

第 7 章 图表对象操作

在前面的章节中，已经详细讲解了如何使用 Excel VBA 处理 Excel 的数据。除此之外，Excel 还可以创建各种不同类型的图表。数据是图表的基础，而图表则可以更加直观的显示数据。在 Excel VBA 中，同样给图表提供多种属性和事件。在本章中，将详细讲解如何使用 VBA 代码编辑图表。

7.1 创建图表

在 Excel 中，用户可以通过向创建所需要的图表类型。同时，用户在开发过程中，通过 VBA 代码创建各种类型的图表。在本小节中，将详细讲解如何使用 VBA 创建图表。

案例 136 创建图表工作表

1. 功能说明

在 Excel VBA 中，用户可以通过 VBA 代码来创建图表工作表。同时可以通过 VBA 代码来定制图表的属性。

2. 语法说明

在 Excel VBA 中，Charts 集合包含工作簿中所有图表工作表的集合。通过 Charts 集合的 Add 方法可向集合中添加新的图表工作表（新建图表工作表），Add 方法的语法格式如下：

表达式.Add(Before, After, Count, Type)

该方法的参数都可省略，各参数的含义如下：

- **Before:** 指定工作表的对象，新建的工作表将置于此工作表之前。
- **After:** 指定工作表的对象，新建的工作表将置于此工作表之后。
- **Count:** 要添加的工作表数。默认值为 1。
- **Type:** 指定要添加的图表类型，可创建的图表类型很多，具体可参考 Excel VBA 的帮助信息。

提示：如果 Before 和 After 两者都被省略，新建的图表工作表将插入到活动工作表之前。

在 Excel VBA 中，通过 Chart 对象的 SetSourceData 方法，可为指定图表设置源数据区域。其语法格式如下：

表达式.SetSourceData(Source, PlotBy)

该方法的两个参数含义如下：

- **Source**: 为一个 Range 对象, 用来指定图表的源数据区域。
- **PlotBy**: 指定数据绘制方式。可使用常量 xlColumns (数据系列在行中) 和 xlRows (数据系列在列中) 之一。

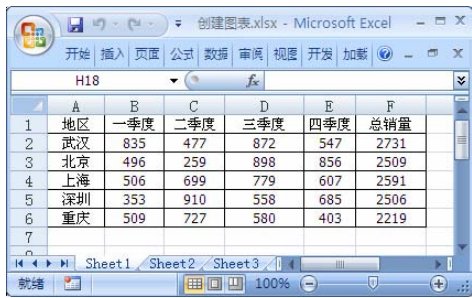
通过 Chart 对象的 ChartTitle 属性, 可返回 ChartTitle 对象, 该对象表示指定图表的标题。通过该对象的属性可控制图表的标题, 如设置标题文本、设置标题的格式等。同时, 要设置图表的内容, 用户需要熟悉关于图表的属性。其常见的属性如表 7.1 所示。

表 7.1 Chart对象的常用属性

名称	意义
BackWall	是一个只读属性, 返回Walls对象, 该对象允许用户单独对三维图表的背景墙进行格式设置。
ChartArea	是一个只读属性, 返回一个ChartArea对象, 该对象表示图表的整个图表区。
ChartStyle	是一个可读写的Variant型属性, 用于返回或设置图表的图表样式。可以使用介于1到48之间的数字设置图表样式。
ChartTitle	是一个只读属性, 用于返回一个ChartTitle对象, 表示指定图表的标题。
ChartType	是一个可读写的XlChartType类型, 用于返回或设置图表类型。
DataTable	是一个只读属性, 用于返回一个DataTable对象, 表示此图表的数据表。
HasAxis	是一个可读写的Variant类型属性, 用于返回或设置图表上显示的坐标轴。
HasDataTable	是一个可读写的Boolean型属性, 如果图表有数据表, 则该属性值为True。
PlotArea	是一个只读属性, 用于返回一个PlotArea对象, 表示图表的绘图区。
PlotBy	是一个Long型可读写, 返回或设置行或列在图表中作为数据系列使用的方式。可为以下XlRowCol常量之一: xlColumns或xlRows。
Visible	返回或设置一个XlSheetVisibility值, 用于确定对象是否可见。
Walls	是一个只读属性, 返回一个Walls对象, 此表示三维图表的背景墙。

3. 案例说明

某公司统计了该公司武汉、北京、上海、深圳和重庆地区的四个季度的销量数据, 根据该数据, 用户需要用 VBA 代码创建图表工作表, 其中原始数据如图 7.1 所示。



	A	B	C	D	E	F
1	地区	一季度	二季度	三季度	四季度	总销量
2	武汉	835	477	872	547	2731
3	北京	496	259	898	856	2509
4	上海	506	699	779	607	2591
5	深圳	353	910	558	685	2506
6	重庆	509	727	580	403	2219

图 7.1 原始数据

4. 编写代码

销量分析图表的 VBA 代码如下：

```
Sub CreatCharts()  
    Dim cht As Chart  
    Set cht = Charts.Add  
    With cht  
        .SetSourceData Source:=Sheets("Sheet1").Range("A1:E6"), PlotBy:=xlRows  
        .ChartType = xlColumnClustered  
        .HasTitle = True  
        .ChartTitle.Text = "销量数据图"  
    End With  
End Sub
```

5. 运行结果

运行程序代码，查看运行的结果，如图 7.2 所示。

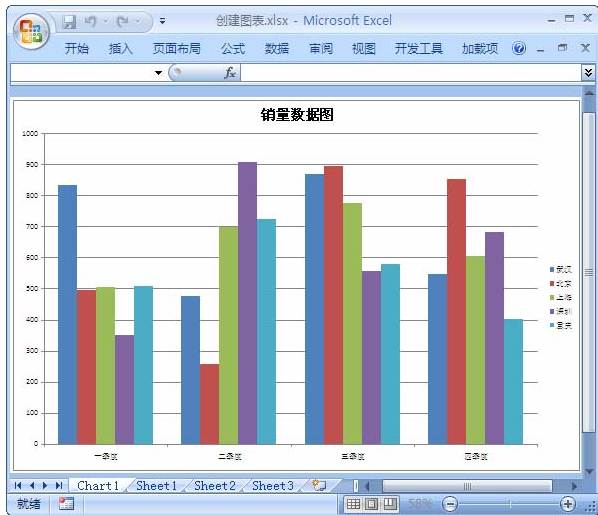


图 7.2 创建图表工作表

6. 程序分析

如果用户对图表的属性不熟悉的时候，可以使用宏功能来创建图表。查看宏代码中的图表属性和代码。

案例 137 创建嵌入式图表

1. 功能说明

在 Excel VBA 中，图表工作表和嵌入式图表是不同的对象。但是，用户同样可以通过 VBA 代码来创建嵌入式图表。

2. 语法说明

在 Excel VBA 中，嵌入到工作表中的图表对象为 **ChartObject** 对象。**ChartObjects** 集合包含指定工作表上所有的 **ChartObject** 对象的集合。每个 **ChartObject** 对象都代表一个嵌入图表。**ChartObject** 对象充当 **Chart** 对象的容器。**ChartObject** 对象的属性和方法控制工作表上嵌入图表的外观和大小。

