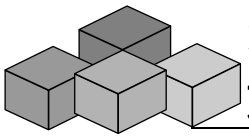

第 5 章 过程与函数设计

VBA 的结构化程序设计中，程序的执行是以过程（Subroutine）和函数（Function）为单位的。从组织结构上分，过程和函数是模块的组成部分，也就是说过程和函数包括在模块之中；同时，过程和函数是由若干条语句构成的、具有特定意图的语句集合。

一个完整的 VBA 项目，通常包括项目的用户界面，以及后台的程序单元。用户界面就是连接用户操作与后台程序的纽带。在这里面，过程和函数的调用占有举足轻重的地位。

虽然在 VBA 各种类型的模块中均可创建过程和函数，但为了便于讲述和理解，同时因为标准模块的作用范围和调用方式也是最方便的，因此本章主要介绍标准模块中的过程和函数的设计方法。

5.1 过程

VBA 中过程的关键字用 Sub 表示，每一个过程必须用 End Sub 作为过程结束标志。如果运行过程期间需要提前退出过程，可以使用 Exit Sub 语句。

5.1.1 创建过程

过程的创建通常有 3 种方法。第一种方法是使用录制宏，Excel 和 Word 各版本都具有录制宏的功能，PowerPoint 高级版本不能录制宏。

第二种方法是在 VBA 编辑器中选择“插入”→“过程”命令，弹出对话框，如图 5-1 所示。

图 5-1“添加过程”对话框

输入过程名称，并单击“确定”按钮，自动生成过程模板：

```
Public Sub Test5()

End Sub
```

以上两种方法只能生成无参数过程。

第三种方法是在模块中直接手工输入代码。对于比较熟练的 VBA 学习者，推荐使用这种直接写代码的方法。

一个 VBA 过程的完整语法格式如下：

```
Public|Private Sub 过程名（参数列表）
    语句块
End Sub
```

对于标准模块中的过程，如果过程名前面既没有 Public，也没有 Private 关键字，则默认是 Public，也就是该过程可以让本模块以外的其他模块调用；如果是 Private，则只能在本模块内部调用。

一个模块中允许书写多个过程。在 VBA 编辑器中，过程和过程之间显示一条横线作为分隔符，并且在代码窗格的右上角下拉列表框中，可以用鼠标快速定位到特定的过程（见图 5-2）。

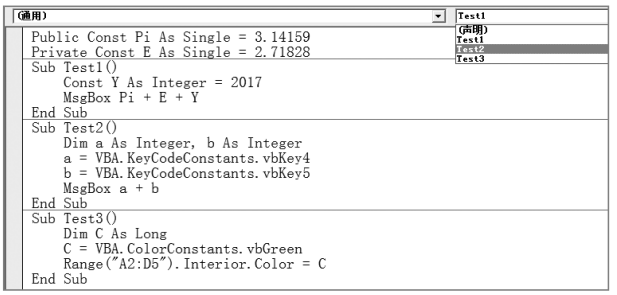

图 5-2　从下拉列表框中选择过程

过程的命名原则与之前讲过的变量命名原则完全相同，可以使用英文单词或者中文作为过程名，但不推荐使用中文作为过程名。同一个模块中不许出现同名过程，否则出现“二义性”错误。

5.1.2　过程的运行和调用

标准模块中的无参数过程，也可以称作宏（Macro）。将光标置于代码中，按下快捷键【F5】运行过程，或者按下快捷键【F8】单步执行过程。

对于窗体模块或者工作表、工作簿事件模块中的事件过程，这些过程不能直接运行，而是通过事件驱动机制，只有触发了特定对象的行为，才能运行过程。

对于带参数过程（至少 1 个参数），不可直接运行过程，必须通过其他过程或语句调用运行。在 Excel VBA 编程中，可以用 Call 或 Application.Run 方法调用其他过程。

源代码：实例文档 95.xlsm/ 过程的调用

```
1.  Public Sub Proc1()
2.      MsgBox Now
3.  End Sub
4.  Public Sub Proc2(S As String)
5.      MsgBox UCase(S)
```

```
6.  End Sub
7.  Public Sub TestCall()
8.      Call Proc1
9.      Proc1
10.     Call Proc2("vba")
11.     Proc2 "excel"
12. End Sub
```

代码分析：以上 3 个过程中，Proc1 是一个无参数过程，Proc2 带有一个参数，TestCall 过程用 Call 关键字去调用上述两个过程。

对于调用无参数过程，Call 关键字可以省略不写。调用带参数过程，使用 Call，过程名后面的参数必须用圆括号括起来。如果不使用 Call 关键字，则过程名后可以不使用圆括号，把参数依次列在后面即可。

Call Proc2 "vba" 是错误的调用方式。

如果访问和调用的过程在其他模块中，则尽量在过程名前加上过程所在的模块名。假设标准模块 m2 中有个 Proc2 过程，那么在其他模块中采用如下调用方式：Call m2.Proc2("vba") 或者 m2.Proc2 "excel" 都可以。

Excel VBA 还允许使用 Run 方法，把过程名也作为字符串传递进去，实现调用。

源代码：实例文档 95.xlsm/ 过程的调用

```
1.  Public Sub TestRun()
2.      Application.Run "Proc1"
3.      Application.Run "Proc2", "excel"
4.  End Sub
```

如果调用其他模块中的过程，别忘记加模块名。例如：

```
Application.Run "m2.Proc2", "excel"
```

如果要调用其他工作簿中的过程，还需要加上工作簿名称。例如：

```
Application.Run "实例文档 95.xlsm!过程的调用 .Proc2", "excel"
```

表示调用实例文档 95.xlsm 中的“过程的调用”模块中的 Pro.c2 过程，参数是 excel。

比较以上两种方式，更推荐使用 Call 这种形式，因为这是 VB6 本来的内置用法。

5.1.3 过程的参数

在过程中使用参数，可以让程序设计变得更加灵活、便捷。过程中的所有参数，称为参数列表。参数定义的格式如下：

```
ByRef|ByVal 参数 1 As 类型 1,参数 2 As 类型 2…
```

