

第 2 章 VBA 语法基础

和其他编程语言类似，VBA 程序代码也是有基础语句组成的。在本章中，将结合具体的例子来介绍 VBA 语法的基础内容。主要包括输入、输出语句、分支结构和循环结构等。在讲解各例子的时候，将详细分析例子的应用范围。

2.1 输入和输出语句

在本小节中，将详细讲解最基础的语法结构：顺序结构。顺序结构不需要使用特殊的控制语句，编辑工具按照用户编写的程序语句依次编译，依次执行。本小节中，将详细讲解最基础的顺序语句：输入和输出语句。

案例 8 输出数据表

1. 功能说明

在 Excel VBA 中，用户可以使用 Print 方法输入各种类型的数据。同时，用户可以在程序代码中设置不同的输出，来检测程序代码。

2. 语法说明

在 Excel VBA 中，Print 方法组要应用于 Debug 对象，其语法格式如下：

```
Debug.Print [outputlist]
```

参数 outputlist 表示需要打印的表达式或列表。如果省略参数，则打印空白行。Print 方法先计算表达式的数值，然后输出结果。在 outputlist 参数中还可以使用分隔符，格式化输出的结果，分隔符有以下几种：

- Spc(n)：插入 n 个空格到输出数据之间；
- Tab(n)：移动光标到适当位置，n 为移动的列数；
- 分号：表示前后两个数据项连在一起输出；
- 逗号：以 14 个字符为一个输出区，每个数据输出到对应的输出区。

3. 案例说明

本例使用 Print 方法向 VBE 的立即窗口中输出数据表。

4. 编写代码

输出数据表的代码如下：

```
Sub GetNumbers()  
Dim i As Integer
```

```

Dim j As Integer

For i = 1 To 5
    For j = 1 To i
        Debug.Print i; "+"; j; "="; i + j; "  ";
    Next
    Debug.Print
Next
End Sub

```

5. 运行结果

按功能键“F5”运行子过程，在“立即窗口”输出数据计算表，如图 2.1 所示。

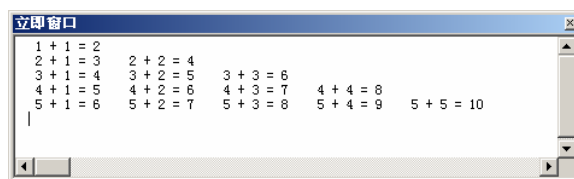


图 2.1 输出结果

6. 程序分析

在上面的程序代码中，利用循环结构依次输出不同情况的数字计算结果。关于循环结构的知识，将在后面章节中详细讲解。

案例 9 输入用户名

1. 功能说明

在 Excel VBA 开发过程中，经常需要和用户进行交互。例如，很多程序代码的结果依赖于用户输入的信息。这个时候，用户需要调用输入函数，提供用户输入信息，并接收用户所输入的信息。

2. 语法说明

在 Excel VBA 中，用户可以使用 InputBox 函数输入信息。该函数将打开对话框，用户可以在对话框中输入数据，并返回所输入的内容。其语法格式如下：

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

各参数的含义如下：

- **Prompt:** 对话框消息出现的字符串表达式。最大长度为 1024 个字符。如果需要在对话框中显示多行数据，则可在各行之间用回车符换行符来分隔。
- **Title:** 对话框标题栏中的字符串。如果省略该参数，则把应用程序名放入标题栏中。
- **Default:** 显示在文本框中的字符串。如果省略该参数，则文本框为空。

- **Xpos:** 和 **Ypos** 成对出现, 指定对话框的左边与屏幕左边的水平距离。如果省略该参数, 则对话框会在水平方向居中。
- **Ypos:** 和 **Xpos** 成对出现, 指定对话框的上边与屏幕上边的距离。如果省略该参数, 则对话框被放置在屏幕垂直方向距下边大约三分之一的位置。
- **Helpfile:** 设置对话框的帮助文件, 可省略。
- **Context:** 设置对话框的帮助主题编号, 可省略。

