

## 第 6 章 数据处理

通过 Excel 相关对象可对工作表中的数据进行操作，如处理单元格区域的公式、对数据进行查询、排序、筛选等操作。本章演示使用 VBA 进行处理数据的实例。

### 6.1 对数据进行排序

在使用 Excel 分析数据的时候，用户经常需要对数据进行排序，得到各种结果。在 Excel VBA 中，用户同样可以对数据进行各种排序。其中包括：按照某数据序列进行排序，自定义排序的条件等。下面将根据条件进行分析。

#### 案例 112 对数据进行排序

##### 1. 功能说明

在 Excel 的基础功能中，对数据进行简单排序是十分强大的功能。用户可以对一组数据按照多个关键字进行排序，可以选择升序，也可以选择降序。这些功能也可以通过 Excel VBA 代码来实现。

##### 2. 语法说明

在 Excel 2007 中，用户对数据进行排序时，在单元格中单击作为关键字的列，选择“开始”选项卡“编辑”组中的“排序和筛选”按钮中的相关命令，可对工作表中的数据进行排序。但是，这种方法有一个明显的缺陷，就是用户排序的数据对象是全部数据表的数据，而无法选择特定区域的数据。

在 VBA 代码中，用户可方便地控制排序的区域，Range 对象的 Sort 方法可对值区域进行排序。其语法格式如下：

```
表达式.Sort(Key1, Order1, Key2, Type, Order2, Key3, Order3, Header, OrderCustom, MatchCase, Orientation, SortMethod, DataOption1, DataOption2, DataOption3)
```

该方面的各参数的含义如下：

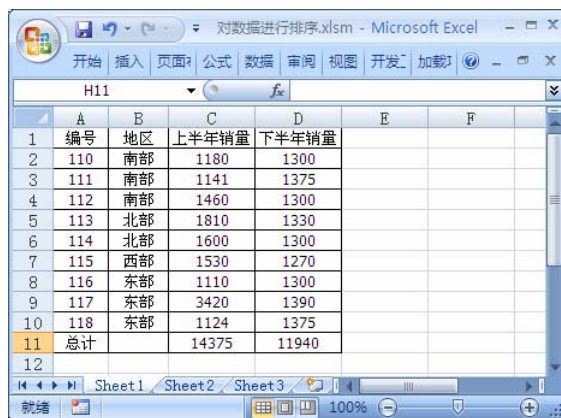
- Key1：指定第一排序字段，作为区域名称（字符串）或 Range 对象；确定要排序的值。
- Order1：确定 Key1 中指定的值的排序次序，可设置为常量 xlAscending（升序）或 xlDescending（降序）。
- Key2：第二排序字段。
- Type：指定要排序的元素。
- Order2：确定 Key2 中指定的值的排序次序。

- **Key3**: 第三排序字段
- **Order3**: 确定 **Key3** 中指定的值的排序次序。
- **Header**: 指定第一行是否包含标题信息。
- **OrderCustom**: 指定在自定义排序次序列表中的基于 1 的整数偏移。
- **MatchCase**: 设置为 **True**, 则执行区分大小写的排序, 设置为 **False**, 则执行不区分大小写的排序; 不能用于数据透视表。
- **Orientation**: 指定以升序还是降序排序。可用常量 **xlSortColumns** (按列排序) 或 **xlSortRows** (按行排序, 这是默认值)。
- **SortMethod**: 指定排序方法。可用常量 **xlPinYin** (按汉语拼音顺序排序, 这是默认值) 或 **xlStroke** (按每个字符的笔划数排序)。
- **DataOption1**: 指定 **Key1** 中所指定区域中的文本的排序方式, 可使用常量 **xlSortNormal** (分别对数字和文本数据进行排序, 这是默认值) 或 **xlSortTextAsNumbers** (将文本作为数字型数据进行排序)。
- **DataOption2**: 指定 **Key2** 中所指定区域中的文本的排序方式。
- **DataOption3**: 指定 **Key3** 中所指定区域中的文本的排序方式。

提示: 使用 **Sort** 方法排序时, 最多只能按三个关键字进行排序。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量, 同时提供了员工所处的地区。本例中, 用户可以对数据按照各数据字段进行排序。基础数据如图 6.1 所示。



	A	B	C	D	E	F
	编号	地区	上半年销量	下半年销量		
1	110	南部	1180	1300		
2	111	南部	1141	1375		
3	112	南部	1460	1300		
4	113	北部	1810	1330		
5	114	北部	1600	1300		
6	115	西部	1530	1270		
7	116	东部	1110	1300		
8	117	东部	3420	1390		
9	118	东部	1124	1375		
10	总计		14375	11940		
11						
12						

图 6.1 销量数据表

### 4. 编写代码

实现按照“上半年销量”数据列 (C 列) 字段进行排序的代码如下:

```
Sub SortNumber()
    Dim rng As Range
    Dim LongRow As Long
```

```

Dim LongCol As Long
LongRow = ActiveSheet.Range("A1").CurrentRegion.Rows.Count
LongCol = ActiveSheet.Range("A2").CurrentRegion.Columns.Count
Set rng = ActiveSheet.Range(Cells(2, 1), Cells(LongRow - 1, LongCol))
rng.Sort key1:=ActiveSheet.Range(Cells(2, 3), Cells(LongRow - 1, 3))
End Sub

```

以上代码首先获取当前工作表中需要排序的单元格区域，对该区域使用 Sort 方法按“上半年销量”列进行排序。

## 5. 程序结果

当用户运行上面的程序代码后，结果如图 6.2 所示。



	A	B	C	D	E
	编号	地区	上半年销量	下半年销量	
1	116	东部	1110	1300	
2	118	东部	1124	1375	
3	111	南部	1141	1375	
4	110	南部	1180	1300	
5	112	南部	1460	1300	
6	115	西部	1530	1270	
7	114	北部	1600	1300	
8	113	北部	1810	1330	
9	117	东部	3420	1390	
10	总计		14375	11940	
11					

图 6.2 排序的结果

## 6. 程序分析

根据上面的程序代码分析，用户可以对数据系列进行其他的排序。只需要修改下面这行代码的具体内容就可以

```

rng.Sort key1:=ActiveSheet.Range(Cells(2, 3), Cells(LongRow - 1, 3))

```

# 案例 113 两个关键字排序

## 1. 功能说明

在 Excel 的排序中，用户除了可以选择数据表中任何一个字段进行排序之外，还可以对数据表按照多个关键字进行排序。其主要方法是使用 Add 方法，添加新的排序信息。在 Excel 操作中，就相当于图 6.3 所示。

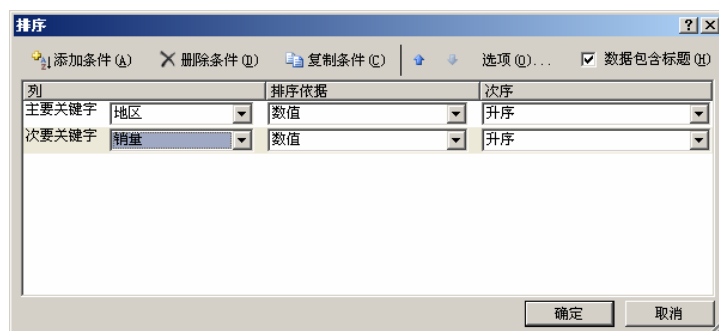


图 6.3 添加排序条件

## 2. 语法说明

在本例中，所使用的语法介绍和前面案例中的 Sort 方法类似。只是需要使用 Add 方法来添加一个新的排序条件。

## 3. 案例说明

某公司统计了部分员工的销量和顾客人数，同时提供了员工所处的地区。本例中，用户可以对数据首先按照地区排序，然后按照销量排序。基础数据如图 6.4 所示。

	A	B	C	D	E
1	编号	地区	销量	顾客人数	
2	110	南部	1180	1300	
3	111	南部	1141	1375	
4	112	南部	1460	1300	
5	113	北部	1810	1330	
6	114	北部	1600	1300	
7	115	西部	1530	1270	
8	116	东部	1110	1300	
9	117	东部	3420	1390	
10	118	东部	1124	1375	
11	总计		14375	11940	