其中，关键字 ByRef 表示按地址传递参数，ByVal 表示按值传递参数。参数 1、参数 2 等叫作形式参数，参数的类型可以是 VBA 基本数据类型，也可以是对象型。

参数的传递方式、参数个数及其类型，要根据编程需要自行指定。

为过程定义了参数，就一定要想到在其他地方如何调用这个过程。对于带参数过程的调

用，要遵循以下两个原则。

❑ 实际参数的个数要与过程中形式参数个数相同。

❑ 实际参数的类型要与过程中形式参数的类型相同或相近。

这里以计算根据两边长和夹角计算三角形面积为例，说明一下过程中参数的定义和调用。三角形的面积公式是 $S=0.5ab\mathrm{Sin}C$，其中 a、b 是两边长，C 是这两条边的夹角（弧度制）。

可以看出，该问题的已知条件是 3 个，根据以往学过的 VBA 知识，可以写出如下普通过程：

源代码：实例文档 95.xlsm/ 过程的参数

```
1.  Sub Test1()
2.      Dim a As Single, b As Single, C As Single
3.      a = 5
4.      b = 7
5.      C = 60 / 180 * 3.14159
6.      MsgBox 0.5 * a * b * Sin(C)
7.  End Sub
```

代码分析：第 5 行代码表示∠ C 是 60° ，需要转换为弧度制。

运行结果是 15.155。

但是可以发现，如果用这个过程去计算另一个三角形的面积时，就需要重新修改原过程的代码，重新赋值，这就很不方便了。为此，可以把上述过程改成带参数的过程，把边长和角度放在过程的参数列表中。

源代码：实例文档 95.xlsm/ 过程的参数

```
1.  Sub Test2(a As Single, b As Single, C As Single)
2.      MsgBox 0.5 * a * b * Sin(C / 180 * 3.14159)
3.  End Sub
4.  Sub Test3()
5.      Call Test2(5, 7, 60)
6.      Test2 20, 20, 90
7.  End Sub
```

代码分析：Test2 是一个带有 3 个参数的过程，a、b、C 都叫作形式参数。在 Test3 中调用 Test2 过程。

第 5 行代码计算两边长为 5 和 7，夹角为 60° 的三角形面积，其中 5、7、60 都叫作实际参数，这行代码的返回结果是 15.155。

第 6 行代码计算两边长为 20 和 20，夹角为 90° 的三角形面积，返回结果是 200。

从第 5 ~ 6 行代码可以看出，这个带有参数的过程使用起来非常方便，只需要更改实际参数的数值，就可以计算另一个三角形的面积。

调用带参数过程时，一般情况下不需要说明实际参数和形式参数的匹配关系，默认是按照从左到右分别传递。例如 Call Test2(5, 7, 60)，就是把 2 传递给 a，7 传递给 b，60 传递给 C。实际上，也可以明确地说明参数的分配情况，例如：

```
Call Test2(a:=5, b:=7, C:=60)
Call Test2(a:=5, C:=60, b:=7)
```

这两行代码的运行结果是一样的，尽管下面那行代码把 b 参数指定到最后。明确参数这种表达形式要注意以下两点。

（1）如果采用明确参数的方式，所有参数都必须采用，不能部分采用。例如 Call Test2(a:=5, 7, 60) 会导致编译错误。

（2）参数名称必须与函数定义的形式参数的名称完全吻合，不能自行改动。例如 Call Test2(m:=5, n:=7, Q:=60) 是不正确的，因为 Test2 过程中不存在 m、n、Q 这些参数名称。

5.1.4 可选和默认参数

前面举过的例子，指定的都是必需参数。那么在调用这种过程，必须恰好匹配参数，参数的个数和类型要严格匹配。

还可以使用 Optional 关键字为过程指定可选参数。所谓可选参数，是指调用这种过程时，这个参数可以提供，也可以不提供。使用 IsMissing 函数可以判断一个参数是否为空。

下面的实例用于计算日平均工资。一般情况下一个月有 22 天是正常上班日，用一个月的工资数除以 22，就得到日平均工资。

源代码：实例文档 95.xlsm/ 过程的参数

```
Sub DailySalary(Total As Integer, Optional Days As Integer = 22)
    MsgBox "平均每天: " & Total / Days
End Sub
Sub Test5()
    DailySalary 4000, 20
End Sub
```

代码分析：DailySalary 过程有一个必须参数 Total 和一个可选参数 Days，Days 的默认值是 22。

在实际的调用过程中，可以使用“ DailySalary 4000, 20”计算出某月出勤 20 天，总工资 4000 元的日平均结果为 200。

如果忽略 Days 参数，例如是 DailySalary 4000 这种方式，则返回 181 元，因为不指定 Days 的情况下，按出勤天数为 22 处理。

此外，还可以使用 IsMissing 来判断是否为过程传递了参数。

下面一个实例用于计算员工一天的出勤时间计算，根据早上来公司的时刻，以及下班离开公司的时刻，就可以计算出上班总时间数，如果中午外出吃午餐，还要减去午休时间 60 分钟。

VBA 的 DateDiff 函数可以计算出两个时间差，例如，DateDiff("n",#8:00:00#,#10:00:00#) 就可以返回 120，因为 n 表示分钟。

源代码：实例文档 95.xlsm/ 过程的参数

```
Sub WorkTime(come As Date, home As Date, Optional launch)
    Dim v As Integer
    If IsMissing(launch) Then
        v = VBA.DateTime.DateDiff("n", come, home)
```

```
5.      Else
6.          v = VBA.DateTime.DateDiff("n", come, home) - 60
7.      End If
8.      MsgBox "工作时长（分钟）: " & v
9.  End Sub
10. Sub Test6()
11.     WorkTime come:=#8:10:00 AM#, home:=#5:35:00 PM#, launch:=True
12. End Sub
```

