消息框只含有一个“确定”按钮。有些其他的消息框用了两个、三个或四个按钮，那就必须从消息框取回一个值，才能知道用户单击了哪个按钮，从而将过程指向到合适的方向上去。

为了从消息框取回一个值，应为它声明一个变量。这做起来很简单，只需要告诉 VBA 变量名和消息框是一样的：

```
Response = MsgBox("Do you want to create the daily report?",  
vbYesNo + vbQuestion, "Create Daily Report")
```

但是，通常希望声明一个适当类型的变量（一个 Long 变量）以包含用户的选择，如本章中用过的例子那样：

```
Dim lngResponse As Long  
lngResponse = MsgBox("Do you want to create the daily report?",  
vbYesNo + vbQuestion, "Create Daily Report")
```

运行该代码时，VBA 将用户选择的按钮以一个值保存起来。随后，用户可以查看该值并采取相应行动。表 13.5 显示出用户可以选择的按钮的列表。可以用常量，也可以用数值

来引用这些按钮。通常，常量比数值更易于阅读。

表 13.5 与被选择的按钮对应的常量

数值	常量	被选择的按钮
1	vbOK	OK (确定)
2	vbCancel	Cancel (取消)
3	vbAbort	Abort (终止)
4	vbRetry	Retry (重试)
5	vbIgnore	Ignore (忽略)
6	vbYes	Yes (是)
7	vbNo	No (否)

例如，为了在 vbYesNo 消息框里查看用户选择了哪个按钮，可以使用一个简洁的 If...Then...Else 语句：

```
Dim lngUserChoice As Long
lngUserChoice = MsgBox("Do you want to create the daily report?", _
    vbYesNo + vbQuestion, "Create Daily Report")
If lngUserChoice = vbYes Then
    Goto CreateDailyReport
Else
    Goto Bye
EndIf
```

这里，如果用户选择“是”按钮，VBA 就转到以 CreateDailyReport 标记来标明的代码行，并从那里继续运行过程；如果不是，它就通过转到结尾处的 Bye 标记以使过程终结。If 条件用来检查用户在消息框内进行选择而产生的响应，看看是否是 vbYes（由单击“是”按钮产生，或在“是”按钮被选择时按下 Enter 键而产生）。如果该响应不是 vbYes——就是说，如果用户选择“否”按钮或者按下 Esc 键，而这个消息框又只有“是”按钮和“否”按钮的话，就运行 Else 语句。

输入框

当需要从用户那里获得一个简单片段的文字信息时，可使用输入框。人们是通过外观而不是通过名称和输入框相熟的：它们看起来常常像是图 13.10 所示的样子。

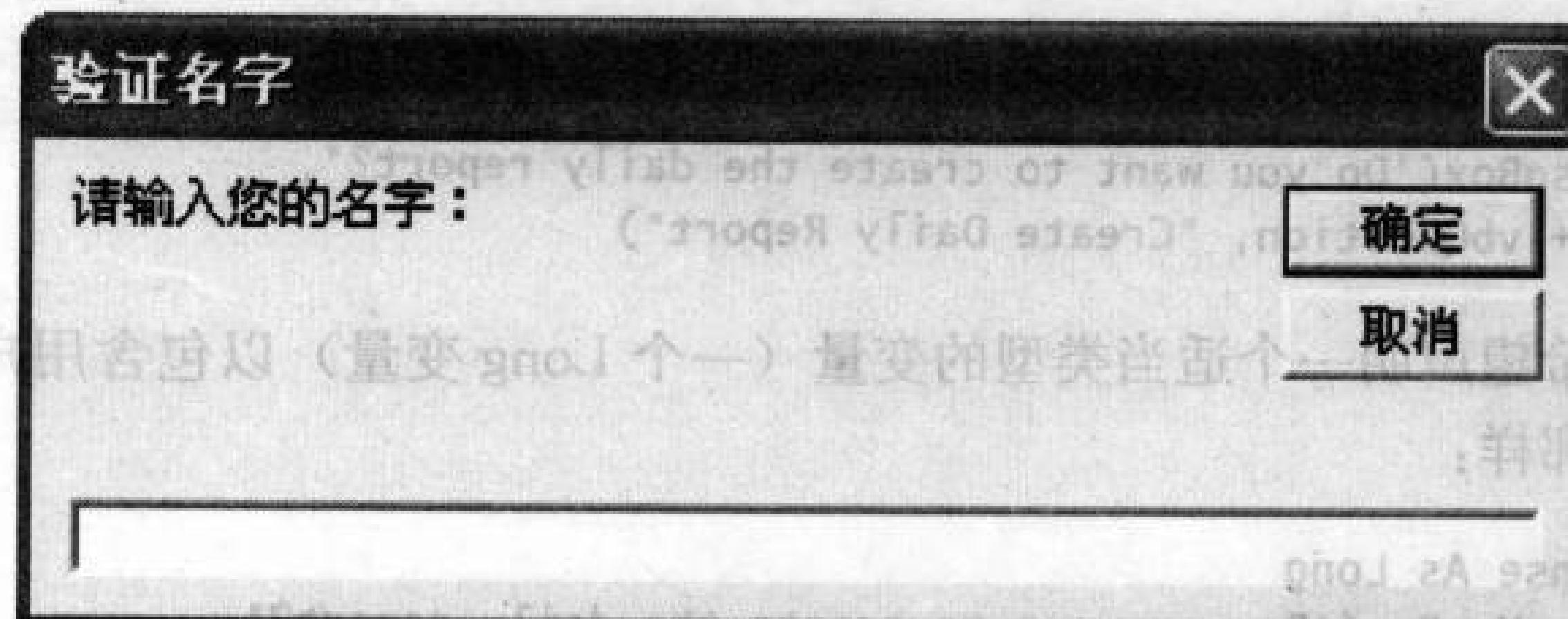
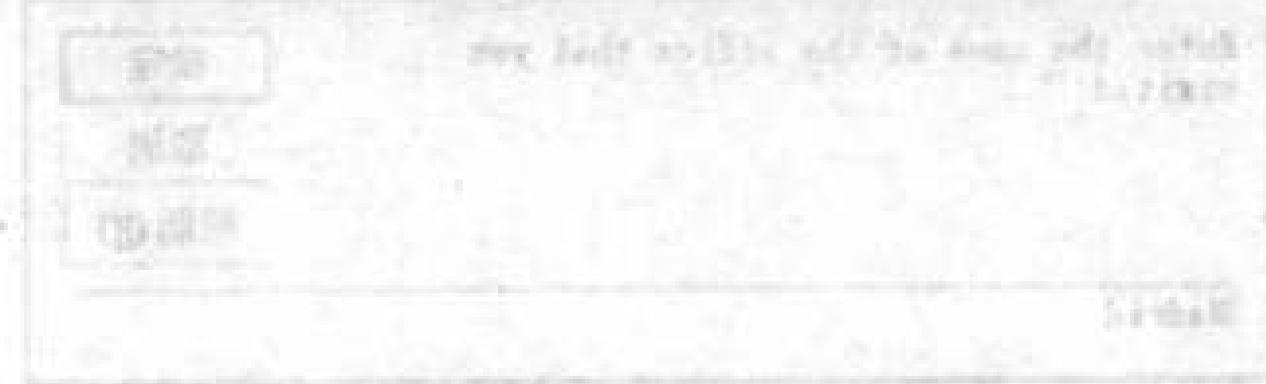


图 13.10 使用输入框以便从用户处获得简单信息

提示：为了从用户处获得两个或多个片段信息，可以连贯地使用两个或多个输入

框，但是，通常使用一个自定义对话框会更好。在第 14 章里将开始学习自定义对话框。



输入框的语法

输入框的语法比较简洁，而且与消息框的语法类似：

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

下面是各参数的含义：

prompt 它是一个必需的字符串，指明在输入框中出现的文字提示。和在 MsgBox 中一样，prompt 最多可为 1024 个字符长，可以使用换行和回车字符使各行分隔。但是，和 MsgBox 中的 prompt 不同的是，输入框的 prompt 不会自动把长字符分拆成两行或多行，必须使用换行和回车字符，使每行不长于 35 个字符。

title 它是一个字符串，用于指定输入框的标题栏内的文本。如果不指定 title 参数，VBA 就输入应用程序的名称作为标题。

default 它是一个字符串，用以说明文本框内的默认文本。在两种情况下，输入 default 参数是个好主意：当默认文本比较合适的时候；当需要显示样本文本，以便用户能够理解何种响应正在期待用户做出的时候。下面是一个合适的默认文本的例子：如果显示的输入框要求用户告诉名称，建议输入来自“选项”对话框“用户信息”选项卡的“名称”数值。

xpos 和 ypos 它们是可选的数字值，用来指定输入框在屏幕上的位置。xpos 控制输入框左边缘离屏幕（不是 Word 窗口）左边缘的水平距离，而 ypos 控制输入框顶部离屏幕顶部的垂直距离。每种量度均以 twip 为单位（参见下注）。如果略去这两个参数，VBA 就在默认位置上，即水平居中，垂直方向上距屏幕下端三分之一处显示输入框。

注意：一个 twip 是 1/1440 英寸。普通的计算机屏幕是每英寸 96 个点 (dpi)，所以每个像素为 15twip，而分辨率为 800×600 的计算机屏幕是 12 000×9000twip。为使输入框和对话框精确定位，应针对不同的屏幕分辨率以 twip 值进行试验，以期获得满意的效果。一般来说，在默认位置显示输入框效果最佳。

helpfile 和 context 它们是可选参数，用以指定“帮助”文件和“帮助”文件中的上下文。如果用户通过输入框请求帮助，显示即跳到该指定处。使用了 helpfile，就一定要使用 context。

任何可选参数均可省略。但是，如果要使用在语法顺序中位于被省略的可选参数后面的可选参数的话，必须用逗号来指明这个省略，或者使用已有名称的参数。

与消息框不同，输入框带有已预先定义的按钮组——“确定”和“取消”，如果指定了 helpfile 和 context 参数，还要加上“帮助”按钮——所以，不必为输入框指定主要按钮。下面的例子先声明字符串变量 strWhichOffice，再将输入框指定给它，所得到的输入框的结果见图 13.11。

```
Dim strWhichOffice As String  
strWhichOffice = InputBox(  
    "Enter the name of the office that you visited:", _  
    "Expense Assistant", "Madrid", , , _  
    "c:\Windows\Help\Procedure Help.chm", 0)
```

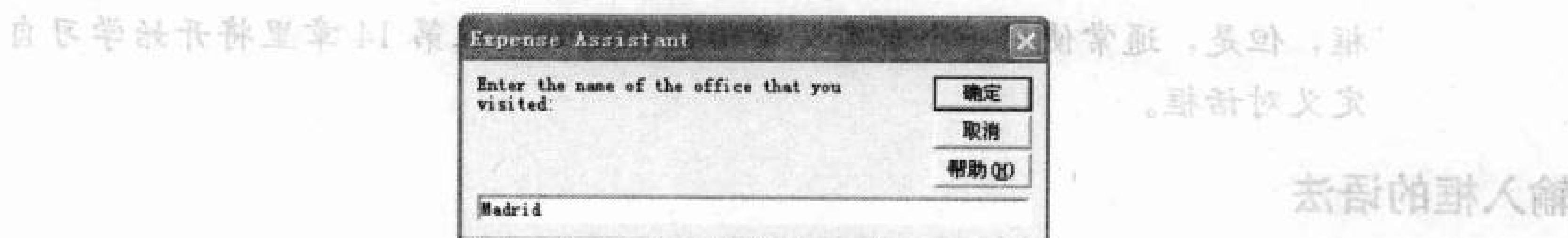


图 13.11 带有预定义按钮组的输入框

从输入框取回输入

为了从输入框取回输入，应声明将要包含该输入的数字变量或字符串变量。这里，变量 strWhichOffice 将要包含用户在输入框中输入的内容：

```
Dim strWhichOffice
strWhichOffice = _
    InputBox("Enter the name of the office that you visited:", _
        "Expense Assistant 2000", "Madrid", , , _
        "c:\Windows\Help\Procedure Help.chm", 0)
```

做完了这些，而且用户已经输入了一个值或一个字符串，并选择了“确定”按钮，VBA 就可以使用这个值或字符串了。为了确认用户已经选择了“确定”按钮，需检查输入框返回的不是零长度字符串（如果用户选择了“确定”按钮但文本框内为空，也返回一个零长度字符串），然后采取相应的行动：

```
strWhichOffice = InputBox(_
    "Enter the name of the office that you visited:", _
    "Expense Assistant 2000", "Madrid", , , _
    "c:\Windows\Help\Procedure Help.chm", 0)
If strWhichOffice = "" Then End
```

当消息框和输入框均不满足需要时

从本章中可以看出，消息框极大地增强了过程，它使得用户能在转折点处做出选择，或者向用户提供重要信息。但是，在用过一段时间消息框之后，就会注意到消息框的局限性：

- ◆ 只能提供有限的信息，因为它受到显示方式的限制。
- ◆ 只能使用不多的几种按钮组，这限制了消息框的作用。

当用户希望在消息框里能输入更复杂的消息以最大限度地使用所提供的按钮时，最好使用自定义对话框。在第 14 章和第 15 章里将会看到，自定义对话框比较容易创建，它比消息框有更强的功效和灵活性。

有些过程通过一系列消息框让用户做出很多选择，而实际上人们希望避免编写这样的过程。同样，要从用户处获得一个片段信息，输入框是有用的，但是，除此之外，它的局限性也很明显，如果要立即连续地使用两个或多个输入框，还是使用一个对话框来代替吧。

第 14 章 生成简单的自定义对话框

- ◆ 使用自定义对话框来做什么
- ◆ 生成自定义对话框
- ◆ 向对话框添加控件
- ◆ 把对话框链接到过程
- ◆ 从对话框返回用户的选择

在本章中，要开始考查 VBA 提供了功能来生成可与用户互动的自定义对话框。对话框是 VBA 最有功效但也最复杂的特性之一。本章只涉及较简洁的对话框成分及如何对这些成分进行操控。下一章要介绍如何生成更复杂的对话框，例如包含有很多带标签页面的对话框和用户单击某一控件时能更新自己的对话框。

什么时候需要使用自定义对话框

当简单的与用户互动的方法已经不能使用时，常常想要使用自定义对话框——例如，当消息框提供的有限的按钮选择已经不能让用户做出合理选择的时候，或者当需要获取更多的来自用户的信息，而简单的输入框已经提供不了的时候，就应该使用自定义对话框了。

还有，当过程要求用户借助于选择或清除复选框以选择非排他性选项的时候，当需要提供通过单选按钮人工进行的排他性选择的时候，或者，当需要向用户提供列表框以让其从中选择的时候，也都需要使用自定义对话框。同样，如果要向用户展示一张图片，或者让用户从一个列表框或一个组合框里选择一个或多个项目时，都需要使用自定义对话框。

自定义对话框可提供范围很广的界面成分，在 Windows 应用程序上工作的用户可能已经熟悉这些成分。可以生成外观和功能与内置对话框非常相像的自定义对话框。

尤其是，可以使用自定义对话框来驱动过程，所以，为了响应用户采取的某个行动，自定义对话框往往就出来了。例如，当用户启动一个过程时，需要该过程显示一个对话框，以提供过程将要做什么样的选择——如选择该过程需要操作的文件。还可以生成由 VBA 触发的对话框，以便对计算机系统中的事件做出响应。例如，对在某一特定时刻发生的事件的响应，或在用户采取某一特定行动（如创建、打开或关闭一个文档）时做出的响应。

由于生成对话框是一件相对复杂和费时的事，想一想其他切实可行的替代办法是明智的。前面学习过消息框和输入框，如果想要针对某些较容易的任务生成自定义对话框，消息框和输入框倒是提供了一个简单的替代办法。有些应用程序如 Word 和 Excel，甚至可以让用户把它们的内置对话框借为己用。如果用户对应用程序熟悉，可能也熟悉这种对话框，那么就可以立即使用它来实施标准行动——例如，打开文件或保存文件。

生成自定义对话框

VBA 使用称为用户窗体的可视对象来构建对话框。用户窗体（有时就称为窗体）是一张空白表，可以在上面放置控件（如复选框和按钮）以生成对话框。

用户窗体包含有代码表，附属于控件的代码即放置其内。代码可以附属于任何控件和用户窗体本身；这种代码就保存在用户窗体的代码表里。可以在 Visual Basic 编辑器的“代码”窗口中，显示用户窗体的代码表，像使用任何其他代码那样使用它。可以像运行过程一样来运行用户窗体（例如选中用户窗体并按下 F5 键，即可运行该窗体），Visual Basic 编辑器将执行窗体幕后的代码。

每个用户窗体都是应用程序的用户界面的一部分，这意味着，实际上可以显示一个用户窗体（一个对话框），让用户与之互动，然后从用户窗体中取回信息，并用 VBA 来处理这些信息。

注意：也可以生成不是对话框的用户窗体。对话框和窗口的区别是可以讨论的。但是，可以简单地认为，窗口是可以调整大小的（拖曳其边框，或单击其最大化按钮，即可调整其大小），而对话框则不行。有些对话框，例如，Word 中的“查找和替换”对话框，有一个初始隐藏部分，可以把它显示出来（在本例中，单击“高级”按钮即可）。但是，除了这种可扩展部分以外，对话框的边框是固定的——使用鼠标，抓住对话框的一角再拖曳它，想这样来扩大对话框是行不通的。

每个用户窗体自身都是一个对象，它又包含有可以分别操控的很多其他对象。例如，可以生成只有两个选项按钮、一个“确定”按钮和一个“取消”按钮的简单对话框。每个选项按钮又是一个对象，“确定”按钮是第三个对象，而“取消”按钮是第四个对象。可以为每个对象设置属性——例如，单击“取消”按钮时采取的行动，或用户使用鼠标指针在某个选项按钮上方游移时显示的屏幕提示——以使对话框尽可能地易于理解、简洁和有用。

注意：屏幕提示也称为工具提示。

为对象设置的大多数属性，既可以在设计阶段（在生成用户窗体时）也可以在运行阶段（显示用户窗体前或显示用户窗体时）设置。例如，可以把复选框控件的 Value 属性设置为 True，以显示复选框处于被选中状态，或把该属性设置为 False，以显示复选框处于清除状态。既可以在生成用户窗体时设置该 Value 属性（这样，每次运行这个用户窗体时都要进行设置），也可以在准备显示这个用户窗体时设置。

下节将说明生成对话框的过程。在本章结束处，给出几个例子，说明如何创建一个过程，并把一个对话框与之相连接。

设计对话框

不做太多计划，粗糙地搞出一个只用很少控件的半凑合的对话框，可能也做得到。但是，通常在生成一个对话框时，应该采用更有序的方法，在开始生成它之前，要计划好需要把哪些东西包含进对话框中去。搞清楚对话框打算具备的功能，列出实现这一功能所必需的各个成分。然后绘出对话框的草图，以便对每个成分置于何处有个大致的想法。

提示：另一个选择是以某个现有的对话框为基础来设计自定义对话框——可以以内置对话框为基础，也可以以自己公司或机构里已经做好的自定义对话框为基础。利用以前的开发成果，不仅避免了从头做起，而且会使用户觉得对生成的自定义对话框很眼熟。完全不能按直觉——几乎和计算机毫不相干——然而没有关系。理解是关键。

插入用户窗体

一旦有了设计想法，生成自定义对话框的第一步，就是在适当的模板或文档中插入一个用户窗体。

1. 显示 Visual Basic 编辑器，如果它还未打开的话。
2. 在“工程资源管理器”窗口内，右击合适的工程，并从快捷菜单中选择“插入”➤“用户窗体”。

注意：单击“工程资源管理器”窗口内的工程，或单击“标准”工具栏上的“插入”按钮，然后从下拉列表中选择“用户窗体”，也可以完成用户窗体的插入。（如果按钮已经显示“插入用户窗体”，单击该按钮即可，不必再用下拉列表。）也可以单击工程，然后选择“插入”➤“用户窗体”。

Visual Basic 编辑器打开一个新的用户窗体，如图 14.1 所示，这个窗体名为 UserForm1（如果工程中已含有名为 UserForm1 的用户窗体，则使用下一个可用序号）。Visual Basic 编辑器还显示出工具箱。（如果在用户窗体中工作而工具箱还处于隐藏状态的话，选择“视图”➤“工具箱”，或单击“标准”工具栏上的“工具箱”按钮，可使工具箱显示。）

VBA 将此用户窗体插入到对应于该工程的 Form 对象（窗体集合）中去。如果所选择的工程还没有包含一个 Form 对象，那么 VBA 会添加一个以包含新的用户窗体。可以看到 Form 对象显示在“工程资源管理器”窗口中。

选择窗体网格设置

Visual Basic 编辑器在每个用户窗体内显示出一个网格，这是为了便于放置控件和对齐控件。要关闭这一网格显示，或关闭 Visual Basic 编辑器的“自动对齐控件到网格”，需遵循如下步骤：

1. 选择“工具”➤“选项”，以显示“选项”对话框。
2. 单击“通用”标签以显示“通用”页（见图 14.2）。
3. 选择希望的设置：
 - A. 如果希望关闭网格显示，需清除“显示网格”复选框。（网格继续起作用，但网格点不显示出来。）
 - B. 如果想停止使用网格，不考虑对齐问题，就清除“对齐控件到网格”复选框。这一特性通常能够节省时间。如果网格太粗略，不利于放置，可以尝试建立网格，再调整其大小。
 - C. 改变“宽度”、“高度”文本框内的点数以调整网格单位的大小。按默认方式，每个单位为 6 点。
4. 单击“确定”按钮以关闭“选项”对话框，并使上述选择起作用。