图 6.4 销量数据表

## 4. 编写代码

实现地区和销量排序的代码如下：

```
S Sub SortMultiNumbers()
    Range("A1:D10").Select
    ActiveWorkbook.Worksheets("Sheet1").Sort.SortFields.Clear
    ActiveWorkbook.Worksheets("Sheet1").Sort.SortFields.Add Key:=Range("B2:B10") _
        , SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
    ActiveWorkbook.Worksheets("Sheet1").Sort.SortFields.Add Key:=Range("C2:C10") _
        , SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
```

```

With ActiveWorkbook.Worksheets("Sheet1").Sort
.SetRange Range("A1:D10")
.Header = xlYes
.MatchCase = False
.Orientation = xlTopToBottom
.SortMethod = xlPinYin
.Apply
End With
End Sub

```

## 5. 程序结果

当用户运行上面的程序代码后，结果如图 6.5 所示。



	A	B	C	D	E	F
1	编号	地区	销量	顾客人数		
2	114	北部	1600	1300		
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141	1375		
8	110	南部	1180	1300		
9	112	南部	1460	1300		
10	115	西部	1530	1270		
11	总计		14375	11940		
12						

图 6.5 排序的结果

## 6. 程序分析

在上面的程序代码中，首先设置排序的条件是“地区”，然后添加排序条件“顾客人数”。最后设置排序的结果。

## 案例 114 多关键字排序

### 1. 功能说明

在用户对数据进行排序的时候，用户可能需要按照多列关键字进行排序。而不仅仅是两个关键字。对于多关键字（多条件）排序的情况，用户可以沿用前面案例的方法。但是，对于多于 3 个关键字的情况，则可以使用前面章节所讲解的程序结构来完成。

### 2. 语法说明

对于超过三个关键字的排序，本案例将使用循环结构，依次对数据按照关键字进行排序。但是，用户需要注意的是，需要先将数据按最后一个关键字排序，接着再将数据按倒

数第二个关键字排序，最后将数据按主要（第一个）关键字排序，即可得到所需要的排列。同时，使用这种方法，用户可使用任意数量的关键字进行排序。

### 3. 案例说明

某公司统计了部分员工的销量和增加的顾客人数，同时提供了员工所处的地区。本例中，用户可以对数据依次按照地区、销量和增加顾客进行排序。基础数据如图 6.6 所示。

	A	B	C	D	E	F
1	编号	地区	销量	增加的顾客		
2	110	南部	1180	130		
3	111	南部	1141	137		
4	112	南部	1460	125		
5	113	北部	1810	133		
6	114	北部	1600	130		
7	115	西部	1530	127		
8	116	东部	1110	130		
9	117	东部	3420	139		
10	118	东部	1124	137		
11	总计		14375	1188		

图 6.6 多关键字排序

### 4. 编写代码

多关键字排序的 VBA 代码如下：

```
Sub MultSortNum()
    Dim rng As Range
    Dim i As Integer

    Application.ScreenUpdating = False

    Dim LongRow As Long
    Dim LongCol As Long
    LongRow = ActiveSheet.Range("A1").CurrentRegion.Rows.Count
    LongCol = ActiveSheet.Range("A2").CurrentRegion.Columns.Count
    Set rng = ActiveSheet.Range(Cells(2, 1), Cells(LongRow - 1, LongCol))

    With rng
        For i = 5 To 3 Step -1
            .Sort key1:=ActiveSheet.Range("B2").Offset(, i - 3)
            MsgBox Range("B2").Offset(, i - 3)
        Next
    End With
    Application.ScreenUpdating = True
End Sub
```

## 5. 程序结果

当用户运行上面的程序代码后，结果如图 6.7 所示。



	A	B	C	D	E	F
1	编号	地区	销量	增加的顾客		
2	114	北部	1600	130		
3	113	北部	1810	133		
4	116	东部	1110	130		
5	118	东部	1124	137		
6	117	东部	3420	139		
7	111	南部	1141	137		
8	110	南部	1180	130		
9	112	南部	1460	125		
10	115	西部	1530	127		
11	总计		14375	1188		

图 6.7 排序的结果

## 6. 程序分析

从图中的结果可以看出，首先按 B 列（地区）排序，部门相同时再按 C 列（销量）排序，基本工资相同再按 D 列（增加的销量）排序。

## 案例 115 自定义排序

### 1. 功能说明

在 Excel 进行分析的时候，用户经常需要设定自己的排序标准。而在 Excel 中，所提供的排序标准是常见的降序、升序或者字母排序等。当用户需要按照自己特色标准排序时，则需要自行编写相应的代码。

### 2. 语法分析

本例演示用 VBA 代码创建自定义序列的方法，主要用 `AddCustomList` 方法添加自定义序列。`AddCustomList` 方法为自定义自动填充和/或自定义排序添加自定义列表。其语法格式如下：

**表达式.AddCustomList(ListArray, ByRow)**

参数的含义如下：

- **ListArray**：将源数据指定为字符串数组或 `Range` 对象。
- **ByRow**：仅当 `ListArray` 为 `Range` 对象时使用。如果为 `True`，则使用区域中的每一行创建自定义列表；如果为 `False`，则使用区域中的每一列创建自定义列表。如果省略该参数，并且区域中的行数比列数多（或者行数与列数相等），则 Excel 使用区域中的每一列创建自定义列表。如果省略该参数，并且区域中的列数比行数多，则 Excel 使用

区域中的每一行创建自定义列表。

注意：如果要添加的列表已经存在，则本方法不起作用。

使用 Application 对象的 GetCustomListNum 方法返回字符串数组的自定义序列号。其语法格式如下：

表达式.GetCustomListNum(ListArray)

参数 ListArray 为一个字符串数组。

### 3. 案例说明

某公司统计了部分员工的销量和增加的顾客人数，同时提供了员工所处的地区。本例中，用户需要对地区按照“东部、南部、西部和北部”的次序进行排序。基础数据如图 6.8 所示。

	A	B	C	D	E	F
1	编号	地区	销量	增加的顾客		
2	110	南部	1180	130		
3	111	南部	1141	137		
4	112	南部	1460	125		
5	113	北部	1810	133		
6	114	北部	1600	130		
7	115	西部	1530	127		
8	116	东部	1110	130		
9	117	东部	3420	139		
10	118	东部	1124	137		
11	总计		14375	1188		

图 6.8 原始数据表

### 4. 编写代码

在本例中，用户首先需要定义自定义的排序序列，如图 6.9 所示。

	A	B	C	D
1	东部			
2	南部			
3	西部			
4	北部			

图 6.9 自定义序列

自定义序列排序的 VBA 代码如下：

```
Sub SelfSortNumbers()  
    Dim rng As Range
```



```

Dim n As Integer
Dim rng1 As Range
Dim arr

Application.ScreenUpdating = False

Dim LongRow As Long
Dim LongCol As Long
LongRow = ActiveSheet.Range("A1").CurrentRegion.Rows.Count
LongCol = ActiveSheet.Range("A2").CurrentRegion.Columns.Count
Set rng1 = ActiveSheet.Range(Cells(1, 1), Cells(LongRow - 1, LongCol))

With Worksheets("Sheet2")
    r = .Range("A1").End(xlDown).Row
    Set rng = .Range(.Cells(1, 1), .Cells(r, 1))
End With

With Application
    arr = .WorksheetFunction.Transpose(rng)
    .AddCustomList ListArray:=arr
    n = .GetCustomListNum(arr1)
End With

rng1.Sort key1:=ActiveSheet.Range(Cells(2, 2), Cells(LongRow - 1, 2)), _
    Order1:=xlAscending, Header:=xlYes, OrderCustom:=n + 1
Application.ScreenUpdating = True
End Sub