代码分析：WorkTime 过程中，launch 是一个可选参数。如果在调用过程中，不为该参数传递数值，则 IsMissing 会返回 True。参数缺失的情况下默认为该员工没有外出就餐，不扣时间。

第 11 行代码，调用过程时为 launch 指定了参数，所以 IsMissing 为 False，会扣掉 60 分钟。

Test6 的运行结果如图 5-3 所示。

图 5-3　运行结果 1

5.1.5　参数的传递方式

VBA 过程的创建，可以为每个参数指定传递方式。参数传递方式分为按引用传递（ByRef）和按值传递（ByVal）两种。如果不指定，则默认为按引用传递。

如果是按引用传递，当把实际参数传递到过程中后，被调用过程中改变形参的值，会自动修改实参的值。反之，如果按值传递，则被调用过程不能修改实参的值。

源代码：实例文档 95.xlsm/ 参数的传递方式

```
1.  Sub Test1(ByRef a As Integer, ByVal b As String)
2.      a = a * 2
3.      b = VBA.Strings.StrReverse(b)
4.      MsgBox a & b
5.  End Sub
6.  Sub Test2()
7.      Dim c As Integer, d As String
8.      c = 21
9.      d = "VBA"
10.     Test1 c, d
11.     Debug.Print c, d
12. End Sub
```

代码分析：Test1 过程中，形参 a 是按引用传递，b 是按值传递。

运行 Test2，执行到第 10 行时，调用 Test1，变量 c 传递给形参 a，a 修改为自身的两倍，c 也随之变成 42。

变量 d 传递给形参 b，b 修改为倒序排列，但由于是按值传递，因此不会造成变量 d 的改变，因此第 4 行代码的输出结果是 42ABV。

第 11 行的打印结果是 42　　VBA。

可以看出，c 发生了改变，d 没发生变化。

5.1.6 参数数量可变的过程

一般情况下，调用过程时传递的数目要与函数定义的形式参数数目相等。使用 ParamArray 关键字，可以创建参数数目不定的过程。

源代码：实例文档 95.xlsm/ 数目不定的参数

```
1.  Sub Test1(agv As Boolean, ParamArray num())
2.      Dim i As Integer
3.      Dim sum As Single
4.      For i = LBound(num) To UBound(num)
5.          sum = sum + num(i)
6.      Next i
7.      If agv = True Then
8.          MsgBox sum / (UBound(num) - LBound(num) + 1)
9.      Else
10.         MsgBox sum
11.     End If
12. End Sub
13. Sub Test2()
14.     Call Test1(True, 3, 4.2, 6)
15.     Call Test1(False, 3, 6)
16. End Sub
```

代码分析：Test1 过程的定义中，agv 是一个必需参数，当该参数为 True 时，显示多个数字的平均值，否则，显示多个数字的总和。

num() 是一个参数数组，代码中从第 4 ~ 6 行遍历参数数组的中的每一个元素，追加到 sum 中。

第 8 行代码计算平均值，其中，(UBound(num) – LBound(num) + 1) 用来获取数组 num 的长度（元素个数）。

运行 Test2 过程，执行到第 14 行时，参数列表中的 3 个数字就构成了一个数组，传递给了形参中的 num。

因此第 14 行的运行结果是 4.4，第 15 行的结果是 9。

5.1.7 数组作为参数

一般情况下，过程的形参、传递的实参，其类型通常是 VBA 基本数据类型或者是对象类型。

但有些时候，需要把数组作为一个参数整体传递。此时，定义函数时的参数类型需要指定为 Variant，调用过程时，只需要传递数组名称即可。

下面的实例用来计算数组的极差。所谓极差就是数组中最大值与最小值的差。使用工作表函数可以快速获取数组的最大值、最小值。

源代码：实例文档 95.xlsm/ 数组作为参数

```
1.  Sub Test1(v)
2.      MsgBox Application.WorksheetFunction.Max(v) - Application.
        WorksheetFunction.Min(v)
```

```
3.  End Sub
4.  Sub Test2()
5.      Dim arr(2 To 6)  As Integer
6.      arr(2) = 226
7.      arr(3) = 26
8.      arr(4) = 13
9.      arr(5) = 28
10.     arr(6) = 82
11.     Test1 arr
12. End Sub
```

代码分析：Test1 带有一个变体型参数 v，调用过程时，可以为该参数传递任意数据类型。

从第 4 行起是主调过程，第 5 ~ 10 行声明数组并为数组赋值。

第 11 行调用 Test1，并把 arr 传递给形参 v。

第 2 行返回数组的极差，结果是 213。

5.2　函数

在 VBA 中，函数（Function）分为内置函数和用户自定义函数（User Defined Function，UDF），这节主要介绍用户自定义函数的设计与应用。

VBA 中的函数与过程的差异非常小，几乎所有的过程都可以改写为函数。

函数与过程的唯一不同之处在于函数可以有返回值，而过程不能有返回值。

函数的书写格式为：

```
Public|Private Function 函数名（参数列表） As 类型
    函数体
End Function
```

其中，函数的作用范围关键字 Public、Private、参数列表，以及函数的返回值类型 As 类型，这三部分都是可有可无的。唯一不能缺失的是函数名称。

过程的创建、运行和调用以及参数方面的细节，同样适用于函数，这里不再重复讲述。

此外，中途退出函数，用的是 Exit Function，而不是 Exit Sub。

在 Excel VBA 中，用户自定义函数，还可以用于工作表的公式计算之中，而其他 Office 组件的自定义函数，只能供 VBA 语句调用。

下面创建一个最简单的函数：

```
Function Funny()

End Function
```

以上 Funny 函数只有函数框架，没有函数内容，是个空函数。该函数既没有作用范围的关键字，也没有参数列表，还没有返回类型，但是仍然可以从工作表函数列表中看到它（见图 5-4）。