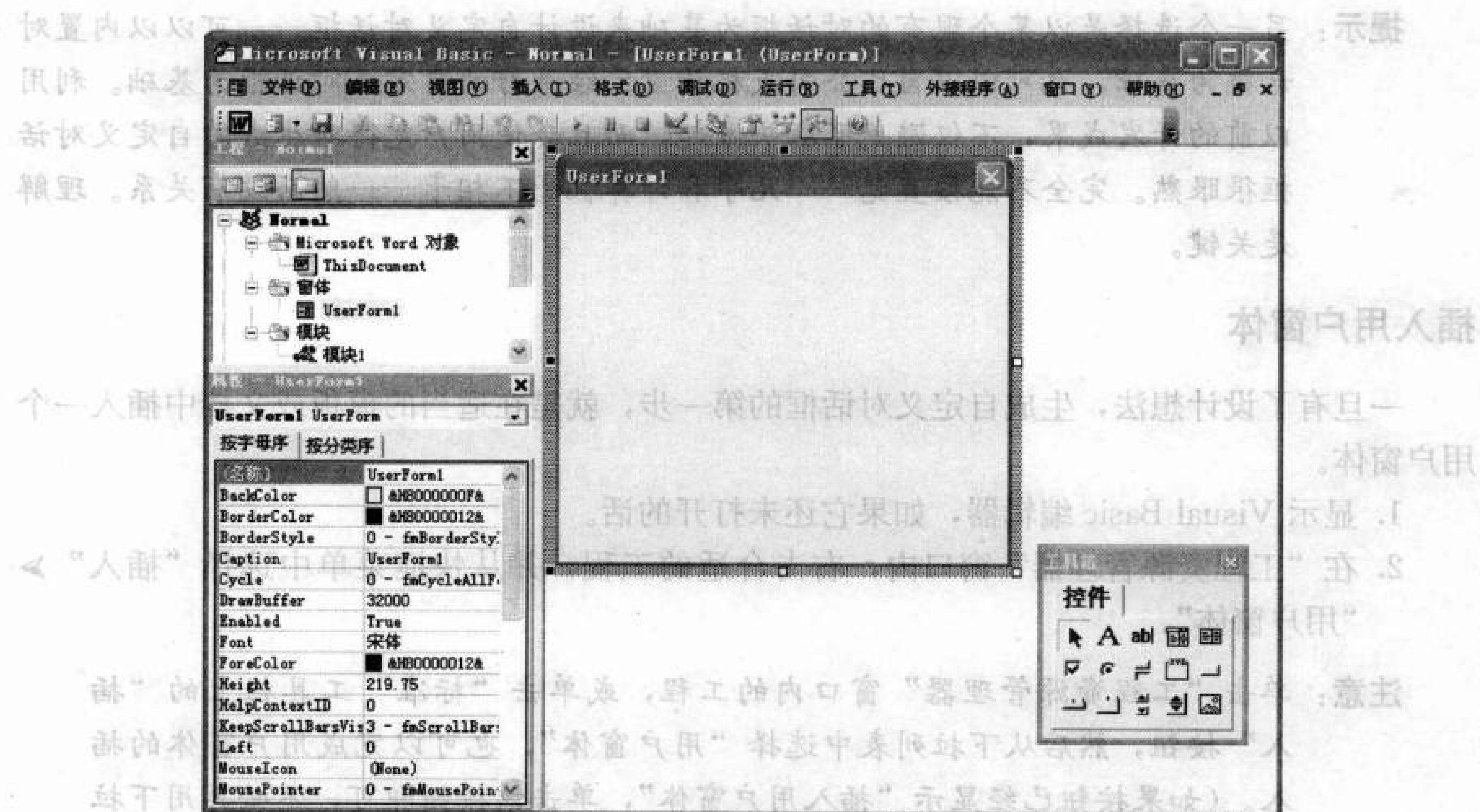


图 14.1 生成新对话框的第一步是启动一个新用户窗体。当用户

窗体是活动窗口时, Visual Basic 编辑器显示工具箱

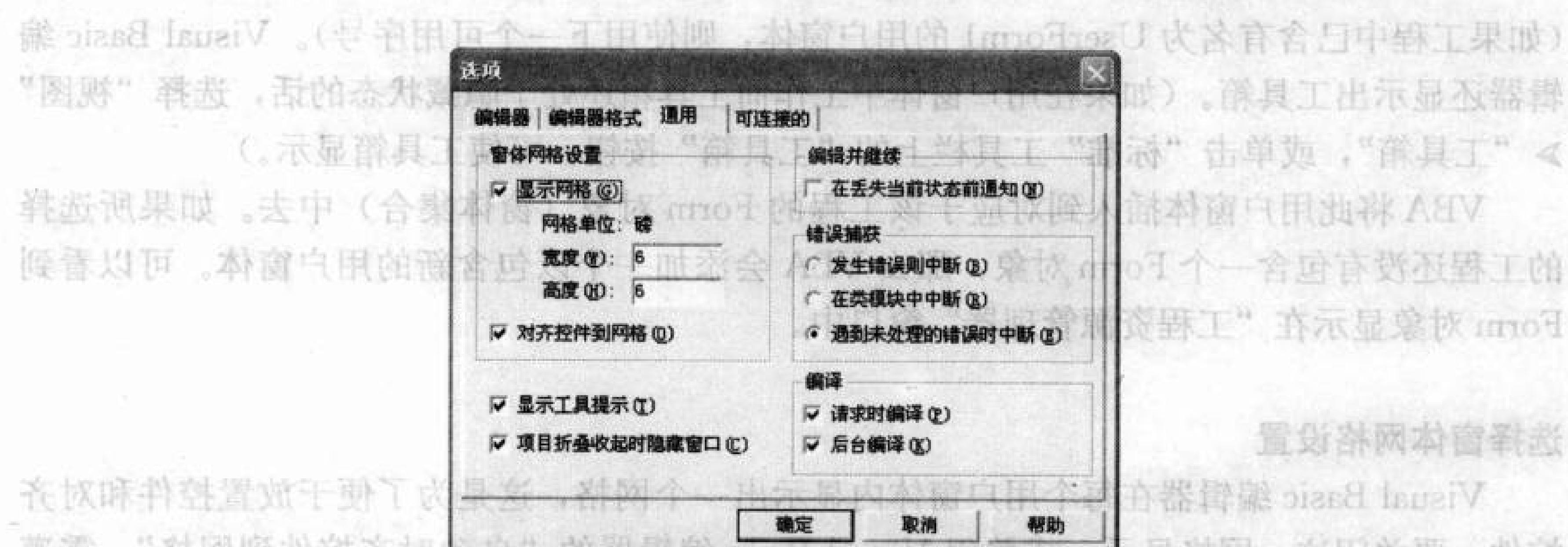


图 14.2 “选项”对话框的“通用”页, 包含有切换网格显示、
调整网格大小和决定是否对齐控件到网格等选项

VBA 命名约定

VBA 中对象的名称最多可有 40 个字符长, 必须以字母开头, 开头字母之后可以是字母、数字和下划线的任意组合。在名称中不能使用空格或符号, 而且在其上下文中, 每个名称必须是唯一的——例如, 在一个工程之内, 每个用户窗体必须有一个唯一的名称, 但是, 任何用户窗体或对话框中的对象, 可以和另一个对话框中的对象同名。

这些都是规则; 也可以使用约定来为 VBA 对象命名, 以便尽可能一致和易于理解。例如, 使用用户窗体名称要以字母 frm 开头的约定, 可以在阅读别人的代码时, 立即识别出某个名称属于用户窗体——代码编写人也能在时隔很久之后重读自己的老代码时识别出名称来。还可以在必要处添加注释, 使代码清楚好读。

针对最常用的 VBA 对象的某些通行的命名约定，显示在下表中。在学习本书后面内容的过程中，会碰到针对其他 VBA 对象的命名约定。

对象	前缀	例子
复选框	chk	chkReturnToPreviousPosition
命令按钮	cmd	cmdOK
窗体（用户窗体）	frm	frmMoveParagraph
框架	fra	fraMovement
列表框	lst	lstConferenceAttendees
组合框	cmb	cmbColor
菜单	mnu	mnuProcedure
选项按钮	opt	optSpecialDelivery
标签	lbl	lblUserName
文本框	txt	txtUserDescription

命名约定是：从小写字母的与每种对象对应的前缀开始，然后，对象名称的其余部分以大写字母开头，以便于阅读。可以在 VBA 名称中使用下划线，把名称分隔成若干独立块，但在任何名称中不可使用空格。

命名约定勉强是个初步正式的东西，而且很多人都希望为自己的 VBA 用户窗体内的对象使用适合自己的名称。但是，如果打算分配 VBA 模块，或者还有其他人使用模块，那就值得花些时间和精力去遵守命名约定——如果要制定自己的约定，必须将其建成文档，让别人都能了解。

重命名用户窗体

下一步，把用户窗体的名称从默认名（UserForm1）改成更具说明性的名称。

提示：作为对选择名称的建议，请参看前面的补充说明“VBA 命名约定”。

1. 如果“属性”窗口没有显示，按下 F4 键以显示它。图 14.3 显示出“属性”窗口的两个页面：按字母序和按分类序。按字母序是当前选择的属性的按字母表次序列表；按分类序则包含若干类，如外观、行为、字体、杂项、图片、位置的属性列表。（有些控件的属性，比这里列出的种类还要多。）单击种类名旁边的加号（+）可使该种类展开，显示出它包含的各属性；单击种类名旁边的减号（-）则使其退回。如果未选择“按分类序”标签，单击它以选择它。

可以使用“属性”窗口的“按分类序”来输入用户窗体的名称和标题——首先要找到“Caption”属性和“Name”（名称）属性的正确位置。“Caption”属性包含在“外观”集合中，而“（名称）”属性包含在“杂项”集合中。

- 确认下拉列表正在显示用户窗体的默认名称。如果没有，从下拉列表中选择该用户窗体。
- 在“（名称）”右边的单元格中选中用户窗体的默认名称（如：UserForm1 或

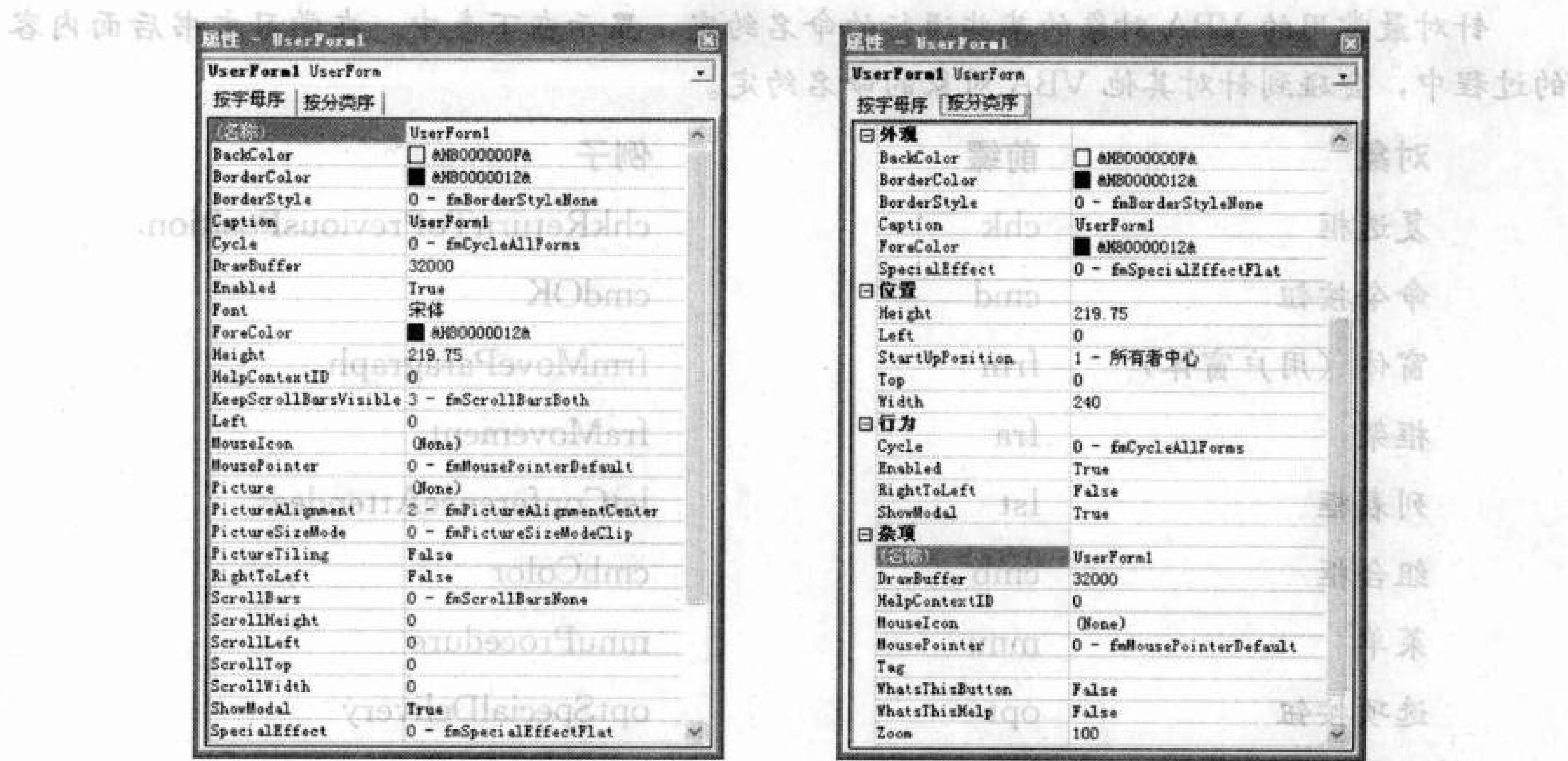


图 14.3 可在“属性”窗口上选择“按字母序”或“按分类序”

UserForm2)，为该用户窗体键入新名，可以是想用的任何名称，但要符合标准 VBA 限制：

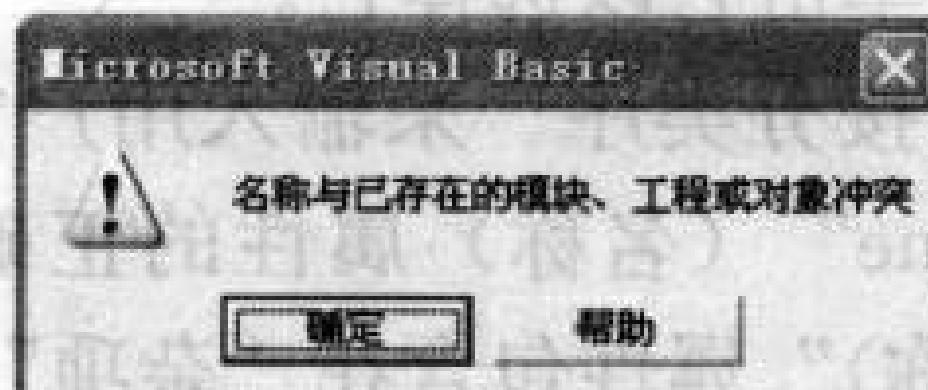
- ◆ 它必须以字母开头。
- ◆ 它可以包含字母、数字和下划线，但不能有空格和符号。
- ◆ 它最多可有 40 个字符长。

4. 单击“Caption”属性以选择用户窗体的默认名称，然后为该用户窗体键入标题——对话框的标题栏中希望出现的文本标签。除了要受到标题栏长度的限制以外，对键入内容没有限制。如果超出长度限制，VBA 在其最大可显示长度处使用省略号来截断它。键入时，名称就出现在用户窗体标题栏上，所以很容易看出，适应用户窗体当前大小的合适的长度是多长。
5. 按下 Enter 键，或在“属性”窗口内其他处（或 Visual Basic 编辑器内其他处）单击以输入用户窗体的名称。

提示：为其他对象命名，遵从此处所述的相同方法。

处理“名称与已存在的模块冲突”错误

如果遇到“名称与已存在的模块、工程或对象冲突”错误（如下图所示），很可能是，刚刚试图给某个用户窗体取的名字与已经指定给某一过程的名称相同。在未使用正式的命名约定时，此事极易发生。



这个错误也可能是重复使用了某个 VBA 工程或对象库的名称，但更像是使用了已经指定过的名称。

向用户窗体添加控件

既然已经完成了该用户窗体的重命名，就要准备使用图 14.4 所示的工具箱向用户窗体添加控件。当用户窗体为活动窗口时，VBA 自动显示出工具箱，但是，如果用户窗体未被激活，通过选择“视图”>“工具箱”，也可以显示出工具箱。

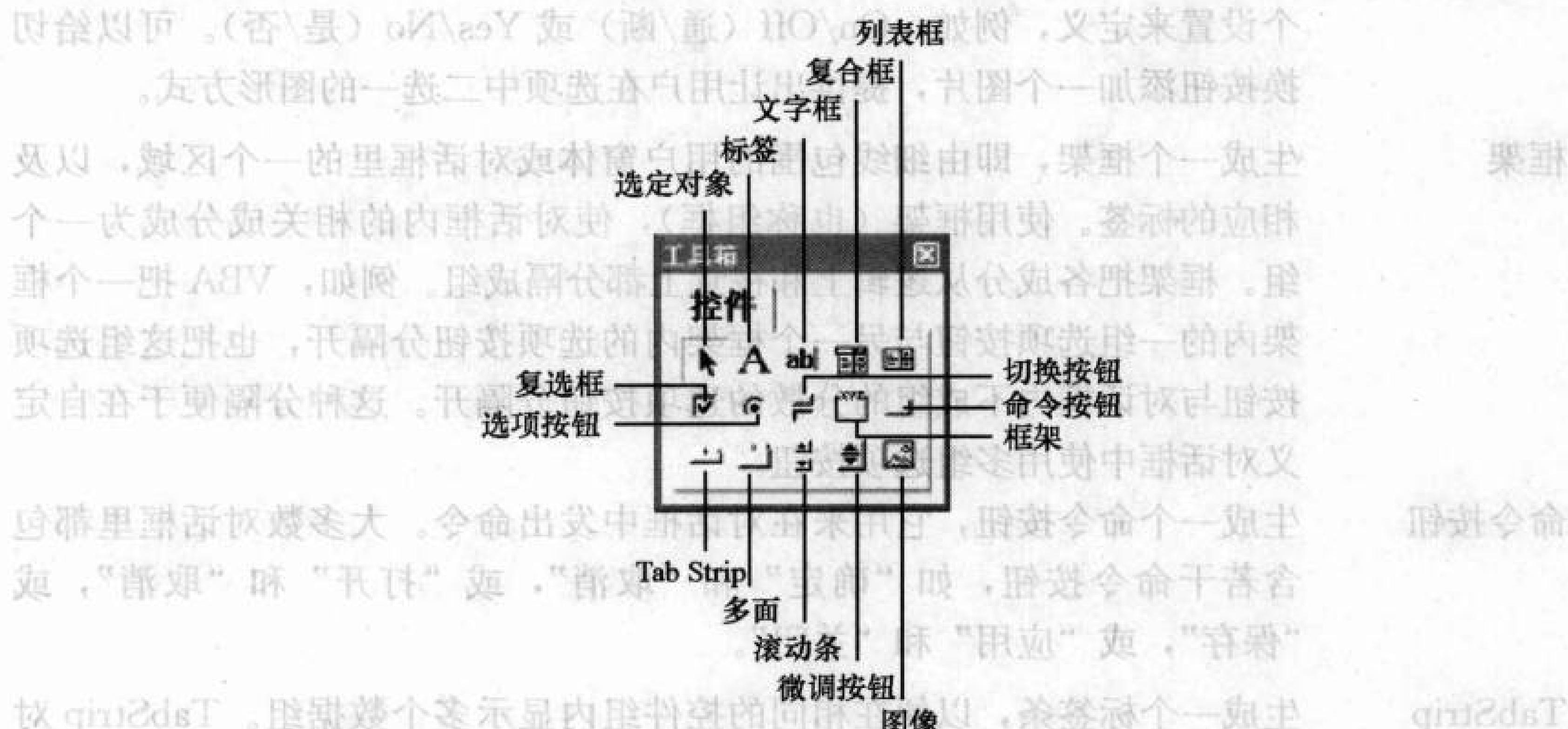


图 14.4 使用工具箱向用户窗体添加控件

以下介绍工具箱内的各按钮的作用：

按钮	行动
选定对象	将鼠标指针恢复到选择模式。一旦放置了一个对象，鼠标指针自动返回到选择模式。通常，只有当选择了另一按钮，然后又决定不用它时，或者双击一个控件，造成了多次放置时，才需要单击“选定对象”按钮。
标签	生成一个标签——文本，用来标识对话框的某一部分，或用来解释用户需知的信息以便更有效地使用对话框。
文本框	生成一个文本框（也称编辑框），用户可以向框内输入文本，也可以使用文本框向用户显示文本，或为用户提供文本，以便复制和粘贴到别处。文本框可以包含一行（默认）或多行，可以显示水平滚动条或垂直滚动条或两种滚动条都显示。
组合框	生成一个组合框，这是把文本框与列表框组合起来的一个控件。用户可以从列表框中选择一个值，或将一个新值输入到文本框。
列表框	生成一个列表框，这是列出多个值的一个控件。用户可以从列表中挑出一个值，但是不能输入新值（与组合框不同）。列表框适于提交已关闭的数据组。
复选框	生成一个复选框和相应的标签。用户可以选择或清除复选框，以使相关的行动进行或停止。

选项按钮	生成一个选项按钮（也称无线电按钮）和相应的标签。此按钮通常是一个圆圈，当某个选项被选中时，圆圈内出现一个黑点，用户只能从选项按钮组中选择一个选项按钮。（“无线电按钮”的名称来自于以按下按钮来选台的无线电接收机，因为一次只能选一个按钮。）
切换按钮	生成一个切换按钮，它显示要不要选择某一个项目。切换按钮可以用两个设置来定义，例如，On/Off（通/断）或 Yes/No（是/否）。可以给切换按钮添加一个图片，提供出让用户在选项中二选一的图形方式。
框架	生成一个框架，即由细线包围的用户窗体或对话框里的一个区域，以及相应的标签。使用框架（也称组框），使对话框内的相关成分成为一个组。框架把各成分从逻辑上和视觉上都分隔成组。例如，VBA 把一个框架内的一组选项按钮与另一个框架内的选项按钮分隔开，也把这组选项按钮与对话框内不成组的分散的选项按钮分隔开。这种分隔便于在自定义对话框中使用多组选项按钮。
命令按钮	生成一个命令按钮，它用来在对话框中发出命令。大多数对话框里都包含若干命令按钮，如“确定”和“取消”，或“打开”和“取消”，或“保存”，或“应用”和“关闭”。
TabStrip	生成一个标签条，以便在相同的控件组内显示多个数据组。TabStrip 对于显示数据库中的记录是特别有用的。数据库中的每个记录都含有相同的信息字段，所以可以显示在相同的控件组内。标签条提供了在记录之间导引的便利方式。
多页	生成一个多页控件，以显示多页对话框。多页对话框含有多个布局，选择不同的标签，便能得到不同的布局。多页对话框的一个例子就是“选项”对话框（“工具”>“选项”），在大多数 Office 应用程序中，它有多个页面。
滚动条	生成一个独立的滚动条。独立的滚动条在对话框中用得比较少。组合框和列表框都有内置滚动条。
微调按钮	生成一个附属于另一控件的微调按钮控件。它通常是很小的矩形按钮，内有一个向上指的箭头和一个向下指的箭头（或者一个向左指的箭头和一个向右指的箭头）。要在合理范围内，以相同间隔来显示一系列的值，微调按钮是很管用的。例如，如果要在一个文本框里显示以 25 美分为步长的价格增减的话，可以使用它来调整价格值，这比让用户把价格直接输入文本框要好。
图像	生成一个图像控件，以便在用户窗体内显示图片。例如，可以使用图像控件，在对话框内放置一张图片或一个公司标志。

注意：本章所述的工具箱包含 VBA 提供的基本工具组合。正如第 2 章“自定义工具箱”里讨论过的那样，可以用多种方法来自定义工具箱：把其他已有的控件添加到工具箱；为控件创建附加页面；将控件从一个页面移到另一个页面；创建自己的自定义控件，以便迅速地将自己常用的工具放置好。

如图 14.5 所示，在用户窗体内单击时，VBA 标准大小的被选控件添加进去。当用户在窗体内某处单击时，VBA 就把控件的左上角置于该处。放置控件时，它会贴紧对齐用户窗体上的网格（除非“对齐控件到网格”复选框被关闭，本章前面“选择窗体网格设置”中有论述）。

提示：双击工具箱内某一控件的按钮，可以实现相同按钮的多次放置。某控件放置了第一次之后，Visual Basic 编辑器并不还原选择指点符，而是停留不动，准备该控件的下一次放置。完成多次放置后，单击“选定对象”按钮，以恢复鼠标指针，或者单击任何其他的工具箱按钮以放置那种类型的控件。

必要时，可以对标准大小控件的尺寸进行调整，如图 14.6 所示。只需选中该控件，然后单击并拖曳控件四周的某个选择“手柄”（白色小方格）即可。拖曳角上的“手柄”时，VBA 使控件按此角的两条边改变大小；拖曳一边的中点上的“手柄”时，控件只在一个方向上改变大小。在任一情况下，VBA 均显示虚线的轮廓线，以指明释放鼠标后，该控件会有多大。

