

VBA 数组基础与数组函数集

完美 Excel 出品
微信公众号: *excelperfect*



在 VBA 中,可以使用数组来方便地存储数据或一系列相关的数据,本电子书《VBA 数组基础与数组函数集》讲解了 Excel VBA 数组的基本知识、相关的数组函数及一组自定义的数组函数集。

在完美 Excel 微信公众号中发送消息：**数组**，即可获得本电子书文档下载链接和密码。

坚持分享最好的Excel与VBA技术知识

微信公众号: *excelperfect*

知嗒知识号: *完美Excel*



目录

1. 用最浅显的介绍来帮你认识数组.....	1
2. 简单的数组操作.....	7
声明数组.....	7
给数组赋值.....	9
获取数组元素的值.....	9
数组的下限与上限.....	10
3. 二维数组	12
声明二维数组.....	12
给二维数组赋值.....	13
获取二维数组元素的值.....	14
二维数组的下限与上限.....	15
4. 应用数组处理工作表数据.....	18
使用工作表数据填充数组.....	18
将数组中的数据输入到工作表.....	20
在数组中使用工作表函数.....	22
5. 动态数组	24
声明动态数组.....	24
保留动态数组中的数据.....	25
Erase 语句.....	28
6. 与数组相关的函数——Array 函数与 IsArray 函数	29
Array 函数语法	29
IsArray 函数语法	34

7. 与数组相关的函数——Split 函数与 Join 函数.....	35
Split 函数语法	35
Join 函数语法	37
8. 与数组相关的函数——Filter 函数.....	40
Filter 函数语法	40
9. 使用数组作为过程参数及从函数返回数组	48
使用数组作为过程的参数.....	48
从函数返回数组.....	53
将一个数组赋值给另一个数组.....	54
10. VBA 数组函数集.....	56
关于完美 Excel	73
完美 Excel 微信公众号使用指南	74

1. 用最浅显的介绍来帮你认识数组

上学时，班主任会把我们一个班的同学分成几组；军训时，教官会把同学们分成一组一组；参加拓展训练时，老师会把学员们分成几组；..... 这样的例子在日常生活中太多，似乎是再平常不过的事了。在组织中将人们分组，便于管理，提高了行动效率。

如果将上述例子放置在计算机环境中，那么我们可以说“他们”可以组成一个个数组。



图 1.1

我们将大小相同的箱子紧挨着排列成一条直线，就构成了“数组”，如图 1.2 所示。

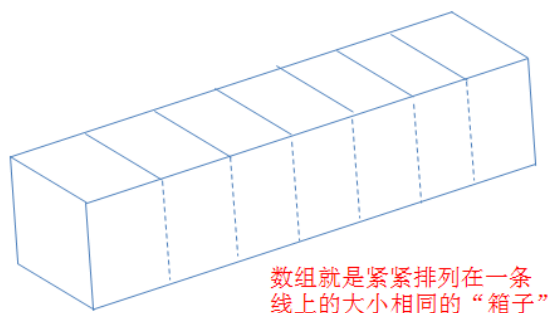


图 1.2

在“箱子”里装的必须是同类型的东西，其它类型的东西是装不进“箱子”里的。

如果我们将“箱子”当作变量，那么**数组就是由相同数据类型的变量连续排列在一起所构成的**。

我们可以使用数组来方便地管理大量同类型的数据。

需要给每个数组取一个唯一的名字，即**数组名**，以便与其他变量或数组相区分，也便于使用。可以给数组取任何名字，但取的名字最好能够反映数组的内容。一般来说，**数组名应遵循以下基本原则**：

- 不能以数字或者下划线开头
- 不能仅由数字组成
- 不能有%、\$、&、#、@等特殊字符
- 名字必须是唯一的，不能与其他变量或数组有相同名称

上文已提到过，数组就是紧紧排列在一条线上的大小相同的“箱子”。我们将“箱子”简化为“方框”，如下图 1.3 所示，一系列连续的方框构成了一个名为 Arr 的数组。其中，每个方框就是**数组元素**，方框的个数就是该**数组的元素数**。

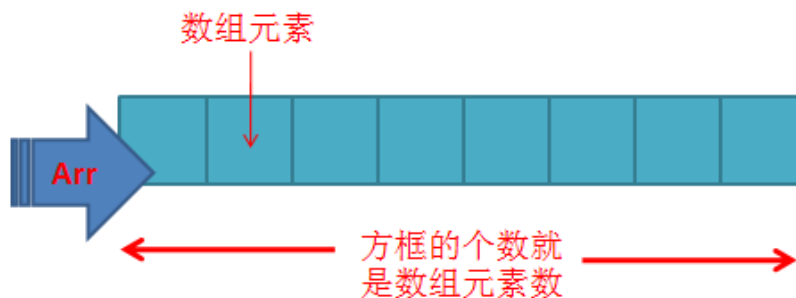


图 1.3

为了识别数组元素，计算机会从左到右给它们编号，一般从“0”或“1”开始编号，如下图 1.4 所示。

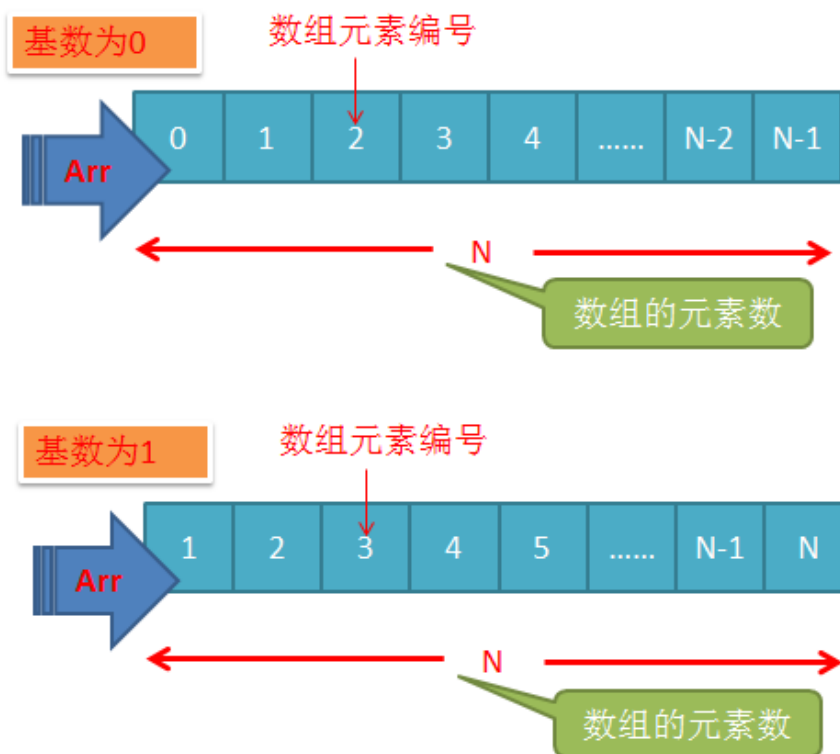


图 1.4

使用数组名和数组元素编号的组合，就可以访问特定的数组元素。例如，如果数组 `Arr` 元素编号的基数为 0，那么 `Arr(0)` 表示该数组的第 1 个元素。即：

数组名 (元素编号)

可以访问该编号代表的数组元素。

通常，我们将数组元素编号称为数组**下标或者索引值**。

下标的最小索引值称为数组的**下限**，最大索引值为数组的**上限**。超过下限或上限来访问数组，就会导致“下标越界”错误。

如果将方框垂直叠加起来，形成一个矩形数组，如下图 1.5 所示，则组成了**二维数组**。

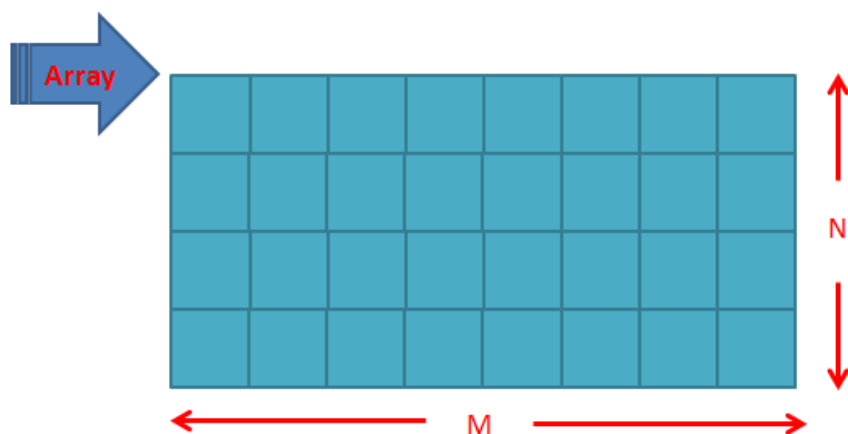


图 1.5

二维数组由水平和垂直方向紧密排列的相同数据类型的变量组成,图 5 就是一个有 N 行, 每行有 M 个元素的名为 Array 的数组。

与一维数组类似, 使用下标来代表所要访问的数组元素:

Array (N) (M)

表示第 N 行的第 M 个元素。

我们可以指定数组开始的基数是 0 还是 1。默认为 0 , 此时, Array(1) (2) 在图 1.6 所示的位置, 也就是指定行和指定列交叉的位置。

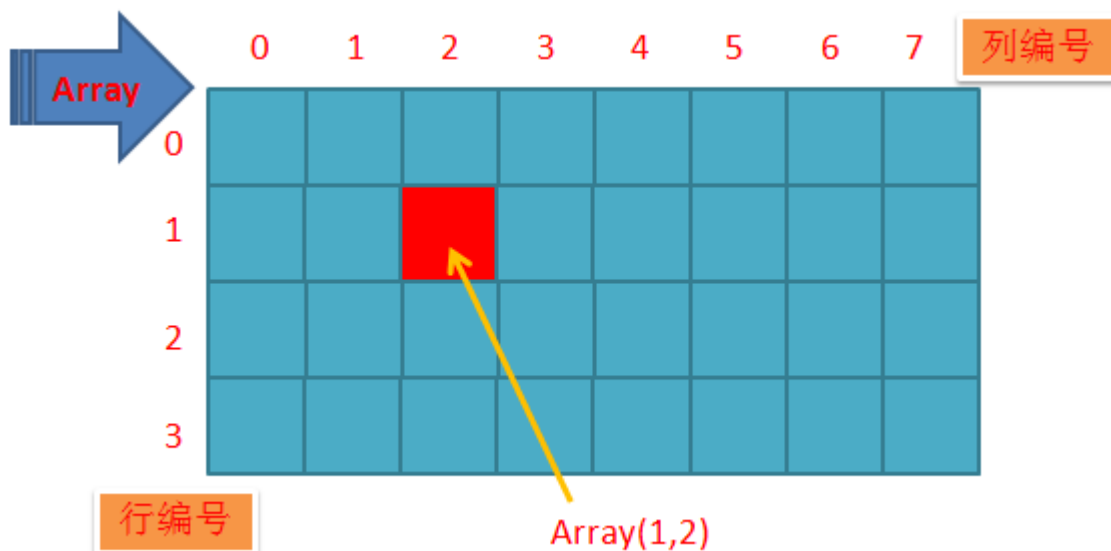


图 1.6

仔细看看图 1.6, 是不是很熟悉!