```

## 5. 程序结果

当用户运行上面的程序代码后，结果如图 6.10 所示。



	A	B	C	D	E	F
1	编号	地区	销量	增加的顾客		
2	116	东部	1110	130		
3	117	东部	3420	139		
4	118	东部	1124	137		
5	110	南部	1180	130		
6	111	南部	1141	137		
7	112	南部	1460	125		
8	115	西部	1530	127		
9	113	北部	1810	133		
10	114	北部	1600	130		
11	总计		14375	1188		

图 6.10 排序结果

## 6. 程序分析

以上代码首先获取需要排序的单元格区域，接着将工作表 Sheet2 中的数据添加到自定义序列中，再使用自定义序列进行排序。这是自定义序列的通用方法。同时，当用户运行该程序代码后，可以查看对应的排序条件如图 6.11 所示。

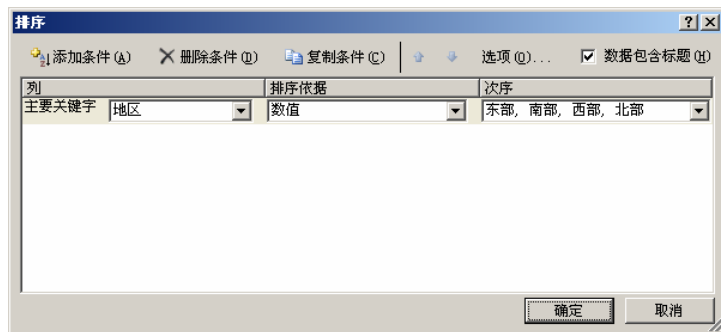


图 6.11 排序条件

如果用户希望删除相应的排序条件，可以使用 DeleteCustomList 方法。使用 Application 对象的 DeleteCustomList 方法删除一个自定义序列。其语法格式如下：

**表达式.DeleteCustomList(ListNum)**

参数 ListNum 为自定义序列数字。此数字必须大于或等于 5（Excel 有四个不可删除的内置自定义序列）。

## 案例 116 随机排序

### 1. 功能说明

在实际分析过程中，用户除了需要对数据进行标准的排序之外，有时还需要一种特殊的排序：随机排序。也就是说，某些数据不需要任何特定的排序，完全随机的序列。这个功能通过 Excel 的基础操作将无法实现，需要通过程序代码实现。

### 2. 语法说明

随机排序是一种特殊的排序，使用 VBA 代码实现随机排序的基本思路是：首先，自行生成随机序列，然后将这个随机序列当作关键字，对其进行排序，最后的结果就是随时排序的数据结果。

### 3. 案例说明

某公司统计了部分员工的销量和工时，同时提供了员工所处的地区。本例中，用户需要对原始数据进行随机排列。基础数据如图 6.12 所示。

	A	B	C	D	E	F
1	编号	地区	销量	工时		
2	110	南部	1180	130		
3	111	南部	1141	137		
4	112	南部	1460	125		
5	113	北部	1810	133		
6	114	北部	1600	130		
7	115	西部	1530	127		
8	116	东部	1110	130		
9	117	东部	3420	139		
10	118	东部	1124	137		
11	总计		14375	1188		
12						

图 6.12 原始数据

### 3. 编写代码

随机排序的 VBA 代码如下：

```
Sub RandSort()  
    Dim rng As Range  
    Dim intCol As Long  
    Dim intRow As Long  
  
    Randomize  
  
    Application.ScreenUpdating = False  
  
    With ActiveSheet  
        intRow = .Range("A1").CurrentRegion.Rows.Count  
        intCol = .Range("A2").CurrentRegion.Columns.Count  
  
        For i = 2 To intRow - 1  
            .Cells(i, intCol + 1) = Int((Rnd * 50) + 2)  
        Next  
  
        Set rng = .Range(Cells(2, 1), Cells(intRow - 1, intCol + 1))  
        rng.Sort key1:=.Range(Cells(2, intRow + 1), Cells(intRow - 1, intCol + 1))  
  
        .Columns(intCol + 1).Clear  
  
    End With  
  
    Application.ScreenUpdating = True  
  
End Sub
```

## 5. 程序结果

当用户运行上面的程序代码后，结果如图 6.13 所示。



	A	B	C	D	E
1	编号	地区	销量	工时	
2	115	西部	1530	127	
3	111	南部	1141	137	
4	116	东部	1110	130	
5	118	东部	1124	137	
6	112	南部	1460	125	
7	117	东部	3420	139	
8	114	北部	1600	130	
9	110	南部	1180	130	
10	113	北部	1810	133	
11	总计		14375	1188	

图 6.13 随机排序的结果

由于是随机排序的，当用户再次运行程序代码时，结果会变化，如图 6.14 所示。



	A	B	C	D	E
1	编号	地区	销量	工时	
2	117	东部	3420	139	
3	115	西部	1530	127	
4	111	南部	1141	137	
5	114	北部	1600	130	
6	118	东部	1124	137	
7	110	南部	1180	130	
8	116	东部	1110	130	
9	112	南部	1460	125	
10	113	北部	1810	133	
11	总计		14375	1188	

图 6.14 再次运行程序的结果

## 6. 程序分析

以上代码中，先在数据的右列添加随机数据，使用的是内置函数 `Rnd`。然后使用 `Sort` 方法按该列的数据进行排序，最后删除增加的随机数据列。

### 案例 117 自动排序

#### 1. 功能说明

本案例的主要功能是实现排序的自动化。自动化的意思是，用户不需要选择排序的操

作就可以实现按照某字段自动排列。例如，某公司的员工花名册中，需要自动按照名字的字母次序进行排列，而不需要每次都进行排序的操作。

## 2. 语法说明

本例需要根据用户对单元格数据的更改及时完成排序，所以需要在工作表的 **Change** 事件过程中编写代码。该事件在用户对工作表进行操作编辑修改的时候触发。另外，本例还使用了 **Application** 对象的 **Intersect** 方法，该方法返回一个 **Range** 对象，该对象表示两个或多个区域重叠的矩形区域。其语法格式如下：

表达式.Intersect(Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7, Arg8, Arg9, Arg10, Arg11, Arg12, Arg13, Arg14, Arg15, Arg16, Arg17, Arg18, Arg19, Arg20, Arg21, Arg22, Arg23, Arg24, Arg25, Arg26, Arg27, Arg28, Arg29, Arg30)

该方法最多可使用 30 个单元格区域作为参数，至少需使用两个参数。

## 3. 案例说明

某公司统计了部分员工的销量和工时，同时提供了员工所处的地区。本例中，当用户输入新员工的地区属性时，需要按照“地区”列进行自动排序。基础数据如图 6.15 所示。

	A	B	C	D	E	F
1	编号	地区	销量	工时		
2	113	北部	1810	133		
3	114	北部	1600	130		
4	116	东部	1110	130		
5	117	东部	3420	139		
6	118	东部	1124	137		
7	110	南部	1180	130		
8	111	南部	1141	137		
9	112	南部	1460	125		
10	115	西部	1530	127		
11						

图 6.15 原始数据

## 4. 编写代码

要完成本例的功能，需要在工作表的 **Change** 事件过程中编写以下代码：

```
Private Sub Worksheet_Change(ByVal Target As Range)

    Dim rng As Range
    Dim intRow As Integer
    Dim intCol As Integer

    If Target.Column <> 2 Then Exit Sub
    If Not Application.Intersect(Target, [B2:B500]) Is Nothing Then
        Set rng = ActiveSheet.Range("A1").CurrentRegion
    End If
End Sub
```

```

intRow = rng.Rows.Count
intCol = rng.Columns.Count

Set rng = rng.Offset(2, 0).Resize(intRow - 2, intCol)

rng.Sort Key1:=Range("B2")

End If
End Sub