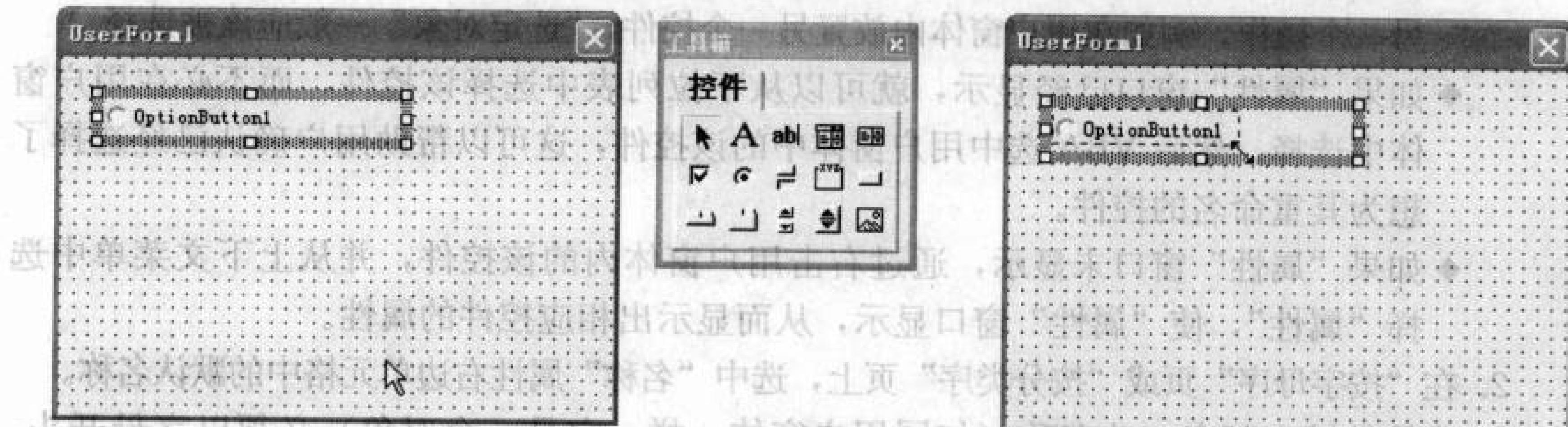


图 14.5 当在用户窗体内单击时，VBA 将标准大小的控件放置其中。如果“对齐控件到网格”复选框被中（默认为选中），VBA 自动使控件与窗体上网格对齐

图 14.6 放置控件后，必要时可通过拖曳选择“手柄”来改变控件的大小

也可以使用在将控件放入用户窗体时单击和拖曳的方法，生成自定义大小的控件（与放置好标准大小控件，再拖曳成所希望的大小的做法不一样）。但是，先放置标准大小控件，再在必要时改变其大小，这样还是比较容易的。

提示：要改变用户窗体自身的大小，需单击其标题栏（或窗体中没有控件的任何空白处）以选中它，然后单击和拖曳出现在它四周的选择手柄即可。

为了删除某一个控件，在用户窗体内右击它，再从上下文菜单中选择“删除”。也可以单击以选中它，然后按下“Delete”键，或选择“编辑”>“删除”。

为了删除多个控件，使用前段所述的方法，先选择它们，再删除它们。具体方法如下：

- ◆ 要选择多个相邻的控件，应先选中第一个控件，保持 Shift 键按下不放，再选中序列中的最后一个控件。
- ◆ 要选择多个不相邻的控件——或者是，使用 Shift 键选择了多个相邻的控件之后，还要选择其他控件——应先选中第一个控件，保持 Ctrl 键按下不放，再选中其余每个控件。
- ◆ 要选择用户窗体内同一区域中的多个控件，先在窗体内控件外面单击，以得到一个选择框，再拖曳此框使之至少包围每个控件的一部分。释放鼠标按键时，Visual Basic 编辑器便选择了这些控件。

重命名控件

和对待用户窗体一样，VBA 给了添加进来的每个控件一个默认名称，这个名称由控件类型和对应该类型的顺序号组成。当在用户窗体内生成第一个文本框时，VBA 给它命名为 TextBox1；再生成另一个文本框时，VBA 给它命名为 TextBox2；等等。对话框中的每个控件都必须有唯一的名称以便于在代码中引用。

为了便于记住控件的功能，通常希望把控件的默认名称更改成能描述其功能的名称。例如，如果 TextBox2 是用来输入用户所在机构的名称的，就希望把它重命名为 txtOrganizationName、txtOrgName、txtO_Name 或类似的名称。

要给某一控件重命名，需遵循如下步骤：

1. 单击用户窗体内的该控件以选中它，并在“属性”窗口内显示它的属性。
 - ◆ 选择控件时，确认工具箱中的“选定对象”按钮已被选择。（除非正在工具箱中执行另一个操作。例如在用户窗体内放置另一个控件，“选定对象”一般应该被选择。）
 - ◆ 如果“属性”窗口已经显示，就可以从下拉列表中选择该控件，而不必在用户窗体中选择。然后 VBA 选中用户窗体中的该控件，这可以帮助用户确认已经选择了想为其重命名的控件。
 - ◆ 如果“属性”窗口未显示，通过右击用户窗体内的该控件，并从上下文菜单中选择“属性”，使“属性”窗口显示，从而显示出相应控件的属性。
2. 在“按字母序”页或“按分类序”页上，选中“名称”属性右边单元格中的默认名称。
3. 为控件键入新名。此名称（如同用户窗体一样，它是一个对象）必须以字母开头，可以包含字母、数字和下划线（但不能有空格和符号），最长可以为 40 个字符。
4. 按下 Enter 键，或在“属性”窗口内或用户窗体内其他地方单击，以设定控件名。

注意：今后任何时候都可以为该控件重新再命名。但是，如果要这样做，还必须更改驱动用户窗体的代码中对该控件的引用。这是为了提醒编程人员，在编写代码前要为自己的控件选择适合的名称。

移动控件

为了移动当前尚未选择的某一控件，先单击控件内的任意地方以选中它，再把它拖曳到希望它出现的地方，如图 14.7 所示。

为了移动一个已选择的控件，需使鼠标指针在包围它的选择边框上方游移，让鼠标指针变成一个带小方框的箭头（如图 14.8 所示），然后再单击该控件，并把它拖曳到希望它出现的地方。

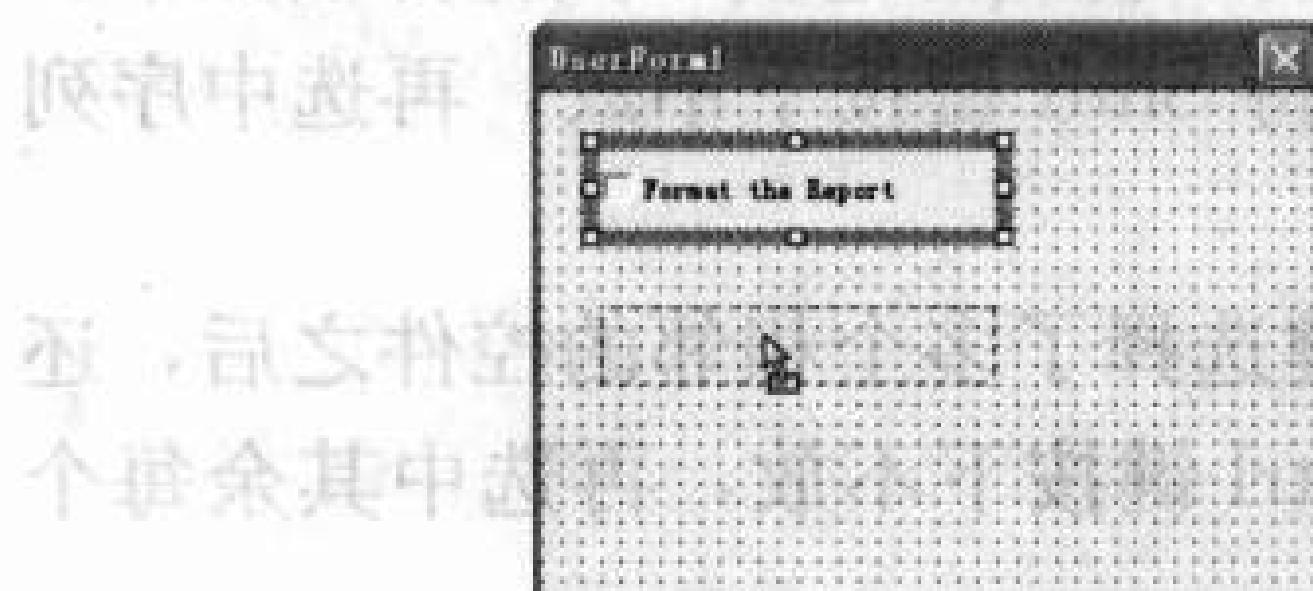


图 14.7 如果某一控件当前尚未被选择，可以通过单击和拖曳来移动它

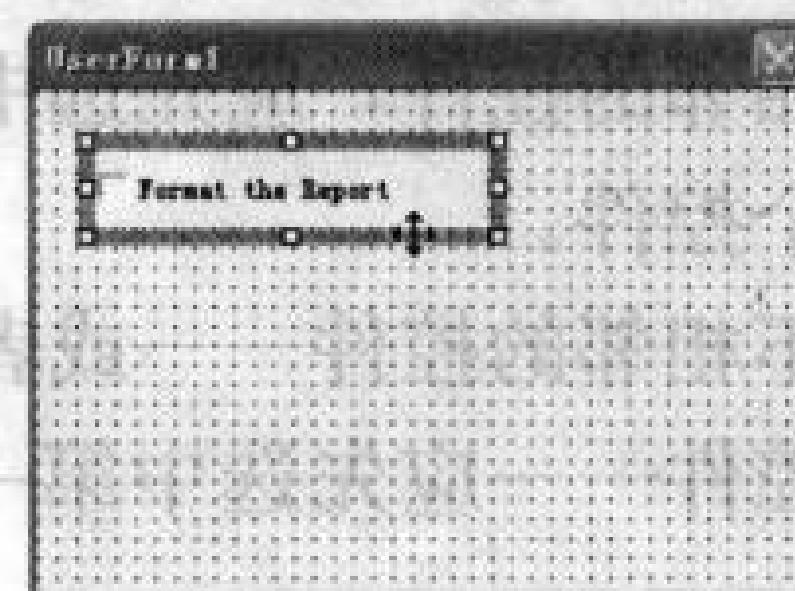


图 14.8 如果控件已被选择，使鼠标指针在其选择边框上方游移，然后单击并拖曳该控件

注意：可以使用“剪切”和“粘贴”命令（这些命令来在“标准”工具栏、“编辑”菜单、上下文菜单或键盘上）来移动控件，但这不是个很好的方法：“粘贴”命令总是将控件放置在用户窗体或容器（例如框架）的正中间，所以还不得不将它拖曳到它的新位置上去。

复制和粘贴控件

可以使用“复制”和“粘贴”命令来复制和粘贴已经添加到用户窗体的控件。可以把它粘贴到同一用户窗体或另一用户窗体中。“粘贴”命令把控件的复印件留在用户窗体或容器的正中间，用户必须把它拖曳到想放置它的地方。与使用“剪切”和“粘贴”命令来移动已有控件相比，使用“复制”和“粘贴”命令来生成新控件的优点是：新控件具备了原控件的特性，所以通过生成一个控件，设置其属性，然后克隆它来节省时间。

下面需要做的是把每个克隆好的复印件移动到合适的位置，把 VBA 给它取的默认名称更改成更具说明性和更易记忆的名称，设置应有别于该控件的“同胞”的属性。对于粘贴到另一用户窗体的复印件，甚至都不必为它改名——它们需要针对与其一道工作的代码来取适合的名字。

提示：应该把经常使用的自定义控件的复印件添加进工具箱。从工具箱里，可以迅速把该控件的多个复印件添加进用户窗体中。进入窗体后，需要为第一个复印件后面的每个复印件取名。但是，除此之外，在工具箱里对控件复印件设置的其他所有属性保持不变。

如果需要为相同类型的每个控件分别设置所有属性，那么你很可能会发现，使用工具箱按钮而不是“复制”和“粘贴”命令，能够更快地插入新控件。

作为使用“复制”和“粘贴”命令的替代方法，可以在单击和拖曳某一控件的同时按下 Ctrl 键来复制该控件。VBA 显示一个十号附着到鼠标指针，如图 14.9 所示，以指明正在复制控件，而不是移动控件。将此复印件放在用户窗体上希望它出现的地方。

更改控件上的标签

当控件有一个已显示的标签时，可用如下方法更改此标签：

1. 单击控件以选中它。
2. 单击标签以选中此标签。VBA 显示淡淡的虚线边框来包围这个标签，如图 14.10 所示。

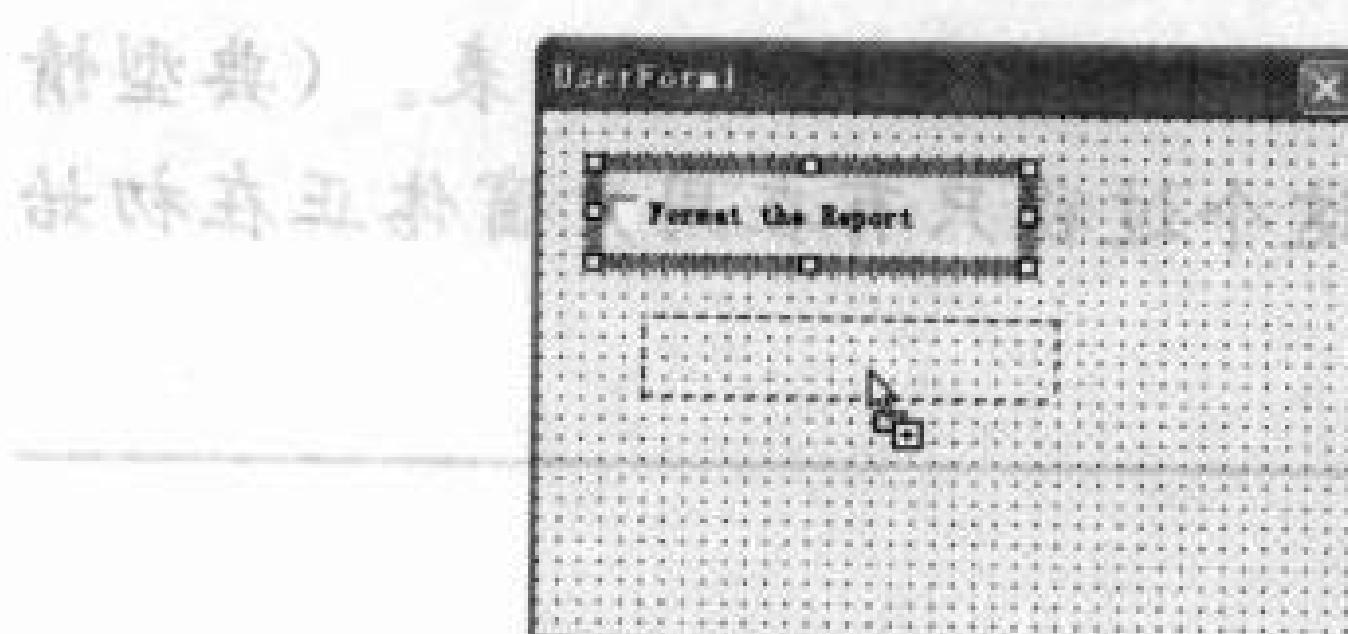


图 14.9 当使用 Ctrl 键和拖曳的方法
复制控件时，VBA 显示一个
十号，附着到鼠标指针

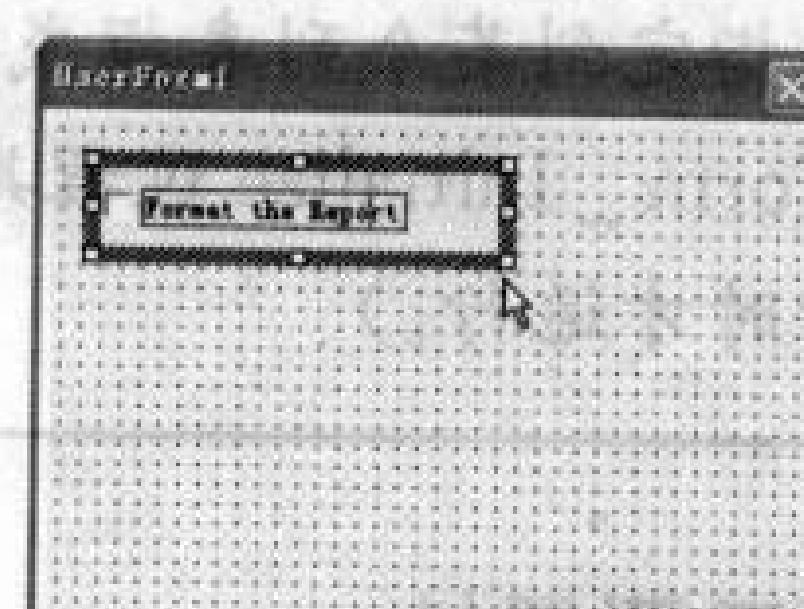


图 14.10 为了更改控件上的标签，选中
该控件，然后在标签内单击，
使淡淡的虚线边框显示出来

警告：当单击标签以选中它，并再次单击以定位插入点时，要确认这两次单击和所谓的双击是有区别的。一次双击将显示与用户窗体对应的代码表，并添加一个对应于标签的 Click 事件的过程。如果此事发生，按下 Shift+F7 键，或选择“视图”>“对象”，以再次观察窗体。然后从步骤 1 重新开始。

3. 在标签内单击，从而定位插入点，以便对标签进行编辑，或从整个标签上拖曳以全选它。
4. 按照自己愿望编辑标签的文本。
5. 按下 Enter 键，或在用户窗体内其他地方单击，使标签更改生效。

提示：也可以通过更改“属性”窗口中标签的“Caption”属性来更改该标签。

什么时候应该为控件设置属性

可以在设计阶段或运行阶段为一个控件设置很多属性。每个属性的设置，都要有一个时间和一个位置。是在设计时设置，还是运行时设置，就看哪个时间设置更合理。

一般来说，总是很想在设计时为那些不会变更的属性进行设置。有些属性，例如用户窗体的“名称”属性，必须在设计时设置——不会在运行用户窗体时才设置此类属性。通常也希望在设计时为控件命名，虽然在运行时可以添加控件，并设置它们的“名称”属性。

在大多数情况下，希望在设计时对那些控制用户窗体本身及其控件的位置和大小的属性进行设置。优点是明显的：可以确认用户窗体看上去与事先的打算相符合，这样比较清楚明了，等等。

也有这种情况，可能要在运行时调整用户窗体的属性或者窗体内某些控件的大小或位置。例如，可能需要向窗体内添加一些选项按钮以处理在窗体基本设计时未考虑到的不测事件。还有，可能创建了一个窗体，它的两个选项按钮组共享相同的空间——实际上是一个组定位在另一个组的上面。运行时才能确定，哪个选项按钮组是需要的，那就让这个组成为可视的，而把另一个组隐藏起来。如果每个组包含相同数目的选项按钮，那就要靠一个组的选项按钮来对付，在运行时为每个按钮指定适当的属性。

在控件的很多属性提供了灵活性的情况下，依靠运行时显示和隐藏不同的控件组，而不是依靠运行时添加或删除控件，常常可以把用户窗体设计成能够处理几种环境。在设计时为一个用户窗体生成完善的控件组，可以避免大多数因在运行时增添附加控件而造成的困难。有时候，可能需要临时创建一个用户窗体，以显示关于用户所处情况的信息。

只要继续在工作中使用控件，就免不了要在运行时为某些控件设置信息。例如，不能在设计时将项目表指定到某个列表框或组合框，必须在运行时才能指定这个项目表。（典型情况是，在 UserForm _Initialize 过程期间指定项目表，而这个过程只有在用户窗体正在初始化以便进行显示时才运行。）

工具箱控件的重要属性

本节将讨论默认工具箱内各控件的重要属性。

首先，要说明用来有效地操控大多数控件的共有属性。随后，把所有控件逐个研究一番，列出每个控件独有的属性。

(注意: 如果是 VBA 新手, 并且发现本节比较难读的话, 可以暂时跳过它, 待到需要引用关于控件属性的信息时, 再回过来读。

共有属性

表 14.1 列出了为所有或大多数控件共有的属性, 它们按种类分组。

表 14.1 大多数或所有控件共有的属性

属性	应用于	说明
信息		
BoundValue	除框架、图像、标签之外的所有控件	包含控件在用户窗体内接受焦点时该控件的值
HelpContextID	除图像、标签之外的所有控件	返回与控件相关联的“帮助”文件主题的上下文标识符
Name	所有控件	包含控件的名称
Object	所有控件	使人可以将与标准属性或方法同名的自定义属性或方法指定给控件
Parent	所有控件	返回包含该控件的用户窗体的名称
Tag	所有控件	用于将附加信息指定给控件
Value	复选框、组合框、命令按钮、列表框、多页、选项按钮、滚动条、微调按钮、TabStrip、文本框、切换按钮	最富变化的属性之一, Value 指定控件的状态与值。对于复选框、选项按钮、切换按钮, 它可为整数值 1 (True), 表示该项被选中, 为 0 (False) 则表示该项被清除。对于滚动条和微调按钮, 返回包含控件当前值的一个 Value。对于组合框、列表框, 返回当前选中的一行 (或多行) 的 BoundColumn 值。对于多页, 返回指明活动页的一个整数。对于文本框, 返回文本框中的文本。命令按钮的 Value 总是 False, 因为选择命令按钮就会触发 Click 事件。但是, 可以将命令按钮的值设置为 True, 这与单击它有相同的效果
大小和位置		
Height	所有控件	控件的高度, 以点 (Point) 计
LayoutEffect	除图像之外的所有控件	指明在窗体的布局改变时控件是否移动
Left	所有控件	控件左边框与包含该控件的窗体或框架左边之间的距离, 以像素 (pixel) 计
OldHeight	所有控件	控件原先的高度, 以像素计
OldLeft	所有控件	控件左边框的原先位置, 以像素计
OldTop	所有控件	控件顶边的原先位置, 以像素计
OldWidth	所有控件	控件原先的宽度, 以点计
Top	所有控件	控件顶边与包含该控件的窗体或框架顶边之间的距离, 以像素计
Width	所有控件	控件的宽度, 以点计
外观		
Alignment	复选框、选项按钮、切换按钮	指明控件与其标题如何对齐位置
AutoSize	复选框、组合框、命令按钮、图像、标签、选项按钮、文本框、切换按钮	一个布尔型属性, 它控制对象是否自动地调整自身的大小以容纳其内容。默认值为 False, 它表示控件不自动地调整自身的大小

(续表)