不错，就是工作表。

工作表是二维数组的典型，它由 1048576 行和 16384 列组成。不同的是，在 VBA 中，我们可以使用 `Cells(1,2)` 访问单元格 B1。

好了，通过上面的介绍，想必你已经对数组有了基本的认识。

下一章我们将开始介绍 VBA 中数组的基本知识。

2. 简单的数组操作

上一章中介绍了数组的一些基本概念，我们了解到数组就是单个元素的集合，可以非常方便地存储和管理大量的数据。其实，数组存储在计算机内存中，因此处理数组比处理工作表中的数据更快，更有效率。

下面，让我们看看如何在 VBA 中使用数组。

声明数组

在 VBA 中，使用 Dim 语句声明数组。例如，语句

```
Dim Arr(7) As Integer
```

声明了一个名为 Arr 的包含 8 个元素的整型数组，如图 1.1 所示。



图 2.1

VBA 默认数组下标索引值以 0 为基数。如果想要声明的数组下标以 1 为基数，那么应该在模块开头放置下面的语句：

```
Option Base 1
```

此时，语句

```
Dim Arr(7) As Integer
```

声明了的 Arr 数组如图 1.2 所示。



图 2.2

当然，也可以使用以下语句，使 Arr 数组的下标索引值从 1 开始。

```
Dim Arr(1 To 8) As Integer
```

更为疯狂的是，可以使用这种方法使数组的下标从任意值开始，但为什么要这样呢？

声明数组的方式

① **Dim 数组名 (数组元素数) As 数据类型**

② **Dim 数组名 (数组元素数)**

其中：

- **数组名**为任何有效的变量名，即遵守变量名的命名规则
- **数组元素数**可以是任意正整数
- 数组下标的起始索引值取决于 **Option Base 语句**，若省略该语句或者设置 Option Base 0，则数组下标起始索引值为 0；若设置 Option Base 1，则数组下标起始索引值为 1
- 可以在数组元素数中使用 **n To m**，来显式声明数组的下标下限和上限
- 数据类型可以是任何有效的 VBA 数据类型，包括整型、字符型、日期型、对象，甚至数组等
- 若省略掉 **As 数据类型**，则认为数组为 Variant 型

注：除特别说明外，下面介绍的内容均为 VBA 默认的情形，即数组下标基数为 0。

给数组赋值

下面的过程给数组 Arr 赋值：

```
Sub testArray()  
    Dim Arr(7) As Integer  
    Dim i As Integer  
  
    For i = 0 To 7  
        Arr(i) = i * i  
    Next i  
End Sub
```

运行程序后，数组 Arr 中各元素的值如图 2.3 所示。

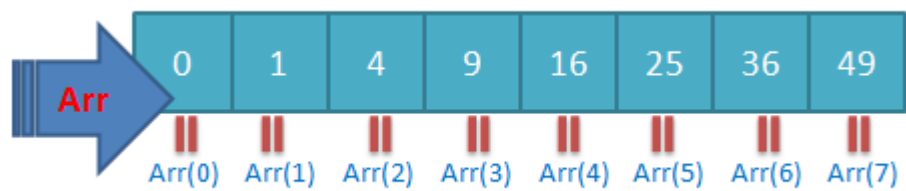


图 2.3

获取数组元素的值

可以使用数组下标索引值来获取数组元素的值，例如下面的过程：

```
Sub testArray1()  
    Dim Arr(7) As Integer  
    Dim i As Integer  
  
    For i = 0 To 7  
        Arr(i) = i * i  
    Next i
```

```

    MsgBox "数组 Arr 的第 2 个值是: Arr(1) = " & Arr(1) & vbCrLf & _
        "数组 Arr 的第 5 个值是: Arr(4) = " & Arr(4)
End Sub

```

运行代码后的结果如图 2.4 所示。

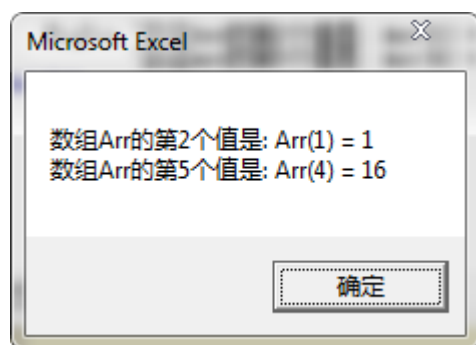


图 2.4

由于默认下标索引值基于 0 开始，因此第 2 个元素是 Arr(1)，第 5 个元素是 Arr(4)。

注意，如果下标超出了数组范围，则会出现运行时错误：下标越界。

数组的下限与上限

可以使用 LBound 函数和 UBound 函数来确定数组的下限和上限，即数组下标的最小索引值和最大索引值。例如，下面的过程：

```

Sub testArray2()
    Dim Arr(7) As Integer
    Dim i As Integer

    MsgBox "数组 Arr 的下限值是: " & LBound(Arr) & vbCrLf & _
        "数组 Arr 的上限值是: " & UBound(Arr)
End Sub

```

运行后的结果如图 2.5 所示。



图 2.5

也可以使用下面的过程给数组 `Arr` 赋值：

```
Sub testArray3()  
    Dim Arr(7) As Integer  
    Dim i As Integer  
  
    For i = LBound(Arr) To UBound(Arr)  
        Arr(i) = i * i  
        Debug.Print Arr(i)  
    Next i  
End Sub
```

对于数组 `myArray` 来说，如果其下限为 0，那么该数组元素的总数为：

`UBound(myArray) + 1`

如果其下限为 1，则数组元素的总数就等于 `UBound` 的返回值。

一般说来，数组元素数可以由下式算出：

`UBound(myArray) - LBound(myArray) + 1`

注意，对未初始化的数组使用 `UBound` 函数和 `LBound` 函数会导致“下标越界”错误。

至此，我们以一维数组为例介绍了一些简单的数组操作，接下来让我们看看二维数组。

3.二维数组

VBA 可以定义多达 60 维的数组，但实际上很少使用三维以上的数组。本文主要讲解二维数组及其使用。

声明二维数组

与一维数组一样，使用 Dim 语句来声明二维数组。例如，语句

```
Dim myArray(3,7) As Integer
```

声明了一个名为 myArray 的包含 4 行 8 列共 32 个元素的二维整型数组，如图 3.1 所示。

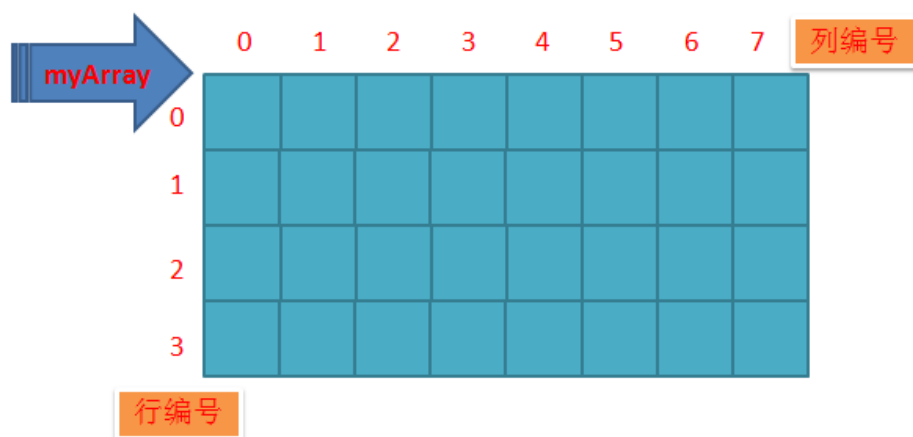


图 3.1

也可以使用以下语句，使 myArray 数组的下标索引值从 1 开始。

```
Dim Arr(1 To 4, 1 To 8) As Integer
```

声明一个 4 行 8 列的二维数组。

声明二维数组的方式

① `Dim 数组名 (数组元素数, 数组元素数) As 数据类型`

② `Dim 数组名 (数组元素数, 数组元素数)`

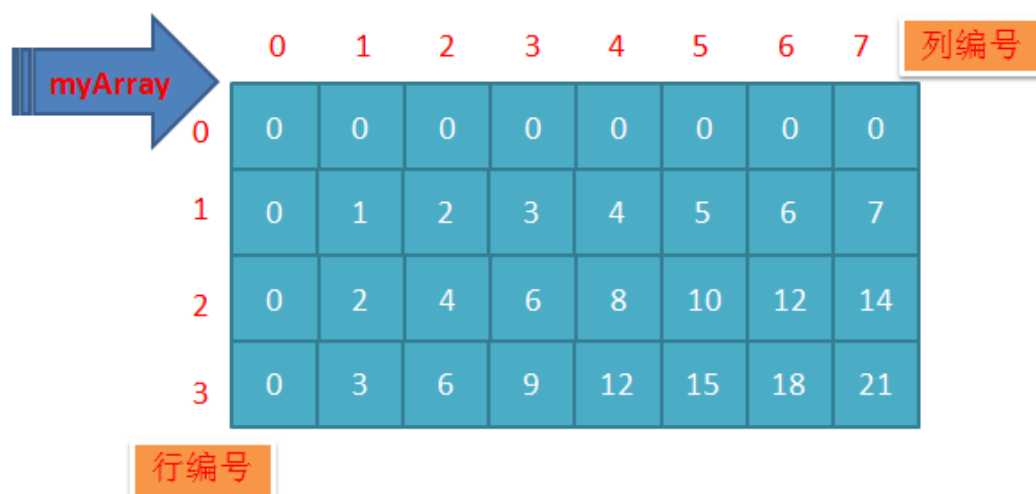
各部分的说明与上篇文章相同。

给二维数组赋值

下面的过程给二维数组 myArray 赋值：

```
Sub testMyArray()  
    Dim myArray(3, 7) As Integer  
    Dim i As Integer, j As Integer  
  
    For i = 0 To 3  
        For j = 0 To 7  
            myArray(i, j) = i * j  
        Next j  
    Next i  
End Sub
```

运行程序后，数组 myArray 中各元素的值如图 3.2 所示。



The diagram illustrates the myArray 2D array. A blue arrow labeled 'myArray' points to a table. The table has 4 rows and 8 columns. The columns are indexed from 0 to 7, and the rows are indexed from 0 to 3. The values in the table are the product of the row index and the column index.

	0	1	2	3	4	5	6	7	列编号
0	0	0	0	0	0	0	0	0	
1	0	1	2	3	4	5	6	7	
2	0	2	4	6	8	10	12	14	
3	0	3	6	9	12	15	18	21	
行编号									

图 3.2

获取二维数组元素的值

如图 3.3 所示，可以看出数组 myArray 的第 2 行第 3 列的元素的值为 2。

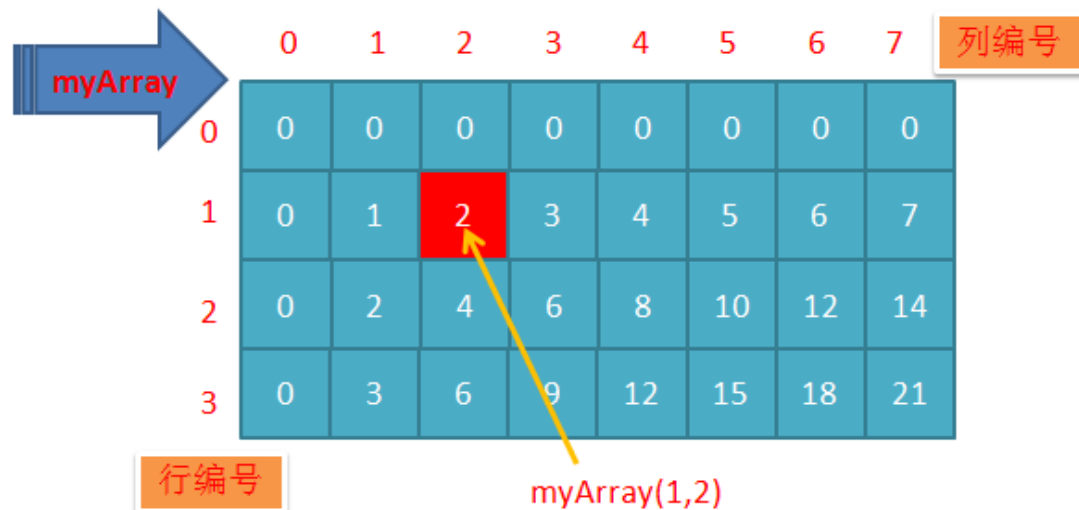


图 3.3

可以使用 `myArray(1, 2)` 来获取其值，如下面的过程：

```
Sub testMyArray1()  
    Dim myArray(3, 7) As Integer  
    Dim i As Integer, j As Integer  
  
    For i = 0 To 3  
        For j = 0 To 7  
            myArray(i, j) = i * j  
        Next j  
    Next i  
  
    MsgBox "数组 myArray 的第 2 行第 3 列元素的值是: " & _  
        "myArray(1,2) = " & myArray(1, 2)  
End Sub
```

运行代码后的结果如图 3.4 所示。

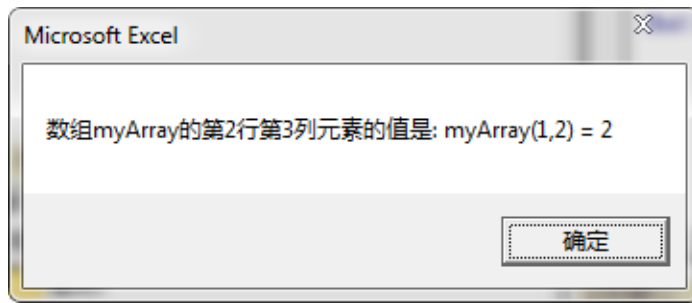


图 3.4

由于默认下标索引值基于 0 开始，因此第 2 行的元素下标索引值是 1，第 3 列元素下标索引值是 2。

注意，如果下标超出了数组范围，则会出现运行时错误：下标越界。

二维数组的下限与上限

可以使用 LBound 函数和 UBound 函数来确定二维数组的下限和上限，即二维数组各维的下标的最小索引值和最大索引值。例如，下面的过程：

```
Sub testMyArray2()  
    Dim myArray(3, 7) As Integer  
    Dim i As Integer, j As Integer  
    Dim k As Integer, str As String  
  
    For i = 0 To 3  
        For j = 0 To 7  
            myArray(i, j) = i * j  
        Next j  
    Next i  
  
    For k = 1 To 2  
        str = str & "数组 myArray 的第" & k & "维的下限是：" &  
LBound(myArray, k)  
        str = str & " 上限是：" & UBound(myArray, k) & vbCrLf &  
vbCrLf  
    Next k  
End Sub
```

```
Next k

MsgBox str

End Sub
```

