第 5 章　VBA 语句

VBA 程序由各种各样的语句构成，其中常用的语句有赋值语句和注释语句，赋值语句用于为程序中的变量保存值，注释语句用于帮助理解程序，不会产生实际的编译代码。常用的语句还有输入语句和输出语句，还可以在程序中控制使用暂停语句暂停程序的运行，使用退出语句终止程序的运行。本章的主要内容和学习目的有：

- 认识 VBA 中的语句，并学习语句的书写规则；
- 理解赋值语句的作用，学习使用赋值语句为变量赋值；
- 掌握 InputBox 函数的使用方法，能够提供用户的输入信息；
- 掌握 MsgBox 函数的使用方法，能够输出程序的结果或者返回提示信息；
- 掌握 Stop 语句和 End 语句，能够使用其暂停或终止程序的运行。

5.1　VBA 中的语句

任何一种程序设计语言都有一整套严格的编程规范。在进行代码的编写前，应该了解这些规则，使自己的代码符合这些规则，这样才能被正确地识别和执行。赋值语句和注释语句是 VBA 程序中经常用到的两类语句。程序功能的实现离不开变量和属性的赋值，为了使大型程序便于阅读，程序中也会经常出现注释语句。本节将介绍 VBA 的语句特点以及复制语句和注释语句的使用方法。

5.1.1　什么是语句

VBA 的语句是执行具体任务的指令，是 VBA 的方法、属性、函数、表达式和所有能被识别的组合。编写代码时必须遵循的规则称为语法。VBA 为了方便语句的输入，提供了语句自动格式化功能，其能够在输入 VBA 语句后，自动按照一定的规则对语句进行简单的格式化。

VBA 语句的自动格式化，包括关键字首字母自动大写、运算符前后自动输入空格以及删除语句中多余的空格等。在书写 VBA 程序代码时，必须遵循一些基本规则。遵循语句书写的基本规则，能够使程序的结构清晰、便于理解且方便调试。

一般情况下，程序中的一个语句占用一行。在 VBA 中，也可以将几个语句放在同一行中构成一个复合语句。复合语句中的各个语句使用冒号“：”来分隔。复合语句代码如下所示。

```
Debug.Print 30 : Debug.Print 31:Debug.Print 32
```

在 Visual Basic 编辑器的“代码”窗口中，每行 VBA 代码可以包含 1023 个字符，但有时语句过长，需要换行，此时可使用续行符来实现。续行符是一个空格后面加一个下划线“_”，其具体的使用方法如下所示。

```
myDocument.Shapes.Range(Array(1, 3)).Fill.Patterned _
msoPatternHorizontalBrick
```

在编写程序代码时，关键字、变量名、常量名、过程名之间一定要使用空格来进行分隔，并且应该使用缩进格式。具有缩进格式的程序有更强的可读性，能够反映程序代码的逻辑关系和嵌套关系，示例如下所示。

```
If IsNumeric(TextBox1.Text) Then                    '判断文本是否为数字
   TempNum = CInt(TextBox1.Text)                    '复合语句
   If TempNum >= 0 And TempNum <= 100 Then          '嵌套 If 语句
       ScrollBar1.SmallChange = TempNum
   Else
    TextBox1.Text = ScrollBar1.SmallChange
   End If
Else                                                '不满足条件时执行
   TextBox1.Text = ScrollBar1.SmallChange
End If
```