属性	应用于	说明
BackColor	所有控件	控件的背景色。这一属性包含代表颜色的一个数
BackStyle	复选框, 组合框, 命令按钮, 框架, 图像, 标签, 选项按钮, 文本框, 切换按钮	指明对象的背景是透明的 (fmBackStyleTransparent) 还是不透明的 (fmBackStyleOpaque, 默认值)。透过透明的控件, 可以看到窗体中控件背后的内容。可以使用透明控件达到有趣的效果——例如, 把一个透明的命令按钮放在图像或另一个控件的上方
BorderColor	组合框, 图像, 标签, 文本框, 列表框	指定控件边框的颜色。可以从“系统”下拉列表或调色板中选择边框的颜色, 或者给 BorderColor 输入一个 8 位数的整数值 (如 16711680 对应中蓝色)。VBA 把“BorderColor”属性保存为一个 16 进制数值 (为 00FF0000)。为使 BorderColor 起作用, BorderStyle 必须设置为 fmBorderStyleSingle
BorderStyle	组合框, 框架, 图像, 标签, 列表框, 文本框, 用户窗体	指定控件或用户窗体的边框的样式。BorderStyle 与 BorderColor 属性同时使用以便设置边框的颜色
Caption	复选框, 命令按钮, 标签, 选项按钮, 切换按钮	一个文本字符串, 包含对控件的说明——该文本可以在标签、命令按钮或切换按钮、复选框或选项按钮旁边出现
Font (对象)	除图像、微调按钮、滚动条之外的所有控件	是一个对象而不是属性——控制对象的标签以何种字体来显示。对于文本框、组合框和列表框, 它控制控件中的文本以何种字体来显示
ForeColor	除图像之外的所有控件	控件 (常常是控件上的文本) 的前景色。这一属性包含代表颜色的一个数
Locked	复选框, 组合框, 命令按钮, 列表框, 选项按钮, 文本框, 切换按钮	一个布尔型属性, 指明用户是否能修改该控件。如果设置为 True, 用户不能修改该控件, 虽然该控件仍能接受焦点 (即被选择) 和触发事件。如果设置为 False (默认值), 该控件打开以便编辑
MouseIcon	除多页之外的所有控件	指定用户使鼠标指针在控件上方游移时要显示的图像。为了使用 MouseIcon 属性, MousePointer 属性必须设置为 99, fmMousePointerCustom
MousePointer	除多页之外的所有控件	指定用户使鼠标指针在控件上方游移时要显示的鼠标指针的类型
Picture	复选框, 命令按钮, 框架, 图像, 标签, 选项按钮, Page, 切换按钮, 用户窗体	指定在控件上要显示的图片。使用 Picture 属性, 可以将一个图片添加到以文本为基础的控件上, 如添加到命令按钮上
PicturePosition	复选框, 命令按钮, 标签, 选项按钮, 切换按钮	指明图片与其标题如何对齐
SpecialEffect	复选框, 组合框, 框架, 图像, 标签, 列表框, 选项按钮, 文本框, 切换按钮	指明控件要使用的可视效果。对于复选框、选项按钮、切换按钮, 可视效果可以是平面的, 即没有特殊效果 (fmButtonEffectFlat), 或“下沉”的 (fmButtonEffectSunken)。对于其他控件, 可视效果可以是平面的 (fmSpecialEffectFlat)、“上升”的 (fmSpecialEffectRaised)、“下沉”的 (fmSpecialEffectSunken)、“带有雕刻”的 (fmSpecialEffectEtched) 或“带有隆起”的 (fmSpecialEffectBump)
Visible	所有控件	指明该控件是否为可视的, 以布尔型值来表示

(表 1)

(续表)

属性	应用于	说明
WordWrap	复选框, 命令按钮, 标签, 选项按钮, 文本框, 切换按钮	一个布尔型属性, 它指明控件内或控件上的文本在行末是否自动换行。对于大多数控件, WordWrap 默认设置为 True; 但常常希望把这一属性改设为 False 以避免文本不恰当地自动换行。如果控件是文本框, 而它的 MultiLine 属性已设置为 True 的话, VBA 忽略 WordWrap 属性
行为		
Accelerator	复选框, 命令按钮, 标签, 选项按钮, Page, Tab, 切换按钮	控件的加速键, 按下加速键 (常与 Alt 组合起来使用) 可访问到控件。例如, 在很多对话框中, 按下 Alt+C 键可访问到“取消”按钮。对应于标签的加速键, 作用于 Tab 顺序中下一个控件, 而不是标签自身
ControlSource	复选框, 组合框, 列表框, 选项按钮, 滚动条, 微调按钮, 文本框, 切换按钮	用来设置或保存控件的 Value 的单元格或字段。默认值是一个空字符串 (" "), 说明该控件没有控件源
ControlTipText	所有控件	用户使鼠标指针保持在控件上方时显示的屏幕提示文本。ControlTipText 的默认值是一个空字符串, 它指明不显示屏幕提示
Enabled	所有控件	一个布尔型值, 它控制该控件能否被访问 (以互动方式或程序方式)
TabIndex	除图像之外的所有控件	控件在用户窗体内 Tab 顺序中的位置, 用一个整数来表示, 整数的取值范围是从 0 (第一个位置) 到用户窗体内的控件数
TabStop	除图像、标签之外的所有控件	一个布尔型值, 它确定用户是否能通过按下 Tab 键来选择该控件。如果将 TabStop 设置为 False, 用户只能使用鼠标来选择该控件。TabStop 的设置不改变对话框的 Tab 顺序

标签

标签控件只是在屏幕上显示文本。使用定位属性来放置标签, 使用“Caption”属性来指定希望标签显示的文本。使用如表 14.2 所示的“ TextAlign”属性来对齐标签文本与标签控件的边框。

表 14.2 标签控件的“ TextAlign”属性值

fmTextAlign 常量	数值	文本对齐
fmTextAlignLeft	1	文本与控件的左边框对齐
fmTextAlignCenter	2	文本居于控件区正中间
fmTextAlignRight	3	文本与控件的右边框对齐

文本框

表 14.3 列出文本框控件的重要属性。

表 14.3 文本框控件的重要属性

属性	说明
AutoTab	一个布尔型属性, 它确定, 当用户已经在文本框或组合框内输入了最大允许数目的字符时, VBA 能否自动移到下一字段 (按下 Tab 键可人工地移到下一字段)

(表头)

(续表)

属性	说明	朗读	于限制	封底
AutoWordSelect	一个布尔型属性。它确定,当用户拖曳鼠标通过文本框或组合框内的文本时, VBA 能否自动选择一个完整的字(完整的单词)。			
DragBehavior	能否允许在文本框或组合框内实施拖放功能: fmDragBehaviorDisabled (0) 不允许拖放; fmDragBehaviorEnabled (1) 允许拖放			
EnterFieldBehavior	它确定,当用户将焦点移到文本框或组合框时, VBA 能否选择文本框或组合框中编辑区内的内容: fmEnterFieldBehaviorSelectAll (0) 能选择文本框或组合框当前行的内容; fmEnterFieldBehaviorRecallSelection (1) 不能改变原先的选择			
EnterKeyBehavior	一个布尔型属性。它确定,当用户按下 Enter 键的同时焦点在文本框时, VBA 做什么。如果 EnterKeyBehavior 为 True, 当用户按下 Enter 键时, VBA 生成一个新行; 如果 EnterKeyBehavior 为 False, VBA 移动焦点到用户窗体上的下一个控件。如果 MultiLine 为 False, VBA 忽略 EnterKeyBehavior 的设置			
HideSelection	一个布尔型属性。它确定 VBA 是否显示文本框或组合框内任何被选择的文本。如果 HideSelection 为 True, 当控件没有焦点时, VBA 显示文本时不对选定文本突出显示。如果 HideSelection 为 False, 无论控件有没有焦点, VBA 总是使选定文本突出显示			
IMEMode	确定输入法编辑器(IME)默认的运行时间模式。这一属性只用于远东地区的应用程序			
IntegralHeight	一个布尔型属性。它确定,如果列表太高,在当前高度内放不下,列表框或文本框是否要在垂直方向上调整自身高度,以便能够显示全部文本行。如为 True 则要调整,为 False 则不调整			
MultiLine	一个布尔型属性。它确定,文本框是否能包含多行文本(True),还是只有一行(False)。如果 MultiLine 为 True, 当内容太多而当前文本框又放不下时, 文本框会添加一个垂直滚动条			
PasswordChar	指明用来代替在文本框中输入的文本的占位符字符。为 PasswordChar 指定了字符时, VBA 显示的是占位符字符,而不是实际键入文本框的字母。这一属性通常用于输入需要保密、不让人读的密码和其他信息			
ScrollBars	指明在文本框中显示哪些滚动条。通常,最好把 WordWrap 属性设置为 True, 让 VBA 在需要时向文本框添加垂直滚动条,而不使用 ScrollBars 属性			
SelectionMargin	一个布尔型属性。它确定,用户能否通过在文本行左边的选择条内单击来选择文本框或组合框中的一行文本			
ShowDropDownWhen	它确定,什么时候显示组合框或文本框的下拉按钮。fmShowDropDownWhenNever (0) 是任何情况下都不显示下拉按钮,对文本框来说,它是默认值。fmShowDropDownWhenFocus (1) 是在文本框或组合框有焦点时显示下拉按钮。fmShowDropDownWhenAlways (2) 是总要显示下拉按钮,对组合框来说,它是默认值			
TabKeyBehavior	一个布尔型属性。它指明用户能否在文本框中输入制表符。如果 TabKeyBehavior 为 True 并且 MultiLine 为 True, 按下 Tab 键,就在文本框中输入了一个制表符。如果 MultiLine 为 False, VBA 忽略 TabKeyBehavior 的 True 设置。如果 TabKeyBehavior 为 False, 按下 Tab 键,会使焦点移到 Tab 顺序中的下一个控件			

组合框和列表框

表 14.4 列出组合框控件和列表框控件的重要属性。这两个控件很相似,它们有很多共有的属性。(组合框和列表框的许多属性是相同的,但它们也有自己的独特属性)

表 14.4 组合框控件和列表框控件的重要属性

属性	说明
AutoTab	见表 14.3
AutoWordSelect	见表 14.3
BoundColumn	一个不定型属性。它确定多列组合框或多列列表框中的数据的来源。默认设置是 1 (第一列)。为了指定另一列, 需指明列的编号 (列从 1 开始编号, 第一列即最左边那一列)。要将 ListIndex 的值指定到 BoundColumn, 可使用 0
ColumnCount	一个长整型属性。它设置或返回在组合框或列表框中显示的列数。如果数据源是未绑定的, 最多可以指定 10 列。为了显示数据源中所有的可用列, 将 ColumnCount 设置为 -1
ColumnHeads	一个布尔型属性。它确定组合框或列表框是否要显示列的标题。设置为 True, 则显示; 为 False, 则不显示
ColumnWidths	一个字符串属性。它设置或返回多列组合框或列表框中每列的宽度
ListRows	(只有组合框有。) 是一个长整型属性。它设置或返回组合框中显示的行数。如果列表中的项数大于 ListRows 的值, 组合框会显示一个滚动条, 以便用户能够使用滚动方式看到此前看不到的项目
ListStyle	确定列表使用的可视效果。对于组合框和列表框, fmListStylePlain 显示常规的、不加装饰的列表。对于组合框, fmListStyleOption 在每个条目左边显示一个选项按钮, 以允许用户从表中选择一个项目。对于列表框, fmListStyleOption 为单选列表显示选项按钮, 为多选列表显示复选框
ListWidth	(只有组合框有。) 是一个不定型属性。它设置或返回组合框中列表的宽度。默认值为 0, 它使该列表与组合框的文本区有相同的宽度
MatchEntry	确定在用户键入字符且焦点在组合框或列表框时, 组合框或列表框使用的匹配形式。fmMatchEntryFirstLetter (0) 与以键入的字母或字符开头的下一条目进行匹配: 如果用户键入两次 t, VBA 选择以 t 开头的第一个条目, 然后选择以 t 开头的第二个条目。fmMatchEntryComplete (1) 与用户键入的每个字母进行匹配: 如果用户键入 te, VBA 选择以 te 开头的条目。fmMatchEntryNone (2) 指明不进行匹配: 用户不能通过在列表框或组合框中键入来选择一个项目, 必须要使用鼠标或箭头键以进行选择。对于组合框, 默认的 MatchEntry 设置是 fmMatchEntryComplete。对于列表框, 默认设置是 fmMatchEntryFirstLetter
MatchRequired	(只有组合框有。) 是一个布尔型属性。它确定用户在退出该控件之前, 是否必须从组合框中选择一个条目。设置为 True, 则是; 设置为 False, 则否。这一属性, 对于确认用户没有把不完全的条目键入组合框的文本框区, 但是忘记了去完成下拉列表区中的选择, 是很有用的。如果 MatchRequired 为 True, 同时用户试图不进行选择就退出组合框, VBA 就会显示出无效的属性值消息框
MultiSelect	(只有列表框有。) 它控制用户能在列表中做单项选择还是做多项选择。fmMultiSelectSingle (0) 让用户只选择一个项目。fmMultiSelectMulti (1) 让用户通过单击并用鼠标, 或通过按下空格键来选择多个项目。fmMultiSelectExtended (2) 让用户使用 Shift + 单击、Ctrl + 单击和 Shift 配合箭头键来扩展或缩减其选定范围
RowSource	一个字符串属性。它指定组合框或列表框的列表的来源
SelectionMargin	见表 14.3
ShowDropButtonWhen	见表 14.3

复选框

复选框控件的大多数属性已经讨论过了, 但是还有一个没有介绍过的重要属性是 Tri-

pleState，这个属性对选项按钮和切换按钮控件也起作用。

TripleState 是一个布尔型属性。它确定复选框、选项按钮或切换按钮，除了 True 和 False 状态之外，是否还可以有 Null 状态。当复选框处于 Null 状态时，它的框能够出现并被选择，但是为灰色。例如，在 Word 中的“字体”对话框中，当一个受复选框控制的属性——如图 14.11 中的“阴影”复选框——只作用到当前选定范围的一部分，而不是作用到整个选定范围时，可以看到这种效果。

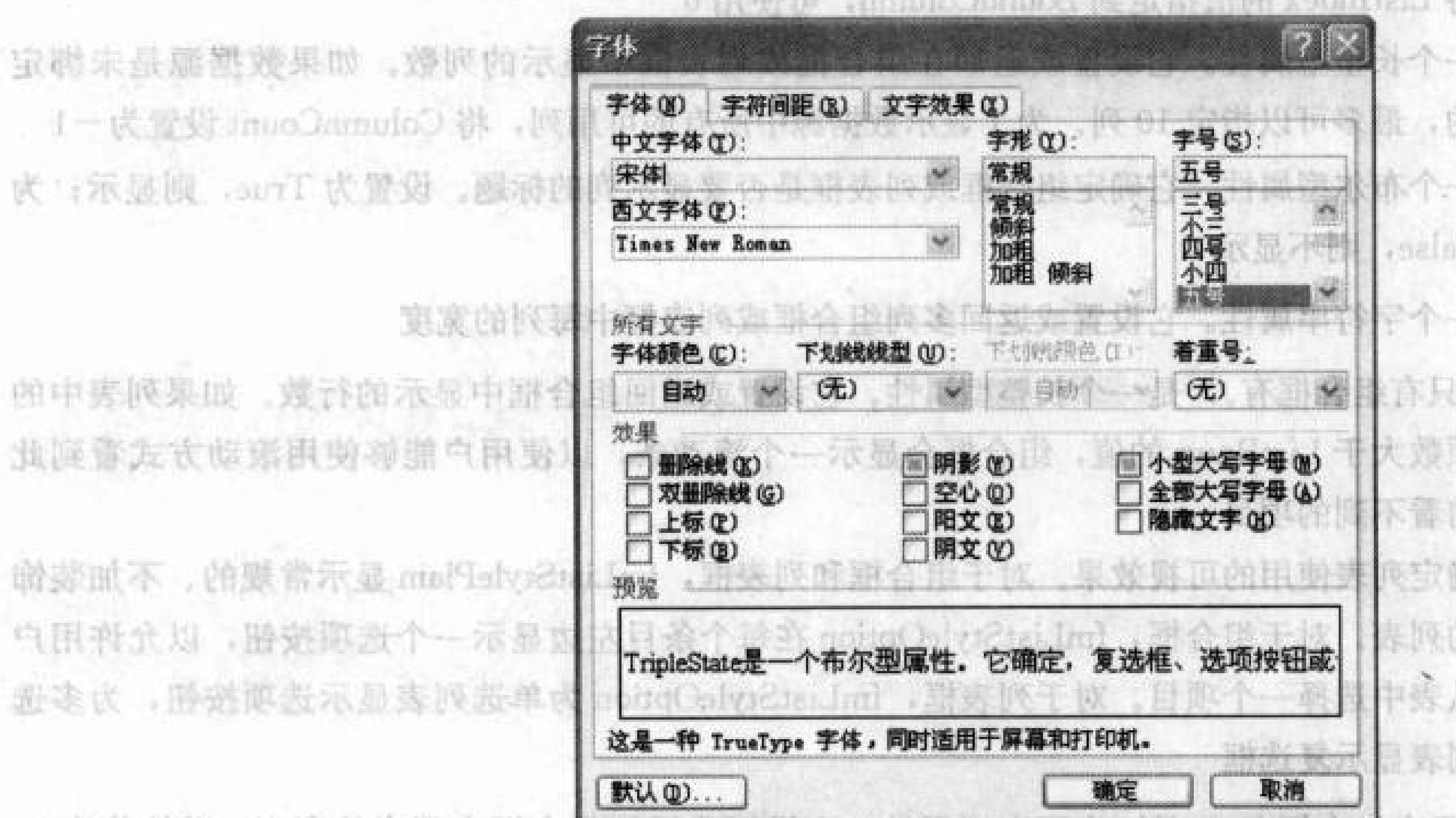


图 14.11 把复选框的 TripleState 属性设置为 True，可以显示 Null 状态中的复选框。当复选框只作用到当前选定范围的一部分而不是全部时，Word 的“字体”对话框显示出 Null 状态中的复选框（带有绿色小方块）

在其他控件中简述过的某些属性，这里要详述一下：

- ◆ SpecialEffect 属性控制复选框的可视外观。默认值为 fmButtonEffectSunken (2)，它显示下沉框。也可以选择 fmButtonEffectFlat (0) 以显示带有平面效果的框。图 14.12 显示了下沉复选框和平面复选框。

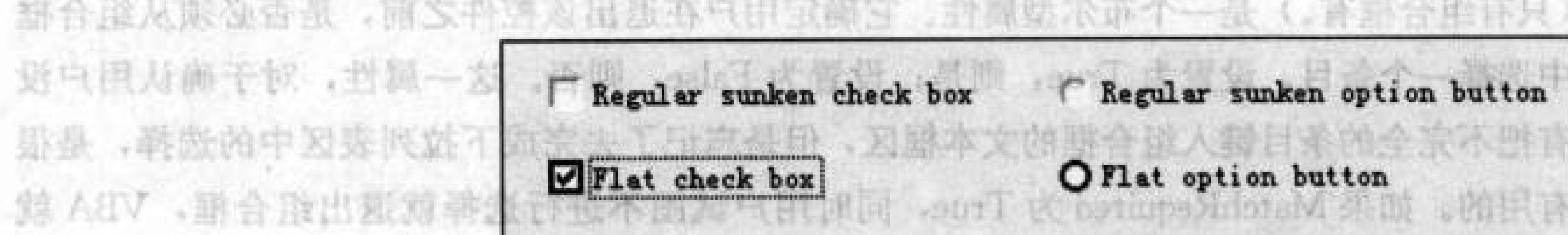


图 14.12 使用 SpecialEffect 属性来显示平面复选框或选项按钮，而不是正常的下沉复选框或选项按钮

- ◆ Value 属性，它指明复选框是被选中 (True)，还是被清除 (False)，它是复选框的默认属性。下面的三条语句有相同的效果：

```
If CheckBox1.Value = True Then  
If CheckBox1 = True Then  
If CheckBox1 Then
```

- ◆ Accelerator 属性提供对复选框的快速访问。给复选框指定一个唯一的加速键，以便用户能够从键盘上对复选框的接通和断开进行快速切换。

选项按钮

和复选框类似，选项按钮控件有一组比较简洁的属性，几乎所有这些属性在本章中都已见到过。本节介绍 `GroupName` 属性，它是选项按钮独有的。另外，还要介绍一些选项按钮工作时用到的重要属性。

`GroupName` 属性是一个字符串属性。它将选项按钮指定到一个选项按钮组。`GroupName` 的默认设置是一个空字符串 (" ")，这意味着在对选项按钮进行显式指定之前，没有把选项按钮指定到一个组。当输入组名时，便建立了这个组。使用 `GroupName` 属性，可以在同一窗体上得到多组选项按钮，不必使用多个框架来分开多个组。但是，必须把选项按钮的逻辑组合相互区分清楚，以便用户能够立即讲出哪些选项按钮组成一个组。实际上，框架往往为从视觉和逻辑上分隔各选项按钮组，提供出简易方式——但是，`GroupName` 提供的灵活性还是很有用的。

下面是选项按钮控件的其他重要属性：

- ◆ `Value` 属性，它指明选项按钮是被选中 (True)，还是被清除 (False)，它是选项按钮的默认属性。根据情况把选项按钮对象或它的 `Value` 设置为 True 或 False，可以设置或返回选项按钮的状态。把一个选项按钮的 `Value` 设置为 True，也就是把同一组或同一框架内的所有其他选项按钮控件设置成了 False。
- ◆ `Accelerator` 属性，它提供对选项按钮的快速访问。给每个选项按钮指定一个唯一的加速键，以便用户能够从键盘上对选项按钮的接通和断开进行切换。
- ◆ `SpecialEffect` 属性，它控制选项按钮的可视外观。`fmButtonEffectSunken` (2) 是默认值，显示下沉式选项按钮。`fmButtonEffectFlat` (0) 则显示平面按钮。图 14.12 显示了下沉选项按钮和平面选项按钮。
- ◆ `TriState` 属性（前节“复选框”中讨论过），生成一个三状态选项按钮：选中 (True)、清除 (False) 和 Null (被选择但变为灰色)。但是 `TriState` 属性不能生效，所以用户不能通过互动方式来设置 Null 状态，只能在需要时通过程序来设置它。

切换按钮

切换按钮控件生成一个切换按钮。它是一个按钮，未被选中时，外观呈“突起”状；被选中时，外观变为“推入”状。切换按钮控件的重要属性与复选框和命令按钮相同：

- ◆ `Value` 属性是切换按钮的默认属性。
- ◆ `TriState` 属性生成一个三状态切换按钮：选中 (True)、清除 (False) 和 Null (被选择但变为灰色)。用户可以通过单击方式，将三状态的切换按钮设置成它的 Null 状态。在 Null 状态中，切换按钮变为灰色。
- ◆ `Accelerator` 属性提供对切换按钮的快速访问。

框架

框架控件比较简单，但它有几个属性值得一谈，这些属性列在表 14.5 中。框架控件与 Page 对象共享其中一些属性。

表 14.5 框架控件的属性

属性	说明
Cycle	确定用户在离开框架中或 Page 上的最后一个控件时所应采取的行动。fmCycleAllForms (0) 将焦点移至用户窗体内或页面上的 Tab 顺序中的下一控件；fmCycleCurrentForm (2) 则保持焦点在框架内或页面上，直至焦点明显地移至另一框架内或另一页面上的控件。这一属性对 Page 对象也起作用
InsideHeight	是一个只读属性。它返回框架内区域的高度（以点计），但不包括显示的任何水平滚动条的高度。这一属性对 Page 对象也起作用
InsideWidth	是一个只读属性。它返回框架内区域的宽度（以点计），但不包括显示的任何垂直滚动条的宽度。这一属性对 Page 对象也起作用
KeepScrollBarsVisible	这个属性确定当用户不需要滚动条来导引框架或页面时，是否仍要框架或页面显示水平和垂直滚动条。fmScrollBarsNone (0) 是在不需要时不显示滚动条。fmScrollBarsHorizontal (1) 是一直要显示水平滚动条。fmScrollBarsVertical (2) 是一直要显示垂直滚动条。fmScrollBarsBoth (3) 是一直要显示水平滚动条和垂直滚动条。对于框架对象，fmScrollBarsNone 是默认的，而对于 Page 对象，fmScrollBarsBoth 是默认的。这一属性对 Page 对象也起作用
PictureTiling	是一个布尔型属性，它确定，是让显示在控件的图片在背景上平铺（True），以使它占据该控件覆盖的整个区域，还是不让图片在背景上平铺（False）。为了设置平铺模式，要使用 PictureAlignment 和 PictureSizeMode 属性。这一属性对 Page 对象和图像控件也起作用
PictureSizeMode	它确定如何显示背景图片。fmPictureSizeModeClip (0) 是默认设置，它要裁掉图片中比页面、框架或图像控件大的部分。使用这一设置，是希望以原始的大小和比例显示图片。fmPictureSizeModeStretch (1) 是在水平或垂直方向上扩展图片，使其填满页面、框架或图像控件。这一设置对彩色背景和装饰效果都是有利的，但是，对要求识别性好的图片可能会造成损害，它也对 PictureAlignment 属性的设置带来不好的影响。fmPictureSizeModeZoom (3) 使图片成比例放大，直到水平方向或垂直方向到达控件的边界，但是，它和扩展图片不一样，在某一方向到达控件边界时，另一方向也到达了它的最大值。这对放大图片尺寸而保留其比例是有利的，但是，必须对未达到最大值的那个方向进行调整，以去掉空白。这一属性对 Page 对象和图像控件也起作用
PictureAlignment	它确定图片的位置。fmPictureAlignmentTopLeft (0) 是把图片放在控件的左上角。fmPictureAlignmentTopRight (1) 是把图片放在控件的右上角。fmPictureAlignmentCenter (2) 是默认设置，把图片放在控件的正中间（水平方向和垂直方向都居中）。fmPictureAlignmentBottomLeft (3) 是把图片放在控件的左下角。fmPictureAlignmentBottomRight (4) 是把图片放在控件的右下角。这一属性对 Page 对象和图像控件也起作用