运行后的结果如图 3.5 所示。

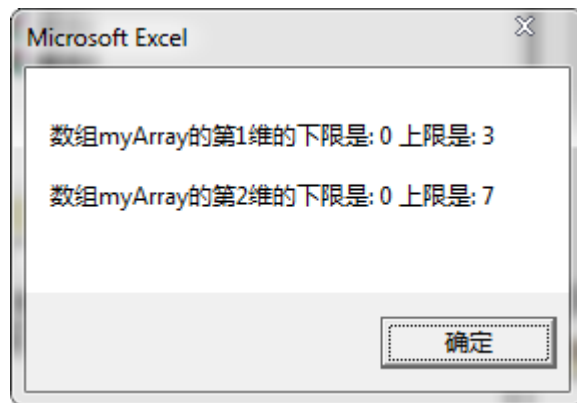


图 3.5

可以看出，在 LBound 函数和 UBound 函数中，第 1 个参数为数组名称，第 2 个参数指定数组的维数。

本章简要介绍了二维数组的基本操作，下一章将重点讲解二维数组在工作表中的应用。

4. 应用数组处理工作表数据

前面我们已经讲过，工作表就是一个二维数组，指定其水平维度（行）和垂直维度（列）就可以找到单元格，例如 `Cells(2,3)` 就是单元格 C2。而在二维数组中，使用其每一维的下标索引值即可从中获取元素，例如 `myArray(1,2)` 就是数组 `myArray` 的第 2 行第 3 列的元素。

在 Excel 中，可以将使用数组来存储单元格区域数据，并进行处理，然后再将数据输入到工作表，这比只是在工作表中处理数据更高效。

使用工作表数据填充数组

如下图 4.1 所示的工作表。

	A	B	C
1	学生姓名	语文	数学
2	张三	98	99
3	李四	96	92
4	王五	99	98
5	赵六	92	89
6	周七	88	90
7	钱八	90	92
8	孙九	97	95
9			

图 4.1

使用下面的语句快速填充数组：

```
Dim myArray As Variant  
myArray = Worksheets("Sheet1").Range("B2:C8")
```

注意到，在声明数组变量时，并没有指定维数，也没有指定具体的数据类型，而是指定为 Variant 型。因此，也可以使用下面的声明语句：

```
Dim myArray
```

下面的过程代码测试刚才创建的数组的维数及其上限和下限：

```
Sub testArray()  
    Dim myArray As Variant, vUBound As Variant  
    Dim i As Integer, str As String  
  
    myArray = Worksheets("Sheet1").Range("B2:C8")  
    i = 1  
  
    Do  
        str = str & "第 " & i & " 维的下限 = " & LBound(myArray,  
i) & _  
            " 上限 = " & UBound(myArray, i) & vbCr & vbCr  
        i = i + 1  
        On Error Resume Next  
        vUBound = UBound(myArray, i)  
        If Err.Number <> 0 Then  
            str = str & "数组 myArray 包含的维数是： " & i - 1  
            Exit Do  
        End If  
        On Error GoTo 0  
    Loop  
  
    MsgBox str  
End Sub
```

运行代码后的结果如下图 4.2 所示。

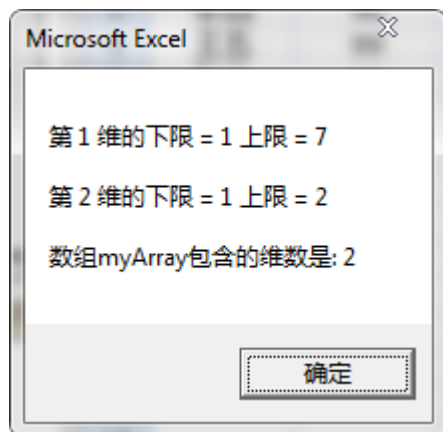


图 4.2

可以看出，与普通数组不同，放置单元格数据的数组的下限索引值是 1。

将数组中的数据输入到工作表

下面的代码对图 4.1 工作表中每位学生的成绩求和并将结果输入到列 D：

```
Sub testArray1()  
    Dim myArray  
    Dim iCount As Integer  
  
    myArray = Worksheets("Sheet1").Range("B2:C8")  
  
    For iCount = LBound(myArray) To UBound(myArray)  
        Worksheets("Sheet1").Range("D" & iCount + 1) _  
            = myArray(iCount, 1) + myArray(iCount, 2)  
    Next iCount  
End Sub
```

运行代码后的结果如图 4.3 所示。

	A	B	C	
1	学生姓名	语文	数学	
2	张三	98	99	
3	李四	96	92	
4	王五	99	98	
5	赵六	92	89	
6	周七	88	90	
7	钱八	90	92	
8	孙九	97	95	
9				

↓

	A	B	C	D	E
1	学生姓名	语文	数学		
2	张三	98	99	197	
3	李四	96	92	188	
4	王五	99	98	197	
5	赵六	92	89	181	
6	周七	88	90	178	
7	钱八	90	92	182	
8	孙九	97	95	192	
9					

图 4.3

下面的过程将数组 myArray 中的数据输入到工作表单元格区域 A1:C3 中：

```
Sub testArray2()
    Dim myArray(1, 2)

    myArray(0, 0) = "姓名"
    myArray(0, 1) = "性别"
    myArray(0, 2) = "成绩"
    myArray(1, 0) = "张三"
    myArray(1, 1) = "男"
    myArray(1, 2) = "95"

    Worksheets("Sheet2").Cells(1, 1). _
        Resize(UBound(myArray, 1) + 1, UBound(myArray, 2) + 1)
    = myArray
End Sub
```

在数组中使用工作表函数

仍以上文所示的工作表为例，下面的代码使用工作表函数 **Average** 分别求语文和数学的平均成绩。

```
Sub testArray3()  
    Dim myArrayChn, myArrayMath  
  
    myArrayChn = Worksheets("Sheet1").Range("B2:B8")  
    myArrayMath = Worksheets("Sheet1").Range("C2:C9")  
  
    MsgBox " 语 文 平 均 成 绩 为 : " &  
WorksheetFunction.Average(myArrayChn) & _  
        vbCr & " 数 学 平 均 成 绩 为 : " &  
WorksheetFunction.Average(myArrayMath)  
End Sub
```

如果要求语文和数学成绩的最高分，可以在语句中使用 **Max** 函数：

```
WorksheetFunction.Max (myArrayChn)  
WorksheetFunction.Max (myArrayMath)
```

一般来说，将工作表单元格数据赋值给 Variant 型变量后，该变量就成了一个二维数组。可以使用 TRANSPOSE 函数将工作表中的一列转换成一列，从而将二维数组转变成一维数组。

仍以上文所示的工作表为例，代码：

```
Sub testArray4()  
    Dim myArrayChn As Variant  
    myArrayChn = WorksheetFunction.Transpose(Range("B2:B8"))  
    MsgBox "第 5 位学生的语文成绩是: " & myArrayChn(5)  
End Sub
```

运行后的结果如图 4.4 所示。

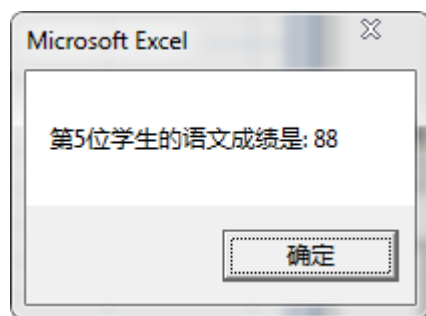


图 4.4

当然，也可以使用 Transpose 函数将一行中数据转换成一维数组，还可以使用 Index 函数将工作表中一行数据转换成一维数组，例如：

```
Dim myArrayTitle As Variant
myArrayTitle = WorksheetFunction.Index(Range("A1:C1").Value, 1, 0)
```

5. 动态数组

前面讲解的数组都是在声明时就指定了维数和大小，它们称之为固定数组。在不知道要使用多大数组的情形下，可以声明一个尽可能大的固定数组。但是，数组是存储在计算机内存中的，如果声明的数组越大，浪费的内存就越多。此时，可以考虑使用动态数组。

例如，使用数组来存储用户输入的数据，但是无法知道用户究竟会输入多个数据，此时，可以声明和使用动态数组来解决。

动态数组是指没有设置大小的数组，允许在程序运行时使用 ReDim 语句重新定义数组的大小。

声明动态数组

使用下面的语句，通过忽略数组维数来声明动态数组：

```
Dim 数组名() As 数据类型
```

当然，也可以使用语句声明 Variant 型的数组：

```
Dim 数组名()
```

然后，在需要使用该数组时，使用 ReDim 语句来设置该数组的大小。

下面的示例将当前工作簿中所有工作表的名字存储在数组 myArray 中。由于事先不知道工作簿中有多少工作表，因此声明一个动态数组。

```
Sub testDynamicArray()  
    '声明一个字符型的动态数组  
    Dim myArray() As String
```

```

Dim iCount As Integer, iWks As Integer

iWks = ActiveWorkbook.Worksheets.Count

'设置数组大小
ReDim myArray(1 To iWks)

For iCount = 1 To iWks
    myArray(iCount) = ActiveWorkbook.Sheets(iCount).Name
Next iCount
End Sub

```

保留动态数组中的数据

每次使用 ReDim 语句时，都会初始化数组，即清除该数组中原来存储的数据内容。如果要保留原来的数据，那么要加上 Preserve 关键字。

实际上，ReDim 语句的作用是创建一个新的数组，而 Preserve 关键字的作用则是指原数组中的数据复制到新数组中。

注意，如果使用 ReDim 语句重新定义的数组大小比原数组要小，则无论是否使用 Preserve 关键字，肯定会丢失掉那部分被减少了的元素数据。

下面的示例，收集用户在文本框 TextBox1 中输入的数据，单击 CommandButton2 按钮时显示用户已经输入过的数据。

```

Dim myArray() As String
Dim iCount As Integer

Private Sub CommandButton1_Click()
    '保留原数组的值并将数组变大
    ReDim Preserve myArray(iCount)

    '将用户输入的文本存储在数组中
    myArray(iCount) = Me.TextBox1.Text

```

```

        iCount = iCount + 1
        Me.TextBox1.Text = ""
End Sub

Private Sub CommandButton2_Click()
    Dim str As String
    Dim i As Integer

    str = "用户已输入的内容是:"

    For i = LBound(myArray) To UBound(myArray)
        str = str & vbCrLf & myArray(i)
    Next i

    MsgBox str
End Sub

```

运行上述代码后的结果如图 5.1 所示。

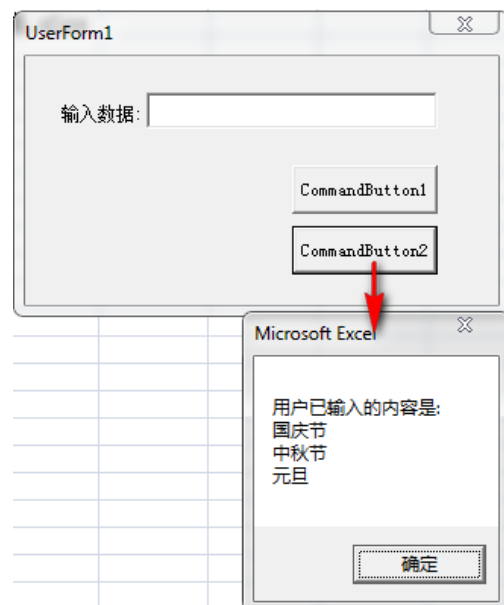


图 5.1

下面的示例搜索指定路径下的所有 Excel 文件，并在数组中存储其文件名。由

于不知道文件夹里面有多少所需要的文件，因此无法指定数组的实际大小。

```
Sub testDynamicArray1()  
    Dim strName As String  
    Dim strFileNames() As String  
    Dim iCount As Integer, i As Integer  
  
    '获取指定类型的文件名  
    '将其存储在数组中  
    strName = Dir("G:\09. Excel\*.xls*")  
    Do Until strName = ""  
        iCount = iCount + 1  
        ReDim Preserve strFileNames(1 To iCount)  
        strFileNames(iCount) = strName  
        strName = Dir  
    Loop  
  
    '将数组中存储的文件名输出到工作表  
    For i = 1 To UBound(strFileNames)  
        Worksheets("Sheet1").Range("A" & i) = strFileNames(i)  
    Next i  
End Sub
```

上述代码首先使用 Dir 函数在指定路径中找到与*.xls*相匹配的第一个文件，将其保存在数组中，接着获取下一个相匹配的文件，并将其保存在数组中，直至找不到相匹配的文件并返回一个零长度字符串。

注意，如果数据很多，那么在使用 Preserve 时，会减慢程序代码的运行速度。

可以定义二维或三维以上的动态数组，并且使用 ReDim 语句可以改变数组的维数，但是使用 Preserve 关键字只能改变多维数组最后一维的大小，而不能改变数组的维数。

Erase 语句

Erase 语句可以清空数组，使其重新成为赋值前的状态。

- 可以使用 Erase 语句一次清空多个数组，数组之间用逗号隔开。
- 使用 Erase 语句后，固定大小的数组将保持其维数，而动态数组将释放其使用的所有内存空间。
- 如果使用 Erase 语句清除了动态数组，那么再次使用该数组之前要使用 ReDim 语句重新定义维数。

本章内容 2017 年 11 月 28 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
VBA 进阶 | 数组基础 06: 与数组相关的函数——Array 函数与 IsArray 函数

6. 与数组相关的函数——Array 函数与 IsArray 函数

在某些情形下，可以使用 Array 函数方便地填充数组。例如，下面的代码：

```
Dim myArray  
myArray = Array("国庆节", "中秋节", "元旦", "春节")
```