```

### 5. 程序结果

本例是事件触发形式的。因此，用户需要首先对工作表进行编辑，例如，在“地区”数据列输入新的数据“东部”，如图 6.16 所示。

	A	B	C	D	E	F
1	编号	地区	销量	工时		
2	113	北部	1810	133		
3	114	北部	1600	130		
4	116	东部	1110	130		
5	117	东部	3420	139		
6	118	东部	1124	137		
7	110	南部	1180	130		
8	111	南部	1141	137		
9	112	南部	1460	125		
10	115	西部	1530	127		
11		东部				

图 6.16 输入新的数据

当用户输入新数据后，工作表的事件会自动触发，得到的结果如图 6.17 所示。

	A	B	C	D	E	F
1	编号	地区	销量	工时		
2	113	北部	1810	133		
3	114	北部	1600	130		
4	116	东部	1110	130		
5	117	东部	3420	139		
6	118	东部	1124	137		
7		东部				
8	110	南部	1180	130		
9	111	南部	1141	137		
10	112	南部	1460	125		
11	115	西部	1530	127		
12						

图 6.17 自动排序的结果

## 6. 程序分析

在上面的代码中，首先判断更改数据的单元格是否为第 2 列（表示“地区”数据），然后判断更改数据单元格是否为“B2:B500”单元格区域中的单元格，然后获取当前区域需要排序的单元格区域，使用 Sort 方法对这个区域进行排序。

## 6.2 对数据进行筛选

在数据分析中，筛选是一个十分常见的功能。当用户需要分析大量数据的时候，对数据进行筛选是很重要的操作。从某种程度上来讲，筛选和分类很类似。在 Excel 2007 中，在“开始”选项卡的“编辑”组中，单击“排序和筛选”按钮，从下拉的菜单按钮中选择相应的命令即可进行数据筛选操作，如图 6.18 所示。



图 6.18 进行数据筛选

用户在 Excel 中进行的筛选操作，都可以通过 VBA 来实现。在本小节中，将详细讲解如何通过 VBA 实现各种筛选功能。

## 案例 118 简单筛选

### 1. 功能说明

本案例的主要功能是使用 VBA 对原始数据进行简单筛选。简单筛选的含义就是，将原始数据按照某个数据字段进行筛选。筛选后的数据结果是仅显示某类数据

2. 语法说明

在 Excel VBA 中，使用 Range 对象的 AutoFilter 方法，可对 Range 区域的数据中使用“自动筛选”筛选一个列表。该方法的语法如下：

表达式.AutoFilter(Field, Criteria1, Operator, Criteria2, VisibleDropDown)

各参数的含义如下：

- **Field**：相对于作为筛选基准字段（从列表左侧开始，最左侧的字段为第一个字段）的字段的整体偏移量。
- **Criteria1**：筛选条件，为一个字符串。使用“=”可查找空字段，或者使用“<>”查找非空字段。如果省略该参数，则搜索条件为 All。如果将 Operator 设置为 xlTop10Items，则 Criteria1 指定数据项个数（例如，“10”）。
- **Operator**：指定筛选类型，可用常量如表 6-1 所示。
- **Criteria2**：第二个筛选条件（一个字符串）。与 Criteria1 和 Operator 一起组合成复合筛选条件。
- **VisibleDropDown**：如果为 True，则显示筛选字段的自动筛选下拉箭头。如果为 False，则隐藏筛选字段的自动筛选下拉箭头。默认值为 True。

提示：如果忽略全部参数，此方法仅在指定区域切换自动筛选下拉箭头的显示。

表 6-1 筛选类型

名称	值	描述
xlAnd	1	条件1和条件2的逻辑与。
xlBottom10Items	4	显示最低值项（条件1中指定的项数）。
xlBottom10Percent	6	显示最低值项（条件1中指定的百分数）。
xlFilterCellColor	8	单元格颜色
xlFilterDynamic	11	动态筛选
xlFilterFontColor	9	字体颜色
xlFilterIcon	10	筛选图标
xlFilterValues	7	筛选值
xlOr	2	条件1和条件2的逻辑或。
xlTop10Items	3	显示最高值项（条件1中指定的项数）。
xlTop10Percent	5	显示最高值项（条件1中指定的百分数）。

3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区。本例中，用户需要对数据按照“地区”数据字段进行筛选，基础数据如图 6.19 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量		
2	114	北部	1600	1300		
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141	1375		
8	110	南部	1180	1300		
9	112	南部	1460	1300		
10	115	西部	1530	1270		
11	总计		14375	11940		

图 6.19 原始数据

#### 4. 编写代码

简单筛选的 VBA 代码如下：

```
Sub FilterNum()
    Dim str As String

    str = Application.InputBox(prompt:="输入要筛选的地区（不输入信息，将显示全部数据）：", _
        Title:="地区筛选", Type:=2)

    If str = "False" Then Exit Sub

    If str = "" Then
        Worksheets("Sheet1").Range("A1").AutoFilter field:=2
    Else
        Worksheets("Sheet1").Range("A1").AutoFilter _
            field:=2, _
            Criteria1:=str
    End If
End Sub
```

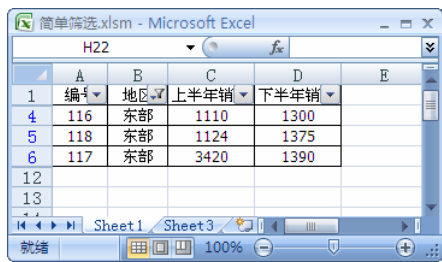
#### 5. 程序结果

运行程序代码，程序将会显示筛选字段的对话框，如图 6.20 所示。



图 6.20 筛选字段

单击对话框中的“确定”按钮，查看筛选后的结果，如图 6.21 所示。



The screenshot shows an Excel window titled '简单筛选.xlsm - Microsoft Excel'. The active sheet is 'Sheet3'. The data table has columns: 编号 (ID), 地区 (Region), 上半年销 (Upper Half Sales), and 下半年销 (Lower Half Sales). The data is filtered to show only records from the '东部' (East) region.

	A	B	C	D	E
	编号	地区	上半年销	下半年销	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	

图 6.21 筛选后的数据结果

当用户需要重新显示全部数据的时候，可以重新运行程序代码，在对话框中不输入任何数据，如图 6.22 所示。



The dialog box is titled '地区筛选'. It contains a text input field with the placeholder text '输入要筛选的地区（不输入信息，将显示全部数据）：'. Below the input field are two buttons: '确定' (OK) and '取消' (Cancel).

图 6.22 显示所有的数据结果

单击对话框中的“确定”按钮，查看最新筛选的结果，如图 6.23 所示。



The screenshot shows the same Excel window as Figure 6.21, but now all data is displayed. The data table includes a '总计' (Total) row at the bottom.

	A	B	C	D	E
	编号	地区	上半年销	下半年销	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	1375	
8	110	南部	1180	1300	
9	112	南部	1460	1300	
10	115	西部	1530	1270	
11	总计		14375	11940	

图 6.23 显示所有数据

## 6. 程序分析

在上面的代码中，首先要求用户输入筛选条件，接着判断用户输入的是否为空，若为空，则显示全部数据。若输入的筛选条件不为空，则筛选等于输入条件的数据。当通过 VBA 代码实现简单筛选时，用户可以检测其筛选条件，如图 6.24 所示。

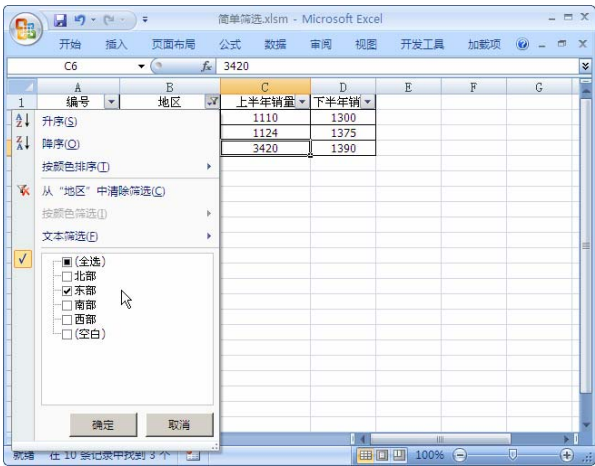


图 6.24 查看筛选条件

## 案例 119 多条件筛选

### 1. 功能说明

相对于小节的案例，本小节案例的主要功能是对同一个字段设置多个条件。在实际应用中，用户对某个字段进行筛选的时候，经常需要设定是多个条件。例如，筛选某个字段的数据范围，或者筛选条件的逻辑组合等。