命令按钮

命令按钮控件有三个独特的属性，如表 14.6 所示。

表 14.6 命令按钮控件的独特属性

属性	说明
Cancel	是一个布尔型属性。它确定该命令按钮是（True）或不是（False）用户窗体上的“取消”按钮。用户窗体的“取消”按钮可以使用任何名称，能把它区别开来的是，它的 Cancel 属性设置为 True。要激活“取消”按钮，用户需按下 Esc 键，或单击该按钮，或将焦点置于该按钮并按下 Enter 键。在任何给定时间，窗体上只有一个命令按钮是“取消”按钮。将一个命令按钮的 Cancel 属性设置为 True，会使 VBA 将以前的 Cancel 属性设置为 True 的按钮的 Cancel 属性变为 False

(续表)

属性	说明
Default	是一个布尔型属性。它确定命令按钮是 (True) 或不是 (False) 用户窗体上的默认按钮。在任何给定时间，窗体上只有一个命令按钮是默认按钮。将一个命令按钮的 Default 属性设置为 True，会使 VBA 将以前的 Default 属性设置为 True 的按钮的 Default 属性变为 False。要激活默认按钮，用户需在任何其他命令按钮都不具有焦点时按下 Enter 键。
TakeFocusOnClick	是一个布尔型属性。它确定，当命令按钮被用户单击时，它是获得焦点 (True)，还是不获得焦点 (False)。这一属性的默认设置是 True，但是，当希望即使用户单击了该命令按钮，焦点仍保持停留在用户窗体内另一控件上时，用户可能会想把这一属性设置为 False。然而，如果用户使用 Tab 键或箭头键来移至某一命令按钮时，即使其 TakeFocusOnClick 属性设置为 False，该命令按钮仍将获得焦点。

提示：可以为窗体上默认命令按钮之外的每个命令按钮设置 Accelerator 属性，以便用户可以从键盘上快速访问这些命令按钮。

注意：窗体上的默认按钮也可以是“取消”按钮。这对于提供删除文本或删除某一文件之类的不可逆行动的窗体会带来明显的好处，但是，这也会搞乱可访问性辅助工具（如屏幕读出程序），并使认知能力差的用户在使用窗体工作时产生困难。基于这些理由，通常最好让窗体的默认按钮是“取消”按钮之外的另一个不同按钮。

TabStrip 和多页

TabStrip 控件有几个独特的属性，还有一些与多页控件共有的属性，表 14.7 列出了这些属性。

表 14.7 TabStrip 和多页控件的属性

属性	说明
ClientHeight	(只有 TabStrip 有。) 是一个单精度浮点型属性。它设置或返回标签条显示区域的高度，以点计。
ClientLeft	(只有 TabStrip 有。) 是一个单精度浮点型属性。它返回标签条的左边缘和其内的控件的左边框之间的距离，以点计。
ClientTop	(只有 TabStrip 有。) 是一个单精度浮点型属性。它返回标签条的顶边和其内的控件的顶边之间的距离，以点计。
ClientWidth	(只有 TabStrip 有。) 是一个单精度浮点型属性。它设置或返回标签条显示区域的宽度，以点计。
SelectedItem	它设置或返回在标签条中当前选中的标签，或在多页控件中当前选中的页面。
TabFixedHeight	是一个单精度浮点型属性，它设置或返回标签的固定的高度，以点计。如果 TabFixedHeight 设置为 0，则标签自动调节大小以容纳其内容。
TabFixedWidth	是一个单精度浮点型属性。它设置或返回标签的固定的宽度，以点计。如果 TabFixedWidth 设置为 0，则标签自动调节大小以容纳其内容。
TabOrientation	它确定标签条中或多页上标签的位置。fmTabOrientationTop (0) 是默认设置，使标签显示于标签条或多页的顶部。fmTabOrientationBottom (1) 使标签显示于标签条或多页的底部。fmTabOrientationLeft (2) 使标签显示于标签条或多页的左边。fmTabOrientationRight (3) 使标签显示于标签条或多页的右边。

(索引)

滚动条和微调按钮

滚动条和微调按钮共享一些属性，这些属性以前未碰到过。表 14.8 列出了这些属性。

表 14.8 滚动条和微调按钮控件的属性

属性	说明
Delay	是一个长整型属性。它设置，当用户多次单击和按住鼠标微调时控件记录的两次单击之间的延迟时间，以毫秒计。默认延迟为 50 毫秒。控件对第一次单击是立即记录的，第二次单击则在 Delay×5 之后记录（附加延迟是为了在仅单击一次时对用户有帮助），第三次和以后各次单击都在 Delay 之后记录
LargeChange	（只有滚动条有。）是一个长整型属性。它确定，当用户在滚动块和滚动条箭头之间的滚动条上单击时，能产生多大移动量。在设置了滚动条的 Max 和 Min 属性之后，才设置 LargeChange 属性
SmallChange	是一个长整型属性。它确定，当用户在滚动条或微调按钮的一个滚动箭头上单击时，有多大移动量产生。SmallChange 需要一个整数值，默认数值是 1
Max	是一个长整型属性。它为滚动条或微调按钮的 Value 属性指定最大值。Max 必须是一个整数，默认值是 1
Min	是一个长整型属性。它为滚动条或微调按钮的 Value 属性指定最小值。Min 必须是一个整数，默认值是 1
ProportionalThumb	（只有滚动条有。）是一个布尔型属性。它确定，滚动块是固定大小 (False)，还是与滚动区的大小成比例 (True)。这使用户对当前能看到多大滚动区有个大致的了解。默认设置是 True

图像

到现在为止，已经见到了图像控件的所有属性，对于用户窗体中使用的大多数图像控件，希望设置的有位置属性、大小属性以及下面这些属性：

- ◆ 使用 Picture 属性来指定希望出现在图像控件中的图片文件。在“属性”窗口内的 Picture 行中单击，然后再单击文本框显示的省略号按钮 (...)。在“加载图片”对话框中，选择图片并单击“确定”按钮以添加图片。图像控件可以显示 BMP、CUR (游标)、GIF、ICO (图标)、JPG 和 WMF 文件，但是没有图形文件，如 TIF 或 PCX。

提示：在图像控件内显示 Windows 屏幕的一部分的最容易方式是使用 PrintScreen 键 (捕获全屏) 或 Alt+PrintScreen 组合键 (捕获活动窗口) 来捕获它，然后把它粘贴进一个应用程序 (如 Paint)，必要时在那里修整它，再把它作为一个 BMP 文件保存起来。

- ◆ 使用 PictureAlignment 属性以确定图片的位置。
- ◆ 使用 PictureSizeMode 属性，确定图片是否要剪裁、扩展或放大，以与图像控件的大小相适应。必要时需调节图像控件的高度和宽度。
- ◆ 如果需要平铺图像以使其占据控件内全部空间的话，使用 PictureTiling 属性来实现。

Page

Page 对象是多页对象内包含的多个页面中的一个页面。在讨论其他控件的属性时，已

已经涉及了它的所有属性，只有 Index 属性除外，这是它和 Tab 对象共有的属性。

Index 属性是一个整型属性，它确定多页控件内 Page 集合中 Page 对象的位置，或 TabStrip 内 Tabs 集合中 Tab 对象的位置。第一个 Page 对象或 Tab 对象的编号为 0（零）；第二个 Page 或 Tab 对象的编号为 1，等等。可以改变某个标签或页面的 Index 属性，从而改变该标签或页面在集合中出现的位置。

Tab

Tab 对象是 TabStrip 对象内包含的多个标签中的一个标签。在讨论其他控件的属性时，已经涉及了它的所有属性。

使用控件组工作

把两个或多个控件组合起来，工作时把它们当作一个单元来使用，可以调节其大小，为其选定格式，或删除它们。

建立控件组

为了建立控件组，通过 Shift+单击、Ctrl+单击或使用拖曳来选择组成控件组的各控件，然后右击并从上下文菜单中选择“生成组”。也可以先选择各控件，再单击用户窗体工具栏上的“组”按钮（首先需要显示这个工具栏——它不是默认显示的），或选择“格式”>“生成组”。VBA 生成包含若干控件的一个新控件组，并把它们放入环绕整个组、阴影边框上带有若干“手柄”的方框内，如图 14.13 中的右图所示。



图 14.13 建立控件组，就可以和多个控件同时工作。VBA 通过放置环绕控件组的方框来指明一个控件组

解散控件组

为了解散控件组，右击组内包含的任何控件，并从上下文菜单中选择“取消组”。也可以先单击组内任何控件以选择控件组，再单击用户窗体工具栏上的“取消组”按钮，或选择“格式”>“取消组”。VBA 去掉带有若干“手柄”的阴影边框，显示一个边框和若干“手柄”来环绕原属控件组的各个独立的控件。

改变组内控件的大小

先选择控件组，再拖曳环绕边框上的调节“手柄”，可以迅速改变组内所有控件的大小。例如，可以选中右边中间的“手柄”，然后向内拖曳，以便缩短各控件，如图 14.14 所示。

各控件将随着控件组轮廓线的改变而成比例地改变其大小。

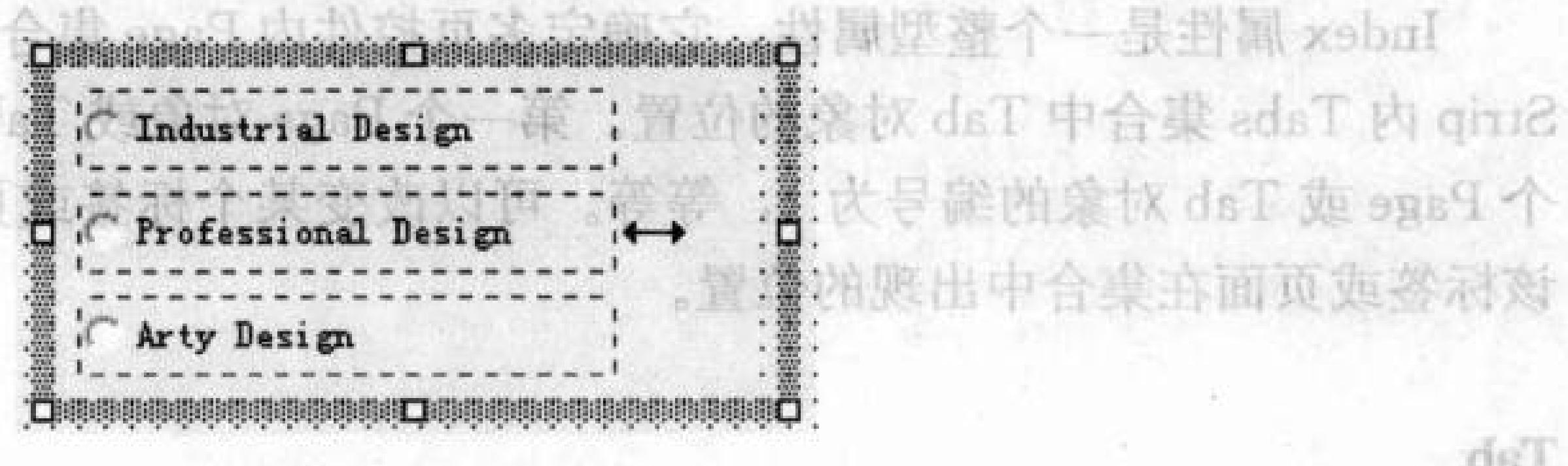


图 14.14 通过拖曳环绕边框上的调节“手柄”来改变组内所有控件的大小

一般来说，如图 14.14 那样，当相同类型的很多控件组合起来时，这一行动工作得最好。例如，改变由几个文本框或几个选项按钮组成的控件组的大小会工作得很好，而改变由一个文本框、一个命令按钮和一个组合框组成的控件组的大小，就很难说是个好主意。

删除控件组

右击控件组内的任意控件，并从上下文菜单中选择“删除”，或选择控件组并按下 Delete 键，可以迅速删除整个控件组。

使用控件组中的一个控件工作

即使已经把很多控件组合起来，必要时还是可以同各控件分别单独工作。为此，首先单击控件组内的任意控件以选择控件组，如图 14.15 中的左图所示。然后单击想与之一起工作的那个控件，如图 14.15 中的右图所示。VBA 显示一个环绕该控件组的深阴影边框（说明该控件组仍存在），并显示一个环绕那个单独的控件的淡阴影边框，说明该控件被选中。

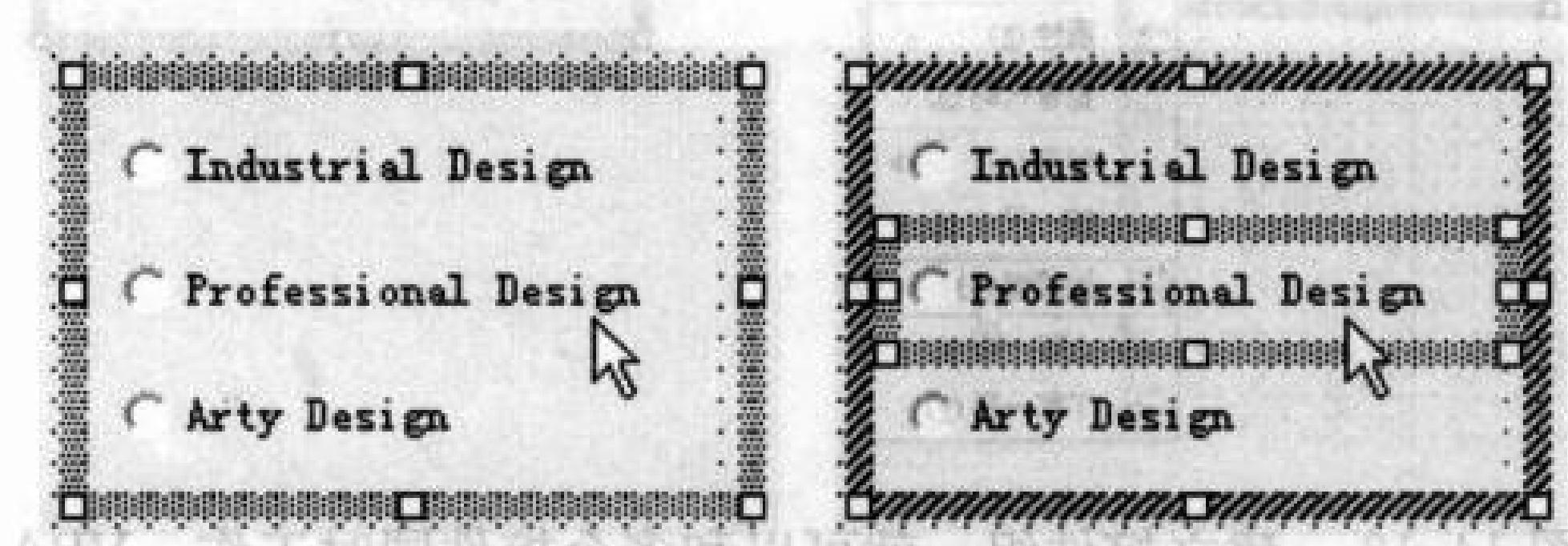


图 14.15 为了和组内的一个控件一道工作，先选择组（如左图所示），再选择控件（如右图所示）

这样，就可以和那个单独的控件一道工作了，如同这个控件没有参加控件组一样。工作结束时，单击控件组内的另一控件或单击用户窗体内的任何地方，即可去掉对那个单独控件的选择。

对齐控件

即使使用了“贴紧对齐到网格”特性，常常还需要人工对齐控件。对齐被选择控件的最容易办法是在多个控件的任意一个控件内右击，并从“对齐”子菜单中选择一个选项：左对齐、居中对齐、右对齐、顶端对齐、中间对齐、底端对齐或对齐到网格。这些选项的作用如下：

左对齐	各控件的左边对齐
居中对齐	各控件的水平中心点对齐
右对齐	各控件的右边对齐
顶端对齐	各控件的顶边对齐
中间对齐	各控件的垂直中心点对齐
底端对齐	各控件的底边对齐
对齐到网格	各控件对齐到网格

VBA 把各控件的边框或中心点与主控件的边框或中心点的当前位置对齐。主控件是四周有白色调节“手柄”，而不是黑色“手柄”的控件。选择了需要人工对齐的各控件后，把已经处于正确位置的控件作为主控件，方法是：单击它，从而使它拥有白色“手柄”。然后选择想要的对齐选项。

警告：要确认所选择的对齐选项适合于选择的各控件。如果告诉 VBA 按不恰当的方式去对齐控件，VBA 也会很乐意去做的。例如，如果选择了很多选项按钮或文本框，再从“对齐”子菜单里选择“顶端对齐”的话，VBA 将顺从地让各控件的顶端相互拥塞在一起，搞得很不好用。（使用“撤销”命令可以从这种小麻烦中脱身。）

放置控件

VBA 在“格式”菜单上提供了几个放置命令：

- ◆ 在“格式”>“统一尺寸”子菜单上，使用“宽度相同”、“高度相同”、“两者都相同”命令，使两个或多个控件一维尺寸相同或二维尺寸相同。
- ◆ 使用“格式”>“正好容纳”命令，让 VBA 以某个成分的标签的大小为根据，为该成分决定一个合适的大小。这种做法对于某些控件，例如带有中等长度标签的切换按钮比较合适，但是 VBA 将会把“确定”按钮缩小到小得不好用的程度。
- ◆ 使用“格式”>“调至网格”命令，增大或减小控件的尺寸，使其与最近的网格点对齐。
- ◆ 在“格式”>“水平间距”和“格式”>“垂直间距”子菜单上，使用“相同”、“递增”、“递减”、“移除”命令来设置两个或多个控件的水平间距和垂直间距。“移除”选项去掉两个控件之间的多余空间，它对于某些控件（例如垂直序列的选项按钮）比较合适（看起来很好地靠近在一起），但是对命令按钮却不适用（命令按钮之间需要留一些空间）。
- ◆ 在“格式”>“窗体内居中”子菜单上，使用“水平对齐”和“垂直对齐”命令，使一个控件或一个控件组居于窗体正中间。让各控件在垂直方向上居中往往并不好，常常希望让一个框架或一组命令按钮水平方向居中。
- ◆ 在“格式”>“排列按钮”子菜单上，使用“底端对齐”和“居右”命令以迅速重新排列对话框中的各个命令按钮。

调节对话框的 Tab 顺序

对话框中的框架内的各控件，借助于按下 Tab 键（前移）或 Shift+Tab 组合键（后移），使 VBA 按某种顺序选择各控件，这种顺序就是一个对话框的 Tab 顺序，或对话框内

的一个框架的 Tab 顺序。用户窗体中的每个框架都有各自的与它所包含的各控件对应的 Tab 顺序：框架按对应于对话框的 Tab 顺序出现，框架内的各控件则按对应于框架的 Tab 顺序出现。

为对话框或框架设置 Tab 顺序，应使之尽可能易于使用。一般来说，对于讲英语的用户，最好把对话框或框架的 Tab 顺序安排为从左到右和从上到下。有些国家的用户，希望把 Tab 顺序安排为从右到左。还有人希望把 Tab 顺序安排成从一个控件移到另一个有关的控件，而这个有关的控件并不是常规的 Tab 顺序中的下一个控件。

VBA 按照添加控件时的顺序，在先来先服务的基础上，将 Tab 顺序指定给对话框或框架中的各控件。但是，由于未必能做到以完善的顺序添加所有控件，这种顺序很难产生出最佳的对应于对话框的 Tab 顺序，所以往往要调节 Tab 顺序——至少要检查它是否正确。对于框架，由于只放少量控件，所以有更好的机会以适当的顺序来添加控件；但是，在将对话框交付使用之前，也要对框架中的 Tab 顺序进行检查。

为了更改对话框或框架中的 Tab 顺序，必须：

1. 在用户窗体或框架内的空白处右击，并从上下文菜单中选择“Tab 顺序”，以显示“Tab 键顺序”对话框，如图 14.16 所示（也可以选择用户窗体或框架，并选择“视图”>“Tab 键顺序”）。

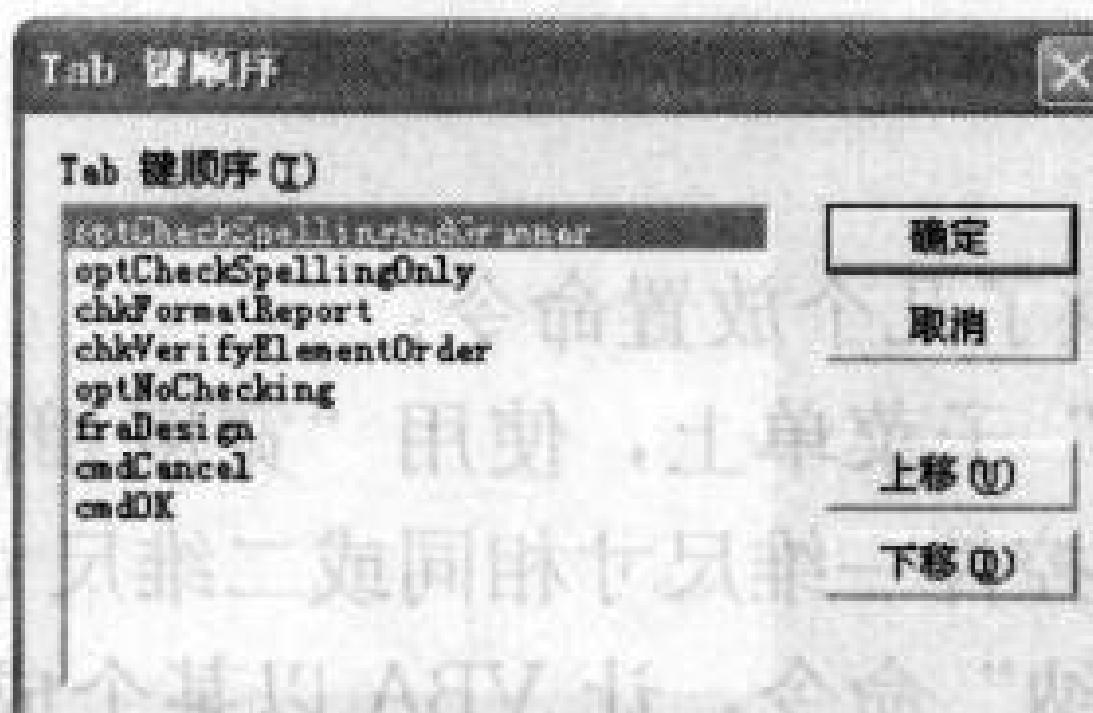


图 14.16 使用“Tab 键顺序”对话框将用户窗体或框架中的各控件按逻辑顺序进行安排

2. 在“Tab 键顺序”列表框内选择各控件，根据情况单击“上移”按钮或“下移”按钮，把各控件按希望的顺序重新安排。可以使用 Shift+单击或拖曳方法来选择控件区，或使用 Ctrl+单击来选择两个或多个不相邻接的控件。
3. 单击“确定”按钮以关闭“Tab 键顺序”对话框。