等价于：

```
Dim myArray(3)  
myArray(0) = "国庆节"  
myArray(1) = "中秋节"  
myArray(2) = "元旦"  
myArray(3) = "春节"
```

因此，Array 函数可以将一组值批量赋给一个数组。

Array 函数语法

Array 函数返回一个 Variant 型数组，该数组由传递给该函数的参数组成。其语法为：

Array([元素 1],[元素 2],...,[元素 n])

其中:

- 元素 1~n 可以是任意数据类型, 代表赋给数组元素的数据。
- 由 Array 函数返回的数组只可赋值给一个 Variant 型变量, 不能赋值给已声明为数组变量的变量。
- Array 函数返回的数组中元素的顺序与传递给函数的参数值的顺序相同。
- Array 函数总是返回 Variant 类型的数组, 但元素的数据类型可以不同, 这取决于传递给该函数的数值类型。例如 Array("One", 2, 3.14) 返回的数组中, 第 1 个元素是 String 型, 第 2 个是 Integer 型, 第 3 个是 Double 型。
- Array 函数创建的数组下限由 Option Base 确定。若忽略该语句, 则数组下限值为 0。
- 若 Array 函数没有任何参数, 则会创建一个空数组。
- Array 函数返回的数组是动态数组, 其初始大小是 Array 函数的参数数量, 可以使用 ReDim、ReDim Preserve 来对所创建的数组重新定义维数。
- 如果使用 VBA.Array(), 例如
Dim myArray As Variant
myArray = VBA.Array("国庆节","中秋节","元旦")
那么数组的基数总是 0, 与 Option Base 的设置无关。即 Option Base 1 时, myArray(1) 的值仍然是“中秋节”。

编程技巧

1. 在已确定元素内容时, 使用 Array 函数更有效率。例如:

```
Dim myArray As Variant  
myArray = Array("国庆节", "中秋节", "元旦", "春节")
```

2. 可以使用 Array 函数创建多维数组，例如：

```
Dim myArray As Variant

myArray = Array(Array("One", "Two", "Three"), _
                Array("Four", "Five", "Six"), _
                Array("Seven", "Eight", "Nine"))
```

创建了一个 3 行 3 列的二维数组。

然而，要访问这个数组中的元素，例如第 2 行第 3 列的元素数据“Six”，不能够使用：

```
myArray(1, 2)
```

这样，将产生“下标越界”错误。

而应该使用：

```
myArray(1) (2)
```

或者，先声明一个 Variant 型变量，然后将相应行的第 1 维赋值给该变量，再使用该变量来访问相应列的元素数据，例如：

```
Dim vElement As Variant
vElement = myArray(1)

MsgBox vElement(2)
```

3. 可以使用 Array 函数来为 ActiveX 列表框或组合框控件赋值，例如：

```
Me.ListBox1.List = Array("国庆节", "中秋节", "元旦", "春节")
```

给列表框添加了 4 个节日名。

4. 可以使用 Array 函数创建控件数组，即将控件名称作为 Array 函数的参数。然后，可以将数组元素当作相应的控件对象来使用，例如：

```
Private Sub CommandButton1_Click()
    Me.ListBox1.Clear

    Me.ListBox1.List = Array("国庆节", "中秋节", "元旦", "春节")

    Me.ListBox1.ListIndex = 0
End Sub
```

```
Private Sub CommandButton2_Click()
    Dim myArray As Variant
    myArray = Array(ListBox1, CommandButton1)
    MsgBox myArray(0).Text
End Sub
```

运行后的结果如下图 6.1 所示。

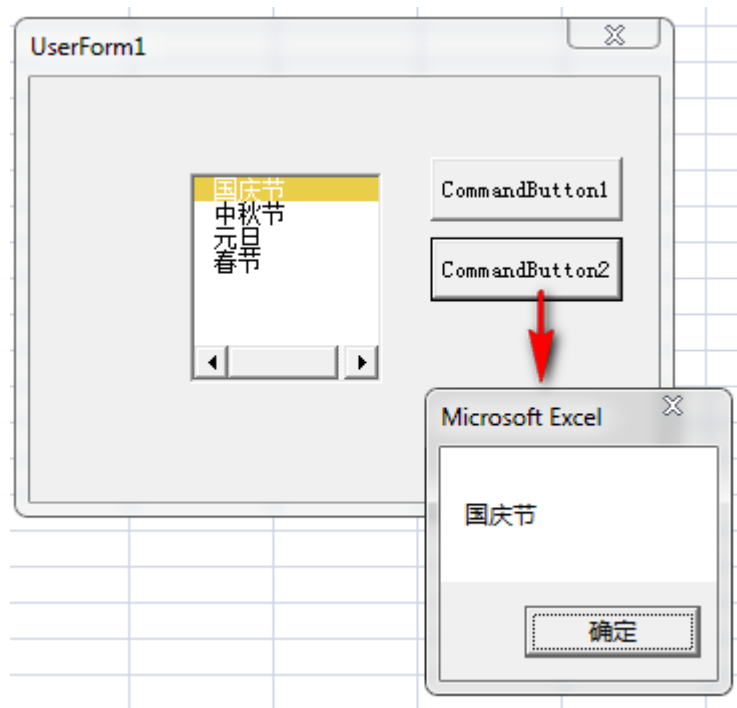


图 6.1

示例

示例 1：使用 Array 函数创建数组

```
Sub TestArray()
    Dim myArray() As Variant

    '从逗号分隔的字符串列表中创建数组
    myArray = Array("One", "Two", "Three")

    '显示数组元素
    MsgBox myArray(0) & vbCr & myArray(1) & vbCr & myArray(2)
```

```

'也可以使用数值作为参数
myArray = Array(10, 20, 30)

'显示数组元素

MsgBox myArray(0) & vbCr & myArray(1) & vbCr & myArray(2)
End Sub

```

示例 2：下面的示例先使用 Array 函数将一组值赋给变量 myArray，然后将该变量中的值输入到工作表 Sheet1 的第 1 行。

```

Option Base 1
Sub ArrayTest()
    Dim myArray As Variant
    Dim iCount As Integer

    myArray = Array("姓名", "性别", "成绩")

    With Worksheets("Sheet1")
        For iCount = 1 To UBound(myArray)
            .Cells(1, iCount).Value = myArray(iCount)
        Next iCount
    End With
End Sub

```

运行代码后的结果如图 2 所示。

	A	B	C
1	姓名	性别	成绩
2			
3			
4			

图 6.2

在某些情形下，需要检验某个变量是否为数组，以便于后续操作，此时，可以使用 IsArray 函数。

IsArray 函数语法

IsArray 函数返回一个 Boolean 值，表明传递给该函数的变量是否为数组。其语法为：

```
IsArray (varname)
```

其中：

- 参数 varname 为等检验是否为数组的变量的名称。
- 如果传递给 IsArray 函数的变量是一个数组或者包含一个数组，则返回值为 True；否则为 False。

编程技巧

在不确定使用的变量是否为数组时，如果试图访问不存在的数组中的元素或者对其使用数组函数（如 LBound 或者 UBound），就会导致错误。此时，可以先使用 IsArray 函数来确定其是否为数组，然后再进行后续相应的操作。

例如，下面的代码段检查变量 myArray 是否为数组，若不是则退出：

```
If Not IsArray(myArray) Then Exit Sub
```

7. 与数组相关的函数——Split 函数与 Join 函数

Split 函数可以将一个包含分隔符的字符串解析成一个数组，更一般地说，是可以把一个字符串拆分并创建一个一维字符串数组。

与 Split 函数相对，Join 函数可以将数组中的各元素数据使用指定的分隔符连接成一个字符串。

Split 函数语法

Split 函数返回一个 Variant 型数组，该数组由传递给该函数的参数根据指定的分隔符分解而成。其语法为：

```
Split(expression,[delimiter[,count[, compare]])
```

其中：

- 参数 expression 必需，为 String 型，要拆分成数组的字符串。
- 参数 delimiter 可选，为 Variant 型，界定 expression 中的子字符串的字符。
- 参数 count 可选，为 Long 型，要返回的字符串数。
- 参数 compare 可选，指定比较的方法，为 vbCompareMethod 常数：
vbBinaryCompare、vbTextCompare 或者 vbDatabaseCompare。默认为 vbBinaryCompare。
- 该函数返回的数组的基数总是 0，与 Option Base 的设置无关。
- 如果没有找到参数 delimiter 指定的分隔符，那么该函数将返回整个字符串。

串。

- 如果省略了参数 `delimiter`，那么就使用空格作为分隔符。
- 如果省略了参数 `count` 或指定其值为-1，那么返回所有的字符串。
- 一旦分解的子字符串达到了 `count` 指定的数量，那么参数 `expression` 指定的字符串的剩余部分就会全部作为数组的下一个元素。
- 该函数返回的数组赋值的变量必须是一个动态的、一维的字符串数组，或者是一个 `Variant` 型变量。
- 数组中的元素顺序按照参数 `expression` 的先后顺序。
- 如果将空字符串传递给 `Split` 函数，那么将返回一个空数组。

示例

下面的过程使用 `Split` 函数创建一个数组：

```
Sub testSplit()  
    Dim SplitArray() As String  
    Dim str As String  
    Dim iCount As Integer  
  
    SplitArray = Split("完美 Excel,excelperfect,Excel,Office", ",")  
  
    For iCount = 0 To UBound(SplitArray)  
        str = str & "SplitArray(" & iCount & ") = " &  
SplitArray(iCount) & vbCr  
    Next iCount  
  
    MsgBox str  
End Sub
```

运行代码后的结果如图 7.1 所示。

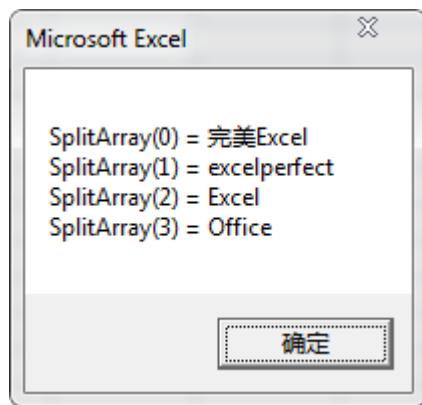


图 7.1

Join 函数语法

Join 函数返回一个字符串，该字符串由传递给该函数的数组根据指定的分隔符连接而成。其语法为：

```
Join(sourcearray, [delimiter])
```

其中：

- 参数 sourcearray 必需，为 String 型或者 Variant 型，指定包含被连接的元素的数组。其下限可以是任意值。如果该参数中的元素为数值型数据，则应该使用 Variant 型数组。
- 参数 delimiter 可选，为 String 型，用来连接数组各元素的分隔符。如果没有指定该参数，那么使用空格作为分隔符。
- 参数 sourcearray 指定的数组中没有被使用的元素在返回的字符串中将由参数 delimiter 指定的分隔符代替。
- 如果传递给 Join 函数的是空数组，那么将返回一个空字符串。

示例

下面的示例先声明一个数组 myArray 并给其前两个元素赋值，然后使用逗号分隔符将该数组中的值连接成一个字符串。

```
Sub testJoin()
```

```

Dim myArray(1 To 10) As String
Dim result As String

myArray(1) = "国庆节"
myArray(2) = "中秋节"

result = Join(myArray, ",")

MsgBox result
End Sub

```

运行代码后的结果如图 7.2 所示。

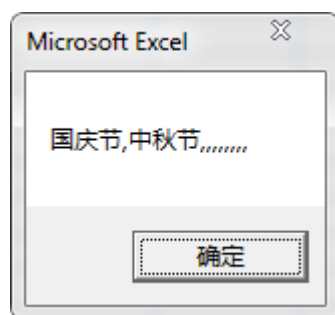


图 7.2

注意到，除前两个元素外，数组剩余的部分由逗号替代。

下面的示例先使用 Split 函数将字符串拆分成数组，然后使用 Join 函数使用指定的连接字符将数组元素连接成字符串。

```

Sub TestSplitJoin()
    Dim myStr As String
    Dim myArray() As String

    ' 由逗号分隔的字符
    myStr = "A1,B2,C3"
    ' 将字符串分成一组子字符串
    myArray = Split(myStr, ",")
    ' 显示数组元素

```

```
MsgBox myArray(0) & vbCr & myArray(1) & vbCr & myArray(2)  
'将数组的所有元素合成一个字符串  
'使用 " and " 连接  
myStr = Join(myArray, " and ")  
'显示字符串  
MsgBox myStr  
End Sub
```

运行后的结果如图 7.3 所示。

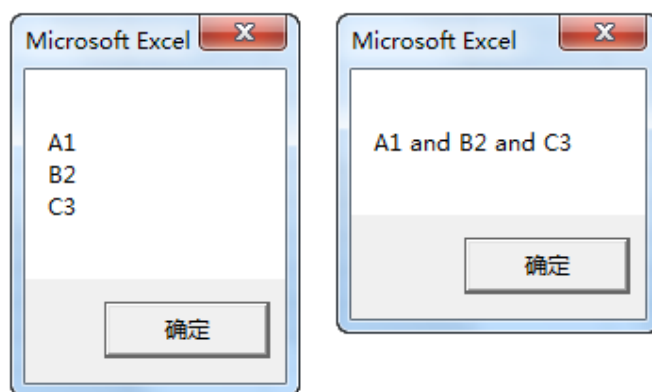


图 7.3

8. 与数组相关的函数——Filter 函数

Filter 函数根据源数组中的元素是否与指定字符串相匹配，从而产生一个与指定字符串相匹配的数组。

Filter 函数语法

Filter 函数返回一个字符串型数组，该数组由传递给该函数的数组根据指定的条件筛选而成。其语法为：

```
Filter (Sourcearray, FilterString[, Switch[, Compare]])
```