### 2. 语法说明

在前面小节所介绍的 `AutoFilter` 方法中，可以通过设置参数 `Operator` 来指定筛选类型，该方法提供多个筛选类型。例如，通过 `Operator` 参数可以设定多个筛选条件的逻辑关系：并、或等运算。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区。本例中，用户需要对数据按照“地区”数据字段进行筛选。需要筛选的条件是：南部和东部的数据，其中基础数据如图 6.25 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	1375	
8	110	南部	1180	1300	
9	112	南部	1460	1300	
10	115	西部	1530	1270	
11	总计		14375	11940	
12					

图 6.25 原始数据

#### 4. 编写代码

多条件筛选的 VBA 代码如下：

```
Sub MultiFilter()  
  
    Range("A1:D11").Select  
    Selection.AutoFilter  
  
    ActiveSheet.Range("A1").AutoFilter Field:=2, _  
        Criteria1:="=东部", Operator:=xlOr, Criteria2:="=南部"  
  
End Sub
```

#### 5. 程序结果

运行程序代码，查看筛选后的结果，如图 6.26 所示。



	A	B	C	D	
1	编号	地区	上半年销量	下半年销量	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	1375	
8	110	南部	1180	1300	
9	112	南部	1460	1300	

图 6.26 筛选后的数据结果

#### 6. 程序分析

在上面的代码中，直接使用 AutoFilter 方法中的 Operator 参数属性，设定两个筛选结

果。用户可以查看对应的筛选结果，如图 6.27 所示。

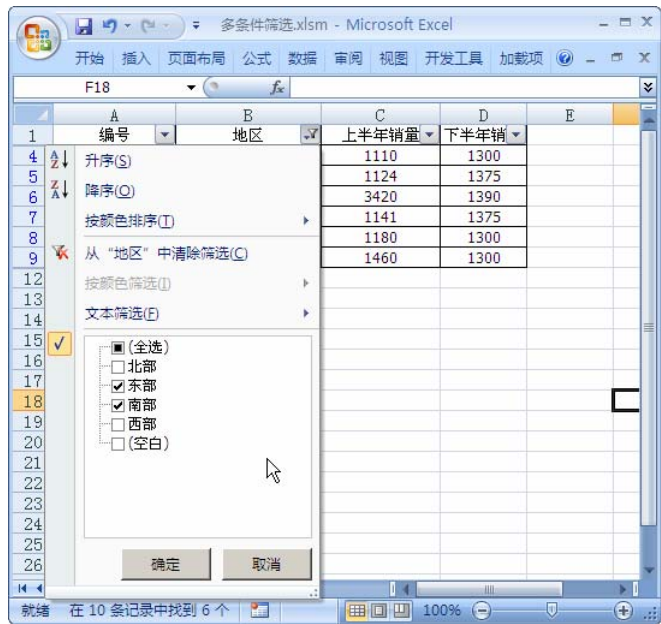


图 6.27 查看筛选条件

## 案例 120 多字段筛选

### 1. 功能说明

本案例的主要功能是对原始数据中的多个数据字段进行筛选。当用户在筛选数据时，有时需要对多个数据字段设置条件。这个时候，用户就需要设置不同的数据字段，同时设置筛选条件，同时设置筛选条件的逻辑关系。

### 2. 语法说明

本小节同样需要用到 `AutoFilter` 方法中的参数 `Operator`。和前面小节例子不同的是，本案例需要首先设定不同的筛选字段，然后设置对应的筛选条件。最后，则需要设置不同筛选条件的逻辑关系。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区。本例中，用户需要获取东部销量大于 1120 的数据信息。其中原始数据如图 6.28 所示。

	A	B	C	D
1	编号	地区	上半年销量	下半年销量
2	114	北部	1600	1300
3	113	北部	1810	1330
4	116	东部	1110	1300
5	118	东部	1124	1375
6	117	东部	3420	1390
7	111	南部	1141	1375
8	110	南部	1180	1300
9	112	南部	1460	1300
10	115	西部	1530	1270
11	总计		14375	11940

图 6.28 原始数据

#### 4. 编写代码

多字段筛选的 VBA 代码如下：

```
Sub MultiDataFilter()

    Range("A1:D11").Select
    Selection.AutoFilter
    ActiveSheet.Range("A1").AutoFilter Field:=2, Criteria1:="东部"
    ActiveSheet.Range("A1").AutoFilter Field:=3, Criteria1:=">1120", _
        Operator:=xlAnd
End Sub
```

#### 5. 程序结果

运行程序代码，查看筛选后的结果，如图 6.29 所示。

	A	B	C	D
1	编号	地区	上半年销量	下半年销量
5	118	东部	1124	1375
6	117	东部	3420	1390

图 6.29 筛选后的数据结果

#### 6. 程序分析

本案例有相当的代表性，当用户需要对其他数据字段添加筛选条件，然后设置这些筛选条件的逻辑关系就可以。

## 案例 121 使用通配符筛选

### 1. 功能说明

本案例的主要功能是对原始数据中的字段使用通配符进行筛选。使用通配符进行筛选，相当于模糊查询。用户如果能灵活通配符，可以高效的实现多种筛选。

### 2. 语法说明

根据前面小节介绍，在 Excel VBA 中，使用 Range 对象的 AutoFilter 方法，可对 Range 区域的数据中使用“自动筛选”筛选一个列表。该方法的语法如下：

**表达式.AutoFilter(Field, Criteria1, Operator, Criteria2, VisibleDropDown)**

在该表达式中，参数 Criteria1 表示筛选条件，一般情况下是字符串。使用“=”可查找空字段，或者使用“<>”查找非空字段。如果省略该参数，则搜索条件为 All。同时，如果用户希望使用通配符，则可以在参数 Criteria1 中使用 Excel 通用的通配符。在 Excel 中，可以通用使用的通配符如表 6.2 所示。

表 6.2 Excel 中的通配符

模式中的字符	在表达式中匹配
?	任何单一字符。
*	零个或多个字符。
#	任何单一数字（09）。
[charlist]	任何位于
[!charlist]	任何不在

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区和姓名。本例中，用户需要获对姓名进行模糊查询。其中原始数据如图 6.30 所示。



	A	B	C	D	E
1	编号	姓名	地区	上半年销量	下半年销量
2	114	王杰	北部	1600	1300
3	113	陈俊	北部	1810	1330
4	116	徐涛	东部	1110	1300
5	118	王军	东部	1124	1375
6	117	陈梁	东部	3420	1390
7	111	张梅	南部	1141	1375
8	110	冯涛	南部	1180	1300
9	112	张康	南部	1460	1300
10	115	王强	西部	1530	1270
11	总计			14375	11940

图 6.30 原始数据

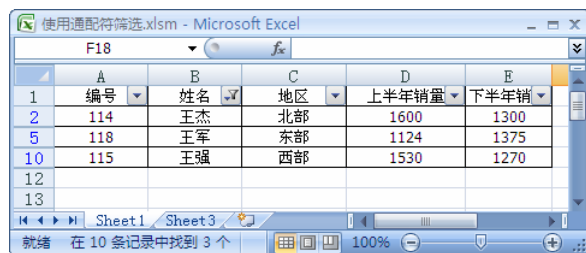
#### 4. 编写代码

通配符筛选的 VBA 代码如下：

```
Sub FluFilter()  
  
    Range("A1:E11").Select  
    Selection.AutoFilter  
    ActiveSheet.Range("A1").AutoFilter Field:=2, Criteria1:="=王*"  
  
End Sub
```