通过 **ChartObjects** 集合的 **Add** 方法，可向集合中添加嵌入式图表。其语法格式如下：
`表达式.Add(Left, Top, Width, Height)`
该方法的 4 个参数指定嵌入式图表尺寸，分别设置左上角的坐标位置和图表的初始大小。使用 **ChartObjects** 集合的 **Delete** 方法可删除指定工作表的嵌入式图表。

3. 案例说明

某公司统计了该公司武汉、北京、上海、深圳和重庆地区的四个季度的销量数据，根据该数据，用户需要用 VBA 代码创建嵌入式图表，其中原始数据如图 7.3 所示。



	A	B	C	D	E	F	G
1	地区	一季度	二季度	三季度	四季度	总销量	
2	武汉	835	477	872	547	2731	
3	北京	496	259	898	856	2509	
4	上海	506	699	779	607	2591	
5	深圳	353	910	558	685	2506	
6	重庆	509	727	580	403	2219	
7							
8							
9							
10							

图 7.3 原始数据

4. 编写代码

创建嵌入式图表的 VBA 代码如下：
`Sub CreatInsertCharts()`

```

Dim cht As ChartObject
On Error Resume Next
ActiveSheet.ChartObjects.Delete

On Error GoTo 0
With Range("A10:F20")
    Set cht = ActiveSheet.ChartObjects.Add( _
        .Left, .Top, .Width, .Height)
End With

With cht
    .Name = "SaleChart"

    With .Chart
        .SetSourceData Source:=Worksheets("Sheet1").Range("A1:E6"), PlotBy:=xlRows
        .ChartType = xlColumnClustered
        .SetElement msoElementChartTitleCenteredOverlay
        .ChartTitle.Text = "销量数据图"
    End With
End With

End Sub

```

5. 运行结果

运行程序代码，查看运行的结果，如图 7.4 所示。



图 7.4 创建嵌入式图表

6. 程序分析

在上面的代码中，首先删除当前工作表中的嵌入图表，如果当前工作表中没有嵌入图表，执行 `Delete` 方法时将出现错误，所以需使用错误捕捉语句获取错误。接着使用 `ChartObjects` 集合对象的 `Add` 方法添加一个嵌入式图表，最后设置图表对象的相关属性。

案例 138 确定图表的位置

1. 功能说明

在用户使用图表分析和显示数据的时候，常常需要根据需要确定图表的位置。用户可以通过 `VBA` 代码实现该功能。

2. 语法说明

在 `Excel VBA` 中，用户可以通过 `Chart` 对象的 `Location` 方法，可改变图表的放置位置。该方法的语法格式如下：

表达式.`Location`(Where, Name)

两个参数的含义如下：

- **Where**：用来设置图表移动的目标位置。可设置为 `xlLocationAsNewSheet`（将图表移动到新工作表）、`xlLocationAsObject`（将图表嵌入到现有工作表中）或 `xlLocationAutomatic`（`Excel` 控制图表位置）三个常量之一。
- **Name**：如果 `Where` 为 `xlLocationAsObject`，则该参数为必选参数。如果 `Where` 为 `xlLocationAsObject`，则该参数为嵌入该图表的工作表的名称。如果 `Where` 为 `xlLocationAsNewSheet`，则该参数为新工作表的名称。

3. 案例说明

某公司统计了该公司武汉、北京、上海、深圳和重庆地区的四个季度的销量数据，根据该数据，用户需要用 `VBA` 代码确定图表的不同位置，其中原始数据如图 7.5 所示。



	A	B	C	D	E	F
1	地区	一季度	二季度	三季度	四季度	总销量
2	武汉	835	477	872	547	2731
3	北京	496	259	898	856	2509
4	上海	506	699	779	607	2591
5	深圳	353	910	558	685	2506
6	重庆	509	727	580	403	2219
7						
8						

图 7.5 原始数据

4. 编写代码

(1) 嵌入图表转图表工作表的 VBA 代码如下:

```
Sub ChartForm1()  
    Dim cht As ChartObject  
    On Error Resume Next  
    Set cht = ActiveSheet.ChartObjects(1)  
    If cht Is Nothing Then Exit Sub  
    cht.Chart.Location xlLocationAsNewSheet, "销量数据图"  
End Sub
```

(2) 图表工作表转嵌入图表的 VBA 代码如下:

```
Sub ChartForm2()  
    Dim cht As Chart  
    Dim chto As ChartObject  
    On Error Resume Next  
    Set cht = Charts("销量数据图")  
    If cht Is Nothing Then Exit Sub  
    cht.Location xlLocationAsObject, ActiveSheet.Name  
    Set chto = ActiveSheet.ChartObjects(1)  
    With Range("A10:F20")  
        chto.Top = .Top  
        chto.Left = .Left  
        chto.Width = .Width  
        chto.Height = .Height  
    End With  
End Sub
```

5. 运行结果

运行将嵌入图表转图表工作表的程序代码, 查看运行的结果, 如图 7.6 所示。



图 7.6 嵌入图表转图表工作表

运行图表工作表转嵌入图表的代码，查看运行的结果，如图 7.7 所示。



图 7.7 图表工作表转嵌入图表

6. 程序分析

在上面第一段代码中,通过工作表的 ChartObjects 集合返回的是一个 ChartObject 对象,要改变其位置,需使用该对象的 Chart 属性返回一个 Chart 对象,通过 Chart 对象的 Location 方法才能改变图表对象的位置。

案例 139 删除图表

1. 功能说明

在 Excel VBA 中,用户除了可以使用代码创建图表,还可以使用代码删除图表。在本小节中,将结合具体的例子来演示如何使用 VBA 删除图表。

2. 语法说明

本例使用了 ChartObject 对象的三个方法来完成相应的功能,各方法的含义如下:

- BringToFront: 将图表放到 z-次序前面,即将图表显示在最前面。
- Activate: 使当前图表成为活动图表(激活图表)。
- Delete: 删除图表。

3. 案例说明

某公司统计了该公司武汉、北京、上海、深圳和重庆地区的四个季度的销量数据，根据该数据，用户需要用 VBA 代码演示如何删除图表，其中原始数据如图 7.8 所示。



图 7.8 原始数据

4. 编写代码

删除图表的 VBA 代码如下：

```
Sub DeleteChart()  
    Dim cht As ChartObject  
    For Each cht In ActiveSheet.ChartObjects  
        cht.BringToFront  
        cht.Activate  
        If MsgBox("删除图表？", vbQuestion + vbYesNo, "删除图表") = vbYes Then  
            cht.Delete  
        End If  
    Next  
End Sub
```

5. 运行结果

运行程序代码，查看运行的结果，如图 7.9 所示。



图 7.9 删除第一个图表

单击“是”按钮后，删除第二个图表，如图 7.10 所示。



图 7.11 删除第二个图表

单击“是”按钮，查看删除后的结果，如图 7.12 所示。

A screenshot of the Microsoft Excel worksheet after the charts have been deleted. The data table is clearly visible, showing sales data for five regions across four quarters and a total sales column. The worksheet title is '删除图表.xlsxm - Microsoft Excel'.