其中：

- 参数 sourcearray 必需，为 String 型或 Variant 型数组，包含要筛选的数据。如果该参数指定一个 String 型数组，Filter 函数将忽略零长字符串；如果该参数指定一个 Variant 型数组，Filter 函数将忽略空元素。
- 参数 FilterString 必需，为 String 型，指定要在 sourcearray 中查找的字符串。
- 参数 Switch 可选，为 Boolean 型。如果为 True，则返回所有匹配的值；如果为 False，则返回所有不匹配的值。默认为 True。
- 参数 compare 可选，指定比较的方法，为 vbCompareMethod 常数：vbBinaryCompare、vbTextCompare 或者 vbDatabaseCompare。默认为 vbBinaryCompare，即区分大小写。

- 无论 Option Base 如何设置，返回的数组的下限基数总是 0。
- Filter 函数返回值所赋给的数组必须声明为一维的 String 或 Variant 动态数组。
- 不能要求 Filter 函数仅查找完全匹配的元素项，因为该函数总是返回包含匹配字符串的所有元素。

编程技巧

1. Filter 函数不仅可用于处理字符串数组，而且可用于处理数值数组。此时，应将参数 Sourcearray 指定的数组指定为 Variant 型。

下面的过程筛选数组 vSource 中含有“1”的元素，并将结果赋值给数组 vResult。

```
Sub testFilter()  
    Dim vSource As Variant, vResult As Variant  
    Dim strFilter As String  
    Dim i As Integer, str  
  
    strFilter = CStr(1)  
    vSource = Array(10, 15, 21, 22, 30, 36, 51)  
    vResult = Filter(vSource, strFilter, True, vbBinaryCompare)  
  
    For i = LBound(vResult) To UBound(vResult)  
        str = str & vResult(i) & " "  
    Next  
  
    MsgBox str  
End Sub
```

代码运行后的结果如图 8.1 所示。

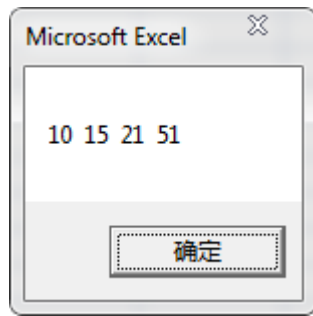


图 8.1

2. Filter 函数与 Dictionary 对象配合使用。例如，下面的代码查找出字典对象键中指定的内容并显示。

```
Sub FilterAndDictionary()  
    Dim iCount As Integer  
    Dim strKeys() As Variant  
    Dim strFiltered() As String  
    Dim strMatch As String  
    Dim blnSwitch As Boolean  
    Dim objDict As Dictionary  
    Dim str As String  
  
    Set objDict = New Dictionary  
  
    '填充字典对象  
    objDict.Add "excelperfect", "完美 Excel"  
    objDict.Add "excelsoft", "MS Excel"  
    objDict.Add "Excellent", "优秀 Excel 专家"  
    objDict.Add "office", "MS Office"  
  
    '字典对象的键组成的数组  
    strKeys = objDict.Keys  
  
    strMatch = "excel"  
    blnSwitch = True
```

```

'查找包括字符串 excel (不区分大小写) 的所有键
strFiltered() = Filter(strKeys, strMatch, blnSwitch,
vbTextCompare)

'显示已找到的字典内容
For iCount = 0 To UBound(strFiltered)
    str = str & strFiltered(iCount) & " | " & _
        objDict.Item(strFiltered(iCount)) & vbCr
Next iCount

MsgBox "键" & " | " & "内容" & vbCr & str
End Sub

```

运行代码后的结果如图 8.2 所示。

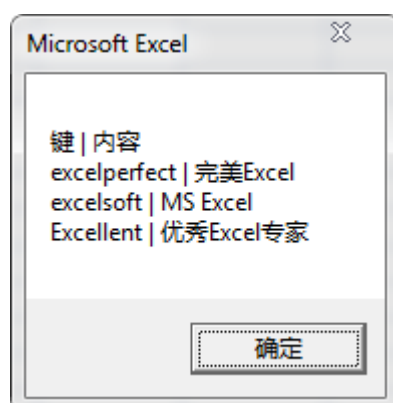


图 8.2

示例

示例 1：下面的过程代码用于测试 Filter 函数

```

Sub TestFilter()
    Dim myArray() As Variant
    Dim myFilteredArray As Variant

    '创建数组
    myArray = Array("One", "Two", "Three")

```

```

'筛选数组中包含"T"的元素
myFilteredArray = Filter(myArray, "T", True)
'显示结果
MsgBox "在数组 ("One", "Two", "Three") 中 " & _
    "筛选含有 "T" 的元素将返回" & _
    vbCr & Join(myFilteredArray, vbCr)
'筛选数组中不包含 "T" 的元素
myFilteredArray = Filter(myArray, "T", False)
'显示结果
MsgBox "在数组 ("One", "Two", "Three") 中 " & _
    "筛选不含有 "T" 的元素将返回" & _
    vbCr & Join(myFilteredArray, vbCr)
'筛选数组中含有 "t" 的元素
myFilteredArray = Filter(myArray, "t", True)
'显示结果
MsgBox "在数组 ("One", "Two", "Three") 中 " & _
    "筛选含有 "t" 的元素将返回" & _
    vbCr & Join(myFilteredArray, vbCr)
'在数值数组中筛选数字 "1"
myArray = Array(1, 2, 3, 10)
myFilteredArray = Filter(myArray, 1)
'显示结果
MsgBox "在数值数组 (1, 2, 3, 10) 中 " & _
    "筛选含有 1 的数组元素返回" & _
    vbCr & Join(myFilteredArray, vbCr)
End Sub

```

运行的结果如图 8.3 所示。

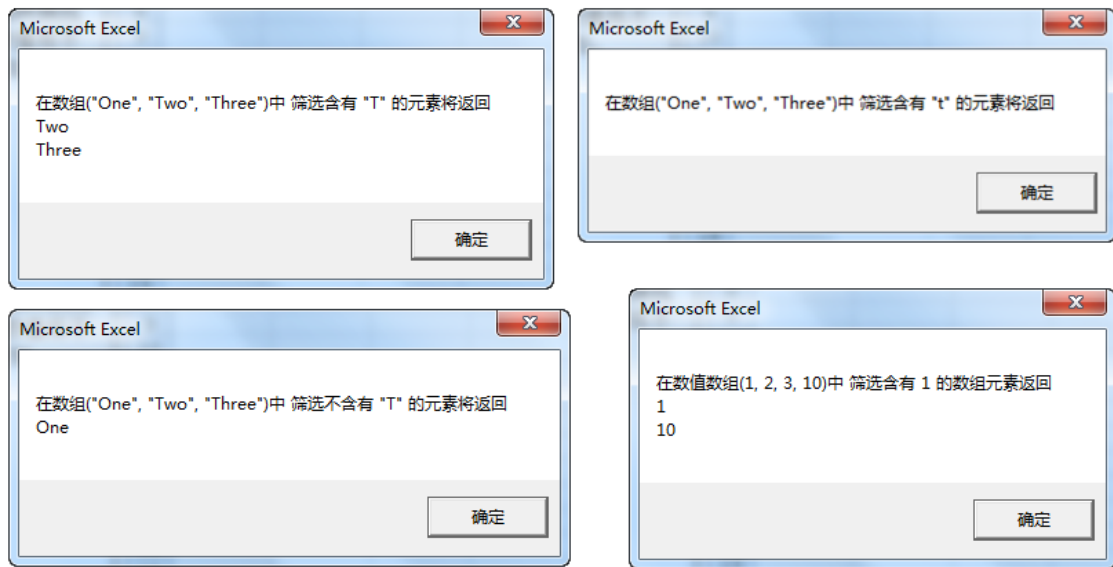


图 8.3

示例 2： 下面的代码过程演示如何仅获取完全匹配的元素

```
Sub FilterExactly()
    Const myMarker As String = "!"
    Const myDelimiter As String = ","
    Dim myArray() As Variant
    Dim mySearchArray As Variant
    Dim myFilteredArray As Variant

    '创建数组
    myArray = Array(1, 2, 3, 10)
    '预先在数组中筛选包含 1 的元素
    myFilteredArray = Filter(myArray, 1)

    If UBound(myFilteredArray) > -1 Then
        '标记每个找到的元素的开始和结束
        'myMarker 和 myDelimiter 必须是字符
        '且该字符不会出现在数组的任何元素中!
        mySearchArray = Split(myMarker & Join(myFilteredArray,
myMarker & _
myDelimiter & myMarker) & myMarker,
```

```

myDelimiter)

'下面筛选修改后的数组
myFilteredArray = Filter(mySearchArray, _
    myMarker & "1" & myMarker)

'从结果中移除标记
myFilteredArray = Split(Replace(Join(myFilteredArray, _
    myDelimiter), myMarker, ""), myDelimiter)

End If

'显示结果
MsgBox "筛选数组(" & Join(myArray, ", ") & _
    ") 以获得含有 1 的完全匹配的元素将返回:" & _
    vbCr & Join(myFilteredArray, vbCr)

End Sub

```

运行后的结果如图 8.4 所示。

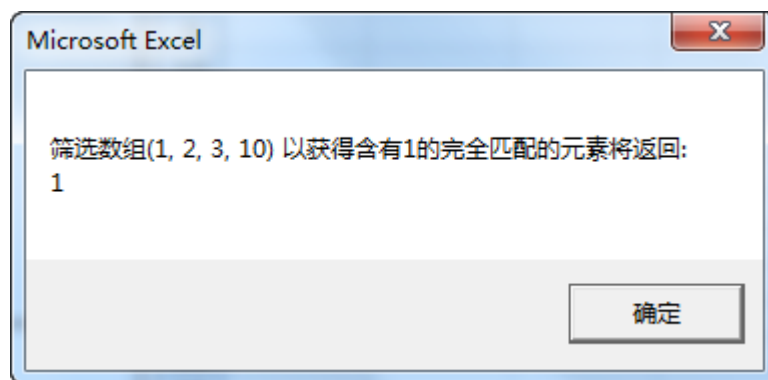


图 8.4

9. 使用数组作为过程参数及从函数返回数组

在 VBA 中，可以使用数组作为过程的参数，也可以从函数过程返回数组，这是一项很有用的技术。

使用数组作为过程的参数

可以传递数组作为过程的参数，如下面的示例代码：

```
Sub test()  
    Dim myArray(2) As Long  
    Dim iCount As Integer  
    Dim str As String  
  
    myArray(0) = 1  
    myArray(1) = 2  
    myArray(2) = 3  
  
    testPassArray passArray:=myArray  
  
    For iCount = LBound(myArray) To UBound(myArray)  
        str = str & "myArray(" & iCount & ") = " & myArray(iCount)  
        & vbCrLf  
    Next iCount  
  
    MsgBox str
```

```

End Sub

Sub testPassArray(ByRef passArray() As Long)
    Dim i As Long
    For i = LBound(passArray) To UBound(passArray)
        passArray(i) = (i + 1) * 100
    Next i
End Sub

```

在代码中：

- 将数组 myArray 传递到被调用的 testPassArray 过程，在该过程中，数组经过处理后，最后的结果如图 9.1 所示。

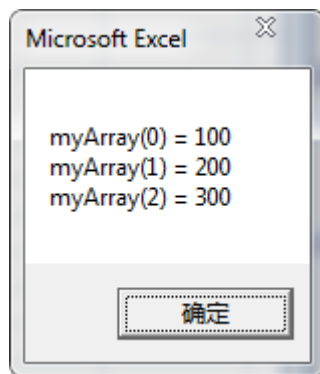


图 9.1

- 注意到 testPassArray 过程名后的关键字 ByRef，表明数组是通过引用传递的，这样，在被调用过程中对数组的修改都是对实际所传递的数组的修改，从图 9.1 所示的结果中我们也可以看出来。
- 在调用过程中的数组的数据类型与被调用过程中传递的参数中的数组的数据类型要相匹配，如本例中的数组 myArray 和 passArray 都是 Long 型。不能简单地将被调用过程的参数声明为 Variant 型，想当然地认为可以接受任意类型的数组。

下面演示传递动态数组给被调用函数的示例代码：

```

Sub testDynamicArray()
    Dim DynArray() As Double

```

```

Dim iCount As Long
Dim str As String

'调用 PopulateArray 过程来调整数组大小并填充相应的数据
PopulateArray myArray:=DynArray,
testRange:=Range("A2:A9"), strName:="张三"

str = "张三的测试成绩分别为: "
For iCount = LBound(DynArray) To UBound(DynArray)
    str = str & vbCr & DynArray(iCount)
Next iCount

MsgBox str
End Sub

Sub PopulateArray(ByRef myArray() As Double, testRange As
Range, strName As String)
    Dim rng As Range
    Dim iIndex As Long

    If testRange Is Nothing Then
        MsgBox "单元格区域为空!"
        Exit Sub
    End If

    '重新定义动态数组的大小为整个单元格区域的单元格数
    ReDim myArray(1 To testRange.Cells.Count)

    '遍历单元格区域,找到相应的单元格后将对应值存储到数组中
    For Each rng In testRange.Cells
        If rng.Value = strName Then
            iIndex = iIndex + 1
            myArray(iIndex) = rng.Offset(0, 1).Value
        End If
    Next rng
End Sub