【范例 5-1】 通过示例熟悉 VBA 程序中语句的书写规范。本程序将实现对工作表中合并单元格个数的统计，同时将以提示对话框的形式显示结果，代码如下所示。

```
Sub 统计合并单元格()
    Dim rng As Range, rngA As Range, i As Byte '声明变量
    For Each rng In ActiveSheet.UsedRange       '遍历使用过的单元格
        If rng.MergeCells Then                  '如果包含合并单元格
            If rng.Address = Split(rng.MergeArea.Address, ":")(0) Then
                                                '获取单元格地址后
                If rngA Is Nothing Then         '如果对象变量为空
                    Set rngA = rng              '合并单元格赋予对象变量
                Else
                    Set rngA = Application.Union(rngA, rng)
                                                '合并单元格区域
                End If
                i = i + 1                       '计数变量加 1
            End If
        End If
    Next
    MsgBox "当前工作表中共有：" & i & "个合并单元格！" _
    & Chr(10) & "合并单元格的地址为：" & Chr(10) & rngA.Address
                                                '显示提示
End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口输入上述代码，按 F5 键运行程序，程序给出提示对话框，对话框显示工作表中合并单元格的数量和地址，如图 5.1 所示。

【代码解析】本示例程序代码用来演示程序中语句的书写规范。在程序中，诸如 Dim、MsgBox 和 If 等关键字，无论输入时是大写还是小写，在输入完成后均会自动更改大小写。程序的第 15~16 行提示对话框的文字内容比较多，在“代码”窗口中可以一行输入，为了阅读方便，也可以使用“_”来对程序进行换行，以多行输入。在程序中，For Each…In Next

循环结构中使用了多重的 If 结构的嵌套。代码在输入时使用缩进格式，这样能够使程序条理清晰，读者能够很容易地理解各层嵌套之间的关系。

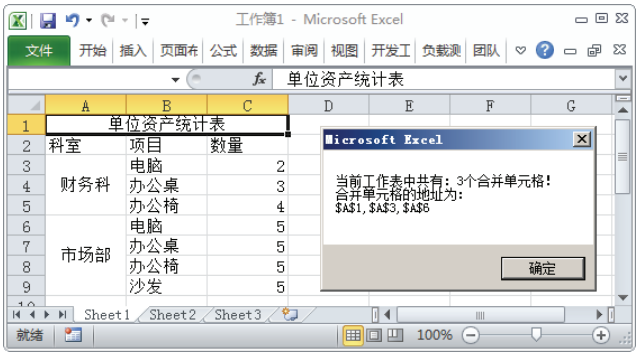

图 5.1　程序运行的效果

提示：程序中，如果 MergeCells 属性值为 True，表示区域包含合并单元格。MergeArea 属性能够返回一个 Range 对象(即工作表对象)代表包含指定单元格的合并区域。第 05 行代码中使用 Slipt 函数来截取符合条件的单元格地址。第 09 行代码使用 Application 对象的 Union 方法来获得合并单元格区域。

5.1.2　使用赋值语句

赋值语句是 VBA 程序中最基本最常用的语句。赋值语句的作用是对表达式进行运算，同时将运算的结果赋予其左侧的变量或对象属性。赋值语句实际上在前面的各章示例中都曾经用到，它的语法格式如下：

```
[Set]<变量名>=<表达式>
```

参数说明如下所示。

- Set：可选。赋值关键字，常常省略不写。
- 变量名：必需。变量或对象属性的名称。
- 表达式：必需。赋给变量或对象属性的值。

在 VBA 中，使用赋值语句应该注意赋值运算符左边只能是变量名或对象属性，不能是常量、表达式。赋值语言执行时，先对右边的表达式进行计算，然后将结果赋给左边的变量名或对象属性。在赋值语句中，关键字 Set 一般省略。语句中的变量名必须遵循标识符命名规则。只有当等号右侧的表达式是一种与左侧变量兼容的数据类型时，赋值才会成功。例如，不能将数值变量的值赋给字符串型的变量。否则在程序编译时就会出错。

注意：赋值语句中的“=”称为赋值符号，其用于对变量的赋值而非数学中通常理解的等于。如在 VBA 中，经常可以看到这种在数学中不可能出现的表达式“m=m+1”，表达式是将变量 m 当前值加 1 后将结果赋于变量 m。

【范例 5-2】 使用赋值语句实现修改选择单元格文字的字体、字号和颜色等。代码如

下所示。

```
01  Sub 设置单元格文字属性()
02      Selection.Font.Name = "黑体"          '设置字体
03      Selection.Font.Size = 22              '设置字号
04      Selection.Font.Bold = True            '设置为黑体
05      Selection.Font.Underline = True       '设置文字带有下划线
06  End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口输入上述代码，按 F5 键运行程序。被选择单元格中文字被设定为程序指定的样式，如图 5.2 所示。

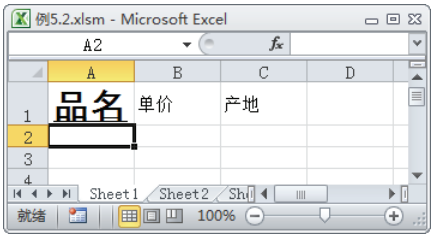

图 5.2　程序运行的效果

【代码解析】本示例程序代码用来设置单元格中文字的样式。代码中的 Name、Size、Bold 和 Underline 是 Font 对象的属性，这里使用赋值语句为这些对象属性赋值，将单元格中的文字设定为指定的样式。