3. 案例说明

在本例中, 将演示使用 **InputBox** 函数对话框输入信息, 然后将信息输出到立即窗口中。

4. 编写代码

在模块中输入以下代码:

```
Sub CetUserName()  
Dim Title As String  
Dim Name As String  
Dim StrName As String  
  
    Title = "输入用户名"  
    Name = "在选框中输入用户名: "  
  
    StrName = InputBox(Name, Title)  
  
    Debug.Print "用户名: "; StrName  
  
End Sub
```

5. 运行结果

(1) 按功能键“F5”运行子过程, 将弹出“输入个人信息”窗口, 如图 2.2 所示。在对话框中输入内容后单击“确定”按钮。

(2) 在“立即窗口”中将输出这些内容, 如图 2.3 所示。

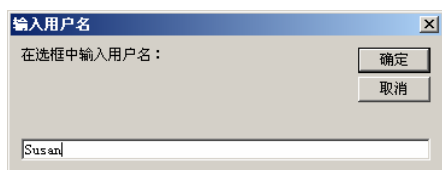


图 2.2 输入用户名



图 2.3 输出结果

6. 程序分析

本案例比较简单, 直接使用 **InputBox** 函数来输入信息。对于 **InputBox** 函数的其他参数, 都直接使用默认数值。在实际开发中, 用户可以设置这些参数的数值, 实现不同的功能。

案例 10 事件确认

1. 功能说明

在 Excel VBA 编码和调试代码的过程中，经常需要获取代码运行的阶段性结果或者信息。例如，在编写数据处理的代码中，用户需要跟踪某变量的数值变化。因此，需要在程序进行的不同地方，显示该变量的数值。

2. 语法说明

在 Excel VBA 中，用户可以使用 **MsgBox** 函数来显示提示信息。**MsgBox** 函数可打开对话框，显示提示信息。并根据用户选择对话框中的按钮，执行不同的程序代码。其语法格式如下：

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

通过函数返回值可获得用户单击的按钮，并可根据按钮数值而选择程序段来执行。函数有 5 个参数，各参数的意义与 **Inputbox** 函数参数的意义类似。

对于 **buttons** 参数，其含义是指定显示按钮的数目及形式、使用提示图标样式以及默认按钮等。其常数值如表 2.1 所示。

表 2.1 按钮常数值

常量	值	说明
vbOkOnly	0	只显示“确定”(Ok)按钮
vbOkCancel	1	显示“确定”(Ok)及“取消”(Cancel)按钮
vbAbortRetryIgnore	2	显示“异常终止”(Abort)、“重试”(Retry)及“忽略”(Ignore)按钮
vbYesNoCancel	3	显示“是”(Yes)、“否”(No)及“取消”(Cancel)按钮
vbYesNo	4	显示“是”(Yes)及“否”(No)按钮
vbRetryCancel	5	显示“重试”(Retry)及“取消”(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	以第一个按钮为默认按钮
vbDefaultButton2	256	以第二个按钮为默认按钮
vbDefaultButton3	512	以第三个按钮为默认按钮
vbDefaultButton4	768	以第四个按钮为默认按钮
vbApplicationModal	0	进入该消息框，当前应用程序暂停
vbSystemModal	4096	进入该消息框，所有应用程序暂停

表 2.1 中的数值（或常数）可分为四组，其作用分别为：

- 第一组值（0~5）：确定对话框中按钮的类型与数量。

- 第二组值（16, 32, 48, 64）：确定对话框中显示的图标。
- 第三组值（0, 256, 512）：设置对话框的默认活动按钮。
- 第四组值（0, 4096）：确定消息框的强制响应性。

buttons 参数由上面 4 组数值组成，其组成方法是：从每一类中选择一个值，把这些数值组合起来，就是 buttons 参数的值。

3. 案例说明

本例的主要功能是当用户激活工作表 Sheet1 时，Excel 显示激活的信息。

4. 编写代码

- （1）进入 VBE，在代码窗口左上方的对象列表中选择“Worksheet”，如图 2.4 所示。
- （2）在代码窗口右上方的事件列表中选择“Activate”，如图 2.5 所示。