把对话框链接到过程

在过程中使用自定义对话框进行工作时，设计自定义对话框仅仅是第一步。下一步是编写代码以显示该对话框，并使其实现各种功能。

用于对话框的代码通常由如下部分组成：

- ◆ 一个过程，它通过加载对话框和使用 Show 方法来显示对话框。这个过程往往被指定给一个菜单项、一个工具栏按钮或一个组合键，以便用户能够调用它。但是，过程也可以自动运行以响应某个系统事件（如在一个指定时刻运行）。
- ◆ 用户窗体，它包含该对话框及其各控件。
- ◆ 附属于用户窗体的代码。这种代码由对应于指定控件的各过程组成。例如，对于只

包含两个选项按钮和两个命令按钮（一个“确定”按钮和一个“取消”按钮）的简单对话框，通常要为“确定”按钮创建一个过程，为“取消”按钮再创建一个过程。为“确定”按钮创建的过程，由“确定”按钮上的 Click 事件来触发（在“确定”按钮上单击，或焦点在“确定”按钮上时按 Enter 键），这个过程确定哪一个选项按钮已被选择，然后采取相应的行动。对于“取消”按钮的过程则用来取消该过程。

注意：可以把过程指定给对应于某一选项按钮的 Click 事件——或者指定给另一个事件；下一章将对此详述——但是，通常这对在静态对话框中捕获命令按钮更有意义。在动态对话框中，可能常常想要捕获某一选项按钮上的单击，并视情况来显示更多的控件。

一旦附属于某一按钮的代码已经运行，执行便返回到对话框（如果它仍在显示的话），或返回到调用对话框的过程。

注意：为响应某一事件而直接运行的代码称为事件过程。一个事件过程在必要时可以调用若干其他过程，所以可以通过一个单个事件来间接地运行多个过程。

加载和卸载对话框

使用 Load 语句加载对话框，使用 Unload 语句卸载对话框。Load 语句将对话框加载进存储器以便使对话框能为程序所用，但它不能显示该对话框；要显示该框，必须使用 Show 方法（下节讨论）。Unload 语句将对话框从存储器中卸出，使得与该对象相关联的存储器恢复原状。如果 Unload 语句运行时对话框被显示，VBA 会把该对话框从屏幕上除去。

Load 和 Unload 语句对应的语法很简单：

```
Load Dialog_Box  
Unload Dialog_Box
```

这里，Dialog_Box 是用户窗体或对话框的名称。例如，下述语句加载名为 frmMyDialog 的对话框：

```
Load frmMyDialog
```

显示和隐藏对话框

要显示对话框，可使用 Show 方法；要隐藏对话框，可使用 Hide 方法。例如，下述语句显示名为 frmMyDialog 的对话框：

```
frmMyDialog.Show
```

运行包含这行代码的过程时，frmMyDialog 对话框就出现在屏幕上，然后就可以向它的文本框里输入文本，选择或清除它的复选框，使用它的下拉列表，以及随意单击该对话框上的按钮。当关闭该对话框时（单击标题栏上的“关闭”按钮，或单击使对话框退出的命令按钮），它便从屏幕上消失，过程则继续运行。但是，在从对话框中取回设置，并就此采取行动之前，对话框对其图形显示之外的事情没有影响。

使用 Show 方法但并不先用 Load 语句来明显地加载对话框，也可以显示出对话框来；

由 VBA 来关心这个隐含的 Load 命令。把 Load 命令包括进去，并没有特别的好处，但它会使代码容易阅读和调试。例如，如下所示的两个过程具有相同的效果：

```
Sub Display_Dialog()
    Load frmMyDialog 'loads the dialog box into memory
    frmMyDialog.Show 'displays the dialog box
End Sub

Sub Display_Dialog()
    frmMyDialog.Show 'loads the dialog box into memory and displays it
End Sub
```

注意：如果运行 Hide 方法，但没有使用 Load 语句或 Show 方法将对话框加载进存储器的话，VBA 会加载该对话框，但不把它显示在屏幕上。

显示了对话框以后，稍待一会，使用 Tab 键移动来检查它的 Tab 顺序。看一看：当打开对话框时，焦点是在合适的控件上吗？当从该控件上前移时，选中的下一控件是用户通常需要使用的下一控件吗？必要时，按照本章前面“调节对话框的 Tab 顺序”所述方法，对 Tab 顺序进行调节。

设置默认命令按钮

为了在对话框内设置一个默认的命令按钮，应把这个命令按钮的 Default 属性设置为 True。VBA 在显示对话框时选择这个默认按钮，这样，如果用户简单地按下 Enter 键来退出对话框，这个按钮就会接收键盘命令。

在任何给定时刻，只能有一个按钮是默认按钮。如果把任何一个按钮的 Default 属性设置为 True，那么 VBA 会自动把另一个原先 Default 属性为 True 的按钮的 Default 属性更改成 False。

从对话框中取回用户的选择

要使对话框采取一个行动，需要从它那里取回用户的选择。本节首先介绍从对话框中取回信息的 VBA 命令，再通过例子来说明如何先从比较简单的对话框，而后从更复杂的对话框中取回用户的选择。

从文本框中返回一个字符串

为了从某个文本框中返回（取回）一个字符串，则需要在用户退出对话框之后检查这个文本框的 Value 属性或 Text 属性。例如，如果有一个名为 txtMyText 的文本框，要返回它的值并在一个消息框内显示它，应使用下列代码：

```
MsgBox txtMyText.Value
```

注意：对于文本框，Value 属性和 Text 属性返回相同的信息；对于大多数其他的 VBA 对象，Value 属性和 Text 属性返回不同的信息。

VBA 既支持单行文本框，也支持多行文本框。为了生成多行文本框，在用户窗体中或属性窗口内的下拉列表中选择文本框，并将其 MultiLine 属性设置为 True。此后用户可向

文本框中输入多行代码，并按下 Shift+Enter 键来开始新行。

要将水平滚动条或垂直滚动条加进文本框，应设置文本框的 ScrollBars 属性：fmScrollBarsHorizontal (1) 显示水平滚动条，fmScrollBarsVertical (2) 显示垂直滚动条，fmScrollBarsBoth (3) 二者均显示。

从选项按钮中返回一个值

常规的选项按钮是一个布尔型控件，它只能有两个值：True 和 False。True 说明按钮被选中，False 说明按钮未被选中。可以使用简单的 If...Then 条件来检查选项按钮的值。例如，如果有两个名为 optSearchForFile 和 optUseThisFile 的选项按钮，要检查它们的值并找出哪个按钮被选中，可使用如下条件：

```
If optSearchForFile = True Then
    'optSearchForFile was selected; take action on this
Else    'optSearchForFile was not selected, so optUseThisFile was
    'take action for optUseThisFile
End If
```

注意：Value 是选项按钮控件的默认属性。上述代码实际上是检查该控件的默认属性的值。代码第一行可以更完整地写为 If optSearchForFile. Value = True Then，也可以更简洁地写为 If optSearchForFile Then，它已隐含了 = True。

当选项按钮多于两个时，使用 If...Then...Else If 条件或 Select Case 语句来确定哪一个按钮被选中。

注意：正如本章前面所述，如果把选项按钮或复选框的 TrippleState 属性设置为 True，它们可以有 Null 状态。如果允许有的话，在过程中也要对此进行检查。不能直接检查控件的值是否为 Null（例如，If opt1. Value = Null 会引起一个错误），所以要先使用 If 语句或 Select Case 语句来测试 True 和 False。如果控件的 Value 既不是 True，也不是 False，那就一定是 Null。

从复选框中返回一个值

与选项按钮一样，常规的复选框要么是 True，要么是 False，所以，可以用 If...Then 条件来检查它的值。例如：

```
If chkDisplayProgress = True Then
    'take actions for chkDisplayProgress
End If
```

同样，这里也是检查控件的默认属性——Value 属性。代码第一行也可写为 If chkDisplayProgress. Value = True Then。

注意：有时候，如果复选框被清除而不是被选中，就有必要采取行动。例如，如果用户清除复选框，就可能有必要关闭一个配置选项。

从列表框中返回一个值

从列表框中返回一个值之前，必须告诉 VBA 哪些选项要显示在列表框中。为此，需创建一个过程去初始化（准备）用户窗体，并向列表框添加项目，然后显示列表框：

1. 右击“工程资源管理器”中用户窗体的名称，并从上下文菜单中选择“查看代码”以显示（在“代码”窗口中）与指定给对话框的各控件相对应的代码。
2. 在“对象”下拉列表中，确认该用户窗体已被选中。
3. 从“过程”下拉列表中，选择“Initialize（初始化）”。Visual Basic 编辑器在代码表上当前已包含的各过程的末尾处，创建一个名为 UserForm_Initialize 的新过程。

```
Private Sub UserForm_Initialize()
End Sub
```

注意：每次用户窗体被调用时，VBA 都运行 UserForm_Initialize 过程。这一过程是向列表框或组合框添加项目的好方法，也是为用户窗体上其他控件设置属性的好方法。

4. 为了向列表框添加项目，使用对应该列表框的对象（这里是 lstBatteries）、双引号中有文本字符串的 AddItem 方法以说明列表框中的各项：

```
lstBatteries.AddItem "Battery #A4601"
lstBatteries.AddItem "Battery #A4602"
lstBatteries.AddItem "Battery #A4603"
lstBatteries.AddItem "Battery #A4604"
```

提示：在初始化窗体时添加项目，可以视情况添加不同数量的项目。例如，如果希望用户从一个特定的文件夹中挑选一个文档，那么可以立即临时创建一个该文件夹中所含各文档的列表，然后把各文档的名称填写进列表框。

为了从单选列表框中取出结果，要返回 Value 属性。例如：

```
MsgBox "You chose this entry from the list box: " & lstBattery.Value
```

当使用 MultiSelect 属性来生成一个可做多项选择的列表框时，可以不再使用 Value 属性来返回列表框中被选择的各项目：如果 MultiSelect 为 True，Value 总是返回一个零值。替代方法是使用 Selected 属性来确定列表框中哪些行已被选择，并使用 List 数组来返回每个被选行的内容。以下语句使用 For…Next 循环来构建一个包含有从多选列表框中选出的各条目的名为 StrMsg 的字符串：

```
strMsg = "You chose the following entries from the list box: " & vbCrLf
For i = 1 To lstBatteries.ListCount
    If lstBatteries.Selected(i - 1) = True Then
        strMsg = strMsg & lstBatteries.List(i - 1) & vbCrLf
    End If
Next i
MsgBox strMsg
```

从组合框中返回一个值

为了从一个组合框中（列表框和文本框的组合）返回一个值，需在 Initialize 过程中向组合框列表添加项目，然后在用户退出对话框后检查组合框的 Value。（组合框控件不提供多项选择能力，因此要检查的属性是 Value。）

```
Private Sub UserForm_Initialize()
    cmbColor.AddItem "Red"
    cmbColor.AddItem "Blue"
    cmbColor.AddItem "Yellow"
End Sub

Result = cmbColor.Value
```

从组合框中取回的项目可以是 Initialize 过程中指定的各项目中的一个，或者是用户键入组合框的文本框部分的一个项目。

把对话框连接到过程的例子

本节举两个例子来说明如何创建一个过程，然后构建一个对话框并加至该过程，使过程用处更大、功能更强。在第一个例子中，先在 Word 中录制一个宏，再把对话框连接到这个宏。第二个例子将可以在任何 VBA 能使用的应用程序中工作，它是从头开始生成一个用户窗体和它的代码。

Word 例子：移动段落过程

第一个过程是在 Word 中将当前段落在文档之内上移或下移一个或两个段落。

录制过程

从录制一个在 Word 中移动当前段落的过程开始。在过程中，需要录制一些命令，它们是为了：

- ◆ 选择当前段落；
- ◆ 剪切选择内容，然后粘贴它；
- ◆ 在文档内上移和下移插入点；
- ◆ 插入一个书签，移动插入点到书签，然后删除书签。

完成后的过程显示一个对话框，内有选项按钮，分别对应于：将当前段落上移一段、上移两段、下移一段或下移两段。对话框还包含一个复选框，用以使插入点在过程结束时返回它的最初位置。是否有人希望过程有此行为，目前只能推测，所以复选框在默认情况下为选中；如果用户不想让插入点返回到它的最初位置，可以清除该复选框。

首先，启动 Word，并生成一个草稿文档，输入三段或四段文本——可以只写打算干些什么事，但要使文本易于识别，这样才便于确认，该过程正在做应该做的事：移动段落。然后将插入点置于刚输入的某一个段落中，再开始按第 1 章讨论过的那样，录制一个宏：

1. 双击状态栏上的“录制”指示符，或选择“工具”>“宏”>“录制新宏”，显示出

“录制宏”对话框。

2. 在“宏名”文本框内，为该宏键入名称：Move_Paragraph，在“说明”文本框内键入说明。
3. 如有必要，在“将宏保存在”下拉列表中，选择模板或文档。
4. 如果愿意的话，使用“工具栏”按钮或“键盘”按钮以生成对应于该宏的工具栏按钮、菜单选项或键盘快捷方式。
5. 单击“确定”按钮以启动录制宏。

在宏中录制以下行动：

1. 在插入点的当前位置处插入一个书签，步骤是：使用“插入”>“书签”命令以显示“书签”对话框，为书签输入名称，再单击“添加”按钮。在本例中，书签取名为Move_Paragraph_Temp以指明这是一个用于Move_Paragraph过程的临时性的书签。
2. 按下F8键4次以选择当前段落。第一次按下F8键是激活扩展模式（状态栏上的扩展指示符切换为通），第二次是选择当前字，第三次是选择当前句子，第四次是选择当前段落。选中段落后，按下Esc键以关闭扩展模式。（状态栏上的扩展指示符切换为断）。
3. 使用“剪切”命令的某种方式（例如，单击“剪切”按钮，或按下Ctrl+X或Shift+Delete键）来剪切被选段落。
4. 按下Ctrl+↑键以将插入点上移一个段落。
5. 使用“粘贴”命令（例如，单击“粘贴”按钮，或按下Shift+Insert键），将被剪切段落粘贴回去。
6. 按下Ctrl+↓键以将插入点下移一个段落。
7. 按下Ctrl+↑键两次以将插入点上移两个段落。

注意：如果从插入点在文档中第一个段落起始处开始，只能将插入点上移一个段落。这不要紧——按下键来录制它就行了。如果Word鸣叫，别理它。

8. 按下Ctrl+↓键两次以将插入点下移两个段落。（按此操作时，第一次敲键之后，如果就到了文档末尾的话，别担心，完成第二次敲击以录制它。Word可能会再次鸣叫。）
9. 打开“书签”对话框（选择“插入”>“书签”），选择Move_Paragraph_Temp书签，再单击“定位”按钮使其定位。然后单击“删除”按钮以删除Move_Paragraph_Temp书签。单击“关闭”按钮以关闭“书签”对话框。
10. 单击“停止录制”工具栏上的“停止录制”按钮，或双击状态栏上的录制指示符以停止宏录制。

单击“工具”>“宏”>“宏”，打开“宏”对话框中，从中选择宏的名称，再单击“编辑”按钮，以便在Visual Basic编辑器中打开已录制的宏。现在应该看到像下面这个样子的宏：

```

1. Sub Move_Paragraph()
2. ' Move_Paragraph Macro
3. ' Macro recorded 5/1/2006 by Jack Ishida
4. '
5. '
```

```

6. With ActiveDocument.Bookmarks
7.     .Add Range:=Selection.Range, Name:="Move_Paragraph_Temp"
8.         .DefaultSorting = wdSortByName
9.         .ShowHidden = False
10.    End With
11.    Selection.Extend
12.    Selection.Extend
13.    Selection.Extend
14.    Selection.Extend
15.    Selection.EscapeKey
16.    Selection.Cut
17.    Selection.MoveUp Unit:=wdParagraph, Count:=1
18.    Selection.PasteAndFormat (wdPasteDefault)
19.    Selection.MoveDown Unit:=wdParagraph, Count:=1
20.    Selection.MoveUp Unit:=wdParagraph, Count:=2
21.    Selection.MoveDown Unit:=wdParagraph, Count:=2
22.    Selection.GoTo What:=wdGoToBookmark, Name:="Move_Paragraph_Temp"
23.    ActiveDocument.Bookmarks("Move_Paragraph_Temp").Delete
24.    With ActiveDocument.Bookmarks
25.        .DefaultSorting = wdSortByName
26.        .ShowHidden = False
27.    End With
28. End Sub

```

这个宏可能比较好读：

- ◆ 行 1 是宏的开始，行 28 是宏的结束。行 2 和行 5 是空白注释行，它们中间的两个注释行显示宏的名称（行 3）和说明（行 4）。
- ◆ 行 6 至行 10 包含一个添加 Move_Paragraph_Temp 书签的 With 语句。行 8 和行 9 在此处是不必要的，但是，宏录制器录制“书签”对话框中的所有设置，包括对“排序依据”选项按钮和“隐藏书签”复选框的设置。
- ◆ 行 11 到行 15 使用“扩展选定范围”特性来选择当前段落。
- ◆ 行 17、行 19、行 20 和行 21 分别录制对应于将插入点上移一段、下移一段、上移两段和下移两段的语法。
- ◆ 行 16 录制“剪切”命令，行 18 录制“粘贴”命令。
- ◆ 行 22 将插入点移到 Move_Paragraph_Temp 书签，行 23 删除书签。行 24 至行 27 录制“书签”对话框中的设置，此处也是不必要的。

可以很快地删除不必要的行，拆除第一个 With 结构，给出一个更简洁的代码：

```

1. Sub Move_Paragraph()
2.     ActiveDocument.Bookmarks.Add Range:=Selection.Range,
3.         Name:="Move_Paragraph_Temp"
4.     Selection.Extend
5.     Selection.Extend
6.     Selection.Extend
7.     Selection.EscapeKey
8.     Selection.Cut
9.     Selection.MoveUp Unit:=wdParagraph, Count:=1
10.    Selection.PasteAndFormat (wdPasteDefault)
11.    Selection.MoveDown Unit:=wdParagraph, Count:=1
12.    Selection.MoveUp Unit:=wdParagraph, Count:=2

```

```

13. Selection.MoveDown Unit:=wdParagraph, Count:=2
14. Selection.GoTo What:=wdGoToBookmark, Name:="Move_Paragraph_Temp"
15. End Sub

```

生成对话框

下面为该过程生成对话框（见图 14.17）。

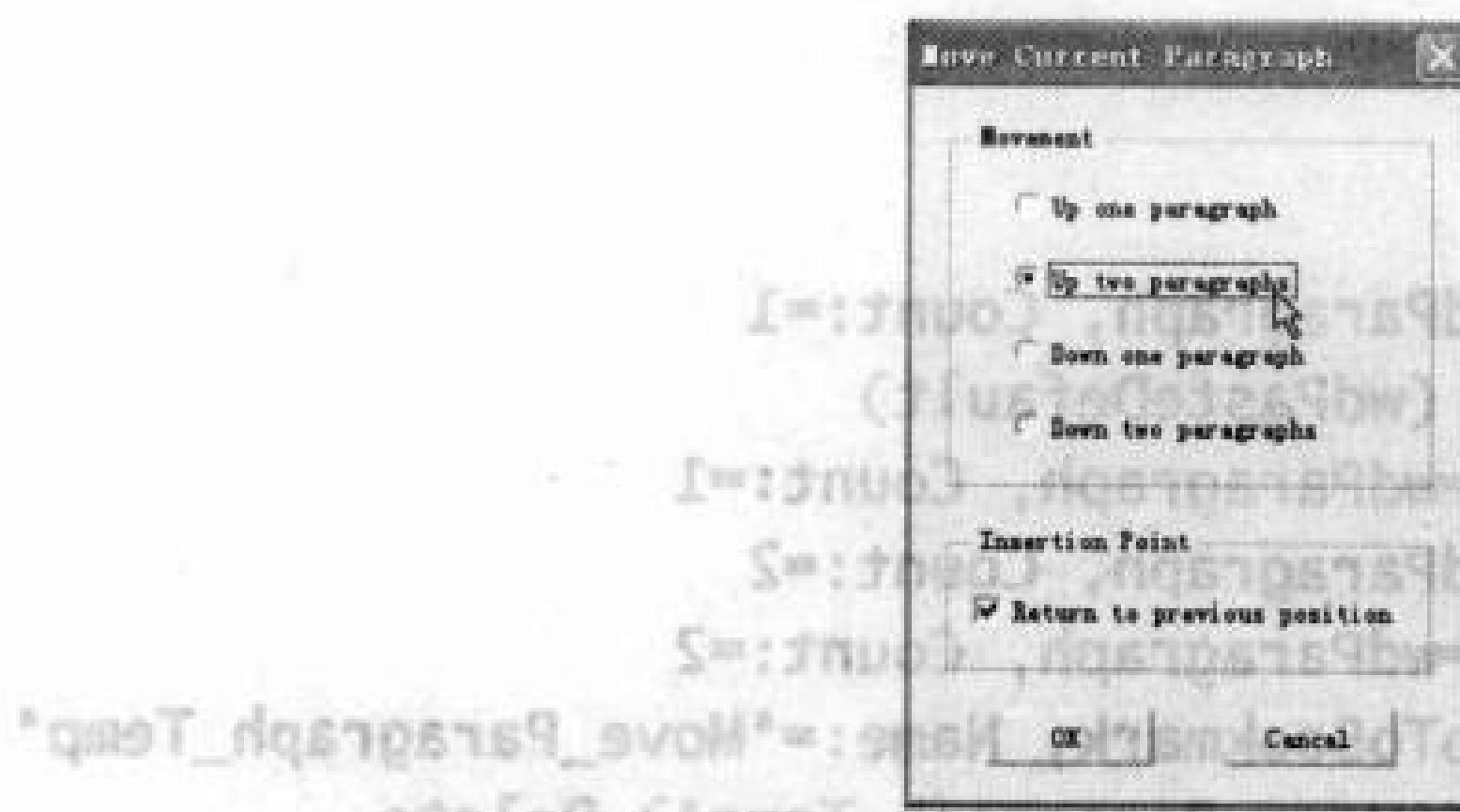


图 14.17 将要连接到 Move_Paragraph 宏的“Move Current Paragraph”对话框

1. 单击“插入”按钮的下拉列表，并选择用户窗体（或仅仅单击“插入”按钮，如果它已显示了用户窗体图标的话），或选择“插入”>“用户窗体”，启动一个用户窗体。
2. 使用对应于该用户窗体的“属性”窗口以设置它的名称和 Caption 属性。在名称旁边的单元格内单击，并输入名称属性，然后在 Caption 旁边的单元格内单击并输入 Caption 属性。作为例子的该用户窗体，取名为 frmMoveCurrentParagraph，且标题为 Move Current Paragraph，这样，窗体的名称和用户将在对话框的标题栏内看到的文本密切相关，但不同于过程名（Move_Current_Paragraph）。