图 7.12 删除图表后的工作表

6. 程序分析

在本例中，通过循环语句依次激活工作表中的图表对象，然后将其激活，提示用户是否删除图表。这种方法十分常见。

7.2 设置图表的属性

在 Excel VBA 中，用户除了可以创建和删除图表，还可以设置图表的属性。在本小节中，将结合具体的例子来讲解如何使用 VBA 设置图表的属性。

案例 140 确定是否是图表工作表

1. 功能说明

在使用 Excel 分析数据的时候，首先需要判断工作表的类型。数据工作表和图表工作表在属性上有明显的不同，因此首先判断工作表是否是图表工作表十分必要。

2. 语法说明

在 Excel VBA 中，用户可以使用 TypeName 函数判断工作表的类型，该函数的语法格式如下：

TypeName(varname)

参数 varname 包含用户定义类型变量之外的任何变量。TypeName 函数的返回值为一个字符串，该字符串可能是数据类型名称，或者是对象名称。例如，在本例中，图表工作表的返回值为“Chart”，普通工作表的返回值为“Worksheet”。

3. 案例说明

某公司统计了该公司武汉、北京、上海、深圳和重庆地区的四个季度的销量数据，根据该数据，用户创建对应的图表。现在需要通过 VBA 判断图表工作表，其中原始数据如图 7.13 所示。



	A	B	C	D	E	F	G
1	地区	一季度	二季度	三季度	四季度	总销量	
2	武汉	835	477	872	547	2731	
3	北京	496	259	898	856	2509	
4	上海	506	699	779	607	2591	
5	深圳	353	910	558	685	2506	
6	重庆	509	727	580	403	2219	
7							
8							

图 7.13 原始数据

4. 编写代码

确定是否是图表工作表的 VBA 代码如下：

```
Sub CheckChartForm()  
  
Dim i As Integer  
  
For i = 1 To ActiveWorkbook.Sheets.Count  
    If TypeName(Sheets(i)) = "Chart" Then  
        Sheets("Sheet1").Activate  
        MsgBox "工作表" & Sheets(i).Name & "是图表工作表！"  
    End If  
Next  
  
End Sub
```

5. 运行结果

运行程序代码，查看运行的结果，如图 7.14 所示。

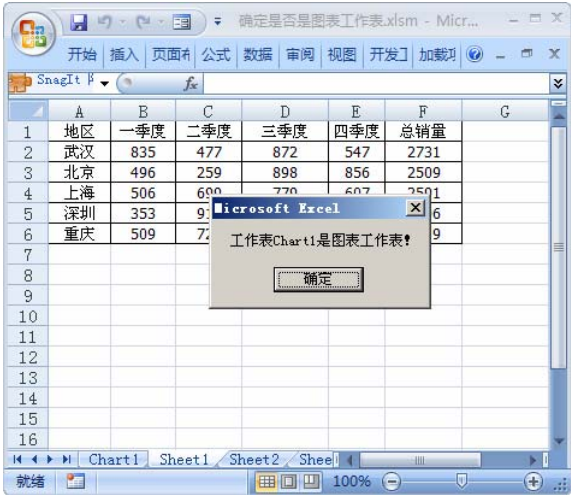


图 7.14 运行程序结果

6. 程序分析

在本例中，首先通过 Sheets 对象的 Count 方法统计该工作簿中的工作表个数。然后使用 For 循环检测各个工作表的类型。

案例 141 添加网格线

1. 功能说明

在 Excel 图表中，网格线是一个重要的工具。为图表添加合适的网格线，用户可以方便的查看图表中的数值。

2. 语法说明

在 Excel VBA 中，设置图表网格线的属性，主要使用 `Axis.HasMajorGridlines` 属性和 `Axis.HasMinorGridlines` 属性。其中，`Axis.HasMajorGridline` 属性用来设置图表的主要网格线。如果坐标轴有主要网格线，则该值为 `True`。只有主要坐标轴组中的坐标轴才能有网格线。其语法表达式如下：

表达式.HasMajorGridlines

表达式表示代表 `Axis` 对象的变量。`Axis.HasMinorGridlines` 属性用来设置图表的次要网格线。如果坐标轴有次要网格线，则该值为 `True`。只有主要坐标轴组中的坐标轴才能有网格线。其语法表达式如下：

表达式.HasMinorGridlines

表达式表示代表 `Axis` 对象的变量。

3. 案例说明

某公司统计了该公司武汉、北京、上海、深圳和重庆地区的四个季度的销量数据，根据该数据，用户创建了对应的图表。现在需要设置图表的网格线属性，其中原始数据和图表如图 7.15 所示。



图 7.15 原始数据

4. 编写代码

(1) 设置所有网格线的代码如下：

```
Sub AddMajorAxis()  
Dim ObjAxis  
For Each ObjAxis In ActiveChart.Axes  
    ObjAxis.HasMajorGridlines = True  
    ObjAxis.HasMinorGridlines = True  
Next ObjAxis  
End Sub
```

(2) 添加主要网格线的代码如下：

```
Sub DeleteMinorAxis()  
Dim ObjAxis  
For Each ObjAxis In ActiveChart.Axes  
    ObjAxis.HasMajorGridlines = True  
    ObjAxis.HasMinorGridlines = False  
Next ObjAxis  
End Sub
```

(3) 删除所有网格线的代码如下：

```
Sub DeleteAllAixs()  
Dim ObjAxis  
For Each ObjAxis In ActiveChart.Axes  
    ObjAxis.HasMajorGridlines = False  
    ObjAxis.HasMinorGridlines = False  
Next ObjAxis  
End Sub
```

5. 运行结果

选中工作表中的图表，运行第一段代码，查看程序运新的结果，如图 7.16 所示。



图 7.16 添加所有网格线

选中工作表中的图表，运行第二段程序代码，查看运行的结果，如图 7.17 所示。



图 7.17 添加主要网格线

选中工作表中的图表，运行第三段程序代码，查看运行的结果，如图 7.18 所示。



图 7.18 删除所有的网格线

6. 程序分析

在本例的代码比较简单，在实际开发过程中，用户可以通过程序代码设置每个坐标轴的网格线属性。有兴趣的读者可以自行测试。

案例 142 自定义数据源

1. 功能说明

在 Excel 中，用户可以根据具体情况来设置图表的数据源。同时，可以使用程序代码动态的修改图表的数据源。

2. 语法说明

在 Excel VBA 中，使用 Chart 对象的 SetSourceData 方法，可为指定图表设置源数据区域。该方法的语法格式如下：

```
表达式.SetSourceData(Source, PlotBy)
```

两个参数的含义如下：

- Source 指定源数据区域的 Range 对象。
- PlotBy 设置数据绘制方式。可为常量 xlColumns（数据系列在行中）或 xlRows（数据系列在列中）之一。

3. 案例说明

某公司统计了该公司武汉、北京、上海、深圳和重庆地区的四个季度的销量数据，根

据该数据，用户创建了对应的图表。现在需要编写 VBA 代码，使用户可以动态修改数据源，其中原始数据和图表如图 7.19 所示。



图 7.19 原始数据

4. 编写代码

自定义数据源的 VBA 代码如下：

```
Sub SetChartData()  
Dim NewData As Range  
  
If ActiveChart Is Nothing Then  
    MsgBox "没有选中图表！"  
    Exit Sub  
End If  
  
Set NewData = Application.InputBox(prompt:="自定义数据源。", Type:=8)  
ActiveChart.SetSourceData Source:=NewData  
End Sub
```