图 5-4“插入函数”对话框 1

此外，通过 VBA 语句也可以调用该函数。

源代码：实例文档 96.xlsm/ 函数的创建和调用

```
1.  Sub Test2()
2.      Call Funny
3.  End Sub
```

运行 Test2 过程后，结果显示什么也不做。

5.2.1 自定义函数的返回值

自定义函数的返回值以及返回值的数据类型，往往与函数的参数有一定的关系。函数参数的类型可以是任意类型，函数的返回值类型同样也可以是任意类型。

例如计算圆柱的体积，函数的参数是圆柱的半径和圆柱的高，返回值就是圆柱的体积值。对于 VBA 初学者，直接写出自定义函数也许有些困难，但是可以先写出解决一个问题的 VBA 过程，然后把过程改写为函数即可。

源代码：实例文档 97.xlsm/ 自定义函数

```
1.  Public Sub CylinderVolume()
2.      Dim radius As Single, height As Single
3.      Dim V As Single
4.      radius = 2
5.      height = 10
6.      V = 3.14159 * radius ^ 2 * height
7.      MsgBox "圆柱的体积是: " & V
8.  End Sub
```

图 5-5 运行结果 2

运行上述过程后，结果如图 5-5 所示。

分析上述过程，该数学问题的参数就是半径和高，计算结果是体积。因此可以把一个问题的所有参数放在函数的括号内，计算结果就是函数的返回结果，而对于参数中实际参数的传递，则是函数被调用时再传递。因此可以改写为如下函数形式。

源代码：实例文档 97.xlsm/ 自定义函数

```
1.  Public Function CylinderVolume(radius As Single, height As Single) As Single
2.      CylinderVolume = 3.14159 * radius ^ 2 * height
3.  End Function
```

由于这个数学问题涉及的数据可能是浮点型的小数，因此参数和函数的返回值都是 Single。

对于有返回值的函数，必须把返回的结果赋给函数名。也就是说，上述函数第 2 行必须用 CylinderVolume 去接收计算的结果。

5.2.2 自定义函数的用途

VBA 的自定义函数，一般情况下是提供给其他 VBA 过程或函数调用的。

在其他过程中调用定义好的自定义函数时，要考虑是否需要获得自定义函数的计算结

果。如果不需要，则和调用 VBA 过程完全一样，用 Call 关键字即可。

下面的过程调用了 5.2.1 节定义的圆柱体积函数。

```
Sub Test1()
    Call CylinderVolume(3, 8)
End Sub
```

Test1 过程计算了半径为 3、高为 8 的圆柱体积，但是计算值并不赋给任何变量。也可以省略 Call 关键字，改写为：

```
Sub Test1()
    CylinderVolume 3, 8
End Sub
```

但一般情形下，有返回值的函数一定要让返回值被其他语句利用。

源代码：实例文档 97.xlsm/ 自定义函数

```
1.  Sub Test2()
2.      Dim v1 As Single, v2 As Single
3.      MsgBox CylinderVolume(2, 10) + CylinderVolume(3, 8)
4.      v1 = CylinderVolume(2, 10)
5.      v2 = CylinderVolume(3, 8)
6.      Debug.Print v2 - v1
7.  End Sub
```

代码分析：第 3 行代码计算了两个不同的圆柱体积之和。

第 6 行代码计算两个圆柱的体积之差。

对于 Excel VBA，还可以在工作表的公式中使用自定义函数。在宏所在的工作簿的工作表中，输入圆柱的基本信息，然后在要计算体积的单元格中输入公式“=CylinderVolume(B2,B3)”，或者单击 Excel 的“插入函数”对话框的“用户定义”类别，找到用户自定义函数 CylinderVolume，使用函数向导对话框输入也可（见图 5-6）。

总而言之，函数的参数可以没有，也可以是多个，而且参数的类型可以互不一样。函数的返回值只能是一个，其类型既可以是 VBA 基本数据类型，也可以是对象类型，还可以是数组。

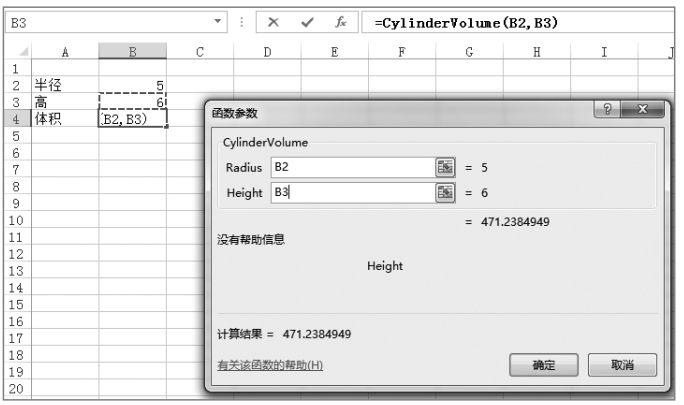

图 5-6　工作表中使用自定义函数

返回对象的函数必须用 Set 关键字为自定义函数赋值；返回数组的自定义函数必须用变体型变量去接收自定义函数。

源代码：实例文档 97.xlsm/ 自定义函数

```
1.  Public Function LastSheet() As Excel.Worksheet
2.      Dim c As Integer
3.      c = ActiveWorkbook.Worksheets.Count
4.      Set LastSheet = ActiveWorkbook.Worksheets.Item(c)
5.  End Function
```

代码分析：以上函数的功能是返回工作簿的最后一个工作表，返回类型是 Worksheet，因此在第 4 行要用 Set 关键字为函数赋值。

下面的过程调用了上述过程。

源代码：实例文档 97.xlsm/ 自定义函数

```
1.  Sub Test3()
2.      Dim w As Excel.Worksheet
3.      Set w = LastSheet
4.      w.Activate
5.  End Sub
```