```

```

End If
Next rng

'重新定义数组大小为已填充的元素数
ReDim Preserve myArray(1 To iIndex)
End Sub

```

在代码中：

- 在主调过程 `testDynamicArray` 中声明了动态数组 `DynArray()`，并将其传递给被调过程 `PopulateArray`。在被调过程 `PopulateArray` 中，根据实际需要调整大小。
- 注意到，在被调过程 `PopulateArray` 中，我们首先将数组大小调整为可能的最大值，待填充完数据后，再将其调整为实际大小。这样，避免了每次增加数据时都要重新调整数组大小而导致会发生的可能问题。
- 注意，在被调用过程的参数列表中声明数组时，不能包括大小。
- 上述代码运行的结果如图 9.2 所示。

	A	B	C	D	E
1	姓名	测试成绩			
2	张三	90			
3	李四	89			
4	张三	95			
5	李四	85			
6	王五	92			
7	张三	98			
8	王五	86			
9	张三	93			
10					
11					
12					
13					



图 9.2

下面的示例传递固定大小的静态数组到 Function 过程：

```

Sub testPassArrayToFunction()
    Dim myArray(1 To 3) As Long
    Dim lngResult As Long

```

```

myArray(1) = 10
myArray(2) = 20
myArray(3) = 30

result = SumToArray(passArray:=myArray)
MsgBox result
End Sub

Function SumToArray(passArray() As Long) As Long
    Dim iCount As Integer
    Dim lngTotal As Long

    For iCount = LBound(passArray) To UBound(passArray)
        lngTotal = lngTotal + passArray(iCount)
    Next iCount

    SumToArray = lngTotal
End Function

```

在代码中：

- 主过程 **testPassArrayToFunction** 调用 **SumToArray** 过程并传递需要求和的数组，**SumToArray** 过程对接收到的数组求和并返回结果。
- 前面我们讲过，在调用过程中的数组的数据类型与被调用过程中传递的参数中的数组的数据类型要相匹配。然而，有时可能不知道传递的数组的类型，那该如何做呢？
如果要传递任意类型的数组，那么在被调用过程的参数声明中不要将要接收数组的参数声明为数组，而应该声明为 **Variant** 型。例如，将上述 **Function** 过程修改如下，使其可以接收其他类型的数组数据。

```

Function SumToArray1(passArray As Variant)

    Dim iCount As Integer
    Dim lngTotal As Long

    If IsArray(passArray) Then
For iCount = LBound(passArray) To UBound(passArray)

```



```

lngTotal = lngTotal + passArray(iCount)
Next iCount
End If
SumToArray = lngTotal
End Function

```

从函数返回数组

下面的示例演示从函数返回一个数组：

```

Sub testReturnArray()
    Dim myArray() As Long
    Dim iCount As Long

    myArray = LoadNumbers(Low:=21, High:=30)

    For iCount = LBound(myArray) To UBound(myArray)
        Debug.Print myArray(iCount)
    Next
End Sub

Function LoadNumbers(Low As Long, High As Long) As Long()
    Dim resultArray() As Long
    Dim lngIndex As Long
    Dim lngVal As Long

    If Low > High Then
        Exit Function
    End If

    ReDim resultArray(1 To (High - Low + 1))
    lngVal = Low

```

```

    For lngIndex = LBound(resultArray) To UBound(resultArray)
        resultArray(lngIndex) = lngVal
        lngVal = lngVal + 1
    Next lngIndex

    LoadNumbers = resultArray()
End Function

```

在代码中：

- 接收数组结果的变量（如示例中的 `myArray()`）必须是动态数组，并且必须与返回的数组有相同的数据类型（如示例中的 `Long` 型），或者声明为 Variant 变量（例如，`Dim myArray As Variant`）。
- 如果接收数组结果的变量（如示例中的 `myArray()`）是静态数组，那么函数返回值的数组必须是动态数组，它将自动调整大小来返回合适的数组。
- 如果接收数组与返回的数组的下标索引的基准值不同，那么将使用返回的数组的下标索引基准值。

将一个数组赋值给另一个数组

VBA 不允许将一个数组直接赋给另一个数组，即便其大小和数据类型都一致。例如，下面的代码：

```

Dim A(1 To 10) As Long
Dim B(1 To 10) As Long
A = B

```

运行代码后将产生编译错误：不能给数组赋值。

然而，可以将包含数组的 Variant 型变量赋值给另一个 Variant 型变量，例如下面的代码：

```

Dim A As Variant
Dim B As Variant
Dim i As Long

```

```
A = Array(11, 22, 33)
B = A
```

通常情况下，想要将一个数组的内容转移到另一个数组中，就必须遍历数组并逐元素赋值：

```
Dim A(1 To 3) As Long
Dim B(0 To 5) As Long
Dim lngIndexA As Long
Dim lngIndexB As Long

A(1) = 11
A(2) = 22
A(3) = 33
lngIndexB = LBound(B)

For lngIndexA = LBound(A) To UBound(A)
    If lngIndexB <= UBound(B) Then
        B(lngIndexB) = A(lngIndexA)
    Else
        Exit For
    End If
    lngIndexB = lngIndexB + 1
Next lngIndexA
```

注意到，上面代码中数组 A 与 B 的维数不同，数组 B 的维数大于数组 A 的维数。如果数组 A 的维数大于数组 B 的维数，则当数组 B 的 UBound 达到时循环终止。

10. VBA 数组函数集

在 VBA 代码中，使用数组来存储数据或一系列相关数据。

本章介绍了可以用于获取数组信息和操控数组的近 40 个函数。这些函数来源于 [cpearson.com](#)，[可以在完美 Excel 微信公众号中发送消息：数组函数集，下载包含这些函数代码的工作簿。](#)

本文使用下列术语：

静态数组是一个数组，在声明数组的 Dim 语句中定义其大小。

```
Dim StaticArray(1 To 10) As Long
```

不能修改静态数组的大小或数据类型。当使用 Erase 语句清空一个静态数组时，不会释放内存。Erase 语句只是简单地设置所有元素为它们的默认值（0、vbNullString、Empty 或 Nothing，取决于数组的数据类型）。

动态数组是一个数组，没有在 Dim 语句中定义大小。相反，它使用 ReDim 语句定义大小，例如：

```
Dim DynamicArray() As Long  
ReDim DynamicArray(1 To 10)
```

可以修改动态数组的大小，但不是数据类型。当使用 Erase 语句清空动态数组时，分配给数组的内存将被释放。被清空之后，必须重新使用 ReDim 语句声明数组才能使用它。

如果数组是静态数组，或者是使用 ReDim 语句调整大小的动态数组，则数组会被分配空间。

静态数组总是会被分配空间且决不为空。

如果是仍没有使用 ReDim 语句调整大小或者使用 Erase 语句清除分配空间的动态数组，那么数组为空或者未被分配。

元素是一组项目中的一个特定项目。

如果你对 VBA（或 VB）数组不熟悉，但是具有其他编程语言（例如 C）的数组方面的经验，就会发现 VBA 数组的工作原理相同。主要区别在于 VBA 数组不仅仅是一系列连续的字节，VBA 数组实际上是一个称为 SAFEARRAY 的结构，它包含有关数组的信息，包括每维中的维数和元素数。

该结构包括指向实际数据的指针变量。在 VBA 代码中的任何数组操作都使用适当的 SAFEARRAY API 函数。虽然这可能会为处理项目添加一些开销，但它可以阻止标准数组中经常发生的常见错误，例如超出数组上限。尝试访问超出数组上限的元素将导致可捕获的运行时错误“9——下标越界”。

VB/VBA 数组与常规（例如，C）数组之间的另一个显著差异是，可以为数组的下限和上限指定任何值。元素 0 不必是数组中的第一个元素索引值。例如，下面是完全合法的代码（只要下限小于或者等于上限——如果下限大于上限，将产生编译错误）：

```
Dim N As Long
Dim Arr(-100 To -51) As Long
Debug.Print "LBound: " & CStr(LBound(Arr)), _
    "UBound: " & CStr(UBound(Arr)), _
    "NumElements: " & CStr(UBound(Arr) - LBound(Arr) + 1)
For N = LBound(Arr) To UBound(Arr)
    Arr(N) = N * 100
Next N
```

虽然我从来没有发现需要使用 0 或 1 之外的下限，但是在某些情况下，这可能是有用的，VB/VBA 数组支持它。

最后的显著区别是，如果没有显示声明数组的下限，则下限将被假定为 0 或 1，这取决于 Option Base 语句的值（如果存在）。如果模块不存在 Option Base，那么假定为 0。例如，代码

```
Dim Arr(10) As Long
```

声明一个 10 或 11 个元素的数组。注意，声明不是指定数组中元素数量，而是指数组的上限。如果模块中没有包含 Option Base 语句，那么下限假设为 0，上面的声明语句与下面的相同：

```
Dim Arr(0 To 10) As Long
```

如果有一个 Option Base 语句指定 0 或 1，那么数组的下限就会被设置为那个值。因此，代码

```
Dim Arr(10) As Long
```

相当于下面两者之一

```
Dim Arr(0 To 10) As Long
```

或

```
Dim Arr(1 To 10) As Long
```

这取决于 Option Base 的值。在我看来，省略下限，只声明上限是非常不好的编程实践。当在模块和工程之间复制/粘贴代码时，忽略下限将导致错误。应该总是在 Dim 语句或者 ReDim 语句中显式地指定数组的下限和上限。

最后，因为动态数组的下限和上限可以使用 ReDim 语句在运行时修改，所以当遍历数组时，应该总是使用 LBound 和 UBound。不要硬编码数组界限。例如：

```
Dim N As Long
Dim Arr(-100 To -51) As Long
For N = LBound(Arr) To UBound(Arr)
    ' do something with Arr(N)
Next N
```

在尝试遍历动态声明的数组之前，应该测试确保数组实际上已被分配空间。可以使用下面介绍的 IsArrayAllocated 函数测试这个条件：

```
Dim Arr() As Long
If IsArrayAllocated(Arr:=Arr) = True Then
    ' loop through the array
Else
    ' code for unallocated array
End If
```

本文后面的内容简要介绍下列子过程：

AreDataTypesCompatible

ChangeBoundsOfArray

CombineTwoDArrays

CompareArrays

ConcatenateArrays

CopyArray

CopyArraySubSetToArray
CopyNonNothingObjectsToArray
DataTypeOfArray
DeleteArrayElement
ExpandArray
FirstNonEmptyStringIndexInArray
GetColumn
GetRow
InsertElementIntoArray
IsArrayAllDefault
IsArrayAllNumeric
IsArrayAllocated
IsArrayDynamic
IsArrayEmpty
IsArrayObjects
IsArraySorted
IsNumericDataType
IsVariantArrayConsistent
IsVariantArrayNumeric
MoveEmptyStringsToEndOfArray
NumberOfArrayDimensions
NumElements
ResetVariantArrayToDefaults
ReverseArrayInPlace
ReverseArrayOfObjectsInPlace
SetObjectArrayToNothing
SetVariableToDefault
SwapArrayRows
SwapArrayColumns
TransposeArray
VectorsToArray

函数说明

(1) AreDataTypesCompatible

```
Public Function AreDataTypesCompatible(DestVar As Variant, SourceVar As Variant) As Boolean
```

这个函数检查两个变量: DestVar 和 SourceVar, 并且确定是否它们是兼容的。如果两个变量具有相同的数据类型, 那么这些变量是兼容的, 或者, 如果在 SourceVar 中的值能够被存储在 DestVar 中, 而不会丢失精度或者遇到溢出错误。例如, Source Integer 与 Dest Long 兼容, 因为 Integer 可以存储在 Long 变量中, 而不会损失精度或者溢出。Source Double 与 Dest Long 不兼容, 因为 Double 会丢失精度 (数字的小数部分会丢失), 并且其转换可能会导致溢出错误。

(2) ChangeBoundsOfArray

```
Public Function ChangeBoundsOfArray(InputArr As Variant, _  
    NewLowerBound As Long, NewUpperBound) As Boolean
```

这个函数修改 InputArray 的上限和下限。保留 InputArr 中现有的数据。InputArr 必须是一个动态的、已分配空间的单维数组。如果该数组新的大小 (NewUpperBound-NewLowerBound+1) 大于原数组大小, 那么该数组多余的元素被设置为该数组的数据类型的默认值 (0, vbNullString, Empty, 或 Nothing)。如果该数组新的大小小于原数组的大小, 那么新数组将仅包含原数组开头的值, 其余的元素将丢失。该数组的元素可以是简单的类型变量 (例如, Long、String)、对象或者数组, 不允许用户定义类型。如果 InputArr 不是数组、或是一个静态数组、或者没有被分配空间、或者 NewLowerBound 大于 NewUpperBound、或者不是单维数组, 那都将导致错误。如果发生错误, 那么该函数返回 False; 若操作成功, 则返回 True。