5. 运行结果

不选中任何图表，运行程序代码，查看程序运行的结果，如图 7.20 所示。

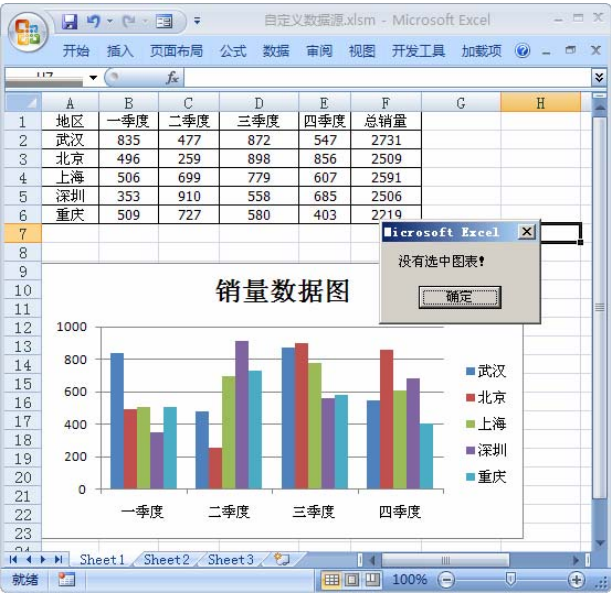


图 7.20 不选中图表运行程序

选中工作表中的默认图表，然后运行程序代码，如图 7.21 所示。



图 7.21 选择新的数据源

单击对话框中的“确定”按钮，查看修改后的图表，如图 7.22 所示。



图 7.22 修改后的图表

6. 程序分析

在上面的代码中，首先判断用户是否选中图表，接着弹出对话框让用户输入或选择新的数据源区域，最后使用 SetSourceData 方法为图表设置新的数据源。

案例 143 添加图表的阴影

1. 功能说明

用户可以使用 VBA 代码设置图表的各种外观属性，在本例中，将演示如何使用代码添加图表的阴影。

2. 语法说明

在 Excel VBA 中，获取对 ShadowFormat 对象的引用后，通过该对象的属性就可控制图表对象的阴影效果。本例使用了以下属性设置阴影效果：

- Visible 属性：设置阴影是否可见；
- Blur 属性：返回或设置指定底纹的模糊度；
- Transparency 属性：返回或设置指定填充的透明度，取值范围为 0.0（不透明）到 1.0（清晰）之间；
- OffsetX 属性：以磅为单位返回或设置指定形状的阴影的水平偏移量。正偏移值将阴影向右偏移，负偏移值将阴影向左偏移；
- OffsetY 属性：以磅为单位返回或设置指定形状阴影的垂直偏移。正偏移值将阴影向

下偏移，负偏移值将阴影向上偏移。

3. 案例说明

某公司统计了该公司武汉、北京、上海、深圳和重庆地区的四个季度的销量数据，根据该数据，用户创建了对应的图表。现在需要编写 VBA 代码，为原始的图表添加阴影，其中原始数据和图表如图 7.23 所示。



图 7.23 原始数据

4. 编写代码

为图表添加阴影的 VBA 代码如下：

```
Sub AddChartShadow()  
    If ActiveChart Is Nothing Then  
        MsgBox "没有选择图表！"  
        Exit Sub  
    End If  
  
    With ActiveChart.ChartArea.Format.Shadow  
        .Visible = msoTrue  
        .Blur = 15  
        .Transparency = 0.2  
        .OffsetX = 6  
        .OffsetY = 6  
    End With  
  
    With ActiveChart.PlotArea.Format.Shadow  
        .Visible = msoTrue  
        .Blur = 4  
        .Transparency = 0.2  
    End With  
End Sub
```

```

.OffsetX = 3
.OffsetY = 3
End With
End Sub

```

5. 运行结果

选中原始图表，运行程序代码，查看程序运行的结果，如图 7.24 所示。



图 7.24 添加图表的阴影

6. 程序分析

以上代码首先检查是否选中了图表，接着设置图表外部区域的阴影效果，最后设置图表中绘制区域的阴影效果。如果用户希望取消图表的阴影，可以使用下面的代码：

```

If ActiveChart Is Nothing Then
    MsgBox "请选择需要设置格式的图表！"
    Exit Sub
End If
ActiveChart.ChartArea.Shadow = False
ActiveChart.PlotArea.Format.Shadow.Visible = msoFalse

```

案例 144 添加数据标签

1. 功能说明

数据标签是图表重要辅助工具，通过数据标签，用户可以直观的查看出图表中的数值。用户可以通过 VBA 代码为图表添加数据标签。

2. 语法说明

在 Excel VBA 中，SeriesCollection 集合对象包含指定的图表或图表组中所有 Series 对象的集合。可用 Chart 对象的 SeriesCollection 方法可返回 SeriesCollection 集合。Series 对象代表图表上的系列，是 SeriesCollection 集合的成员。使用该对象的属性和方法可控制图表中的每个系列。常见的方法和属性如下：

- ApplyDataLabels 方法：向系列应用数据标签。
- Points 方法：返回一个对象，该对象表示数据系列中单个数据点（Point 对象）或所有数据点的集合（Points 集合）。
- HasDataLabels 属性：如果数据系列具有数据标签，则该属性值为 True。

在 Excel VBA 中，Point 对象代表图表系列中的单个数据点。通过该对象可控制图表中数据系列的每一个数据点，如本例使用 DataLabel 设置数据点的标签的显示内容。

3. 案例说明

某公司统计了该公司武汉、北京、上海、深圳和重庆地区的四个季度的销量数据，根据该数据，用户创建了对应的图表。现在需要编写 VBA 代码，为原始的图表添加阴影，其中原始数据和图表如图 7.25 所示。



图 7.25 原始数据

4. 编写代码

添加数据标签的 VBA 代码如下：

```
Sub 显示标签()  
    Dim i As Integer, n As Integer, ser As Series
```

```
If ActiveChart Is Nothing Then
    MsgBox "请选择需要设置格式的图表！"
    Exit Sub
End If

For Each ser In ActiveChart.SeriesCollection
    ser.ApplyDataLabels Type:=xlDataLabelsShowValue, _
        AutoText:=True, LegendKey:=False

    n = ser.Points.Count
    For i = 1 To n
        ser.Points(i).DataLabel.Text = ser.Values(i)
    Next
Next
End Sub
```

5. 运行结果

选中图表，运行程序代码，查看程序运行的结果，如图 7.26 所示。



图 7.26 查看添加的数据标签

6. 程序分析

在上面的代码中，通过一个循环嵌套，对图表中的系列进行逐个处理。每个系列都有多个数据点，内循环完成显示数据的操作。隐藏标签的 VBA 代码如下：

```
If ActiveChart Is Nothing Then
```



```
MsgBox "请选择需要设置格式的图表！"  
Exit Sub  
End If  
  
For Each ser In ActiveChart.SeriesCollection  
    ser.HasDataLabels = False  
Next
```