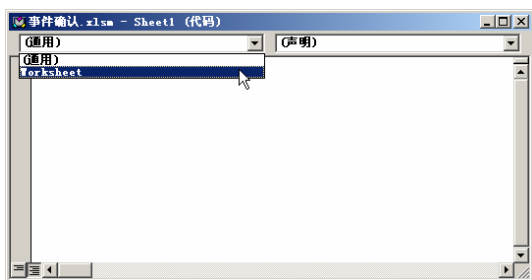


图 2.4 对象列表

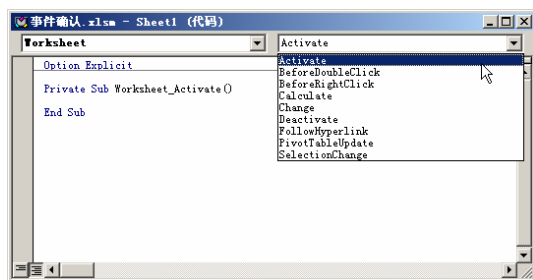


图 2.5 事件列表

- （3）在上面生成的事件过程中输入以下代码：

```
Private Sub Worksheet_Activate()  
MsgBox ("你激活了 Sheet1 工作表！")  
End Sub
```

5. 运行结果

打开工作簿，然后选择 Sheet1 工作表，结果如图 2.6 所示。

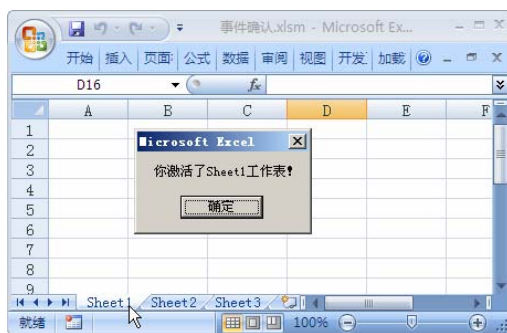


图 2.6 显示的提示信息

6. 程序分析

在本例中，使用的是 `MsgBox` 函数的最简单形式，也就是直接使用 `MsgBox` 函数显示一个字符串信息。在实际开发中，用户可以设置各参数，得到不同的显示结果。

2.2 选择结构

在 Excel VBA 中，除了常见的顺序结构之外，还有一种常见结构：选择结构。选择结构的程序将根据给定的条件选择执行后续的代码。选择结构在日常生活中应用十分广泛，在本小节将结合具体的例子来讲解如何使用选择结构。

案例 11 判断优秀员工

1. 功能说明

在实际开发中，当用户只需要进行一种情况的判断时，最方便的方法是 `If ... Then` 语句。`If` 后面的语句就是判断的条件，`Then` 后面的语句是需要执行的操作。

2. 语法说明

在 Excel VBA 中，使用 `If...Then` 语句可有条件地执行语句。其语法格式如下：

```
If 逻辑表达式 Then  
    语句 1  
    语句 1  
    ... ..  
    语句 n  
End If
```

逻辑表达式可以是计算数值的表达式，VBA 将为 0 看作为 `False`，而非零数值都被看作 `True`。该语句的执行顺序是：当逻辑表达式的值是 `True`，则执行位于 `Then` 与 `End If` 之间的语句；当逻辑表达式的值是 `False`，则不执行 `Then` 与 `End If` 之间的语句，直接跳出循环结构，其流程图如图 2.7 所示。

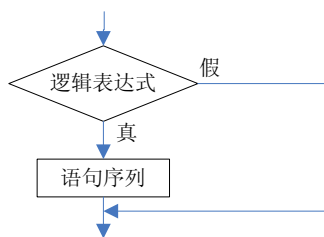


图 2.7 If ... Then 语句流程图

If ... Then 结构还有单行结构条件语句。其语法格式如下：

If 逻辑表达式 Then 语句

该语句的功能为：当逻辑表达式的值是 True，则执行 Then 后的语句；当逻辑表达式的值是 False，则不执行 Then 后的语句。

3. 案例说明

某公司统计了员工上个月的销量，根据销量数值，公司判定销量数值大于 350 的员工为优秀员工。对于优秀员工将其对应的单元格标红，原始数据如图 2.8 所示。



	A	B	C	D
1	员工编号	销量		
2	111	169		
3	112	271		
4	113	367		
5	114	322		
6	115	364		
7	116	418		
8	117	123		
9	118	396		
10	119	403		