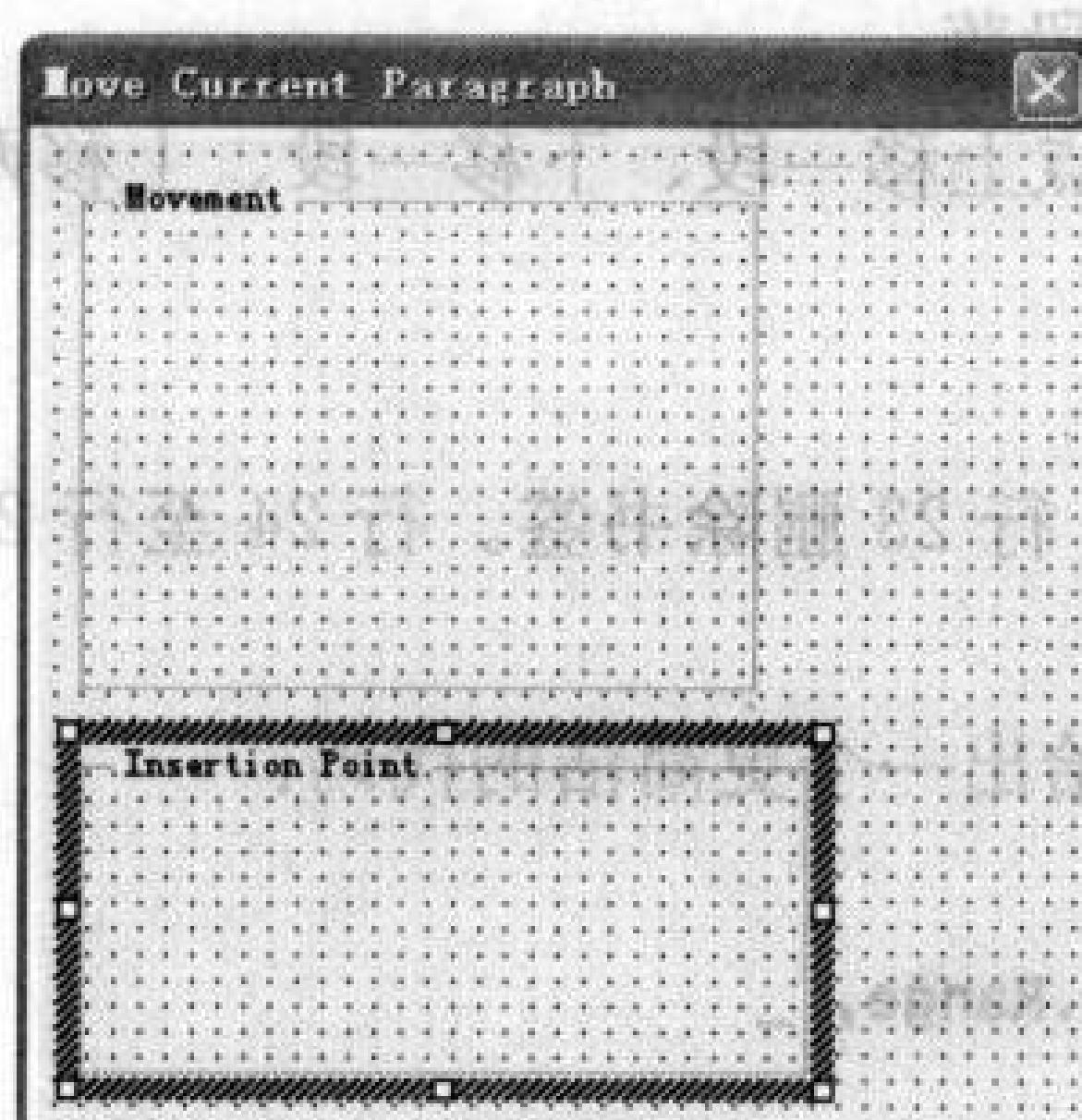


图 14.18 从把两个框架放入用户窗体开始

3. 将两个框架放入用户窗体（如图 14.18 所示），以作为对话框中的组框使用。

- A. 双击工具箱内的“框架”工具，然后在用户窗体内单击并拖曳以放置每个框架。单击“选定对象”按钮以恢复选择指点符。
- B. 选中这两个框架，并选择“格式”>“对齐”>“左对齐”以对齐两个框架。
- C. 使两个框架保持选中，选择“格式”>“统一尺寸”>“宽度相同”（此处不要选择“格式”>“统一尺寸”>“高度相同”或“格式”>“统一尺寸”>“两者都相同”——顶框架应比底框架更高。）

- D. 依次选择每个框架，再为它们在“属性”窗口内设置 Caption 属性，顶框标题取名为 Movement，而底框标题取名为 Insertion Point。然后将顶框名称取名为 fraMovement，而底框名称取为 fraInsertionPoint。
4. 在 Movement 框架内放置四个选项按钮，如图 14.19 所示。

- A. 双击工具箱内的选项按钮工具，然后在 Movement 框架内单击以放置每个选项按

钮。此时，不要单击并拖曳——仅单击即可放置常规宽度的选项按钮。

B. 放置好四个选项按钮时，单击工具箱内的“选定对象”按钮以恢复选择指点符。

然后选中这四个选项按钮，通过选择“格式”>“对齐”>“左对齐”使它们互相对齐。为使它们的间距均衡，选择“格式”>“垂直间距”>“相同”。如有必要，可使用“格式”>“垂直间距”子菜单上的其他项——“增加”、“减少”、“移除”——以调节选项按钮之间的间距。

C. 在“属性”窗口内设置 Caption 属性，为每个选项按钮更改标题，像图 14.19 所示那样，将它们的标题取为：Up one paragraph, Up two paragraphs, Down one paragraph 和 Down two paragraphs。这些选项按钮将控制，该过程把当前段落上移或下移多远。

D. 如果需要调整这四个选项按钮的大小，可以通过右击并从上下文菜单中选择“生成组”，或选择“格式”>“生成组”，或单击用户窗体工具栏上的“组”按钮，选中这四个选项按钮并使其成为控件组。然后选中这个组，并拖曳一个“手柄”，以均匀地调整所有选项按钮的大小。例如，如果需要使所有选项按钮变长以容纳输入的文本，向外拖曳右边中心点上的“手柄”即可。

E. 在“属性”窗口中依次更改每个选项按钮的名称属性，从而将这些选项按钮的名称分别取为 optUpOne、optUpTwo、optDownDne 和 optDownTwo。

提示：按照默认方式，一个用户窗体内的所有选项按钮都是同一选项组的一部分。

这意味着，每次只有一个选项按钮可被选中。如果希望一个用户窗体内有多个选项按钮组，就必须说明各单个组。做到这一点的最容易方法是把每个组放入一个单独的框架控件之内。也可以为每个选项按钮设置 GroupName 属性。

F. 然后，将第一个选项按钮的 Value 属性设置为 True，这需要在“属性”窗口中选择默认的 False 值，再输入 True 以取代之。这样做了之后，就在指定的用户窗体中选中了这个选项按钮，当对话框显示时，这个按钮就被选为选项组的默认选择。输入 U 作为第一个选项按钮的 Accelerator 属性，这样就设置了它的加速键为 U。将第二个选项按钮的 Accelerator 属性设置为 t，第三个设置为 D，第四个设置为 w。

注意：当某一控件的标题包含有同一字母的大写体和小写体时，Accelerator 属性可分辨出来。

5. 在 Insertion Point 框架内，放置一个复选框，如图 14.20 所示。



图 14.19 在 Movement 框架中放置
四个选项按钮

图 14.20 在 Insertion Point 框
架中放置一个复选框

A. 单击工具箱内的复选框工具，然后在用户窗体内的 Insertion Point 框架中单击，以放置默认尺寸的一个复选框。

B. 在“属性”窗口中，将该复选框的名称设置为 chkReturnToPreviousPosition（名称有点长，但有说明性）。再把它的 Caption 属性设置为 Return to previous position。输入 R 作为它的 Accelerator 属性，以将其加速键设置为 R。最后，输入 True 作为它的 Value 属性，使该复选框为默认选中。

6. 然后，为该窗体插入命令按钮（见图 14.21）：

A. 双击工具箱内的命令按钮工具，再单击以便把第一个命令按钮放置在用户窗体底部。单击以放置第二个命令按钮，然后单击“选定对象”按钮，以恢复选择鼠标指针。

B. 使用“格式”菜单上的命令，对这两个命令按钮进行调整。例如，可以使这两个按钮成为控件组，再使用“格式”>“窗体内居中”>“水平对齐”命令，使它们位居正中而横向成对。在这样做之前，必须让它们成为一个控件组——如果只是简单地选中这两者，VBA 会使一个居中，并使其位于另一个的顶上，这样就只有一个按钮看得见。

C. 设置命令按钮的属性如下：对左边的按钮（它将成为 OK 按钮），将其 Name 属性设置为 cmdOK，Caption 属性设置为 OK，Acceleration 属性设置为 O（是 OK 中的 O，不是零），而 Default 属性设置为 True。对右边的按钮（它将成为 Cancel 按钮），将其 Name 属性设置为 cmdCancel，Accelerator 属性设置为 A，Caption 属性设置为 Cancel，Cancel 属性设置为 True，而 Default 属性设置为 False。

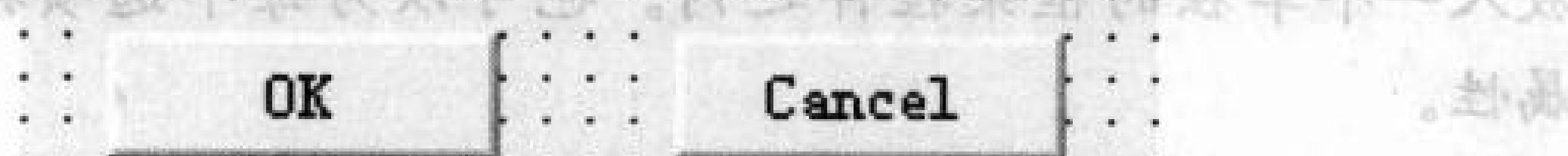


图 14.21 添加两个命令按钮并设置其属性

7. 双击“Cancel”按钮，以显示与之相关联的代码：

```
Private Sub cmdCancel_Click()
End Sub
```

在两行间键入一个 End 语句：

```
Private Sub cmdCancel_Click()
End
End Sub
```

这个 End 语句将对话框从屏幕上除去，并结束当前过程——在本例中，就是 Move_Current_Paragraph 过程。

现在来设置 OK 按钮。当用户单击 OK 按钮时，该过程需要继续，并做以下事情：

- ◆ 通过隐藏对话框或卸载对话框（最好二者都用）使对话框不显示。正如本章前面所述，选择由你定，但两个命令都用最有把握。
- ◆ 检查复选框的 Value 属性，看它是被选中还是被清除。
- ◆ 依次检查每个选项按钮的 Value 属性，看看在单击 OK 按钮时，哪一个选项按钮被

8. 双击 OK 按钮以显示附属于它的代码。(如果此时仍工作在附属于 Cancel 按钮的代码之中, 可从 Private Sub cmdCancel_Click() 代码中作上滚或下滚, 以找出 Private Sub cmdOK_Click() 代码。)

```
Private Sub cmdOK_Click()
    If objCaption = True Then
        Selection.MoveUp Units:=wdParagraph, Count:=1
    ElseIf objCaption.MoveDown Units:=wdParagraph, Count:=1
        Selection.MoveDown Units:=wdParagraph, Count:=1
    Else
        Selection.MoveDown Units:=wdParagraph, Count:=5
    End If
End Sub
```

首先, 在 Private Sub 和 End Sub 两行之间输入以下两行:

```
frmMoveParagraph.Hide
Unload frmMoveParagraph
```

frmMoveParagraph.Hide 激活对应于 frmMoveParagraph 用户窗体的 Hide 方法, 隐藏该窗体使之不在屏幕上显示。Unload frmMoveParagraph 行将对话框从存储器中卸出。

注意: 隐藏或卸载对话框, 对于继续执行过程不是必要的, 但是, 如果不这样做, 用户可能会迷茫。例如, 如果在 Windows 某一应用程序的“打印”对话框上单击了“确定”按钮, 会期待对话框消失而打印命令执行。如果对话框不消失(但它启动了后台打印作业), 可能会使人认为没有记录下单击, 便在对话框消失前一次又一次地单击。

9. 然后, 该过程需要检查 chkReturnToPreviousPosition 复选框的 Value 属性以弄清是否将一个书签插入文档, 以作为插入点当前位置的标志。为此, 输入一个简单的 If ... Then 语句:

```
If chkReturnToPreviousPosition = True Then
    End If
```

如果 chkReturnToPreviousPosition 为 True ——这就是说, 如果该复选框被选中——就运行 Then 语句后面的各代码行。Then 语句包括对应于前已录制的插入一个书签的几个代码行。从过程中剪切这几行, 并将它们粘贴到 If ... Then 语句, 就成了:

```
If chkReturnToPreviousPosition = True Then
    With ActiveDocument.Bookmarks
        .Add Range:=Selection.Range, Name:=" Move_Paragraph_Temp"
    End With
End If
```

如果复选框被选中, 该过程插入一个书签; 如果复选框被清除, 该过程跳过这几行。

10. 然后, 把选择当前段落和将其剪切到剪贴板的代码粘贴进去:

```
Selection.Extend
Selection.Extend
Selection.Extend
Selection.Extend
Selection.Cut
```

11. 在这之后，需要从选项按钮中取回 Value 属性，看一看，用户在对话框中选择“确定”按钮时，是哪一个选项按钮被选中。为此，再次使用一个 If 条件——这次是一个 If…Then ElseIf…Else 条件，并把来自已录制过程的有关插入点—移动的代码行粘贴进去：

```
If optUpOne = True Then
    Selection.MoveUp Unit:=wdParagraph, Count:=1
ElseIf optUpTwo = True Then
    Selection.MoveUp Unit:=wdParagraph, Count:=2
ElseIf optDownOne = True Then
    Selection.MoveDown Unit:=wdParagraph, Count:=1
Else
    Selection.MoveDown Unit:=wdParagraph, Count:=2
End If
Selection.Paste
```

这里，optUpOne、optUpTwo、optDownOne 和 optDownTwo（它在这里使用了 Else 语句，所以程序清单中没有以名称来指明）是来自对话框的四个选项按钮，分别表示将当前段落上移一段、上移两段、下移一段和下移两段四种选择。条件很简单：如果 optUpOne 为 True（即该按钮若被选中），就运行第一个 Then 条件，将插入点从它的当前位置上移一个段落（剪切当前段落之后，插入点将位于原在当前段落之后的那个段落的起始处）。如果 optUpOne 为 False，再判断第一个 ElseIf 条件；如果它为 True，就运行第二个 Then 条件；如果它为 False，再判断下一个 ElseIf 条件。如果这次还是 False，则运行 Else 代码。在本例中，Else 语句的含义是，在对话框中选中了 optDownTwo 选项按钮，所以，Else 代码将插入点下移两个段落。

关注了选项按钮之后，不管插入点最后停留何处，下行代码（Selection.Paste）都要把已剪切段落从剪贴板粘贴进去。

12. 最后，如果 chkReturnToPreviousPosition 复选框被选中，该过程必须使插入点返回到它的最初位置。同样可以使用 If…Then 条件对此进行测试，这个条件组合了来自已录制过程的转到书签和删除书签代码行。

```
If chkReturnToPreviousPosition = True Then
    Selection.GoTo What:=wdGoToBookmark, _
        Name:="Move_Paragraph_Temp"
    ActiveDocument.Bookmarks("Move_Paragraph_Temp").Delete
End If
```

如果 chkReturnToPreviousPosition 复选框被选中，VBA 就把插入点移至临时书签，然后删除此书签。

程序清单 14.1 列出了对应于 OK 按钮的完整清单。

程序清单 14.1

1. Private Sub cmdOK_Click()
2. frmMoveCurrentParagraph.Hide
3. Unload frmMoveCurrentParagraph
4. If chkReturnToPreviousPosition = True Then

```

5.     With ActiveDocument.Bookmarks
6.         .Add Range:=Selection.Range, _
    Name:=" Move_Paragraph_Temp"
7.     End With
8. End If
9. Selection.Extend
10. Selection.Extend
11. Selection.Extend
12. Selection.Extend
13. Selection.Cut
14. If optUpOne = True Then
15.     Selection.MoveUp Unit:=wdParagraph, Count:=1
16. ElseIf optUpTwo = True Then
17.     Selection.MoveUp Unit:=wdParagraph, Count:=2
18. ElseIf optDownOne = True Then
19.     Selection.MoveDown Unit:=wdParagraph, Count:=1
20. Else
21.     Selection.MoveDown Unit:=wdParagraph, Count:=2
22. End If
23. Selection.Paste
24. If chkReturnToPreviousPosition = True Then
25.     Selection.GoTo What:=wdGoToBookmark,
    Name:=" Move_Paragraph_Temp"
26.     ActiveDocument.Bookmarks("Move_Paragraph_Temp").Delete
27. End If
28. End Sub

```

通用例子：从列表框中打开文件

这个例子展示了一个用户窗体，该窗体使用了一个列表框，以便让用户选择一个文件以打开它。用户窗体及其代码都较简单，代码中使用了循环和汇集文件夹中各文件名称的数据，然后在列表框中显示文件名。用户可以选择一个文件，并单击“打开”按钮来打开它。图 14.22 显示出用于 Excel 文件的使用中的用户窗体。

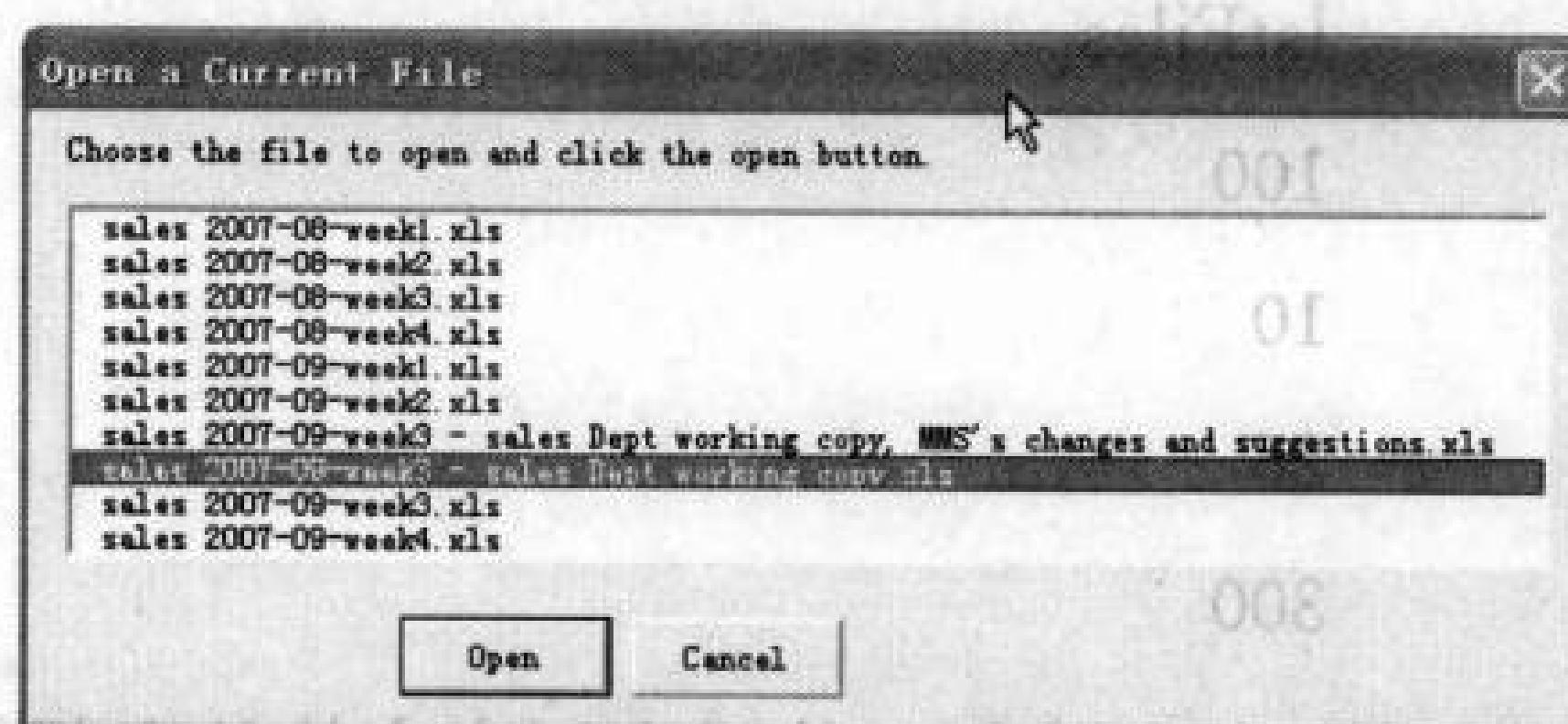


图 14.22 本例中将要构建的用户窗体包含一个可使用户迅速访问所有当前文件的列表框

只要把文件名改成适当的类型，再改动若干关键语句，就可以把本例配置到本书讨论过的任何一种应用程序中去。本例说明如何在 Excel 中生成代码，但是可以把它改编成适用于想要使用的其他应用程序。

构建用户窗体

可以通过如下步骤来构建用户窗体：

- 启动想要在其中工作的应用程序，本例使用 Excel。
- 按下 F11 键，或选择“工具”>“宏”>“Visual Basic 编辑器”以显示 Visual Basic 编辑器。
- 在“工程资源管理器”中，单击想要向它添加用户窗体的那个工程，并从上下文菜单中选择“插入”>“用户窗体”，以便在该工程中插入一个默认尺寸的用户窗体。
- 向右拖曳用户窗体右下角的“手柄”以使用户窗体更宽一些。
- 将窗体的名称设置为 frmOpen_a_Current_File，将它的 Caption 属性设置为 Open a Current File。检查其 Width 属性：使其宽度约为 350 像素。
- 单击工具箱内的“标签”按钮，然后在用户窗体的左上角单击，以便将一个默认尺寸的标签置于该处。激活“属性”窗口，为标签设置属性，如下所示：

属性	值
Name (名称)	lblInfo
AutoSize	True
Caption	Choose the file to open and click the Open button.
Left	10
Top	6
wordWrap	False

- 单击工具箱内的“列表框”按钮，然后在用户窗体内的标签下面单击，以便将一个默认尺寸的列表框置于该处。设置其属性如下：

属性	值
Name (名称)	lstFiles
Height	100
Left	10
Top	25
Width	300

- 双击工具箱内的“命令按钮”按钮，然后在用户窗体的底部单击两次，以便将两个默认尺寸的命令按钮置于该处。设置它们的属性如下：

属性	第一个按钮的值	第二个按钮的值
Name (名称)	cmdOpen	cmdCancel
Cancel	False	True
Caption	Open	Cancel
Default	True	False

属性	第一个按钮的值	第二个按钮的值
Height	21	21
Width	55	55

9. 安排命令按钮如下：

- 单击 cmdCancel 按钮以选中它，然后拖曳它，使它靠近 cmdOpen 按钮。
- 使 cmdCancel 按钮保持被选中，通过 Ctrl+单击 cmdOpen 以使其添加到选定范围。
- 选择“格式”>“组”以使所有按钮成组。
- 选择“格式”>“窗体内居中”>“水平对齐”，使这两个按钮在水平方向上位于窗体正中间。
- 必要时将此控件组向上或向下拖曳。

生成对应于该用户窗体的代码

遵循如下步骤来生成对应于该用户窗体的代码：

- 保持用户窗体被选中，按下 F7 键，以显示该用户窗体的代码表。
- 在代码表的声明部分，输入一条语句 Option Base 1，以使数组从 1 而不是从 0 起开始编号：

```
Option Base 1
```

- 确认在“对象”下拉列表中该窗体已被选中，然后在“过程”下拉列表中选择“Initialize（初始化）”。Visual Basic 编辑器将 Initialize 过程的头和尾输进代码表：

```
Private Sub UserForm_Initialize()
```

```
End Sub
```

- 输入程序清单 14.2 所示的与 Initialize 过程对应的各语句。
- 在“对象”下拉列表中选择 cmdCancel。Visual Basic 编辑器输入 Click 过程的头和尾，如下所示。（Click 是对应于命令按钮控件的默认事件，所以 Visual Basic 编辑器认为需要创建一个 Click 过程。）

```
Private Sub cmdCancel_Click()
```

```
End Sub
```