案例 145 将图表保存为图片

1. 功能说明

图表是特殊的对象，在 Excel 中用户可以灵活处理图表对象。而为了在其他程序中显示或者编辑图表，用户可以将图表保存为图片。

2. 语法说明

在 Excel VBA 中，使用 ChartObjects 对象的 CopyPicture 方法，可将选中图表作为图片复制到剪贴板。该方法的语法格式如下：

表达式.CopyPicture(Appearance, Format)

参数 Appearance 设置图片的复制方式。可设置为以下两个常量之一：

- xlScreen：图片尽可能按其屏幕显示进行复制，这是默认值。
- xlPrinter：图片按其打印效果进行复制。

参数 Format 设置图片的格式。可设置为以下两个常量之一：

- xlBitmap：位图（.bmp、.jpg、.gif）。
- xlPicture：绘制图片（.png、.wmf、.mix）。

3. 案例说明

某公司统计了该公司武汉、北京、上海、深圳和重庆地区的四个季度的销量数据，根据该数据，用户创建了对应的图表。现在需要编写 VBA 代码，将图表转换为图片保存，如图 7.27 所示。



图 7.27 原始数据

4. 编写代码

将图表转换为图片的 VBA 代码如下：

```
Sub SaveASPic()
    If ActiveChart Is Nothing Then
        MsgBox "请选择图表！"
        Exit Sub
    End If

    ActiveChart.CopyPicture Appearance:=xlScreen, Format:=2
    ActiveWindow.Visible = False
    Range("I8").Select
    ActiveSheet.Paste
End Sub
```

5. 运行结果

选中图表，运行程序代码，查看程序运行的结果，如图 7.28 所示。



图 7.28 运行的结果

6. 程序分析

在上面的代码中，为了简单起见，选择将图表保存为位图的形式。

案例 146 设置图表颜色

1. 功能说明

图表具有直观的特点，用户可以设置图表颜色，来增加图表的对比度。用户可以通过 VBA 代码设置图表的颜色。

2. 语法说明

本例通过图表的相关属性获取对各子对象的引用，然后逐个设置子对象的填充颜色。本例用到的子对象如下：

- ChartArea: 图表区域；
- PlotArea: 绘图区；
- Legend: 图例；
- ChartTitle: 图表标题；
- Axes(xlValue): 数值轴；
- Axes(xlCategory): 分类轴；
- SeriesCollection: 序列。

3. 案例说明

某公司统计了该公司武汉、北京、上海、深圳和重庆地区的四个季度的销量数据，根据该数据，用户创建了对应的图表。现在需要编写 VBA 代码，设置图表的数据系列，如图 7.29 所示。



图 7.29 原始数据

4. 编写代码

设置图表颜色的 VBA 代码如下：

```
Sub SetChartColors()  
    Dim i As Integer  
  
    If ActiveChart Is Nothing Then  
        MsgBox "请选择图表！"  
        Exit Sub  
    End If  
  
    With ActiveChart  
        .ChartArea.Format.Fill.ForeColor.RGB = RGB(0, 255, 255)  
        .PlotArea.Format.Fill.ForeColor.RGB = RGB(255, 0, 255)  
    End With  
  
End Sub
```

5. 运行结果

选中图表，运行程序代码，查看程序运行的结果，如图 7.30 所示。

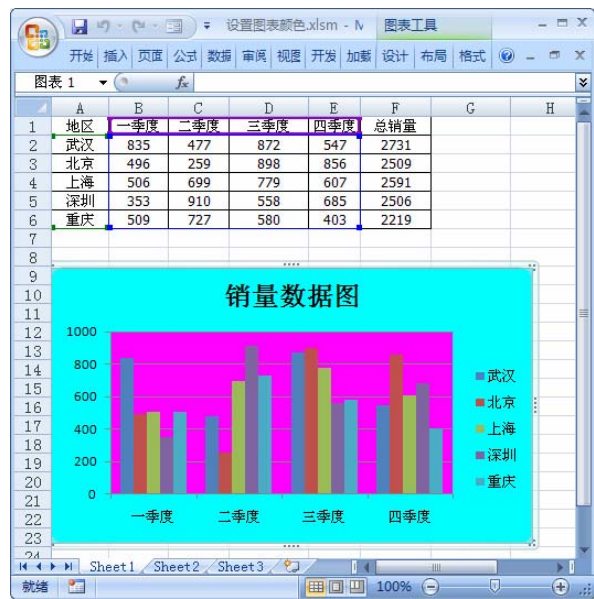


图 7.30 设置图表颜色

6. 程序分析

在 Excel VBA 中，用户可以设置图表多个子对象的颜色，不仅仅是绘图区和图表区。本小节限于篇幅，就不详细列出各种对象的属性。

案例 147 按值显示颜色

1. 功能说明

在 Excel 图表中，数据系列的颜色通常采用默认颜色。用户可以通过 VBA 代码设置数据系列的颜色。

2. 语法说明

在 Excel VBA 中，通过 Interior 对象获取每个数据点对象（Point）的对象内部，再通过 Interior 对象的属性 ColorIndex 设置每个数据点的填充颜色。

3. 案例说明

某公司统计了该公司武汉、北京、上海、深圳和重庆地区的四个季度的销量数据，根据该数据，用户创建了对应的图表。现在需要编写 VBA 代码，设置图表中数据系列的颜色，原始图表如图 7.31 所示。



图 7.31 原始图表

4. 编写代码

按值显示颜色的 VBA 代码如下：

```
Sub SetColorByValue()  
    Dim Ser As Series  
    Dim iTempColor As Integer  
    Dim i As Long  
    Dim IColor As Integer  
  
    If ActiveChart Is Nothing Then  
        MsgBox "请选择图表！"  
        Exit Sub  
    End If  
  
    For Each Ser In ActiveChart.SeriesCollection  
        For i = 1 To Ser.Points.Count  
            Ser.Points(i).Interior.ColorIndex = xlNone  
            iTempColor = Ser.Values(i)  
  
            Select Case iTempColor  
                Case Is < 400  
                    IColor = 7  
                Case Is >= 600  
                    IColor = 5  
            End Select  
        Next i  
    Next Ser  
End Sub
```

```
Case Else
    IColor = 8
End Select

Ser.Points(i).Interior.ColorIndex = IColor
Next i
Next Ser

End Sub
```