代码分析：第 3 行代码调用了 LastSheet 函数，由于该函数不含参数，因此无须括号。

运行上述过程，自动激活工作簿的最后一个工作表。

下面是一个返回数组的自定义函数。

源代码：实例文档 97.xlsm/ 自定义函数

```
1.  Public Function SplitTime() As Variant
2.      Dim arr(1 To 6) As Integer
3.      Dim dt As Date
4.      dt = Now
5.      arr(1) = Year(dt)
6.      arr(2) = Month(dt)
7.      arr(3) = Day(dt)
8.      arr(4) = Hour(dt)
9.      arr(5) = Minute(dt)
10.     arr(6) = Second(dt)
11.     SplitTime = arr
12. End Function
```

代码分析：返回数组的函数，一般可以用 VBA 代码中的变体型变量去接收，从而成为一个 VBA 数组，也可以在单元格中以数组公式的形式调用自定义函数。

上述 SplitTime 函数的功能是把当前系统时间拆分为 6 部分：年、月、日以及小时、分钟、秒。这 6 部分放入一个整型数组中。

在第 11 行代码，把数组 arr 整体赋给函数名，从而使得在调用该函数时，自动获取时间的拆分结果。

为测试该函数的实用性，可以选中水平方向的 6 个单元格，然后输入数组公式“{ =SplitTime() }”，并按下快捷键【 Ctrl+Shift+Enter 】，结果如图 5-7 所示。

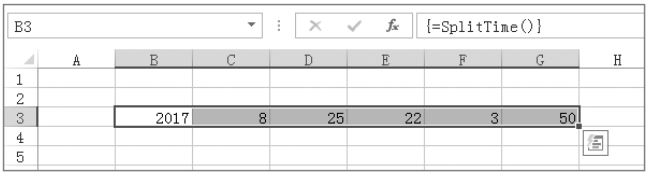

图 5-7　运行结果 3

5.2.3　设置自定义函数的说明信息

在实际编程过程中，一个自定义函数往往包含很多行代码，算法和逻辑也相当复杂，例如世界三大智力游戏之一的数独（Sudoku）问题，就可以理解为函数，函数的参数就是数独题目的初始画面，而函数的返回值就是数独的最终解（见图 5-8）。

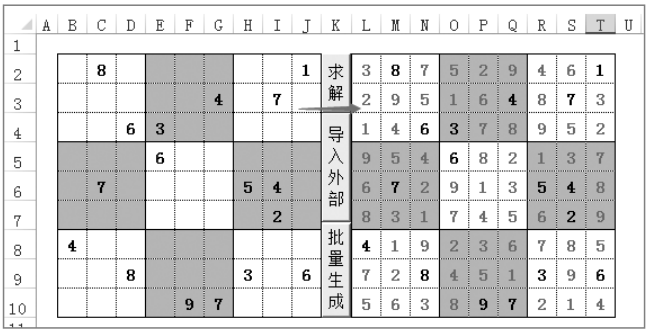

图 5-8　数独

作为自定义函数的开发者，一定要确保函数的正确性。自定义函数通过测试后，就要发布给用户，除了把函数交给用户之外，还要把函数的使用文档提供给用户。也就是说，你要告诉你的函数该怎么用，当然，用户无须知道自定义函数的编制原理和过程。

Excel VBA 的 Application 有一个 MacroOptions 方法，可以为自定义函数设置有关说明信息（作用是帮助用户更便利地使用自定义函数）。

MacroOptions 方法的主要参数如下。

❑ Macro：宏名，也就是函数名称，字符串类型。

❑ Description：函数描述文字，字符串类型。

❑ Category：函数分类，整数或字符串。

❑ ArgumentDescriptions：参数说明，字符串数组。

❑ HelpFile：帮助文档地址，字符串。

❑ HelpContextID：帮助文档页面索引值，整数。

其中，Category 规定了自定义函数在 Excel 函数对话框中的所属类别（见表 5-1）。

表 5-1　Excel 函数类别划分表

类 别 编 号	类 别 名 称
0	全部

续表

类 别 编 号	类 别 名 称
1	财务
2	日期与时间
3	数学与三角函数
4	统计
5	查找与引用
6	数据库
7	文本
8	逻辑
9	信息
10	命令
11	自定义
12	宏控件
13	DDE/ 外部
14	用户定义
15	工程
16	Cube
17	兼容性

ArgumentDescriptions 参数必须是一个数组，数组的每一项内容恰好是自定义函数每一个参数的说明文字。

对于 5.2.1 节中的圆柱体积函数 CylinderVolume，运行如下过程，为该函数指定函数类别，以及函数的描述、各个参数的说明。

源代码：实例文档 97.xlsm/ 自定义函数

```
1.  Sub Test4()
2.      Dim args(1 To 2) As String
3.      args(1) = "圆柱的半径，浮点型。"
4.      args(2) = "圆柱的高，浮点型。"
5.       Application.MacroOptions Macro:="CylinderVolume", Description:="本函数根据半径和高，计算圆柱的体积。", Category:=3, ArgumentDescriptions:=args
6.  End Sub
```

代码分析：该过程中的 args 是一个字符串数组，用来把它赋给 MacroOptions 方法的 ArgumentDescriptions 参数。由于圆柱体积函数只有 2 个参数，所以 args 数组也只需要 2 个元素。

第 5 行是核心代码，为 CylinderVolume 函数指定了函数的描述、函数类别为 3（数学与三角函数），使用数组 args 作为参数说明。

运行过程 Test4 后，回到 Excel 工作表界面，单击“插入函数”按钮，弹出对话框（见图 5-9）。

类别选择“数学与三角函数”，从中可以找到自定义函数 CylinderVolume，同时可以看到该函数的描述文字：“本函数根据半径和高，计算圆柱的体积。”