#### 5. 程序结果

运行程序代码，查看筛选后的结果，如图 6.31 所示。



	A	B	C	D	E
1	编号	姓名	地区	上半年销量	下半年销量
2	114	王杰	北部	1600	1300
5	118	王军	东部	1124	1375
10	115	王强	西部	1530	1270
12					
13					

图 6.31 筛选后的数据结果

#### 6. 程序分析

在本例中，在程序代码中使用了“王\*”，这个通配符筛选条件说明需要筛选“姓名”数据列中的包含“王”的数据。

### 案例 122 动态筛选

#### 1. 功能说明

动态筛选是一种高级筛选功能，这种筛选的条件是动态的，而不是前面案例所涉及到的静态条件。例如，用户需要筛选所有数据中，大于平均值的数据。数据的平均值会随着基础数据的增加而变化，因此是动态筛选。

#### 2. 语法说明

根据前面小节的介绍，在 Excel VBA 中，使用 Range 对象的 AutoFilter 方法，可对 Range 区域的数据中使用“自动筛选”筛选一个列表。该方法的语法如下：

**表达式.AutoFilter(Field, Criteria1, Operator, Criteria2, VisibleDropDown)**

当用户使用参数 Operator 设置为 xlFilterDynamic 时，可以进入动态筛选。同时，参数 Criteria1 可以指定动态筛选的对应条件，具体可以取值如表 6.3 所示。

表 6.3 xlFilterDynamic 对应的筛选条件



名称	值	描述
xlFilterAboveAverage	33	筛选所有高于平均值的值。
xlFilterAllDatesInPeriodApril	24	筛选所有四月的日期。
xlFilterAllDatesInPeriodAugust	28	筛选所有八月的日期。
xlFilterAllDatesInPeriodDecember	32	筛选所有十二月的日期。
xlFilterAllDatesInPeriodFebruary	22	筛选所有二月的日期。
xlFilterAllDatesInPeriodJanuary	21	筛选所有一月的日期。
xlFilterAllDatesInPeriodJuly	27	筛选所有七月的日期。
xlFilterAllDatesInPeriodJune	26	筛选所有六月的日期。
xlFilterAllDatesInPeriodMarch	23	筛选所有三月的日期。
xlFilterAllDatesInPeriodMay	25	筛选所有五月的日期。
xlFilterAllDatesInPeriodNovember	31	筛选所有十一月的日期。
xlFilterAllDatesInPeriodOctober	30	筛选所有十月的日期。
xlFilterAllDatesInPeriodQuarter1	17	筛选所有第一季度的日期。
xlFilterAllDatesInPeriodQuarter2	18	筛选所有第二季度的日期。
xlFilterAllDatesInPeriodQuarter3	19	筛选所有第三季度的日期。
xlFilterAllDatesInPeriodQuarter4	20	筛选所有第四季度的日期。
xlFilterAllDatesInPeriodSeptember	29	筛选所有九月的日期。
xlFilterBelowAverage	34	筛选所有低于平均值的值。
xlFilterLastMonth	8	筛选所有与上月相关的值。
xlFilterLastQuarter	11	筛选所有与上一季度相关的值。
xlFilterLastWeek	5	筛选所有与上周相关的值。
xlFilterLastYear	14	筛选所有与去年相关的值。
xlFilterNextMonth	9	筛选所有与下月相关的值。
xlFilterNextQuarter	12	筛选所有与下一季度相关的值。
xlFilterNextWeek	6	筛选所有与下周相关的值。
xlFilterNextYear	15	筛选所有与明年相关的值。
xlFilterThisMonth	7	筛选所有与本月相关的值。
xlFilterThisQuarter	10	筛选所有与本季度相关的值。
xlFilterThisWeek	4	筛选所有与本周相关的值。
xlFilterThisYear	13	筛选所有与今年相关的值。
xlFilterToday	1	筛选所有与今天相关的值。
xlFilterTomorrow	3	筛选所有与明天相关的值。
xlFilterYearToDate	16	筛选到今天为止一年的所有值。
xlFilterYesterday	2	筛选所有与昨天相关的值。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区和姓名。本例中，用户需要对上半年销量的数据列进行筛选，筛选出大于平均销量的数据。其中原始数据如图 6.32 所示。

	A	B	C	D	E
1	编号	姓名	地区	上半年销量	下半年销量
2	114	王杰	北部	1600	1300
3	113	陈俊	北部	1810	1330
4	116	徐涛	东部	1110	1300
5	118	王军	东部	1124	1375
6	117	陈梁	东部	3420	1390
7	111	张梅	南部	1141	1375
8	110	冯涛	南部	1180	1300
9	112	张康	南部	1460	1300
10	115	王强	西部	1530	1270

图 6.32 原始数据

#### 4. 编写代码

动态筛选的 VBA 代码如下：

```
Sub DynamicFilter()

    Range("A1:E10").Select
    Selection.AutoFilter
    ActiveSheet.Range("$A$1:$E$10").AutoFilter Field:=4, Criteria1:= _
        xIFilterAboveAverage, Operator:=xlFilterDynamic

End Sub
```

#### 5. 程序结果

运行程序代码，查看筛选后的结果，如图 6.33 所示。

	A	B	C	D	E
1	编号	姓名	地区	上半年销量	下半年销量
2	114	王杰	北部	1600	1300
3	113	陈俊	北部	1810	1330
6	117	陈梁	东部	3420	1390

图 6.33 筛选后的数据结果

#### 6. 程序分析

在本例的程序代码中，参数 Criteria1 取值是 xIFilterAboveAverage，表示的筛选条件是大于平均值，其对应的筛选条件如图 6.34 所示。

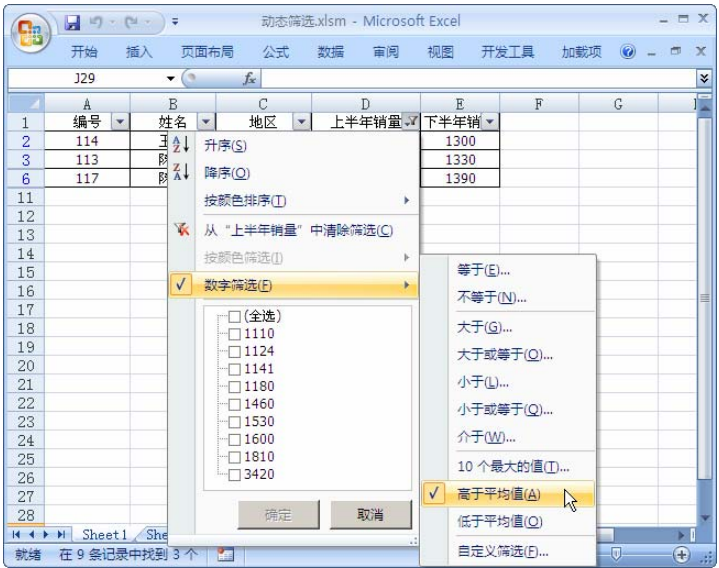


图 6.34 查看筛选条件

## 案例 123 高级筛选

### 1. 功能说明

前面小节已经详细介绍了 Excel 中的简单筛选，尽管 Excel 提供了多种简单筛选功能，但是，当用户需要进行复杂筛选的时候，则需要使用高级筛选。当用户使用高级筛选的时候，用户可以针对多个字段，设置多个不同的筛选条件。因此，Excel 同时也提供了高级筛选的功能。在本小节中，将介绍如何使用 VBA 实现高级筛选。

### 2. 语法说明

Excel 的高级筛选可用 VBA 代码来实现，使用 Range 对象的 AdvancedFilter 方法即可进行高级筛选。

高级筛选必须在工作表中定义一个条件区域，通过该条件从列表中筛选或复制数据。如果初始选定区域为单个单元格，则使用单元格的当前区域。AdvancedFilter 方法的语法格式如下：