5. 运行结果

选中图表，运行程序代码，查看程序运行的结果，如图 7.32 所示。

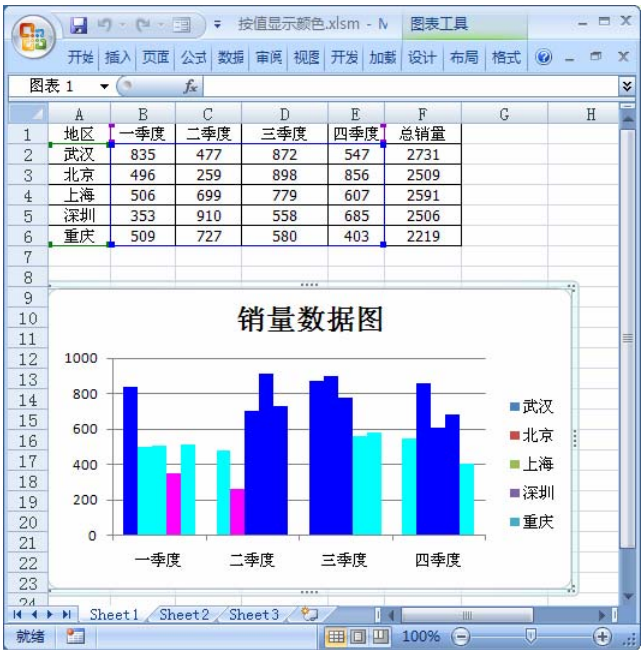


图 7.32 按值显示颜色

6. 程序分析

在上面的代码中，根据每个序列不同数据点的值生成一个颜色值，再将该值赋值给每个数据点作为颜色值。

案例 148 设置图表类型

1. 功能说明

Excel 中提供了多种图表类型，用户可以根据数据特点，选择不同的图表类型。用户

可以通过 VBA 代码来设置不同的图表类型。

2. 语法说明

在 Excel VBA 中，用户可以私用 ChartType 属性设置图表的类型。在 Excel 中，常见的图表类型 ChartType 属性如表 7.2 所示。

表 7.2 xlChartType 的常见取值

xlChartType	图表类型
xl3DArea	三维面积图
xl3DLine	三维折线图
xl3DPie	三维饼图
xlArea	面积图
xl3DLine	三维折线图
xlBubble	气泡图
xlColumnStacked	堆积柱形图
xlLine	折线图
xlXYScatter	散点图
xlPie	饼图

3. 案例说明

某公司统计了该公司武汉、北京、上海、深圳和重庆地区的四个季度的销量数据，根据该数据，用户创建了对应的图表。现在需要编写 VBA 代码，修改图表的类型，原始图表如图 7.33 所示。



图 7.33 原始图表

4. 编写代码

设置图表类型的 VBA 代码如下：

```
Sub SetChartForm()  
If ActiveChart Is Nothing Then  
    MsgBox "没有选择图表！"  
    Exit Sub  
End If  
With ActiveChart  
    .ChartType = xl3DLine  
  
    With .ChartArea.Font  
        .Name = "Arial"  
        .FontStyle = "Regular"  
        .Size = 10  
        .Bold = True  
        .Italic = True  
    End With  
  
    .HasLegend = True  
    .Legend.Position = xlLegendPositionBottom  
    .HasTitle = True  
    .ChartTitle.Text = "销售数据图"  
End With  
End Sub
```

5. 运行结果

选择默认的图表，然后运行程序代码，结果如图 7.34 所示。

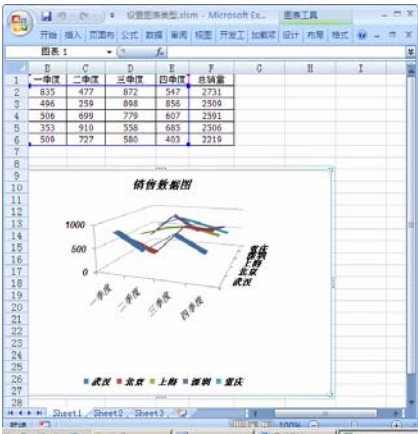


图 7.34 修改图表类型

6. 程序分析

在本例中，首先将图表设置为三维折线图，然后依次设置图表的标题、图例和图表的字体属性。用户可以根据自己的喜好来设置其他的属性数值。

案例 149 设置数据系列的图表类型

1. 功能说明

在 Excel 中，用户除了可以设置图表的类型，还可以为不同的数据系列设置图表类型。用户可以使用 VBA 设置不同数据系列的图表类型。

2. 语法说明

本例所涉及的主要技术是 Series 数据序列对象是 SeriesCollection 集合的成员。SeriesCollection 集合中包含的 Series 对象的个数与图例中包含的图例项个数对应。Series 对象表示绘图区中某个数据序列。需要注意的是，如果需要修改数据序列中单个图形，则需要使用 Series 对象的子对象 Points 集合。因为数据序列中单个图形的类型为 Point。

3. 案例说明

某公司统计了该公司武汉、北京、上海、深圳和重庆地区的上下半年的销量数据，根据该数据，用户创建了对应系列的图表。现在需要编写 VBA 代码，修改其中数据系列的图表类型，原始图表如图 7.35 所示。

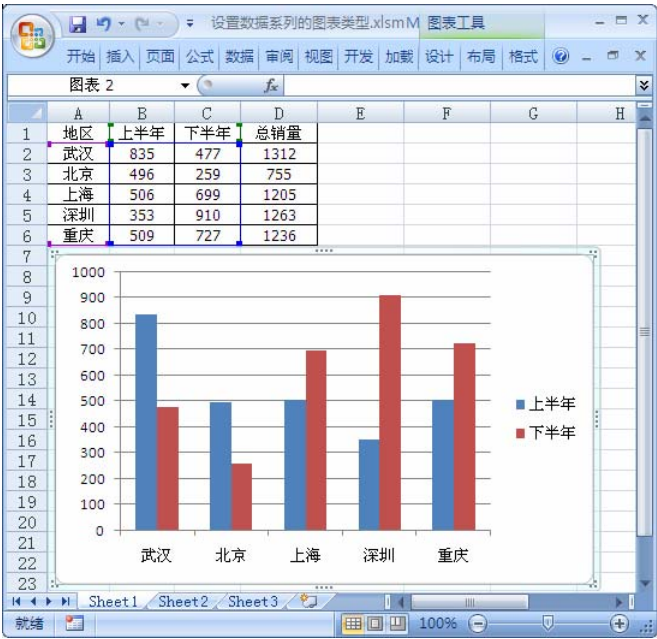


图 7.35 原始图表

4. 编写代码

设置数据系列图表类型的 VBA 代码如下：

```
Sub ChartSeriesChange()  
    Dim Ser As Series  
    Dim Pt As Point  
  
    Set Ser = Worksheets("Sheet1").ChartObjects(1).Chart.SeriesCollection(1)  
    Ser.Type = xlLine  
  
    Set Ser = Worksheets("Sheet1").ChartObjects(1).Chart.SeriesCollection(2)  
    Ser.Type = xlColumn  
  
End Sub
```

5. 运行结果

选择默认的图表，然后运行程序代码，结果如图 7.36 所示。

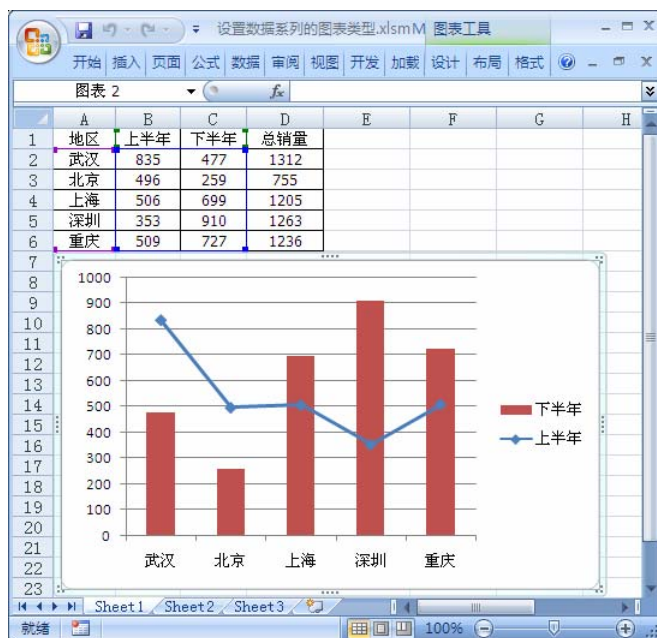


图 7.36 修改数据系列的图表类型

6. 程序分析

在改变图表数据序列类型时，用户需要注意，不可以在同一图表中设置两种不同维度的数据序列。折线图、XY 散点和图气泡等都属于二维图表类型。3D 柱状图、3D 饼图和 3D 气泡图等都属于三维图表类型。