- 输入程序清单 14.2 所示的与 cmdCancel_Click 过程对应的各语句。
- 在“对象”下拉列表中，选择 cmdOpen。Visual Basic 编辑器输入 Click 过程的头和尾。
- 输入程序清单 14.2 所示的与 cmdOpen_Click 过程对应的各语句。
- 自定义行 9（在 Initialize 过程中）和行 32（在 cmdOpen_Click 过程中），以便代码将能与下面所述的正在使用的应用程序一道工作。所创建的过程是要在 Excel 上运行的，所以如果要用于其他应用程序，就必须更改目标文件的路径。

对于 Word，要把 Workbooks.Open 语句改成 Documents.Open：

```
If 1stFiles.Value <> "" Then Documents.Open _  
    Filename:="z:\transfer\" & 1stFiles.Value
```

对于 PowerPoint，要把 Workbooks.Open 语句改成 Presentations.Open：

```
If lstFiles.Value <> "" Then Presentations.Open _
    Filename:="z:\transfer\" & lstFiles.Value
```

程序清单 14.2 显示了用户窗体幕后的代码的完整版本。

程序清单 14.2

```
1. Option Base 1
2.
3. Private Sub UserForm_Initialize()
4.
5. Dim strFileArray() As String
6. Dim strFFile As String
7. Dim intCount As Integer
8.
9. strFFile = Dir("z:\transfer\spreads\*.xls")
10. intCount = 1
11.
12. Do While strFFile <> ""
13.     If strFFile <> "." And strFFile <> ".." Then
14.         ReDim Preserve strFileArray(intCount)
15.         strFileArray(intCount) = strFFile
16.         intCount = intCount + 1
17.         strFFile = Dir()
18.     End If
19. Loop
20.
21. lstFiles.List() = strFileArray
22.
23. End Sub
24.
25. Private Sub cmdCancel_Click()
26.     Me.Hide
27.     Unload Me
28. End Sub
29.
30. Private Sub cmdOpen_Click()
31.     Me.Hide
32.     If lstFiles.Value <> "" Then Workbooks.Open _
        Name:="z:\transfer\spreads" & lstFiles.Value
33.     Unload Me
34. End Sub
```

程序清单 14.2 包含了出现在对应于 frmOpen_a_Current_File 用户窗体的代码表上的所有代码：一个声明部分和三个事件过程。

在声明部分，行 1 为 Option Base 1 语句，它使代码表上所用的任何数组都从 1 而不是从 0 开始编号。行 2 是空白行。

以下说明在 UserForm_Initialize 过程中有何事发生（行 3 至 23）：

- ◆ 行 3 开始这个对应于该用户窗体的 Initialize 过程。行 4 是空白行。

- ◆ 行 5 声明字符串数组变量 strFileArray。行 6 声明字符串变量 strFFile。行 7 声明整型变量 intCount。行 8 是空白行。

- ◆ 行 9 把在放有若干带扩展名.xls 的文件的指定文件夹上（这里是 z:\transfer\spreads\）进行目录操作的结果，指定给 strFFFile。
- ◆ 行 10 把 intCount 设置为 1。注意：如果不使用 Option Base 1 声明，就需要将 intCount 设置为 0（或与所用的另一个 Option Base 对应的值）。对 Dir 的第一次调用，以一个参数来说明路径名，它返回在文件夹中找到的第一个文件（假定它至少找到一个文件）。随后的每次调用都没有参数，它返回文件夹中下一个文件，直至 Dir 不再找出文件为止。
- ◆ 行 11 是空白行。行 12 到行 19 包含一个 Do While...Loop 循环，它在 strFFFile 不为空字符串 ("") 时运行：
- ◆ 行 13 把 strFFFile 与用于表示文件夹（或目录，在 DOS 中表示目录）的单句号和双句号进行比较，确认它是不是文件夹。如果 strFFFile 不是文件夹，行 14 就使用 ReDim Preserve 语句将 strFileArray 数组的大小增加到 intCount 中的数量，同时在数组中保持当前信息，这样来构建文件夹中的文件的列表。
- ◆ 行 15 将 strFFFile 的当前内容指定给 strFileArray 数组的 intCount 下标。
- ◆ 行 16 随后使 intCount 加 1，行 17 将 strFFFile 设置成 Dir 函数的结果（与指定文件夹中*.xls 型式匹配的第一个文件名）。
- ◆ 行 18 终止 If 条件。行 19 包含 Loop 关键字，只要 Do While 语句为 True，循环就将继续。行 20 是空白行。
- ◆ 当循环终止时，行 21 将对话框中 lstFiles 列表框的 List 属性设置成 strFileArray 的内容，它现在包含文件夹中所有文件的列表。
- ◆ 行 22 是空白行，行 23 结束该过程，行 24 又是一个空白行。
以下说明在 cmdCancel_Click 过程中有何事发生（行 25 至行 28）：
 - ◆ 行 25 开始 cmdCancel_Click 过程，行 28 结束该过程。
 - ◆ 行 26 隐藏用户窗体，使用 Me 关键字来引用它。
 - ◆ 行 27 使用用户窗体从存储器中卸出。
- 以下说明在 cmdOpen_Click 过程中有何事发生（行 30 至行 34）：
 - ◆ 行 30 开始 cmdOpen_Click 过程，行 34 结束它。
 - ◆ 行 31 再次使用 Me 关键字，隐藏用户窗体。
 - ◆ 行 32 进行检查，以确认 lstFiles 列表框的 Value 属性是不是空字符串 ("")，如果不是，就使用 Documents 集合中的 Open 方法来打开列表框中选中的文件。这条语句把列表框的 Value 属性添加到路径（z:\transfer\spreads\）以产生完整的文件名。
 - ◆ 行 33 使用用户窗体从存储器中卸出。

使用应用程序的内置对话框

某些应用程序，例如 Word 和 Excel，让人通过 VBA 来使用它们的内置对话框。如果内置对话框能提供所需要的功能，使用它会是一个好的解决方法：不必再构建自定义对话框，只需要把内置对话框连接到代码就行了；不必去调试这种对话框，过程用户通过在应用程序中的工作，就会熟悉这种对话框。

显示内置对话框

为了显示一个内置对话框，需要知道它的名称和常量，也必须决定使用哪种方法来显示该对话框。

找出对话框名称和常量 中夹书文回取古，矮途育处暗用周尤善由司翻。(书文个一

Word 的内置对话框是按常量来识别的，这些常量以几个字母 wdDialog 开头，后面跟着对话框的名称。对话框名称可从要求显示对话框的菜单命令中导出。例如，要引出“打开”对话框，可以使用常量 wdDialogFileOpen，由于需选择“文件”>“打开”才能显示该对话框。又如，要显示“打印”对话框（“文件”>“打印”），可使用常量 wdDialogFilePrint；而要显示“选项”对话框（“工具”>“选项”），可使用常量 wdDialogToolsOptions。

Excel 遵从一个类似的但更带弹性的规则。Excel 的内置对话框也是按常量来识别的，这些常量以几个字母 xlDialog 开头，后随对话框的名称。对话框名称可从要求显示对话框的菜单命令中导出，或从对话框的标题中导出。例如，要引出“打开”对话框，可使用常量 xlDialogOpen（不是 xlDialogFileOpen）。

找出所需要的内置对话框的名称的最容易办法是查看从 Word 或 Excel 可进入的 VBA 帮助文件中的“内置对话框参数列表”主题。从 Visual Basic 编辑器中，选择“帮助”>“Microsoft Visual Basic 帮助”，然后搜索“内置对话框参数”即可。这些表包括可以在对话框中设置的参数。

注意：显示出对象浏览器，再在“搜索”文本框内键入 wddialog 或 xldialog（视情况而定），也可以查看到内置对话框列表。

在使用 Dialogs 属性时使用这些常量，就返回 Dialogs 集合对象，它包含主应用程序中的所有内置对话框。例如，要返回 Word 的“另存为”对话框，并使用 Show 方法来显示它，应使用如下语句：

```
Dialogs(wdDialogFileSaveAs).Show
```

注意：在 Word 中，Dialogs 集合是一个可创建对象，可以直接访问它而无需通过 Application 对象。在 Excel 中，Dialogs 集合不可创建对象，必须总是使用 Application 对象来访问——如 Application.Dialogs(xlDialogOptions-General).Show。

在 Show 方法和 Display 方法之间做出选择

VBA 提供了两种方法，可以在屏幕上显示内置对话框：Show 和 Display。

◆ Show 方法显示指定的 Dialog 对象，然后执行用户在对话框中采取的行动。例如，使用 Show 方法来显示 wdDialogFileSaveAs 对话框时，用户在文件名框内为文件输入名称，并单击“保存”按钮，VBA 便以给定的名称（以及用户选择的任何选项）在指定的文件夹内保存该文件。

◆ Display 方法在屏幕上显示对话框，但不执行用户在对话框中采取的行动。但是，它允许从用户退出的对话框中返回设置，并为自己的目的使用这些设置。

显示 Word 对话框的特定标签

如果想要显示的对话框有标签，可以指定 DefaultTab 属性来显示选择的标签。使用对话框的名称，加上 Tab 这个字以及标签的名称来引用该标签。例如，对应于“项目符号和编号”对话框的常量是 wdDialogFormatBulletsAndNumbering，而对应于“编号”标签的常量是 wdDialogFormatBulletsAndNumberingTabOutlineNumbered。类似地，“字体”对话框被引用为 wdDialogFormatFont，而它的“字符间距”标签被引用为 wdDialogFormatFontTabCharacterSpacing。使用如下语句可以显示这个标签：

```
With Dialogs(wdDialogFormatFont)
    .DefaultTab = wdDialogFormatFontTabCharacterSpacing
    .Show
End With
```

要得到所有标签常量的列表，在对象浏览器中搜索 wdWordDialogTab 即可。

使用 Show 方法来显示对话框并执行行动

Show 方法显示指定的对话框，并执行用户在对话框中采取的行动。当需要用户在某一过程运行时实施惯常的交互式行动时，Show 是一个有用的方法。举一个简单的例子：在一个过程中需要对当前文档实施一些格式化任务，那就要先检查一下，确认文档已经打开，然后才能实施格式化操作；如果没有打开文档，就需显示“打开”对话框，让用户能够打开一个文件。（在显示“打开”对话框之前，可能已有说明此问题的消息框出现。）程序清单 14.3 列出了对应于该过程这一部分的代码。

程序清单 14.3

```
1. If Documents.Count = 0 Then
2.     Proceed = MsgBox("There is no document open." _ 
        & vbCrLf & vbCrLf & _
        "Please open a document for the procedure to work on.", _
        vbOKCancel + vbExclamation, "Format Report")
3.     If Proceed = vbOK Then
4.         Dialogs(wdDialogFileOpen).Show
5.         If Documents.Count = 0 Then End
6.     Else
7.     End If
8. End If
9. 'rest of procedure here'
```

看看代码是如何工作的：

- ◆ 行 1 检查 Documents 集合的 Count 属性，看看文档是否打开；如果没有文档打开，运行行 2 至行 8 的各语句。
- ◆ 行 2 显示一个消息框，告知用户没有文档打开，并要求用户为所工作的过程打开一个

文档。消息框有“确定”按钮和“取消”按钮，并将被选按钮保存在变量 Proceed 中。

- ◆ 行 3 检查“确定”按钮是否被选中；如果是的话，行 4 显示“打开”对话框，以便让用户选择文件，当用户在“打开”对话框内单击“打开”按钮时，VBA 将打开用户选择的文件。
- ◆ 此时，如果用户在“打开”对话框内单击“取消”按钮，可以取消这个过程，所以行 5 再次检查 Documents 集合的 Count 属性，如果发现仍无文档打开，便使用 End 语句来终止过程的执行。
- ◆ 如果“确定”按钮未被选择，执行就从行 3 移到行 6 的 Else 语句；行 7 的 End 语句使过程的执行终止。
- ◆ 行 8 包含对应于嵌套的 If 语句的 End If 语句；行 9 包含对应于外层 If 语句的 End If 语句。
- ◆ 行 10 为注释行，指明过程的其余部分从此处起运行，当然，只有文档是打开的才能到达此处。

使用 Display 方法来显示对话框

与 Show 方法不同，Display 方法显示一个内置对话框，但不执行用户在对话框中采取的行动。不过可以返回用户在对话框中做出的设置，并在过程中使用想要用的任何一种设置。用户能够与其熟悉的各种对话框一道工作，但是只使用确实为过程所需要的那些设置。

例如，常常需要找出某一过程应该是在哪一個文件夹内工作，当用户希望操控很多文档，并且需要知道这些文档的位置时，便是如此。为了找出这些文件夹，当然可以显示一个简单的输入框，提示用户键入通往该文件夹的正确路径——如果用户知道此路径并能正确键入的话。另一个可能的解决办法是显示一个包含驱动、文件夹和文件的树形结构的列表框；这样做需要选定一个数组，将各个文件夹和各文件的名称填入数组，而且，每当用户在树形结构内上下移动时，都得使显示更新——这工作量太大了。但是，使用具有内置树形结构的内置对话框（例如“打开”对话框），并返回设置以为己用，可以很轻松地达到相同的结果。

提示：如果需要执行使用 Display 方法显示的内置对话框中的设置（例如，为了实现用户的选择，需要检查用户的选定范围），可以使用 Execute 方法。

在内置对话框中设置和恢复选项

大多数 Word 和 Excel 内置对话框都有若干参数，可以使用这些参数在对话框中返回或设置某些值。例如，Word 中的“打开”对话框有对应于 Name、ConfirmConversions、ReadOnly、LinkToSource、AddToMru（添加文档至文件菜单根部的“最近用过的文档列表”）、PasswordDoc 等的一些参数。有些参数是在“打开”对话框中就能见到的选项；另一些参数则与在“选项”对话框的各种标签上可以找到的选项相关联。从对话框中相应控件的名称上可以猜测出某些参数的名称，但还有些名称却并不直接有关系。使用“VBA 帮助”中的“内置对话框参数列表”主题，可以找到这些名称。

例如，下述语句设置 Word 中“另存为”对话框内“文件名”文本框的内容，然后显示该对话框：

```
With Dialogs(wdDialogFileSaveAs)
    .Name = "Yellow Paint Primer"
    .Show
End With
```

如果要更改对话框中令人费解的设置，一个好办法是把它们改成用户下次使用对话框时不会得到不期望的结果的那些设置。

返回用户在对话框中选择的按钮

为了找出用户在对话框中单击了哪一个按钮，需检查 Show 方法或 Display 方法的返回值。返回值如下：

返回值	单击过的按钮
-2	关闭
-1	确定
0	取消

1 第一个命令按钮
2 第二个命令按钮
 >2 此后的命令按钮

例如，如果用户在对话框中单击了“取消”按钮，可能会将一个过程取消：

```
If Dialogs(wdDialogFileOpen).Show = 0 Then End
```

为对话框指定超时

在某些应用程序中，包括在 Word 中，某些内置对话框只有在指定时间内才显示，而不让它们在用户退出之前一直保持打开。为此，可在 Show 方法或 Display 方法中使用 TimeOut 变体参数。可以把 TimeOut 指定为若干单位，每个单位约为一毫秒。（如果系统还有很多其他任务要忙，可把这些时间设长一些。）这样，使用如下语句，就可以使“选项”对话框的“用户信息”标签显示约 10 秒钟——足以让用户检查名称设置并在必要时更改它：

```
With Dialogs(wdDialogToolsOptions)
    .DefaultTab = wdDialogToolsOptionsTabUserInfo
    .Show (15000)
End With
```

警告：TimeOut 不为自定义对话框工作，只为内置 Word 对话框工作。而且，某些内置 Word 对话框——例如“新建”对话框（wdDialogFileNew）和“自定义”对话框（wdDialogToolsCustomize）——不响应 TimeOut。

对于类似于本例中的用户名称之类的不重要信息来说，让对话框在超时时退出是很有好处的。因为即使用户离开了计算机，也允许过程继续运行。同样，因为过程会在“另存为”对话框中建议使用一个可行的文件名称，但又允许用户在场时弃用这个名称，所以，也希望让“另存为”对话框在超时时退出。但是，如果在某一过程中用户的输入是至关重要的，那么就不希望使用 TimeOut 参数。

```

    With Dialogs(WizardPictureSaveAs)
        .Name = "Yellow Paint Printer."
        .Caption = "Save Picture"
        .EndWith = 0
    End With

```

第 15 章 生成复杂对话框

- ◆ 什么是复杂对话框
- ◆ 更新对话框以反映用户的选择
- ◆ 显示和隐藏对话框的某些部分
- ◆ 生成多页对话框
- ◆ 使用 TabStrip 来驱动对话框
- ◆ 生成无模型对话框
- ◆ 使用用户窗体事件进行工作

简单对话框常被看做是静态对话框，很多复杂对话框则是动态的。当用户单击复杂对话框内的某些成分时，对话框以动态方式产生变化。这些变化可包括如下内容：

- ◆ 应用程序改变对话框中的信息以反映用户已经做出的选择。例如，如果用户已经选择了一个特定的复选框，那么应用程序可能会使另一个复选框变为不可用，这是因为，当用户使用由第一个复选框控制的选项时，由第二个复选框控制的选项是不可用或不适用的。
- ◆ 当用户单击对话框主区中的一个按钮时，对话框将隐藏的次级选项部分显示出来。
- ◆ 应用程序使用对话框，通过显示适当的指令和激活有关的控件来保持对过程的跟踪，以及引导用户到下一步骤。在本章中，可以看到有关这一技术的一个简单例子。

本章一开始将研究如何生成动态对话框。生成动态对话框比生成静态对话框要多做一些工作，但是，对于在过程中提供信息和进行选择来说，这两种对话框都是很好的方式。

然后从动态对话框转到多页对话框。多页对话框能向用户提供更多的信息和选项。随后研究如何生成无模型对话框。

本章最后一部分将说明如何与 UserForm 对象及在该对象上使用的控件所支持的很多事件一道工作。使用事件，可以监控用户做了什么，并能采取相应的行动，甚至可以阻止用户去干一些不算好的事情。

生成和使用复杂对话框

在简单对话框就能胜任而用户又更容易使用的场合，不必使用复杂对话框。在所有过程只需要一对复选框和一个选项按钮的场合，就没有必要放进动态更新控件的多个页面——这样做也得不到什么好处。但是，为了向用户提供过程要求的灵活性，还是有必要来生成复杂对话框的（如本章开头给出的例子）。

更新对话框以反映用户的选择

更新对话框以反映用户在选项中做出的选择还是比较容易的。完成这一工作的主要工具

是 Click 事件，对话框中的大多数控件都能对 Click 事件起反应，而且可以把附属于对话框的代码表上的过程加至该事件。有些控件有不同于 Click 事件的默认事件；当使用复杂对话框时，会碰到 Change 事件，在本章后半部分还会碰到大量其他事件。

下节中的程序清单 15.1 显示了更新对话框的一个例子。

显示对话框的附加部分

隐藏复杂对话框的一部分，是使用户与对话框的初始互动得以简化的一个好办法。考虑 Word 中的“查找和替换”对话框：当它第一次显示出来时（通过选择“编辑”>“替换”，或按下 Ctrl+H 键），能看到的只是如图 15.1 上图中显示出来的对话框的那部分。（如果选择“编辑”>“查找”，看到的也是同样较小的对话框部分。）如果需要使用比“查找和替换”对话框所提供的更复杂的选项，可以单击“高级”按钮，以显示出对话框的下部分，如图 15.1 下图所示。

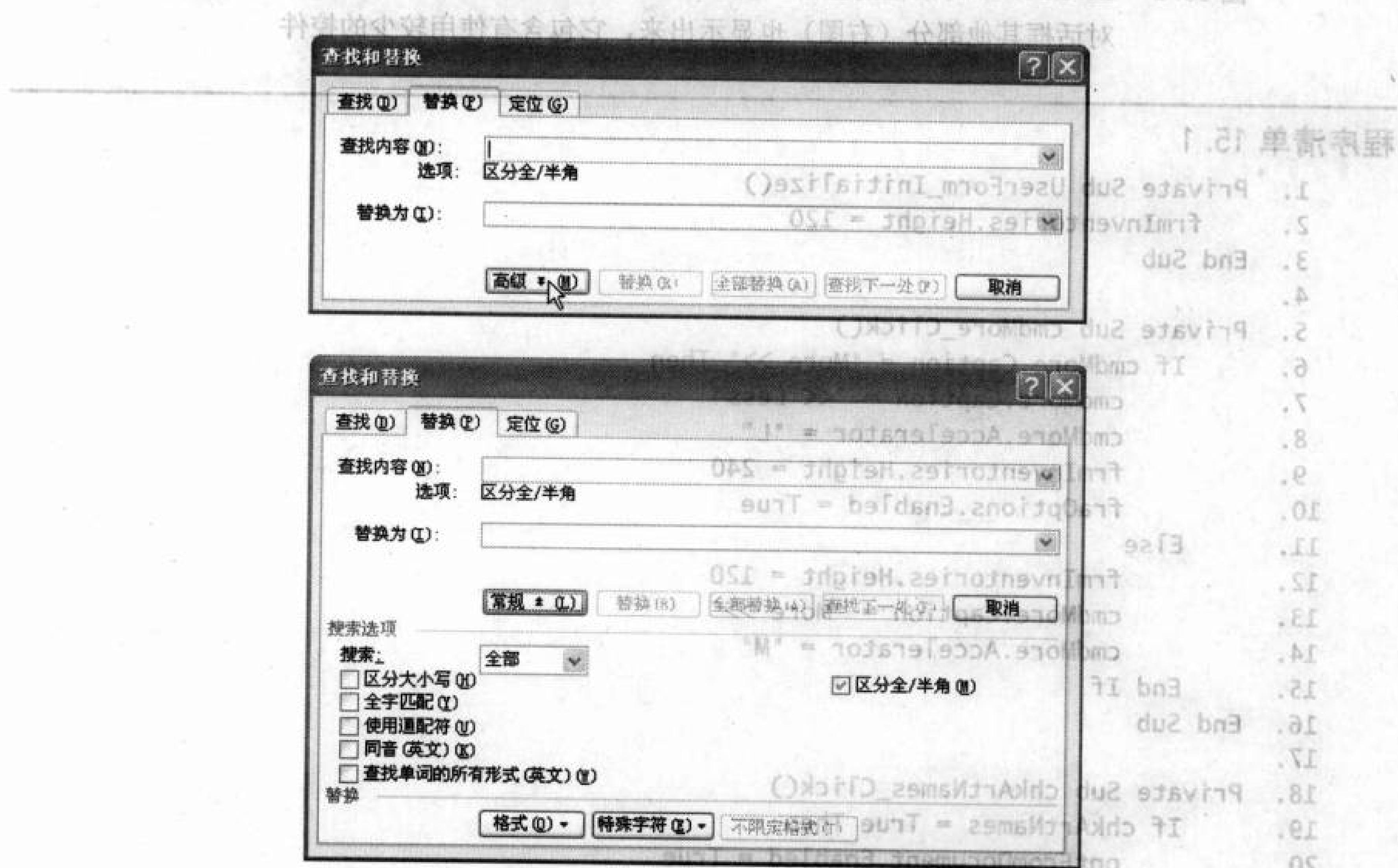


图 15.1 Word 的“查找和替换”对话框隐藏了部分选项（上图），
单击“高级”按钮则可显示其下一半（下图）

有些复杂对话框包含有一个行动子集，大多数用户只在这个子集上投入大量时间，对于这些复杂对话框，可以采纳类似的方法。为此，可以使用两项技术，即分离或级联：

- ◆ 将 Visible 属性设置为 False，以便将出现在对话框的已显示部分的某一控件隐藏起来。想要显示该控件时，则把 Visible 属性设置为 True。
- ◆ 增加对话框的高度或宽度（或高宽都增加）以显示出包含更多控件的区域。

提示： 使用上述任一技术时，特别希望在不显示某些控件时，把这些隐藏着的控件的 Enabled 属性设置为 False，这样，用户就不能移到这些他们看不到的控件上。