**表达式.AdvancedFilter(Action, CriteriaRange, CopyToRange, Unique)**

该方法各参数的含义如下：

- **Action:** 指定是否就地复制或筛选列表，可使用常量 xlFilterCopy（将筛选出的数据复制到新位置）或 xlFilterInPlace（保留数据不动）。
- **CriteriaRange:** 条件区域。如果省略该参数，则没有条件限制。
- **CopyToRange:** 如果 Action 为 xlFilterCopy，则该参数为复制行的目标区域。否则，忽略该参数。

- **Unique**: 如果为 **True**, 则只筛选唯一记录。如果为 **False**, 则筛选符合条件的所有记录。默认值为 **False**。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量, 同时提供了员工所处的地区和姓名。本例中, 用户需要对源数据中的多个数据列进行筛选, 其中原始数据如图 6.35 所示。



	A	B	C	D	E
	编号	姓名	地区	上半年销量	下半年销量
1	114	王杰	北部	1600	1300
2	113	陈俊	北部	1810	1330
3	116	徐涛	东部	1110	1300
4	118	王军	东部	1124	1375
5	117	陈梁	东部	3420	1390
6	111	张梅	南部	1141	1375
7	110	冯涛	南部	1180	1300
8	112	张康	南部	1460	1300
9	115	王强	西部	1530	1270

图 6.35 原始数据

同时, 对于高级筛选而言, 用户需要首先设置对应的条件区域, 在本例中, 条件区域的单元格区域如图 6.36 所示。



	A	B	C	D	E	F
	编号	姓名	地区	上半年销量	下半年销量	
1	114	王杰	北部	1600	1300	
2	113	陈俊	北部	1810	1330	
3	116	徐涛	东部	1110	1300	
4	118	王军	东部	1124	1375	
5	117	陈梁	东部	3420	1390	
6	111	张梅	南部	1141	1375	
7	110	冯涛	南部	1180	1300	
8	112	张康	南部	1460	1300	
9	115	王强	西部	1530	1270	
10						
11						
12						
13	条件区域					
14	编号	姓名	地区	上半年销量	下半年销量	
15						

图 6.36 设置条件区域

### 4. 编写代码

高级筛选的 VBA 代码如下:

```

Sub AdvFilter()

    Dim CriterRng As Range
    Dim DataRng As Range

    Application.ScreenUpdating = False
    Application.Calculation = xlCalculationManual

    Set CriterRng = Worksheets("Sheet1").Range("A13").CurrentRegion
    Set CriterRng = CriterRng.Offset(1, 0).Resize(CriterRng.Rows.Count - 1, CriterRng.Columns.Count)

    Set DataRng = Worksheets("Sheet1").Range("A1").CurrentRegion

    DataRng.AdvancedFilter Action:=xlFilterInPlace, CriteriaRange:=CriterRng
    Application.Calculation = xlCalculationAutomatic
    Application.ScreenUpdating = True

End Sub

```

## 5. 程序结果

在运行该程序代码之前，首先设置条件区域中的条件，如图 6.37 所示。

The screenshot shows an Excel window titled '高级筛选.xlsm - Microsoft Excel'. The main data table is in Sheet1, with columns A to E. Below it, in Sheet3, is the criteria range. The criteria range has two rows: the first row contains the column headers, and the second row contains the filter conditions: '地区' (Region) set to '东部' (East) and '上半年销量' (First Half Sales) set to '>1120'.

	A	B	C	D	E
1	编号	姓名	地区	上半年销量	下半年销量
2	114	王杰	北部	1600	1300
3	113	陈俊	北部	1810	1330
4	116	徐涛	东部	1110	1300
5	118	王军	东部	1124	1375
6	117	陈梁	东部	3420	1390
7	111	张梅	南部	1141	1375
8	110	冯涛	南部	1180	1300
9	112	张康	南部	1460	1300
10	115	王强	西部	1530	1270
11					
12					
13	条件区域				
14	编号	姓名	地区	上半年销量	下半年销量
15			东部	>1120	

图 6.37 设置高级筛选的条件

设置筛选条件后，运行程序代码，查看筛选结果，如图 6.38 所示。



图 6.38 查看筛选结果

同时，用户可以修改筛选的条件区域，如图 6.39 所示。



图 6.39 设置新的筛选条件

再次运行程序代码，得到的结果如图 6.40 所示。



图 6.40 筛选结果

## 6. 程序分析

用户可以自行尝试其他更加复杂的筛选条件，综合应用高级筛选的功能，可以很便利的实现多种复杂的筛选。

## 案例 124 筛选非重复值

### 1. 功能说明

筛选非重复值在数据处理和分析中是十分常见的一种应用。在很多情况下，用户需要了解某类数据的不同值，具体数值的个数反而不重要。这个时候，用户需要筛选出某类数据的非重复值。

### 2. 语法说明

在 Excel VBA 中，用户可以使用多种方法来筛选非重复值。在本例中，使用 Range 对象的 AdvancedFilter 方法筛选非重复值。根据前面小节的介绍，AdvancedFilter 方法的语法格式如下：

表达式.AdvancedFilter(Action, CriteriaRange, CopyToRange, Unique)

其中，参数 Unique 表示筛选唯一值。这是筛选非重复值的最简单的方法。

### 3. 案例说明

某公司统计了部分员的销量，同时提供了员工所处的地区和姓名。本例中，用户需要筛选地区的非重复值，其中原始数据如图 6.41 所示。



	A	B	C	D	E
1	编号	姓名	地区	销量	
2	114	王杰	北部	1600	
3	113	陈俊	北部	1810	
4	116	徐涛	东部	1110	
5	118	王军	东部	1124	
6	117	陈梁	东部	3420	
7	111	张梅	南部	1141	
8	110	冯涛	南部	1180	
9	112	张康	南部	1460	
10	115	王强	西部	1530	
11					

图 6.41 原始数据

### 4. 编写代码

筛选非重复值的 VBA 代码如下：

```
Sub FilterUni()  
Dim i As Long, rng As Range
```



```

Application.ScreenUpdating = False

With ActiveSheet
    i = .Range("C1").End(xlDown).Row
    If i > 1001 Then Exit Sub
    Set rng = .Range(Cells(1, 3), Cells(i, 3))

    rng.AdvancedFilter Action:=xlFilterCopy, _
    CopyToRange:=.Range("F1"), Unique:=True
End With

Application.ScreenUpdating = True
End Sub

```

### 5. 程序结果

运行程序代码，查看筛选后的结果，如图 6.42 所示。

	A	B	C	D	E	F
1	编号	姓名	地区	销量		地区
2	114	王杰	北部	1600		北部
3	113	陈俊	北部	1810		东部
4	116	徐涛	东部	1110		南部
5	118	王军	东部	1124		西部
6	117	陈梁	东部	3420		
7	111	张梅	南部	1141		
8	110	冯涛	南部	1180		
9	112	张康	南部	1460		
10	115	王强	西部	1530		
11						

图 6.42 筛选结果

### 6. 程序分析

用户还可以用 Countif 函数来筛选非重复值，但是其代码会比高级筛选复杂。在其他章节中，将介绍其他方法来筛选非重复值。

## 案例 125 取消筛选

### 1. 功能说明

当用户对数据进行分析后，需要保留或者分析原始数据。这个时候，用户需要取消对数据的筛选。通过 Excel VBA 代码，用户可以取消该筛选。



## 2. 语法说明

如果当前在工作表上显示有“自动筛选”下拉箭头，则 `AutoFilterMode` 属性值为 `True`。设置该属性值为 `False` 可取消自动筛选状态。

注意：不能将该属性设置为 `True`。使用 `AutoFilter` 方法可筛选列表并显示下拉箭头。

## 3. 案例说明

某公司对其原始数据进行筛选，现在需要通过 VBA 代码实现取消筛选，其中筛选状态的工作表如图 6.43 所示。



	A	B	C	D
1	编号	姓名	地区	销量
4	116	徐涛	东部	1110
5	118	王军	东部	1124
6	117	陈梁	东部	3420

图 6.43 筛选后的工作表

## 4. 编写代码

取消筛选的 VBA 代码如下：

```
Sub DisFilter()
    Dim ws As Worksheet
    For Each ws In Worksheets
        ws.AutoFilterMode = False
    Next
End Sub
```