单击“插入函数”对话框的“确定”按钮，进入函数向导界面（见图 5-10）。

图 5-9“插入函数”对话框 2

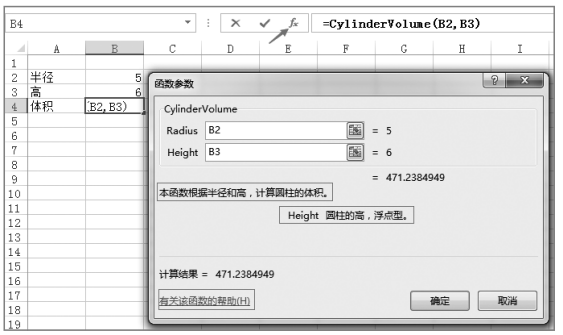

图 5-10　函数向导

“函数参数”对话框中，如果单击 Height 参数，可以看到显示该参数的说明文字：“Height 圆柱的高，浮点型。”。

但是，单击该对话框左下角的“有关该函数的帮助（H)”，会弹出“没有该函数的帮助”。5.2.4 节将会介绍使用 HTML Help Workshop 制作帮助文档。

注意：一个自定义函数只能处于某一特定函数类别中，也就是说上述函数能够在“数学与三角函数”中找到，那么就意味着从其他函数类别中找不到该函数。当然，从“全部”类别中能找到任何一个函数（内置的，以及用户自定义的）。

对于刚刚创建的自定义函数，如果不通过 MacroOptions 方法重新指定所属类别，那么默认类别是 14（用户定义）。通过 MacroOptions 方法还可以把自定义函数放在一个用户定义的单独类别中。

源代码：实例文档 97.xlsm/ 自定义函数

```
1.  Sub Test5()
2.      Application.MacroOptions Macro:="SplitTime",
        Category:=" 刘永富的函数列表 "
3.      Application.MacroOptions Macro:="LastSheet",
        Category:=" 刘永富的函数列表 "
4.  End Sub
```

代码分析：该过程的 Category 参数不是一个整数，而是一个字符串，这表示要创建一个新的函数类别。此时，在 Excel 工作表界面中再次插入函数，可以看到多了一个用户自定义的函数类别：“刘永富的函数列表”，从该类别中可以找到两个用户自定义函数：LastSheet 和 SplitTime，如图 5-11 所示。

图 5-11　自定义函数类别

5.2.4　为自定义函数创建帮助文档

HTML Help Workshop 是微软公司开发的专门制作帮助文件的软件，用该软件可以方

便地制作扩展名为 .chm 的帮助文件。本书配套资源可以下载到该软件，文件名为 HHWorkshop474.zip。

chm 帮助文件也可以称作“电子书”，看起来有点像 Word 文档，呈现为树形级联结构（见图 5-12）。

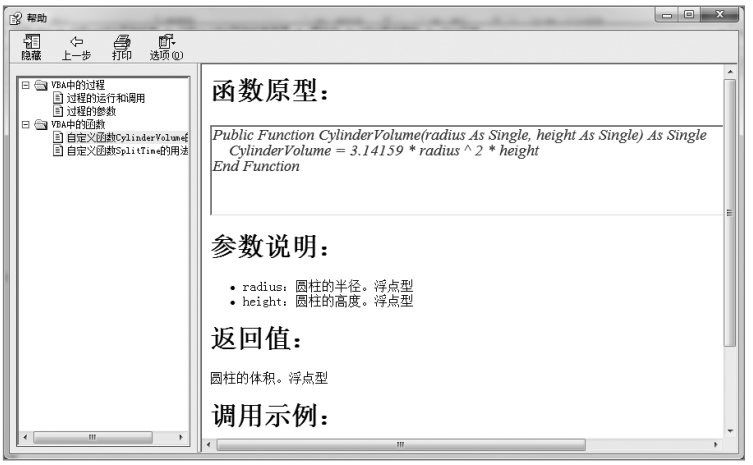

图 5-12　chm 帮助文件

chm 帮助文件实质上是多个 HTML 网页的集合，也就是把多个有关系的 HTML 网页整合成一个扩展名为 .chm 的单独文件。

以下假定 chm 帮助文件中有两大节，每一大节包含两小节。对于 chm 帮助文件，大节既可以有关联的 HTML 页面，也可以没有。在启动 HTML Help Workshop 软件之前，需要在一个文件夹中准备以下 6 个 htm 文件（可以使用网页编辑工具制作）。

- VBA 中的过程（文件：VBA 中的过程 .htm）。
  - 过程的运行和调用（文件：过程的运行和调用 .htm）。
  - 过程的参数（文件：过程的参数 .htm）。
- VBA 中的函数（文件：VBA 中的函数 .htm）。
  - 自定义函数 CylinderVolume 的用法（文件：自定义函数 CylinderVolume 的用法 .htm）。
  - 自定义函数 SplitTime 的用法（文件：自定义函数 SplitTime 的用法 .htm）。