注意：本段代码体现了一种最常见也是最基本的程序结构，那就是顺序结构。这种结构的程序在执行时是从上向下顺序执行的。顺序结构是比较简单的结构，比较符合人们日常生活中大多数发生的事情。

5.1.3　使用注释语句

在程序中，为了增强程序的可读性，方便程序的维护，往往需要为特定的语句添加各种说明，这种说明文字就是注释语句。注释语句对于读者来说并不陌生，在前面章节中编写有关程序代码时，已经多次使用了注释语句。注释语句一般使用下面两种格式：

```
'注释文本
```

或

```
Rem 注释文本
```

在使用单撇号（'）来添加注释语句时，只需要在注释语句前加上单撇号（'）即可，注释语句结束处不需要添加单撇号（'）。注释语句使用的示例如下面代码所示。

```
Selection.Font.ColorIndex= xlAutomatic '将文字颜色设置为自动颜色
```

使用 Rem 关键词来注释语句，应注意需要语句在程序语句和注释语句间加上冒号(:)，代码如下所示：

```
Selection.Font.ColorIndex= xlAutomatic : Rem 将文字颜色设置为自动颜色
```

提示：注释语句在程序中不会产生执行代码，其只是作为对程序的说明而存在。调试程序时，在不希望执行的代码前添加注释符号使其不被执行，这是调试程序查找语句错误的一个常用技巧。

【范例 5-3】 本实例演示注释语句的作用。其功能是在工作表中选择单元格区域，运行程序将自动选择工作表中除选择区域以外的已用单元格区域，代码如下所示。

```
01  Sub 反向选择单元格区域()
02      Application.DisplayAlerts = False   '禁用警告提示
03      Application.ScreenUpdating = False  '禁用屏幕刷新
04      Dim a As String, t As String        '声明变量
05      a = Selection.Address               '获得选区地址
06      t = ActiveSheet.UsedRange.Address   '获得已使用单元格地址
07      With Sheets.Add                     '添加一个新工作表
08          .Range(t) = 0                   '新工作表中对应已用地址单元格赋值
09          .Range(a) = "=0"                '新工作表中对应选区单元格赋值
10          a = .Range(t).SpecialCells(xlCellTypeConstants, 1) _
11          .Address                        '将变量 a 设置为含有常量的单元格地址
12          .Delete                         '删除新工作表
13      End With
14      ActiveSheet.Range(a).Select         '反向选择已用单元格区域
15      Application.ScreenUpdating = True   '重新启用屏幕刷新
16      Application.DisplayAlerts = True    '重新启用警告提示
17  End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口输入上述代码。切换到 Excel 2010 工作表中，选择单元格区域，如图 5.3 所示。切换回 Visual Basic 编辑器，在“代码”窗口中单击，将输入点光标放置于程序中。按 F5 键运行程序，切换到 Excel 2010 工作表中，可以看到已用单元格中未被选择的单元格区域被选择，如图 5.4 所示。

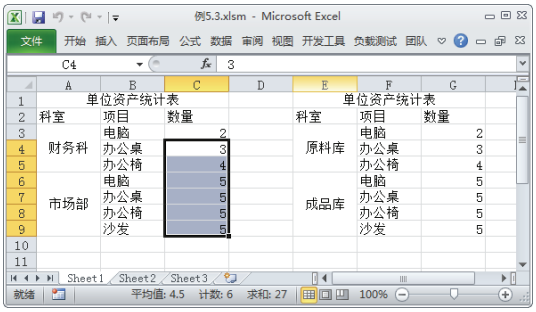

图 5.3　选择单元格区域

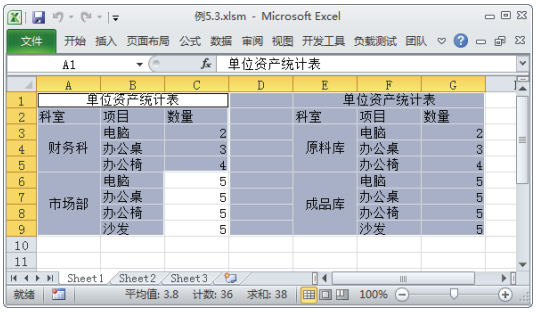

图 5.4　非选择单元格被选择

【代码解析】本示例程序代码用来演示注释语句在程序中的使用方法和意义。在这段程序中，代码较多，同时程序实现的流程较为复杂，为了能够理解代码，添加注释是一个好方法。为了使注释便于阅读，每行语句后添加注释时，可以使用 Tab 键来对齐注释。

5.2　数据的输入和输出

程序是用来处理数据的，在处理数据时，程序需要知道处理什么数据以及处理的结果如何告知用户，这就是数据的输入和输出的问题。在基于 Excel 的 VBA 程序中，数据输入和输出的方式很多，如使用工作表和用户窗体等。本节将介绍 VBA 中常见的数据输入和输出方法。

5.2.1　输入对话框

在 VBA 中，InputBox 函数能产生一个接受用户输入的对话框，用户可以在对话框中输入需要的数据，数据将传递给程序。Input 函数的语法格式如下：