(3) CombineTwoDArrays

```
Public Function CombineTwoDArrays(Arr1 As Variant, _  
    Arr2 As Variant) As Variant
```

这个函数将两个二维数组合并成一个单独的数组。该函数返回一个 Variant, 包含 Arr1 和 Arr2 组合成的数组。如果发生错误, 那么结果为 NULL。Arr1 和 Arr2 的两维都必须具有相同的 LBound, 也就是说, 所有 4 个下限都必须相等。结果数组是 Arr2 添加到 Arr1 后面。例如, 数组 {a,b;c,d} 和 {e,f;g,h}, 合并后创建的数组为 {a,b;c,d;d,f;g,h}。

可以嵌套调用 CombineTwoDArrays 来合并多个数组成单独的数组, 例如:


```
V = CombineTwoArrays(CombineTwoArrays(CombineTwoArrays(A, B), C), D)
```

(4) CompareArrays

```
Public Function CompareArrays(Array1 As Variant, Array2 As Variant, _  
    ResultArray As Variant, Optional CompareMode As VbCompareMethod =  
    vbTextCompare) As Boolean
```

这个函数比较两个数组 Array1 和 Array2，使用 Array1 和 Array2 中相对应的元素比较的结果填充 ResultArray。在 Array1 中的每个元素与 Array2 中相应的元素比较，如果 Array1 中的元素小于 Array2 中的元素，那么 ResultArray 中相应的元素被设置为-1；如果两个元素相等，则为 0；如果 Array1 中的元素大于 Array2 中的元素，则为+1。Array1 和 Array2 具有相同的 LBound 且有相同数量的元素。ResultArray 必须是数字数据类型（通常为 Variant 或 Long）的动态数组。如果 Array1 和 Array2 是数字类型，那么使用 “>” 和 “<” 运算符进行比较。如果 Array1 和 Array2 是字符串数组，那么使用 StrComp 和由 CompareMode 参数确定的文本比较模式（区分大小写或不区分大小写）进行比较。

(5) ConcatenateArrays

```
Public Function ConcatenateArrays(ResultArray As Variant, ArrayToAppend As Variant,  
_  
    Optional NoCompatibilityCheck As Boolean = False) As Boolean
```

这个函数将 ArrayToAppend 追加到 ResultArray 的后面。保存其原始值和该数组末尾的 ArrayToAppend 的值的 ResultArray，必须是动态数组。ResultArray 将会调整大小以容纳其原始数据加上 ArrayToAppend 数组中的数据。ArrayToAppend 可以是静态的或者动态的数组。ResultArray 和 ArrayToAppend 数组的任一个或者两个都可能未分配空间。如果 ResultArray 未分配空间，ArrayToAppend 分配了空间，那么 ResultArray 被设置成与 ArrayToAppend 相同的大小，并且 ResultArray 的 LBound 和 UBound 与 ArrayToAppend 相同。如果 ArrayToAppend 未分配空间，那么 ResultArray 保持原样，函数终止。如果两个数组都未分配空间，那么不会有任何操作，数组保持不变，程序终止。

默认情况下，ConcatenateArrays 确保 ResultArray 和 ArrayToAppend 数组的数据类型相同或者兼容。如果源元素的值可以存储在目标元素中而不损失精度或溢出，那么目标元素与源元素兼容。例如，目标 Long 与源 Integer 兼容，因为在 Long 中可以存储 Integer 而不丢失信息或溢出。目标 Long 与源 Double 不兼容，因为在 Long 中不能够存储 Double 而不丢失信息（小数部分将丢失）或者可能溢出。函数 AreDataTypesCompatible 用于测试兼容的数据类型。可以通过设置 NoCompatibilityCheck 参数为 True 来跳过兼容测试。注意，这可能会导致信息丢失（当复制 Single 或 Double 给 Integer 或

Long 时，小数位可能会丢失)，或者可能会遇到溢出情况，此时，目标数组的元素会被设置为 0。如果发生溢出错误，那么该过程将忽略它，并将目标数组元素设置为 0。

(6) CopyArray

```
Public Function CopyArray(DestinationArray As Variant, SourceArray As Variant, _  
    Optional NoCompatibilityCheck As Boolean = False) As Boolean
```

这个函数将 SourceArray 复制到 DestinationArray。不巧的是，VBA 不允许你使用单个的赋值语句将一个数组复制到另一个数组。必须逐元素复制数组。如果 DestinationArray 是动态的，它会调整大小以容纳 SourceArray 中的所有值。DestinationArray 具有与 SourceArray 相同的下限和上限。如果 DestinationArray 是静态的，并且 SourceArray 比 DestinationArray 的元素更多，那么仅 SourceArray 的开头部分的元素被复制来填充 DestinationArray。如果 DestinationArray 是静态的且 SourceArray 具有比 DestinationArray 更少的元素，那么 DestinationArray 后面的元素将保持原样。DestinationArray 不会调整大小来匹配 SourceArray。如果 SourceArray 为空（未分配），那么 DestinationArray 保持原样。如果 SourceArray 和 DestinationArray 都未分配，那么函数退出且不会修改任何数组。

默认情况下，CopyArray 确保 SourceArray 和 DestinationArray 的数据类型是相同的或者兼容的。如果源元素的值可以存储在目标元素中而不损失精度或溢出，那么目标元素与源元素兼容。例如，目标 Long 与源 Integer 兼容，因为在 Long 中可以存储 Integer 而不丢失信息或溢出。目标 Long 与源 Double 不兼容，因为在 Long 中不能够存储 Double 而不丢失信息（小数部分将丢失）或者可能溢出。函数 AreDataTypesCompatible 用于测试兼容的数据类型。可以通过设置 NoCompatibilityCheck 参数为 True 来跳过兼容测试。注意，这可能会导致信息丢失（当复制 Single 或 Double 给 Integer 或 Long 时，小数位可能会丢失），或者可能会遇到溢出情况，此时，目标数组的元素会被设置为 0。如果发生溢出错误，那么该过程将忽略它，并将目标数组元素设置为 0。

(7) CopyArraySubSetToArray

```
Public Function CopyArraySubSetToArray(InputArray As Variant, ResultArray As  
Variant, _  
    FirstElementToCopy As Long, LastElementToCopy As Long, DestinationElement  
As Long) As Boolean
```

这个函数将 InputArray 的子集复制到 ResultArray 中的某个位置。InputArray 的 FirstElementToCopy 和 LastElementToCopy（包括）之间的元素被复制到 ResultArray，开始于 DestinationElement。在

ResultArray 中已有的数据被覆盖。如果 ResultArray 不足以存储新数据，那么如果它是动态数组，则会适当调整大小。如果 ResultArray 是静态数组并且不够容纳新数据，那么会发生错误且该函数返回 False。InputArray 和 ResultArray 可以都是动态数组，但是 InputArray 必须分配空间，ResultArray 可以不分配空间。如果 ResultArray 未分配，它使用 LBound 为 1 且 UBound 为 DestinationElement + NumElementsToCopy - 1 调整大小。DestinationElement 左侧的元素是数组数据类型的默认值 (0、vbNullString、Empty 或 Nothing)。当从一个数组复制元素到另一个数组时，不会执行类型检查。如果 InputArray 与 ResultArray 不兼容，不会触发错误，在 ResultArray 中的值将是该数组的数据类型的默认值 (0、vbNullString、Empty 或 Nothing)。

(8) CopyNonNothingObjectToArray

```
Public Function CopyNonNothingObjectsToArray(ByRef SourceArray As Variant, _  
ByRef ResultArray As Variant, Optional NoAlerts As Boolean = False) As Boolean
```

这个函数将 SourceArray 中不为 Nothing 的所有对象复制到新的 ResultArray。必须将 ResultArray 声明为 Object 或 Variant 型的动态数组。SourceArray 必须包含所有对象类型变量（虽然对象类型可能会混合——数组可能包含多种类型的对象）或者 Nothing 对象。如果在 SourceArray 中找到非对象变量，则会发生错误。

(9) DataTypeOfArray

```
Public Function DataTypeOfArray(Arr As Variant) As VbVarType
```

这个函数返回指定数组的数据类型 (vbVarType 值)。如果指定的数组是简单数组，单或多维，那么该函数返回其数据类型。指定的数组可以未分配空间。如果传递到 DataTypeOfArray 的变量不是数组，那么该函数返回 -1。如果指定的数组是一组数组，那么结果是 vbArray。例如：

```
Dim V(1 to 5) As String  
Dim R As VbVarType  
R = DataTypeOfArray(V) ' returns vbString = 8
```

(10) DeleteArrayElement

```
Public Function DeleteArrayElement(InputArray As Variant, ElementNumber As Long, _  
Optional ResizeDynamic As Boolean = False) As Boolean
```

这个函数从 InputArray 中删除指定的元素，将删除的元素的右侧的所有元素

向左移动一个位置。该数组的最后一个元素被设置成合适的缺省值（0、vbNullString、Empty 或 Nothing），这取决于该数组中数据的类型。数据类型由数组中的最后一个元素确定。默认情况下，数组的大小不会改变。如果 ResizeDynamic 参数是 True，且 InputArray 是动态数组，那么它将缩小一个元素以删除该数组最后一个元素。

(11) ExpandArray

```
Function ExpandArray(Arr As Variant, WhichDim As Long, AdditionalElements As Long, _  
    FillValue As Variant) As Variant
```

这个函数扩展二维数组的任一维度。它返回一个带有添加了行或列的数组。在数组的底部添加行，在数组的右侧添加列。Arr 是原始数组，不能以任何方式修改该数组。WhichDim 指明是否添加额外的行（WhichDim=1）或者额外的列（WhichDim=2）。AdditionalElements 指示要添加到 Arr 的额外的行或列的数量。新的数组元素使用 FillValue 中的值初始化。如果发生错误，那么该函数返回 NULL。该函数可以嵌套以添加行和列。下面的代码对数组 A 添加 3 行和 4 列并在 C 中放置结果数组。

```
Dim A()  
Dim B()  
Dim C()  
  
' .....  
' Redim A, B, and C, and give them some values here.  
' .....  
  
C = ExpandArray(ExpandArray(Arr:=A, WhichDim:=1,  
AdditionalElements:=3, FillValue:="R"), _  
    WhichDim:=2, AdditionalElements:=4, FillValue:="C")
```

(12) FirstNonEmptyStringIndexInArray

```
Public Function FirstNonEmptyStringIndexInArray(InputArray As Variant) As Long
```

这个函数返回不等于 vbNullString 的元素的字符串数组中第一个条目的索引号。当处理按升序排列的字符串数组时非常有用，它将 vbNullString 条目放置在数组的开头。一般来说，InputArray 将按升序排列。例如：

```
Dim A(1 To 4) As String  
Dim R As Long  
A(1) = vbNullString
```

```

A(2) = vbNullString
A(3) = "A"
A(4) = "B"
R = FirstNonEmptyStringIndexInArray(InputArray:=A)
' R = 3, the first element that is not an empty string
Debug.Print "FirstNonEmptyStringIndexInArray", CStr(R)

```

(13) GetColumn

```

Function GetColumn(Arr As Variant, ResultArr As Variant, ColumnNumber As Long) As Boolean

```

这个函数使用一维数组填充 ResultArr，该数组是通过输入数组 Arr 的 ColumnNumber 指定的列。ResultArr 必须是一个动态数组。ResultArr 的已有内容会被销毁。

(14) GetRow

```

Function GetRow(Arr As Variant, ResultArr As Variant, RowNumber As Long) As Boolean

```

这个函数使用一维数组填充 ResultArr，该数组是通过输入数组 Arr 的 RowNumber 指定的行。ResultArr 必须是一个动态数组。ResultArr 的已有内容会被销毁。

(15) InsertElementIntoAnArray

```

Public Function InsertElementIntoArray(InputArray As Variant, Index As Long, _
    Value As Variant) As Boolean

```

这个函数在 InputArray 中插入值 Value 在 Index 位置。InputArray 必须是一个单维动态数组。它将会调整大小以适应新的数据元素。要在数组的结尾插入元素，设置 Index 为 UBound(Array)+1。

(16) IsArrayAllDefault

```

Public Function IsArrayAllDefault(InputArray As Variant) As Boolean

```

这个函数返回 TRUE 或者 FALSE 来指明数组中所有元素是否具有特定数据类型的默认值。取决于数组的数据类型，默认值可能是 vbNullString、0、Empty 或 Nothing。

(17) IsArrayAllNumeric

```
Public Function IsArrayAllNumeric(Arr As Variant, _  
    Optional AllowNumericStrings As Boolean = False) As Boolean
```

这个 函数返回 TRUE 或者 FALSE 来指明是否数组中所有元素是数字。默认情况下，字符串不被视为数字，即便它们包含数字值。要允许数字字符串，设置参数 AllowNumericStrings 为 True。

(18) IsArrayAllocated

```
Public Function IsArrayAllocated(Arr As Variant) As Boolean
```