网页文件制作完毕后，接下来解压缩“开发资源 /HHWorkShop474.zip”中的文件，安装好 HTML Help Workshop 软件并启动。

选择“文件”→“新建”命令，在弹出的对话框中选择“方案”选项，单击“确定”按钮（见图 5-13）。

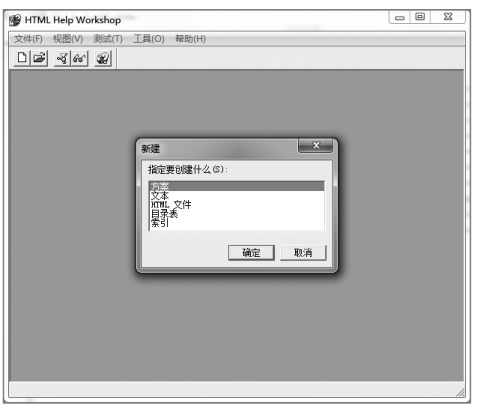

图 5-13　新建方案

在新建方案对话框中，单击“浏览”按钮，新建一个 chm 工程文件，与前面的 htm 网页文件

处于同一路径，工程名重命名为 FunctionsHelp.hhp（见图 5-14）。

单击“下一步”按钮后，创建工程完成，可以看到左侧界面有“方案”“目录”“索引”3 个窗格，如图 5-15 所示。

图 5-14　指定方案保存路径

图 5-15　工程界面

单击“方案”窗格，并单击从上往下数的第 2 个按钮，为工程添加前面设计好的 6 个网页文件（见图 5-16）。

接下来切换到“目录”窗格，这个窗格的设计对于 chm 最终文件具有非常重要的作用，因为这个窗格决定这个文档的目录层次。

首先创建大标题，单击文件夹形状的按钮，输入标题“VBA 中的过程”，如果该标题与一个 htm 文件关联，则单击“添加”按钮，选择一个网页文件。如果该标题只是一个标题，单击标题不切换到任何页面，则只需要输入标题（见图 5-17）。

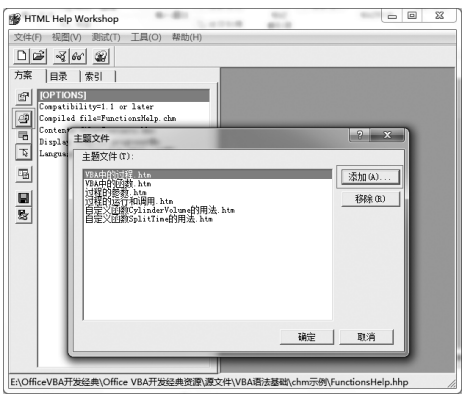

图 5-16　添加网页文件

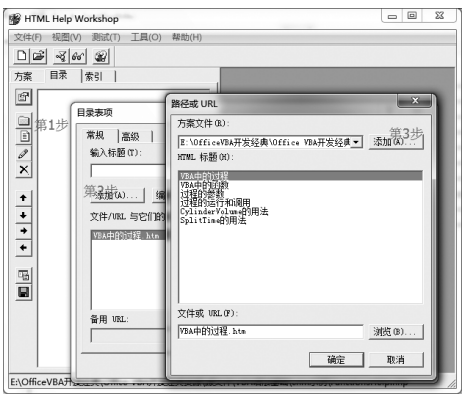

图 5-17　创建目录

仿照上面的做法，单击从上往下数的第 3 个按钮（文件形状的按钮），为大标题添加小标题，单击“添加”按钮，如图 5-18 所示。

添加完成后，可以清晰地看到目录层次是两个大标题，每个大标题包含两个小标题，如图 5-19 所示。

到此，一个简单的 chm 工程就创建好了，接下来就可以把 hhp 工程文件编译为 chm

帮助文件了。

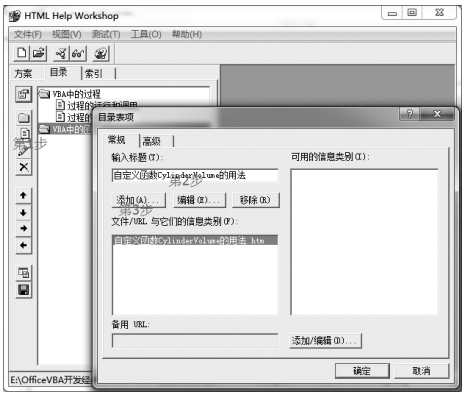

图 5-18　添加二级目录标题

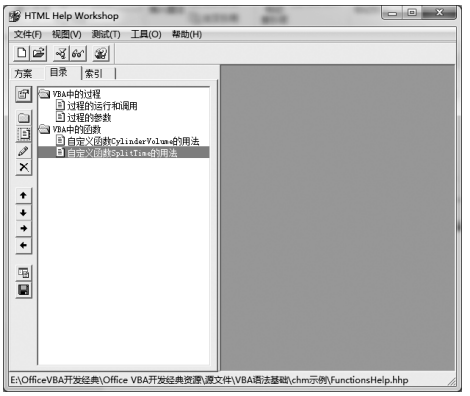

图 5-19　工程初步完成

单击工具栏中从左到右数的第 3 个编译按钮，出现界面如图 5-20 所示。

单击“编译”按钮，稍等片刻在文件夹中可以看到生成了一个 FunctionsHelp.chm 文件，这个就是最终的成品。双击这个 chm 文件，就可以查看。

从以上的讲解可以看出，chm 文件就是把多个 HTML 网页文件整合在一起，用户通过单击左侧的文档结构图，就可以实现页面的跳转。

chm 帮助文件还有一个重要的概念，就是页面的 ID 值。在程序开发中，很多情况下需要自动跳转到某 chm 文件的某一特定页面，这就需要在编译为 chm 文件之前，为网页文件设置编号值。在 HTML Help Workshop 软件中选择“文件”→“全部关闭”命令，然后在文件夹中找到 FunctionsHelp.hhp，用记事本打开该文件。

在 [INFOTYPES] 上方插入以下 4 行文本：

```
[ALIAS]
#include ALIAS.h

[MAP]
#include MAP.h
```