```
InputBox (prompt[, title] [,default] [,Xpos] [,Ypos] [,helpfile,context])
```

参数说明如下所示。

- prompt：该参数的值是对话框中显示的提示信息，一般使用的是字符串表达式，字符串最大长度是 1024 个字符。
- title：该参数决定对话框标题栏中显示的内容，其值为字符串型数据。省略该参数，对话框标题栏将显示应用程序名。
- Default：该参数对话框中输入文本框中显示的字符串。省略该参数，输入文本框为空。
- Xpos：该参数决定了对话框的上边界距离屏幕左侧的水平距离。
- Ypos：该参数决定了对话框的上边界距离屏幕上边界的垂直距离。
- helpfile：该参数用于设置对话框的帮助信息，此参数可以省略。
- context：该参数用于设置对话框帮助主题编号，此参数可以省略。

【范例 5-4】 编程创建一个用于输入产品编号的文本框，代码如下所示。

```
01  Sub 使用 InputBox 函数()
02      Dim msg As String, title As String
03      Dim default As String, MyValue As String   '声明变量
04      msg = "请输入产品编号"                      '对话框的提示信息
05      title = "产品编号输入系统"                  '对话框标题栏文字
06      default = "A0132008803"                    '输入文本框的默认值
07      MyValue = InputBox(msg, title, default, 100, 150)  '显示输入对话框
08      Debug.Print MyValue                        '显示输入值
09  End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口输入上述代码，按 F5 键运行程序。在屏幕的左上角会显示一个输入对话框，如图 5.5 所示。在文本框中输入数据后，单击“确定”按钮关闭对话框，可以在“立即窗口”中查看输入的值，如图 5.6 所示。

【代码解析】在代码中，InputBox 函数需要的参数放置于变量中，这样便于程序的修改。在第 07 行代码中 InputBox 函数中的数字参数 100 和 150 指定了对话框显示在屏幕上

的位置。由于设置了函数的 default 参数，运行时对话框中输入框的默认值是蓝色显示，此时只需要按键即可开始输入。

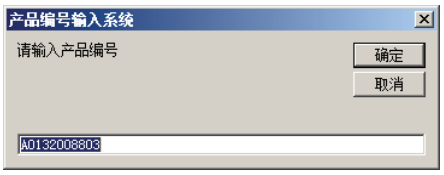

图 5.5　程序运行时获得的输入对话框

图 5.6 “立即窗口”中显示的输入值

注意：在默认情况下，InputBox 函数的返回值是字符串类型数据，如果需要使用该函数来输入数字，需要使用 Val 函数将返回值转换为相应数据类型。如果单击了对话框中的“取消”按钮，则对话框将返回一个空字符串，可以根据这个空字符串来判断用户是否有输入。InputBox 函数一次只能实现一个数据的输入，如果需要要输入多个数据，则需要多次调用该函数。

5.2.2　提示对话框

对话框是实现程序与用户交互的常用方法，使用 MsgBox 函数能产生一个对话框用于显示信息。通过单击对话框中的按钮，将能够返回数值表明用户单击的是哪个按钮，而程序也将继续执行。MsgBox 函数使用的语法格式如下：

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

MsgBox 函数的参数和 InputBox 函数的参数的用法相同，这里不再赘述。这里要说明的是，如果不需要通过返回值获得用户单击的按钮，可以直接使用函数而不要上述语句中的对参数赋值部分。

在 MsgBox 函数中增加了一个 buttons 参数，该参数用于指定按钮显示的数目、样式和消息对话框显示的图标样式等。要了解有哪些常量可以使用，可以在 VBA 自带的帮助文档查询 MsgBox 的提示信息。在帮助文档中查询到的部分 buttons 参数的意义，如图 5.7 所示。