图 2.8 原始数据

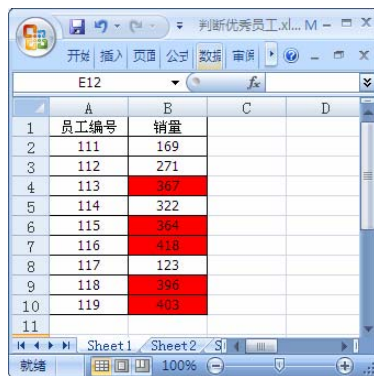
4. 编写代码

判断优秀员工的代码如下：

```
Sub ShowExcel()  
  
    Dim i As Integer  
  
    For i = 2 To 10  
        If Sheets(1).Cells(i, 2).Value > 350 Then  
            Cells(i, 2).Interior.ColorIndex = 3  
        End If  
    Next  
  
End Sub
```

5. 运行结果

打开工作簿，然后运行程序代码，结果如图 2.9 所示。



	A	B	C	D
1	员工编号	销量		
2	111	169		
3	112	271		
4	113	367		
5	114	322		
6	115	364		
7	116	416		
8	117	123		
9	118	396		
10	119	403		
11				

图 2.9 运行结果

6. 程序分析

从上面案例的结果中可以看出，当程序处理的只有一个条件时，使用 If...Then 语句可以很便利的解决问题。

案例 12 根据编号分组

1. 功能说明

当用户在实际开发时，经常需要进行多条件判断。例如，当满足条件 1 时，执行第一种操作；当不满足条件 1 时，执行另外一种操作。当用户需要编写这些类型的代码时，前面案例中的 If ... Then 语句将无法满足，用户需要使用 If ... Then ... Else 语句。

2. 语法说明

在 Excel VBA 中，用户可以使用 If ... Then ... Else 语句，根据条件是否成立分别执行两段不同的代码，其语法格式如下：

```
If 逻辑表达式 Then  
    语句序列 1  
Else  
    语句序列 2  
End If
```

该语句的执行过程是：当逻辑表达式的值是 True 时，将执行“语句序列 1”中的各条语句；当“逻辑表达式”的值为 False 时，就执行“语句序列 2”中的各条语句，其流程图如图 2.10 所示。

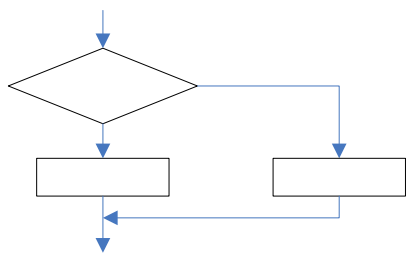


图 2.10 If Then Else 语句流程图

3. 案例说明

某公司统计了员工上个月的销量，公司需要根据员工编号判断员工的组别。当编号是偶数的时候，员工属于女组；当编号是奇数时，员工属于男组，原始数据如图 2.11 所示。

逻辑表达式
真
语句序列1

	A	B	C	D
1	员工编号	销量	组别	
2	111	169		
3	112	271		
4	113	367		
5	114	322		
6	115	364		
7	116	418		
8	117	123		
9	118	396		
10	119	403		

图 2.11 原始数据

4. 编写代码

分组的程序代码如下：

```
Sub ShowTeam()  
    Dim i As Integer  
    For i = 2 To 10  
        If Sheets(1).Cells(i, 1).Value Mod 2 Then  
            Cells(i, 3).Value = "男组"  
        Else  
            Cells(i, 3).Value = "女组"  
        End If  
    Next  
End Sub
```

5. 运行结果

打开工作簿，运行程序代码，得到的结果如图 2.12 所示。



	A	B	C	D
1	员工编号	销量	组别	
2	111	169	女组	
3	112	271	男组	
4	113	367	女组	
5	114	322	男组	
6	115	364	女组	
7	116	418	男组	
8	117	123	女组	
9	118	396	男组	
10	119	403	女组	
11				