这个函数返回 TRUE 或 FALSE 来指明是否指定的数组被分配空间（不为空）。数组是静态数组，或者是使用 ReDim 语句分配的动态数组，返回 TRUE。如果数组是仍没有使用 ReDim 调整大小或者使用 Erase 语句清空的动态数组，返回 FALSE。这个函数基本上与 ArrayIsEmpty 相反，例如：

```
Dim V() As Long  
Dim R As Boolean  
R = IsArrayAllocated(V) ' returns false  
ReDim V(1 To 10)  
R = IsArrayAllocated(V) ' returns true
```

(19) IsArrayDynamic

```
Public Function IsArrayDynamic(ByRef Arr As Variant) As Boolean
```

这个函数返回 TRUE 或者 FALSE 来指明指定的数组是否是动态数组。如果该数组是动态数组返回 TRUE，如果该数组是静态数组则返回 FALSE。例如：

```
Dim DynArray() As Long  
Dim StatArray(1 To 3) As Long  
Dim B As Boolean  
B = IsArrayDynamic(DynArray) ' returns True  
B = IsArrayDynamic(StatArray) ' returns False
```

(20) IsArrayEmpty

```
Public Function IsArrayEmpty(Arr As Variant) As Boolean
```

这个函数返回 TRUE 或者 FALSE 来指明指定的数组是否为空（未分配）。该函数

基本上与 IsArrayAllocated 相反。

```
Dim DynArray() As Long
Dim R As Boolean
R = IsArrayEmpty(DynArray) ' returns true
ReDim V(1 To 10)
R = IsArrayEmpty(DynArray) ' returns false
```

(21) IsArrayObjects

```
Public Function IsArrayObjects(InputArray As Variant, _
    Optional AllowNothing As Boolean = True) As Boolean
```

这个函数返回 TRUE 或 FALSE 来指明指定的数组是否包含了所有的对象变量。对象可能是混合类型。默认情况下，函数允许 Nothing 对象。也就是说，一个为 Nothing 的对象仍然被考虑是一个对象。要使对象是 Nothing 时返回 False，设置参数 AllowNothing 为 False。

(22) IsArraySorted

```
Public Function IsArraySorted(TestArray As Variant, _
    Optional Descending As Boolean = False) As Variant
```

这个函数返回 TRUE 或 FALSE 来指明数组 TestArray 是否按排序顺序排列(升序或降序，取决于参数 Descending 的值)。如果发生错误，则返回 NULL。TestArray 必须是单维的已分配的数组。由于排序是一项消耗很大的操作，特别是含有大量 String 或 Variant 型的数组，可以调用该函数来确定在调用 Sort 过程之前数组是否已经排序。如果该函数返回 TRUE，那么你不需要重新排序数组。

(23) IsNumericDataType

```
Public Function IsNumericDataType(TestVar As Variant) As Boolean
```

这个函数用来指明变量是否是数字数据类型 (Long、Integer、Double、Single、Currency 或 Decimal)。如果输入是一个数组，那么该函数测试数组的第一个元素 (注意，在一组变体中，后续元素可能不是数字)。对于变体数组，使用 IsVariantArrayNumeric。

(24) IsVariantArrayConsistent

```
Public Function IsVariantArrayConsistent(Arr As Variant) As Boolean
```

如果在一组 Variant 中所有数据类型都具有相同的数据类型,那么返回 TRUE。否则,返回 FALSE。如果数组由对象类型变量组成,那么是 Nothing 的对象将跳过。如果所有的非对象变量有相同类型,那么该函数返回 TRUE。

(25) IsVariantArrayNumeric

Public Function IsVariantArrayNumeric(TestArray As Variant) As Boolean

这个函数返回 TRUE 或 FALSE 来看 Variant 数组是否包含所有数字数据类型。数据类型不需要是相同的,可以是 Integer、Long、Single、Double 和 Empty 的混合。只要所有的数据类型是数字的(由 IsNumericDataType 函数确定),结果将是 True。如果数据类型不全为数字或者传入的参数不是数组或者是未分配的数组,则该函数返回 FALSE。这个过程处理多维数组。

(26) MoveEmptyStringsToEndOfArray

Public Function MoveEmptyStringsToEndOfArray(InputArray As Variant) As Boolean

这个函数移除数组开始和结尾的空字符串,数组中的元素向左移。这在处理数组开头放置有空字符串的已排序的文本字符串数组时很有用。例如:

```
Dim S(1 to 4) As String
Dim N As Long
S(1) = vbNullString
S(2) = vbNullString
S(3) = "abc"
S(4) = "def"
N = MoveEmptyStringsToEndOfArray(S)
' resulting array:
For N = LBound(S) To UBound(S)
    If S(N) = vbNullString Then
        Debug.Print CStr(N), "Is vbNullString"
    Else
        Debug.Print CStr(N), S(N)
    End If
Next N
```

(27) NumberOfDimensions

Public Function NumberOfArrayDimensions(Arr As Variant) As Integer

这个函数返回指定数组的维数。如果数组是未分配的动态数组，那么返回 0。

```
Dim V(1 to 10, 1 to 5) As Long
Dim N As Long
N = NumberOfDimensions(V) ' returns 2
```

(28) NumElements

Public Function NumElements(Arr As Variant, Optional Dimension = 1) As Long

这个函数返回指定数组的指定维度中元素数。如果存在错误条件，那么返回 0（例如，未分配数组）。

```
Dim V(1 to 10) As Long
Dim Dimension As Long
Dim N As Long
Dimension = 1
N = NumElements(V, Dimension) ' returns 10
```

(29) SetVariableToDefault

Public Sub SetVariableToDefault(ByRef Variable As Variant)

这个过程设置参数 Variable 为其数据类型的合适的默认值，可能是 0、vbNullString、Empty 或 Nothing。注意，不能重置用户定义类型。可以通过声明该类型的第二个变量来轻松地将用户定义类型设置回其默认状态，例如 Dim DefaultType As MyType，并让这些元素取其默认值。然后使用 LSet 将 UDT 的另一个实例设置为 DefaultType：

```
Public Type MyType
    X As Long
    Y As Long
    S As String
End Type

Dim DefaultType As MyType
Dim DataT As MyType
DataT.X = 1
DataT.Y = 2
```

```
DataT.S = "Test"
' set variables of T back to defaults.
LSet DataT = DefaultType
```

(30) SwapArrayRows 与 (31) SwapArrayColumns

```
Function SwapArrayColumns(Arr As Variant, Col1 As Long, Col2 As Long) As Variant
```

```
Function SwapArrayRows(Arr As Variant, Row1 As Long, Row2 As Long) As Variant
```

这些函数接收一个数组 Arr，返回该数组的副本，其行或列被交换。

(32) ResetVariantArrayToDefaults

```
Public Function ResetVariantArrayToDefaults(InputArray As Variant) As Boolean
```

这个函数将 Variant 数组的所有元素重置为合适的默认值(0、vbNullString、Empty 或 Nothing)。数组可能由几种不同的数据类型组成(例如，一些是 Long 型、一些是 Object、一些是 String，等等)，每个元素都将被重置为合适的默认值。

(33) ReverseArrayInPlace

```
Public Function ReverseArrayInPlace(InputArray As Variant, _
    Optional NoAlerts As Boolean = False) As Boolean
```

这个子函数颠倒数组的顺序。也就是说，调用过程的数组将被反转。输入数组必须是单维数组。如果数组被成功反转，那么该函数返回 True；如果发生错误，则为 False。

```
Dim V(1 to 10) As String
Dim B As Boolean
' load V with some values
B = ReverseArrayInPlace(V)
```

(34) ReverseArrayOfObjectsInPlace

```
Public Function ReverseArrayOfObjectsInPlace(InputArray As Variant, _
    Optional NoAlerts As Boolean = False) As Boolean
```

这个子函数颠倒数组的顺序。也就是说，调用过程的数组将被反转。该函数返回 True 或 False 来指明数组是否被成功反转。如果数组元素不是对象（允许 Nothing 对象），那么将发生错误。

```
Dim V(1 to 10) As Object
Dim B As Boolean
' load V with some objects
B = ReverseArrayOfObjectsInPlace(V)
```

(35) SetObjectArrayToNothing

```
Public Function SetObjectArrayToNothing(InputArray As Variant) As Boolean
```

这个函数设置指定数组的所有元素为 Nothing。InputArray 必须被声明为一组对象，或者是指定的对象类型、或者是通用的 Object，或者是 Variant。如果数组中的元素不是对象或 Nothing，那么将发生错误。数组不可调整大小——保持同样的大小。在处理 variant 数组时，使用该函数替代 Erase，因为 Erase 将设置数组中每个元素为 Empty，而不是 Nothing，并且元素将不再被视为 Object。

(36) SetVariableToDefault

```
Public Sub SetVariableToDefault(ByRef Variable As Variant)
```

这个过程设置 Variable 为其数据类型的合适的默认值。默认值将是 0、vbNullString、Empty 或 Nothing，这取决于 Variable 的数据类型。

(37) TransposeArray

```
Public Function TransposeArray(InputArr As Variant, OutputArr As Variant) As Boolean
```

这个过程转置二维数组，创建的结果数组的行数等于输入数组的列数，列数等于输入数组的行数。保留 LBound 和 UBound。OutputArr 必须是动态数组，会被清空或重新定义，因此将销毁现有的内容。

(38) VectorsToArray

```
Public Function VectorsToArray(Arr As Variant, ParamArray Vectors()) As Boolean
```

这个过程接收任意数量的单维数组，并将它们组合成单个二维数组。输入数组在参数 ParamArray Vectors() 中，并且它们被放置到的数组由 Arr 指定。Arr 必须是动态数组，其数据类型必须与所有向量中所有元素兼容。Arr 被清空，然后被重新定义，因此会销毁其所有现有的内容。在 Vectors 中的每个数组必须是单维已分配的数组。如果 Vector 是一个未分配的数组，该函数将退出且结果为 False。

Vectors 中的每个数组是 Arr 的一行。Arr 的行数是传入的 Vectors 的数量。

Arr 的每一行是一个向量。列数是 Vectors 的大小的最大值。由于 Arr 被清空，在 Arr 中未使用的元素保持为 Arr 的数据类型的默认值（缺省值是 0、vbNullString 或 Empty，取决于 Arr 是如何被分配的）。每个向量的元素都必须是简单的数据类型，不允许对象、数组和用户定义类型。Arr 的行和列都是基于 0 的，而不管 Arr 最初的设置，每一向量的 LBound 和 Option Base 语句。向量可以有不同的大小及不同的 LBound。

关于完美 Excel

完美 Excel 是我创建的一个微信公众号，自 2017 年 5 月 15 日开始，每天推送一篇 Excel 与 VBA 技术和应用方面的文章。目前，共有 300 余篇实用文章可供广大 Excel 爱好者和使用者学习交流。这本电子书就是根据完美 Excel 上发表的《玩转 Excel 数据有效性》系列文章整理而成的。

每天早晨，完美 Excel 微信公众号：*excelperfect* 都会推送一篇关于 Excel 与 VBA 的相关文章。如果你有兴趣学习 Excel 和 VBA 的相关知识和实战技巧，可以关注完美 Excel 微信公众号，绝对不会让你失望！

可以通过下列方法关注[完美 Excel]微信公众号：

方法 1—在通讯录中搜索“完美 Excel”或者“*excelperfect*”后点击关注。

方法 2—扫一扫下面的二维码



完美 Excel 微信公众号使用指南

下图 1 为完美 Excel 微信公众号的界面。公众号名称显示在屏幕正上方，屏幕底部显示有“菜单栏”，目前设置的菜单为“技术精粹”、“VBA 精选”、“联系 me”。在底部左侧的小键盘图标为消息框入口，单击可进入消息框界面给完美 Excel 公众号发送消息。



图 1

下图 2、图 3、图 4 分别为底部 3 个菜单的子菜单。目前，菜单“技术精粹”中设置有“VBA 学习经验”、“玩转数据验证”、“快速学会 30 个函数”、“全部文章合集 1”等 4 个子菜单；菜单“VBA 精选”中设置有“最最基础入门”、“Range 对象详解”、“工作表对象详解”等 3 个子菜单；菜单“联系 me”中设置有“知识分享架构”、“个人声明”、“答疑解惑”、“坚持的美好”、“爱沐智养亲子中心”等 5 个子菜单。



图 2



图 3



图 4

单击这些子菜单会进入详细的文章页面或者文章整理的入口页面，方便读者浏览或查阅本公众号的文章。同时，这些子菜单会随着完美 Excel 微信公众号内容的增加而适时调整。

可以单击底部左侧的小键盘图标，进入发送消息界面，如图 5 所示。在文本框中输入想要发送的文字，单击底部的“发送”按钮，就可以将消息发送给完美 Excel 微信公众号。



图 5

大家应留意完美 Excel 微信公众号推送的文章中的一些信息，例如，我会在百度网盘中存放一些文档资料或者示例工作簿文件，并在文章中给出进入百度网盘下载的文本信息，你只需在发送消息框中输入我给出的文本，单击发送后，就会收到一条关于下载链接和密码的信息。单击链接并按提示输入密码后，即可获得相关的文档资料或示例工作簿文件了。

例如，在图 5 所示的界面中输入“Excel 动画图 2”后，会自动收到图 6 所示的信息，根据信息即可获取这个 Excel 动画图表文件。



图 6

希望大家在完美 Excel 微信公众号中能够学习到所需要的知识，获取到所需要的 Excel 应用技巧，提高自己的水平。