【范例 5-5】 使用消息对话框显示单元格中输入的内容，对话框包含“重试”按钮和“取消”按钮，对话框使用警告样式，代码如下所示。

```
01  Sub 使用MsgBox函数()
02      Dim x As String                                    '声明变量
03      x = Selection.Value                                '获取单元格中的文字
04      y = MsgBox("你选择的单元格中的文字是" & "“" &
05      x & "”", vbRetryCancel + vbExclamation, "提示")'显示提示信息
06      Debug.Print y                                          '显示单击的按钮的值
07   End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口输入上述代码，按 F5 键运行程序。在屏幕的左上角会显示一个对话框。对话框包含“重试”和“取消”按钮，同时将显示选择单元格中文字内容，如图 5.8 所示。单击“重试”按钮关闭对话框，在“立即窗口”中

可以查看“重试”按钮对应的返回值，如图 5.9 所示。

常数	值	描述
vbOKOnly	0	只显示 OK 按钮。
VbOKCancel	1	显示 OK 及 Cancel 按钮。
VbAbortRetryIgnore	2	显示 Abort、Retry 及 Ignore 按钮。
VbYesNoCancel	3	显示 Yes、No 及 Cancel 按钮。
VbYesNo	4	显示 Yes 及 No 按钮。
VbRetryCancel	5	显示 Retry 及 Cancel 按钮。
VbCritical	16	显示 Critical Message 图标。
VbQuestion	32	显示 Warning Query 图标。
VbExclamation	48	显示 Warning Message 图标。
VbInformation	64	显示 Information Message 图标。
vbDefaultButton1	0	第一个按钮是缺省值。
vbDefaultButton2	256	第二个按钮是缺省值。
vbDefaultButton3	512	第三个按钮是缺省值。
vbDefaultButton4	768	第四个按钮是缺省值。
vbApplicationModal	0	应用程序强制返回；应用程序一直被挂起，直到用户对消息框作出响应才继续工作。
vbSystemModal	4096	系统强制返回；全部应用程序都被挂起，直到用户对消息框作出响应才继续工作。
vbMsgBoxHelpButton	16384	将Help按钮添加到消息框

图 5.7　帮助信息中的 buttons 参数列表

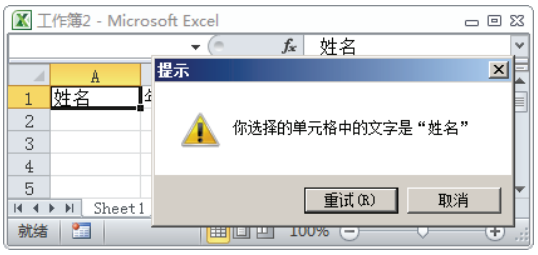

图 5.8　程序运行时获得的提示对话框

图 5.9　“立即窗口”中显示的结果

【代码解析】本段代码用于演示 MsgBox 函数的使用特点。为了使单元格中文字在对话框中显示时能够放置于引号中，在设置 MsgBox 的显示信息文字时，使用&运算符来连接左引号“ “”、单元格中文字内容（即变量 x 值）和右引号“ ””。在设置 MsgBox 函数的 button 参数值时，如果需要使用多个按钮，可以像示例中那样用“＋”连接多个常量来进行设置。单击对话框中的不同按钮对应的返回值如图 5.10 所示。这个返回值同样可以在 VBA 自带的帮助文档中查到。

返回值

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

图 5.10　按钮对应的返回值

5.2.3　显示程序运行结果

Print 方法可用于在程序调试时显示程序运行结果，这个结果在 Visual Basic 编辑器的“立即窗口”中显示。在 VBA 中，Print 方法的语法结构如下所示：