案例 150 设置坐标轴的属性

1. 功能说明

坐标轴是图表中的重要对象，对于不同范围的数据，用户应该选择不同的刻度、单位和间隔等。用户可以使用 VBA 设置坐标轴的属性。

2. 语法说明

在 Excel VBA 中，通过 Chart 对象获取坐标轴对象的语法格式如下所示。

```
Chart.Axes(Type[,AxisGroup])
```

其中 Type 参数用于指定轴类型，该参数可以接受 XlAxisType 枚举值中的三个常量值。AxisGroup 参数指定坐标轴主次之分。当设置为 xlSecondary 时，说明坐标轴为辅助轴。当设置为 xlPrimary 时，说明坐标轴为主坐标轴。XlAxisType 枚举值如表 7.3 所示。

表 7.3 XlAxisType 枚举值表

常量名	常量值	描述
xlCategory	1	表示坐标轴显示的类别。常用于设置图表的水平轴。
xlSeriesAxis	3	表示坐标轴显示的数据系列。该常量只能用于 3D 图表。
xlValue	2	坐标轴显示值。常用语设置图表的垂直轴

3. 案例说明

某公司统计了该公司武汉、北京、上海、深圳和重庆地区的上下半年的销量数据，根据该数据，用户创建了对应系列的图表。现在需要编写 VBA 代码，修改原始图表中的坐标轴属性，原始图表如图 7.37 所示。



图 7.37 原始图表

4. 编写代码

设置坐标轴线属性的 VBA 代码如下：

```
Sub SetAxes()  
    Dim Axs As Axis  
  
    With Worksheets("Sheet1").ChartObjects(1).Chart  
        Set Axs = .Axes(xlValue)  
        With Axs  
            .HasTitle = True  
            .AxisTitle.Caption = "上半年销售数据"  
            .AxisTitle.Orientation = xlVertical  
            .TickLabels.NumberFormat = "0"  
        End With  
  
        Set Axs = .Axes(xlCategory)  
  
        With Axs  
            .HasTitle = True  
            .AxisTitle.Caption = "地区分布刻度"  
            .TickLabels.Orientation = xlVertical  
        End With  
        .SeriesCollection(2).AxisGroup = 2  
  
        Set Axs = .Axes(xlValue, 2)  
        With Axs  
            .HasTitle = True  
            .AxisTitle.Caption = "下半年销售数据"  
            .AxisTitle.Orientation = xlVertical  
            .TickLabels.NumberFormat = "0"  
            .HasMajorGridlines = True  
            With .MajorGridlines.Border  
                .Color = RGB(0, 255, 255)  
                .LineStyle = xlDot  
            End With  
        End With  
  
    End With  
  
End With  
  
End Sub
```

5. 运行结果

选择默认的图表，然后运行程序代码，结果如图 7.38 所示。

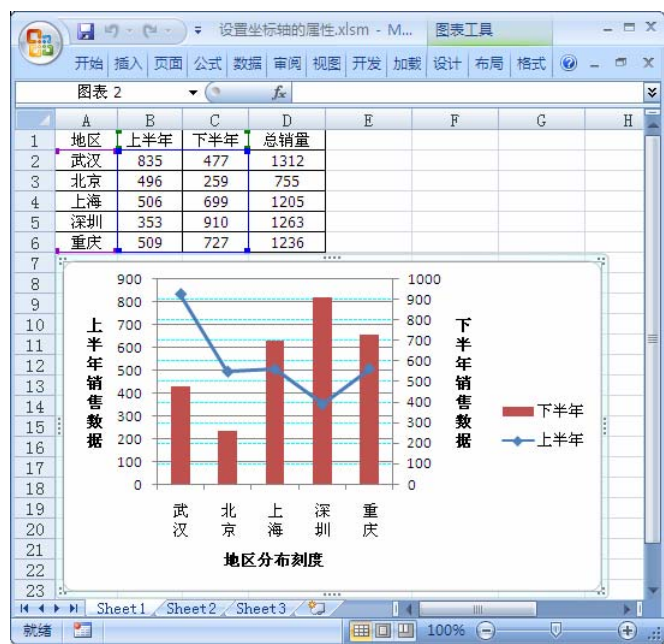


图 7.38 修改数据系列的图表类型

6. 程序分析

在上面的代码中，共对三个坐标轴进行了设置。指定了垂直轴的标题文字和方向以及其刻度的数字格式。设置了水平轴的标题文字以及其月份刻度文字的方向，另外还将图表中第二个数据序列关联到辅助轴。

7.3 图表事件

和其他 Excel 对象类似，图表也有对应的事件。使用这些图表事件，用户以编写针对图表操作的事件。在本小节中，将结合具体的例子来分析如何编写图表事件。

案例 151 激活图表工作表

1. 功能说明

图表工作表是一种特殊的工作表，但是对于图表工作表，Excel 同样提供了激活事件，用户可以在激活事件中编写对应的代码，实现相应的操作。

2. 语法说明

在 Excel VBA 中，激活图表触发的事件和前面章节中讲解的激活工作表或者工作簿的

事件很类似。都是当用户激活该对象时，反应的程序代码。

3. 案例说明

某公司统计了该公司武汉、北京、上海、深圳和重庆地区的上下半年的销量数据，根据该数据，用户创建了对应系列的图表工作表。现在需要编写 VBA 代码，对激活图表工作表编写代码，原始图表如图 7.39 所示。

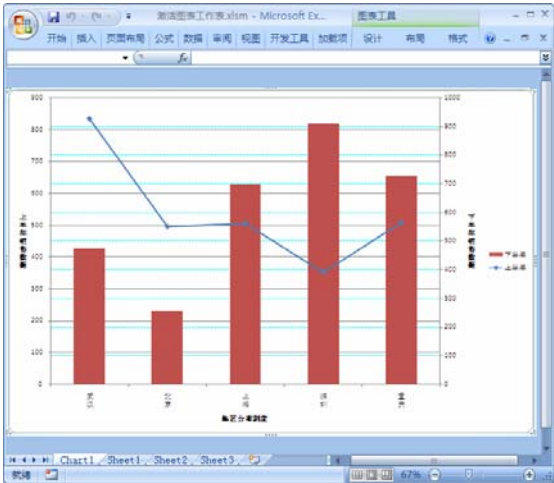


图 7.39 原始图表

4. 编写代码

本例在图表工作表的 Activate 事件过程中编写以下代码：

```
Private Sub Chart_Activate()  
MsgBox "这是图表工作表！"  
End Sub
```