结果如图 5-21 所示。

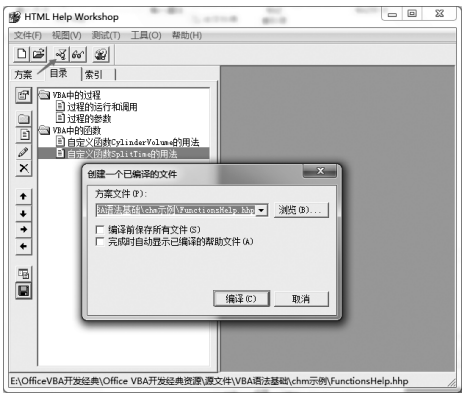

图 5-20　编译工程

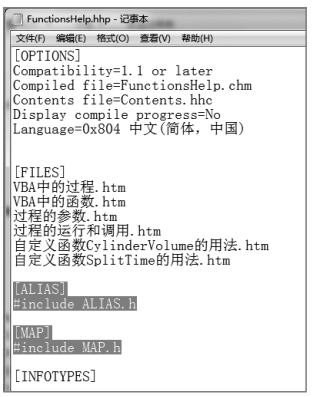

图 5-21　在记事本中编辑工程文件

然后关闭并保存记事本文件。

接下来，用记事本新建一个文本文件，重命名为 ALIAS.h，该文件的内容如图 5-22 所示。用类似的方法创建 MAP.h，文件内容如图 5-23 所示。

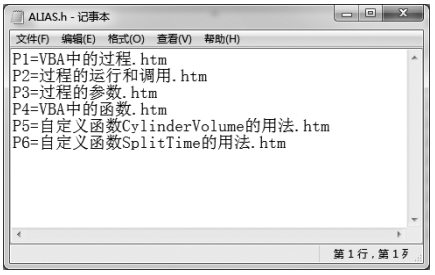

图 5-22　编辑 ALIAS.h 文件

图 5-23　编辑 MAP.h 文件

我们仔细观察上面两个文件，可以发现 P5 对应的网页是“自定义函数 CylinderVolume 的用法 .htm”，对应的 ID 是 2001。那么在代码中调用到 2001，就意味着自动跳转到它对应的网页文件。

编辑并保存好以上两个文件后，用 HTML Help Workshop 再次编译一次，生成新的 chm 帮助文件。

制作好 chm 帮助文件后，在 VBA 中可以通过多种方式去利用帮助文件。

源代码：实例文档 97.xlsm/ 自定义函数

```
1.  Sub Test6()
2.      Dim chm As String
3.      chm = "E:\ chm 示例 \FunctionsHelp.chm"
4.      Application.Help HelpFile:=chm, HelpContextID:=2000
5.  End Sub
```

代码分析：字符串变量 chm 表示帮助文件的路径。

运行上述过程，会自动打开 chm 帮助文件，并且首先显示 ID 为 2000 的网页页面。

也可以把过程中第 4 行代码更换为：

```
    MsgBox "Hello CHM ! ", vbOKCancel + vbQuestion + vbMsgBoxHelpButton,
HelpFile:=chm, Context:=2001
```

重新运行 Test6，首先弹出输出对话框（见图 5-24）。

单击“帮助”按钮，会自动打开帮助文件，并自动显示 ID 为 2001 的网页页面（也就是自定义函数 CylinderVolume 的那页），如图 5-25 所示。

注意：单击 Msgbox 的“帮助”按钮后，并不会关闭 MsgBox 对话框。必须单击“确定”或“取消”按钮才能关闭对话框。

在 Excel VBA 中除了用 Application.Help 方法和 Msgbox 来利用帮助文件外，还可以为自定义函数（UDF）提供帮助信息。

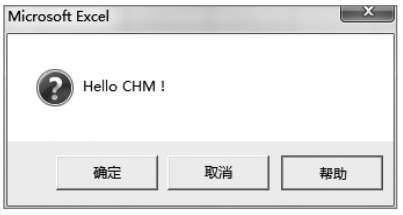

图 5-24　带有帮助按钮的输出对话框

图 5-25　跳转到指定 ID 的页面

源代码：实例文档 97.xlsm/ 自定义函数

```
1.  Sub Test7()
2.      Dim chm As String
3.      chm = "E: \chm 示例 \FunctionsHelp.chm"
4.      Application.MacroOptions Macro:="CylinderVolume", HelpFile:=chm,
        HelpContextID:=2001
5.  End Sub
```

代码分析：第 4 行代码，通过宏选项规定自定义函数 CylinderVolume 的帮助文件，以及该帮助文件中的页面位置。

运行 Test7 过程后，在工作表中再次插入函数时，在函数向导对话框的左下角，单击“有关该函数的帮助 (H)”按钮，就自动弹出 chm 帮助文件，显示该函数的信息（见图 5-26）。

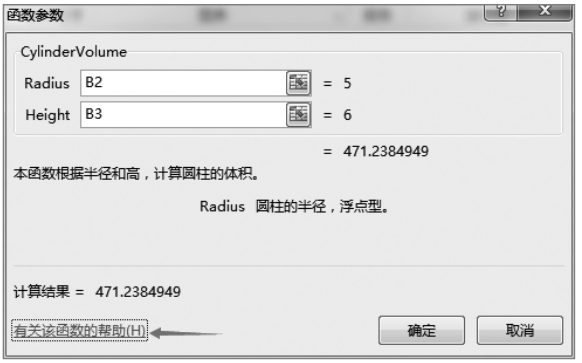

图 5-26　指定函数参数的说明文字

提示：如果对编译完成的 chm 文件不满意，可以在 HTML Help Workshop 中选择“文件”→“打开”命令，选择方案文件，重新编辑修改。

至此，chm 帮助文件的制作讲述完毕。如果要设计更加美观、完善的帮助文件，请读者

查阅其他相关资料。

本节 chm 帮助文件相关源文件路径为：源代码文件 \ 过程与函数设计 \chm 示例。

习题

1．设计一个带参数的过程，并通过另一个过程调用并执行它。

2．设计一个根据出生年份判断属相的函数，例如 2017 年出生的属鸡。制作完成后，从工作表中输入公式，使用该自定义函数。

3．海伦公式根据三角形的三边长，计算三角形面积，适合于任意形状的三角形。其公式为：

$$S=\sqrt{p(p-a)(p-b)(p-c)}$$

其中，$p=\dfrac{a+b+c}{2}$。

请基于上述公式设计一个自定义函数 Area，用于计算任意三角形的面积。设计完后，调用 Area 函数计算三边长分别为 8、15、17 的三角形面积。