```
Debug.Print 显示内容
```

【范例 5-6】 编程在“立即窗口”中显示 3 个间隔的 o(∩_∩)o，代码如下所示。

```
01  Sub Print方法的使用()
02      Debug.Print "o"; "("; "∩"; "_"; "∩"; ")"; "o", "o"; "("; _
03       "∩"; "_"; "∩"; ")"; "o", "o"; "("; "∩"; "_"; "∩"; ")"; "o"
                                                    '显示第一行图案
04      Debug.Print Spc(8); "o"; "("; "∩"; "_"; "∩"; ")"; "o"; _
05      Spc(3); "o"; "("; "∩"; "_"; "∩"; ")"; "o"
                                                    '显示第二行图案
06      Debug.Print Spc(14); "o"; "("; "∩"; "_"; "∩"; ")"; "o"
                                                    '显示第三行图案
07      Debug.Print Spc(8); "o"; "("; "∩"; "_"; "∩"; ")"; "o"; _
08      Spc(3); "o"; "("; "∩"; "_"; "∩"; ")"; "o"
                                                    '显示第四行图案
09      Debug.Print "o"; "("; "∩"; "_"; "∩"; ")"; "o", _
10      "o"; "("; "∩"; "_"; "∩"; ")"; "o", "o"; "("; "∩"; "_"; "∩"; ")";
"o"                                                 '显示第五行图案
11  End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口输入上述代码，按 F5 键运行程序。在“立即窗口”中可以看到程序运行的结果，如图 5.11 所示。

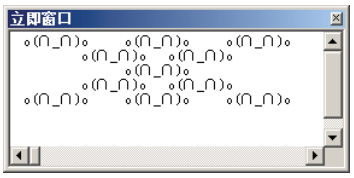

图 5.11　程序运行的效果

【代码解析】在前面章节的很多范例中都使用了 Print 方法在“立即窗口”中显示运行结果，本段代码最大的亮点在于使用了分隔符来格式化输出的数据。本例中使用了两个分隔符，一个是分号（;），其用于将前后两个数据连接在一起输出。“立即窗口”中显示的每一个 o(∩_∩)o 都是用分号(;)连接各个字符得到的。第二个使用的分隔符就是逗号(,)，其以 14 个字符为一个输入区，将每个数据输出到对应的输入区中。代码中就是使用逗号(,)创建了 3 个输入区，得到了规则排列的输出效果。

显示的第二行图案，只需要打印 2 个 o(∩_∩)o。在第 04~05 行代码中，使用 Spc（8）在这一行的前面添加 3 个空格后再拼接 2 个 o(∩_∩)o，并使它们的间隔为 3 个空格。同理，根据 o(∩_∩)o 摆放的位置，分别添加空格来使它们对齐，最终产生图 5.11 的显示效果。

提示：对于 Print 方法来说，除了上面介绍的 2 个分隔符外，其还包括下面 2 个分隔符：一个是 Spc(n)，将其放置于 2 个需显示的字符之间，可以在这 2 个字符间插入 n 个空格；另一个是 Tab(n)，该分隔符能够移动光标，就像按 Tab 键一样，光标移动的距离由 n 决定。

5.3　程序的中断

程序开始运行后，有时需要查看运行到某个语句时的运算中间值，这时就需要暂停程序。而当程序结果满足某个条件时，有时又需要结束程序的运行。要在程序中实现上面提到的两个功能，需要使用暂停语句和退出语句。

5.3.1　暂停程序

当需要程序在执行到某个语句暂停时，可以在该语句处放置一个 Stop 语句使程序的运行暂停。这里要注意的是，Stop 语句是暂停程序的运行而不是退出程序，因此其不会关闭程序，变量也不会被清除，在需要时可以从暂停处开始继续程序的运行。

Stop 语句相当于在程序中设置了一个断点，因此其常用在程序的调试过程中，下面举一个示例来了解 Stop 语句在程序中的作用。

【范例 5-7】 使用 Stop 语句来实现循环的暂停，代码如下所示。

```
01  Sub 使用暂停语句()
02      Dim n As Integer                                    '声明变量
03      For n = 1 To 10
04          n = n + 1                                       '变量加 1
05          Debug.Print n                                   '显示变量值
06          Stop                                            '暂停程序
07      Next
08  End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口输入上述代码，按 F5 键运行程序，程序运行到 Stop 语句处会暂停，在“代码”窗口中会标示出程序暂停的位置。同时，在“立即窗口”中可以看到当前 n 的值。再次按 F5 键程序将继续运行，在“代码”窗口中再次显示新的 n 的值，如图 5.12 所示。

图 5.12　程序运行的效果

【代码解析】本段程序将显示 Stop 语句所起的作用。在程序运行时，运行到循环体中的 Stop 语句时，程序将暂停，暂停前 Print 语句在“立即窗口”中显示 n 的当前值。此时，变量 n 的值并没有清除，当程序结束暂停再次运行时，变量的值将在原有值的基础上增加 2。

5.3.2　停止程序

在 VBA 中，可以使用 End 语句来停止程序的运行。此时，程序的运行将被终止，返回到 VBA 编辑器。此时，VBA 程序将卸载所有窗体，在没有其他程序引用当前程序的公共类模块所创建的对象且无代码执行的情况下，变量内存将被清空，程序将立即关闭。下面对范例 5-7 的程序代码进行修改，使用 End 语句来停止程序的运行，比较 End 语句与 Stop 语句的不同。

【范例 5-8】 使用 End 语句来实现循环的暂停，代码如下所示。