5. 运行结果

激活图表工作表，查看程序运行的结果，如图 7.40 所示。

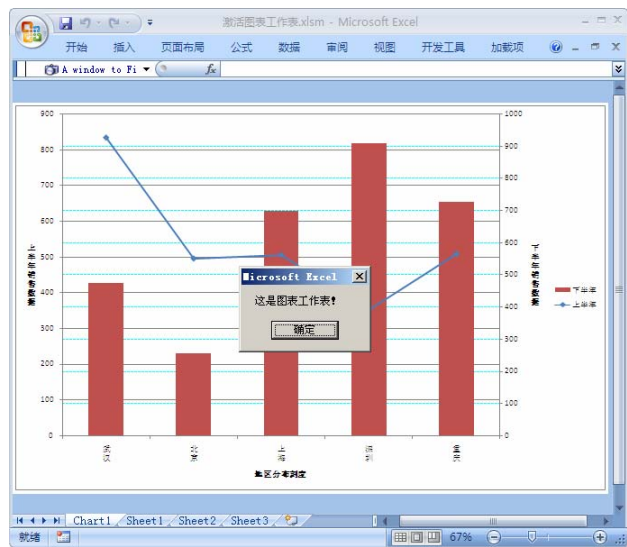


图 7.40 激活图表工作表的结果

6. 程序分析

本例十分简短，和 Excel 其他对象的激活事件一样，图表的激活事件可以编写所有在激活操作时触发的操作。

案例 152 显示子对象名称

1. 功能说明

对于图表的操作，Excel 中提供了对象选取事件。当用户单击图表中的子对象时，会触发相应的事件。用户可以在该事件中编写对应的代码。

2. 语法说明

在 Excel VBA 中，当用户在图表元素上单击选择子对象时，将产生 **Select** 事件，该事件过程的结构如下：

```
Private Sub Chart_Select(ByVal ElementID As Long, ByVal Arg1 As Long, ByVal Arg2 As Long)

End Sub
```

事件过程共有三个参数，用来在事件发生时将相关对象传给事件过程进行处理。各参数的含义如下：

- **ElementID**：选定的图表元素，此参数的值确定 **Arg1** 和 **Arg2** 的期望值。
- **Arg1** 和 **Arg2**：这两个参数的含义取决于 **ElementID** 值，具体值可查询帮助。在本例中不查询这两个参数的值。

本例查询事件发生时参数 **ElementID** 的值，即可获知用户单击的是哪个图表子对象。

3. 案例说明

某公司统计了该公司武汉、北京、上海、深圳和重庆地区的上下半年的销量数据，根据该数据，用户创建了对应系列的图表工作表。现在需要编写 VBA 代码，显示用户选择的图表对象，原始图表如图 7.41 所示。

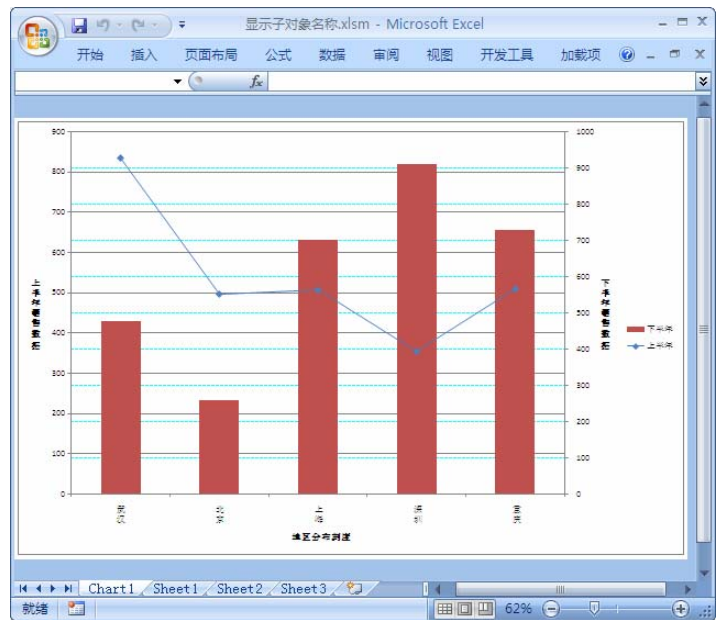


图 7.41 原始图表

4. 编写代码

本例在图表的 Select 事件过程中编写代码，具体代码如下：

```
Private Sub Chart_Select(ByVal ElementID As Long, ByVal Arg1 As Long, ByVal Arg2 As Long)
    Dim str As String
    Select Case ElementID
        Case xlChartArea: str = "图表区"
        Case xlChartTitle: str = "图表标题"
        Case xlLegendEntry: str = "图例项"
        Case xlLegendKey: str = "图例标示"
        Case xlPlotArea: str = "绘图区"
        Case xlLegend: str = "图例"
        Case xlAxisTitle: str = "坐标轴标题"
        Case xlSeries: str = "系列"
        Case xlAxis: str = "坐标轴"
    End Select

    MsgBox "用户选择的对象是： " & str
End Sub
```


End Sub

5. 运行结果

激活图表工作表，选择数据系列，查看程序运行的结果，如图 7.42 所示。

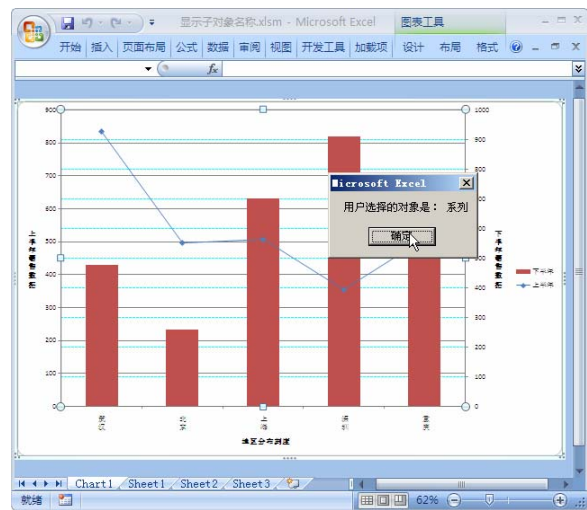


图 7.42 选择数据系列

当用户选择坐标轴对象的时候，查看程序运行的结果，如图 7.43 所示。

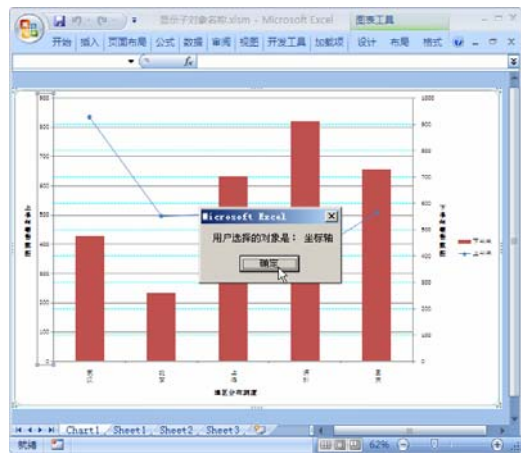


图 7.43 选择坐标轴

6. 程序分析

本例上面的代码比较简单，用户可以列出所有图表的子对象名称。程序可以获取这些对象的属性名称。