图 2.12 运行结果

6. 程序分析

从上面的典型例子中可以看出，当在实际情况中需要根据某条件执行两种不同的操作时，可以使用 If ... Then ... Else 语句便利的完成任务。

案例 13 计算消费金额

1. 功能说明

在实际开发中，用户可能需要处理多条件问题。例如，某公司根据多个销量，将折扣分为多个档次。在计算销售金额的时候，需要判断具体销售属于哪个档次，然后根据该档次的折扣值，计算销售金额。这个时候，用户可以使用 Select Case 语句。

2. 语法说明

在 Excel VBA 中，Select Case 结构的语法格式如下：

```
Select Case 测试表达式
Case 表达式列表 1
    语句序列 1
Case 表达式列表 2
    语句序列 2
    ...
Case Else
    语句序列 n
End Select
```

根据上面的程序结构，代码首先计算“测试表达式”的值，然后将表达式的值与结构中的 Case 的值进行比较。如果相等，就执行与该 Case 语句下面的语句块，执行完毕再跳

转到 End Select 语句后执行，其流程图如图 2.13 所示。

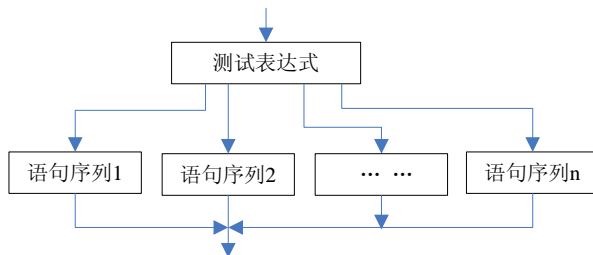


图 2.13 Select Case 语句流程图

在 Select Case 结构中，“测试表达式”通常是数值或字符型的变量。“表达式列表”可以是一个或几个值的列表。如果在列表中有多值，需要用逗号将各值分开。表达式列表可以分下面几种情况：

- 表达式：表示具体的取值。例：Case 5,8,9。
- 表达式 A To 表达式 B：表示数据范围。例，Case 8 To 12 表示 8~12 之间的值。
- Is 比较运算符 表达式：表示范围。例，Case Is>45 表示所有大于 45 的值。
- 以上三种情况的混合。例，Case 8 To 12, 17, Is>35。

3. 案例说明

某销售公司根据商品的销量实行不同的折扣，公司需要根据销量和价格，计算其销量的总额，其中原始数据如图 2.14 所示。

	A	B	C	D	E
1	价格	数量	总额		
2	15	100			
3	20	150			
4	30	200			
5	35	250			
6	20	300			
7	15	150			
8	24	100			
9	30	200			
10	25	400			
11					

图 2.14 原始数据

4. 编写代码

计算销量金额的代码如下：

```
Sub GetIncome()
```

```

Dim IntSale As Integer
Dim discount As Single
Dim SingleMoney As Single
Dim i As Integer
Dim IntPric As Integer
For i = 2 To 10

    IntPric = Cells(i, 1).Value
    IntSale = Cells(i, 2).Value

    Select Case IntSale
        Case Is <= 100
            discount = 0.95
        Case Is <= 150
            discount = 0.85
        Case Is <= 200
            discount = 0.7
        Case Is <= 300
            discount = 0.65
        Case Else
            discount = 0.6
    End Select

    SingleMoney = IntSale * IntPric * discount
    Cells(i, 3).Value = SingleMoney

Next i
End Sub

```

5. 运行结果

打开工作簿，运行程序代码，得到的结果如图 2.15 所示。



	A	B	C	D
1	价格	数量	总额	
2	15	100	1425	
3	20	150	2550	
4	30	200	4200	
5	35	250	5687.5	
6	20	300	3900	
7	15	150	1912.5	
8	24	100	2280	
9	30	200	4200	
10	25	400	6000	

图 2.15 计算所销售金额

6. 程序分析

用户可以使用 If ... Then ... Elseif 结构来重新编写上面的案例，然后和 Select Case 结构进行比较，可以发现 Select Case 结构在处理多条件的情况下，要简洁很多。