```
01  Sub 退出程序()
02      Dim n As Integer                          '声明变量
03      For n = 1 To 10
04          n = n + 1                             '变量加 1
05          Debug.Print n                         '显示变量值
06          End                                   '退出程序
07      Next
08  End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口输入上述代码，按 F5 键运行程序，在“立即窗口”中可以看到当前 n 的值。再次按 F5 键程序将重新启动，此时在“代码”窗口中显示的 n 的值仍然是 2，如图 5.13 所示。

图 5.13　程序运行的效果

【代码解析】在程序运行时，变量 n 的初始值为 0。在运行第 05 行的 End 语句前的语句后“立即窗口”中显示 n 的值为 2。执行 End 语句时，程序将退出，变量将被清除。当再次运行这段程序时，程序中变量的初始值仍然为 0，与上一次程序运行一样执行的是相同的语句，因此得到的结果与第一次执行的结果相同。

提示：实际上在 VBA 中，使用 Quit 方法同样能够退出程序，但此时将退出 Excel 系统而不只是回到 VBA 编辑器。在使用此方法时，如果处于打开状态的工作簿还没有保存，Excel 会给出提示对话框提示保存文档。

5.4　小　　结

本章介绍了 VBA 编程的基础知识，包括 VBA 的语法规则、赋值语句和注释语句的使用要点、数据的输入和输出的方法以及在代码中结束程序运行的方法。通过本章的学习，读者将能熟练地对变量进行赋值，能够使用 InputBox 函数实现用户数据的输入，能够采用不同的方法输出程序处理结果，同时能够在程序调试中灵活使用 Stop 语句来暂停程序。

到目前为止，读者已经能够使用 VBA 来解决很多问题，特别是数值计算的问题。但仅仅掌握这些是不够的，还需要掌握各种程序结构及其实现方式，才能编写功能更为强大的应用程序。第 6 章中，我们将一起学习 VBA 程序的常见控制结构，在 VBA 编程技术上更上层楼。

5.5　本 章 习 题

1．下列语句哪个在程序中会被视为注释？（　　）

A．声明变量　　B．'start A　　C．--sleep　　D．“join me”

2．下面哪个语句能够显示一个标题栏为“我的输入框”的输入对话框？（　　）

A．a=InputBox("请在此输入数值","我的输入对话框",30)

B．a=InputBox("我的输入对话框",)

C．a=InputBox("我的输入对话框",30)

D．a=InputBox("我的输入对话框",30,30)

3．下面哪个语句能够获得一个提示对话框，对话框的标题为“重要提示”，提示内容是“输入错误！”，对话框包含“是”和“否”按钮。（　　）

A．a=MsgBox("重要提示","输入错误！",vbYesNo)

B．a=MsgBox("重要提示","输入错误！",vbOKCancel)

C．a=MsgBox("输入错误！",vbYesNo, "重要提示")

D．a=MsgBox("输入错误！",vbOKCancle, "重要提示")

4. 下面哪个语句能够在“立即窗口”中显示同一行中 3 个等距排列字符“OK”。（　　）

A．Debug.Print "OK"_"OK"_"OK"　　B．Debug.Print "OK";"OK";"OK"

C．Debug.Print OK,OK,OK　　D．Debug.Print "OK","OK","OK"

5．编写一个简单的乘方计算器，用户输入底数和指数，运算结果在提示对话框中显示。

【提示】代码中需要注意的是 InputBox 函数返回的是字符串数据，而运算符^对数值型数据进行计算。因此这里使用 Val 函数来进行数据类型的转换。

6．编写一个程序，使程序在“立即窗口”中绘制数字三角形，绘制三角形使用的数字由用户设置。

【提示】此练习主要巩固 InputBox 函数的使用方法和 Print 方法的分隔符的使用技巧。