2.3 循环结构

循环结构是 Excel VBA 中经常使用的一种程序结构，当用户需要使用程序代码反复完成同一任务的时候，则需要使用循环结构。在本小节中，将结合具体例子来讲解如何使用循环结构。

案例 14 计算自然数之和

1. 功能说明

当用户在进行循环运算的时候，有时可能了解具体循环的次数。这个时候可以使用 For...Next 循环语句依次完成循环运算。其中，最典型的例子就是计算自然数的总和。当用户需要计算自然数之和时，循环次序已经由自然数的大小决定。

2. 语法说明

在 Excel VBA 中，如果知道循环的次数，可以使用 For...Next 循环语句来执行循环。For 循环的语法如下：

```
For 循环变量=初始值 To 终值 [Step 步长值]
    语句序列 1
    [Exit For]
    [语句序列 2]
Next [循环变量]
```

在上面的结构中，循环变量控制循环，每重复一次循环之后，循环变量的值将以步长值相加。步长的默认数值是 1，并且可正可负。如果步长值为正，则初始值必须小于等于终值，才执行循环体。如果步长值为负，则初始值必须大于等于终值，才能执行循环体。For...Next 循环结构的流程图如图 2.16 所示。

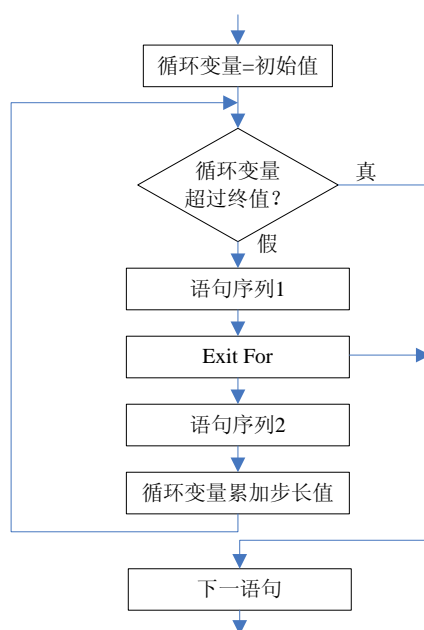


图 2.16 For ...Next 流程图

3. 案例说明

根据循环结构计算 $1+2+3+\dots+1000$ 的数值。

4. 编写代码

计算自然数之和程序代码如下：

```
Sub GetSums()  
    Dim clock As Integer  
    Dim sum As Long  
    Dim counter As Integer  
  
    clock = 1  
    sum = 0  
    counter = 1  
    For counter = 1 To 1000  
        sum = sum + clock  
        clock = clock + 1  
    Next  
    MsgBox "1+2+3+...+1000 = " & sum, vbOKOnly, "计算自然数之和"  
End Sub
```

5. 运行结果

运行程序代码，结果如图 2.17 所示。

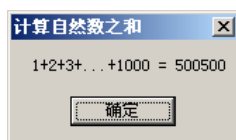


图 2.17 运行结果

6. 程序分析

从上面的结果中可以看出，由于用户计算的是自然数之和。所以，在代码中不需要设置参数 Step 的数值。

案例 15 为单元格赋值

1. 功能说明

在 Excel 中，由一种特殊的循环。在其循环结构中，其循环运算的范围是一个区域。例如，用户需要在某工作表的单元格区域中进行循环。这个时候，用户可以使用 For Each ... Next 循环语句。

2. 语法说明

在 Excel VBA 中，For Each ... Next 循环语句的语法格式如下：

```
For Each 元素 In 对象集合
    [语句序列 1]
    [Exit For]
    [语句序列 2]
Next
```

该循环结构可在对象集合每个元素中执行循环体。集合中必须至少有一个元素，才会进入 For Each 循环体。循环结构先对“对象集合”中的第一个元素执行循环语句，然后对“对象集合”其他的元素执行循环语句，当“对象集合”中的所有元素都执行结束，会退出循环。在循环体中，用户可以设置多个 Exit For 语句，退出循环。。

3. 案例说明

本例的主要功能是利用循环结构为单元格区域赋值。

4. 编写代码

为单元格赋值的程序代码如下：

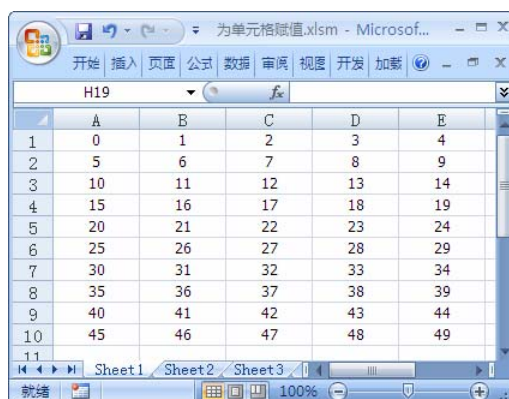
```
Sub GetValues()
    Dim counter As Integer
    Dim rng As Range

    Worksheets("sheet1").Activate
    For Each rng In Range("A1:E10")
```

```
rng.Value = counter  
counter = counter + 1  
Next  
  
End Sub
```

5. 运行结果

运行程序代码，结果如图 2.18 所示。



	A	B	C	D	E
1	0	1	2	3	4
2	5	6	7	8	9
3	10	11	12	13	14
4	15	16	17	18	19
5	20	21	22	23	24
6	25	26	27	28	29
7	30	31	32	33	34
8	35	36	37	38	39
9	40	41	42	43	44
10	45	46	47	48	49

图 2.18 为单元格区域赋值

6. 程序分析

从上面例子的结果中可以看出，循环结构首先填充第一行单元格的数值，然后依次填充其他行的数值，直到循环结束。

案例 16 计算阶乘和

1. 功能说明

在实际开发中，用户也许不会仅仅面对循环问题，或者仅仅面对选择问题。很多复杂问题都是嵌套结构，循环结构和分支结构的嵌套，或者循环结构之间的嵌套等。

2. 语法说明

在 Excel VBA 中，编写嵌套循环的代码时，要注意循环语句的配对情况。如图 2.19 所示，左图是正确的嵌套关系，Next 关闭了内层的 For 循环，而 Loop 关闭了外层的 Do 循环。同样，在嵌套的 If 语句中，End If 语句自动与最靠近的 If 语句配对。嵌套的 Do...Loop 结构的工作方式也是一样的，最内圈的 Loop 语句与最内圈的 Do 语句匹配。图 2.19 右图则是错误的嵌套关系。

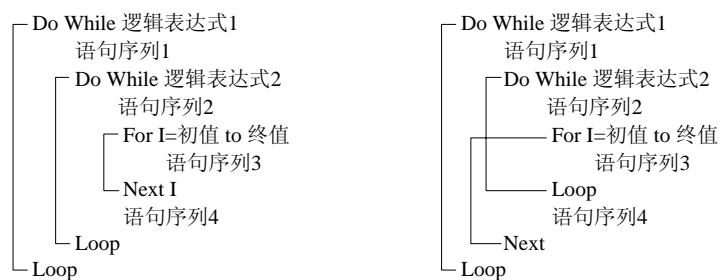


图 2.19 正确的嵌套（左）与错误的嵌套（右）

3. 案例说明

本例将演示计算 $1! + 2! + 3! + \dots + 10!$ 的结果，并输出结果。

4. 编写代码

计算结果的程序代码如下：

```
Sub SumProduct()
    Dim sum As Double
    Dim i As Integer
    Dim DouProduct As Double
    Dim j As Integer

    sum = 0
    For i = 1 To 10
        DouProduct = 1
        For j = 1 To i
            DouProduct = DouProduct * j
        Next
        sum = sum + DouProduct
    Next
    MsgBox "1! + 2! + 3! +...+ 10! = " & sum, vbOKOnly, "求阶乘的和"
End Sub
```

5. 运行结果

运行程序代码，得到的结果如图 2.20 所示。

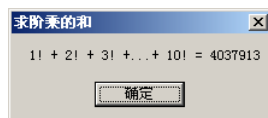


图 2.20 计算阶乘的和

6. 程序分析

使用 Excel VBA 的嵌套结构，用户可以处理许多复杂问题，只是在使用嵌套结构时，用户需要特别注意匹配问题。