## 5. 程序结果

运行程序代码，查看取消筛选后的结果，如图 6.44 所示。



	A	B	C	D
1	编号	姓名	地区	销量
2	114	王杰	北部	1600
3	113	陈俊	北部	1810
4	116	徐涛	东部	1110
5	118	王军	东部	1124
6	117	陈梁	东部	3420
7	111	张梅	南部	1141
8	110	冯涛	南部	1180
9	112	张康	南部	1460
10	115	王强	西部	1530

图 6.44 取消筛选后的结果

## 6. 程序分析

在 Excel 中，Worksheet.FilterMode 属性同样显示用户的筛选状态。如果工作表处于筛选模式，则为 True。但是该属性是只读的，不能通过代码修改。

## 6.3 查询数据

在数据处理中，当数据量很大的时候，查询数据是十分常用的操作。在 Excel 的基本操作中，用户可以选择“开始”选项卡的“编辑”组中单击“查找和选择”按钮，从下拉的菜单按钮中选择相应的命令即可进行查询操作，如图 6.45 所示。

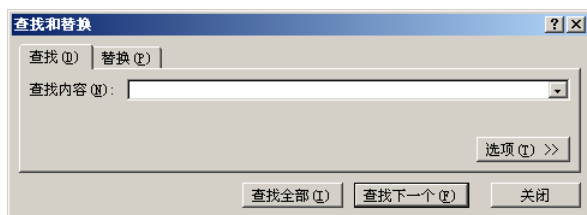


图 6.45 查找内容

在 VBA 中，用户可以使用 Find 方法进行查询相关的操作，本节实例演示常见的数据查询的方法。

### 案例 126 查找数值

#### 1. 功能说明

对于查询数据而言，最常见的形式是在工作表范围内容查询。在 Excel 中，用户可以查询各种类型的数据：包括文本、数值等。在本小节中，将详细讲解如何使用 VBA 编写查询的代码。

#### 2. 语法说明

本例的查找使用了 Range 对象的两个方法：Find 方法和 FindNext 方法。下面详细讲解这两种方法。Find 方法可以在区域中查找特定信息。其语法格式如下：

```
表达式.Find(What, After, LookIn, LookAt, SearchOrder, SearchDirection, MatchCase, MatchByte, SearchFormat)
```

该方法的参数很多，其中 What 参数是必须指定的，其余参数都可省略。各参数的含义如下：

- What: 要搜索的数据。可为字符串或任意 Excel 数据类型。

- **After**: 表示搜索过程将从其之后开始进行的单元格。此单元格对应于从用户界面搜索时的活动单元格的位置。**After** 必须是区域中的单个单元格。要记住搜索是从该单元格之后开始的；直到此方法绕回到此单元格时，才对其进行搜索。如果不指定该参数，搜索将从区域的左上角的单元格之后开始。
- **LookIn**: 信息类型。
- **LookAt**: 设置匹配文本的方式。可为常量 **xlWhole**（匹配全部搜索文本）或 **xlPart**（匹配任一部分搜索文本）。
- **SearchOrder**: 指定搜索区域的次序。可为常量 **xlByRows**（按行）或 **xlByColumns**（按列）搜索。
- **SearchDirection**: 搜索的方向。可为常量 **xlNext**（在区域中搜索下一匹配值）或 **xlPrevious**（在区域中搜索上一匹配值）。
- **MatchCase** : 如果为 **True**，则搜索区分大小写。默认值为 **False**。
- **MatchByte**: 只在已经选择或安装了双字节语言支持时适用。如果为 **True**，则双字节字符只与双字节字符匹配。如果为 **False**，则双字节字符可与其对等的单字节字符匹配。
- **SearchFormat**: 搜索的格式。

使用该方法将返回一个 **Range** 对象，它代表第一个在其中找到该信息的单元格。如果未发现匹配项，则返回 **Nothing**。**Find** 方法不影响选定区域或当前活动的单元格。

提示：每次使用此方法后，参数 **LookIn**、**LookAt**、**SearchOrder** 和 **MatchByte** 的设置都将被保存。如果下次调用此方法时不指定这些参数的值，就使用保存的值。设置这些参数将更改“查找”对话框中的设置，如果省略这些参数，更改“查找”对话框中的设置将更改使用的保存值。要避免出现这一问题，每次使用此方法时最好明确设置这些参数。

**FindNext** 方法继续由 **Find** 方法开始的搜索。查找匹配相同条件的下一个单元格，并返回表示该单元格的 **Range** 对象。该操作不影响选定内容和活动单元格。其语法格式如下：

**表达式.FindNext(After)**

参数 **After** 指定一个单元格，查找将从该单元格之后开始。此单元格对应于从用户界面搜索时的活动单元格位置。

注意：**After** 必须是查找区域中的单个单元格。搜索是从该单元格之后开始的；直到本方法环绕到此单元格时，才检测其内容。如果未指定本参数，查找将从区域的左上角单元格之后开始。

当查找到指定查找区域的末尾时，**FindNext** 方法将环绕至区域的开始继续搜索。发生环绕后，为停止查找，可保存第一次找到的单元格地址，然后测试下一个查找到的单元格地址是否与其相同。

### 3. 案例说明

某公司统计了部分员的销量，同时提供了员工所处的地区和姓名。本例中，用户需要查询该数据表中的数据，其中原始数据如图 6.46 所示。

	A	B	C	D	E
1	编号	姓名	地区	销量	
2	114	王杰	北部	1600	
3	113	陈俊	北部	1810	
4	116	徐涛	东部	1110	
5	118	王军	东部	1124	
6	117	陈梁	东部	3420	
7	111	张梅	南部	1141	
8	110	冯涛	南部	1180	
9	112	张康	南部	1460	
10	115	王强	西部	1530	
11					

图 6.46 原始数据

#### 4. 编写代码

数据查询的 VBA 代码如下：

```
Sub FindNumbers()
    Dim Number_Find As String
    Dim str As String
    Dim rng As Range

    Number_Find = Application.InputBox(prompt:="输入查询的数值：", Title:="数据查询",
    Type:=2)
    If Number_Find = "False" Or Number_Find = "" Then Exit Sub

    Application.ScreenUpdating = False
    Application.DisplayAlerts = False

    With ActiveSheet.Cells
        Set rng = .Find(Number_Find, , , xlWhole, xlByColumns, xlNext, False)
        If Not rng Is Nothing Then
            str = rng.Address
            Do
                rng.Interior.ColorIndex = 3
                Set rng = .FindNext(rng)
            Loop While Not rng Is Nothing And rng.Address <> str
        End If
    End With

    Application.ScreenUpdating = True
    Application.DisplayAlerts = True
End Sub
```

## 5. 程序结果

运行上面的程序代码，会显示对话框，提示需要查询的数据，如图 6.47 所示。



图 6.47 数据查询

单击对话框中的“确定”按钮，查看查询的结果如图 6.48 所示。



图 6.48 查询的结果

## 6. 程序分析

在上面的代码中，首先让用户输入查找的值，接着使用 **Find** 方法查找第一个满足条件的单元格，再使用循环查找工作簿中下一个满足条件的单元格，并在循环中对满足条件的单元格设置底纹。

## 案例 127 格式查询

### 1. 功能说明

在前面的案例中，用户使用 **Find** 方法在 Excel 查询数值。在 Excel 中，除了查询数据之外，还可以根据单元格的格式进行查询。在某些工作表中，用户可能为某些单元格设置了特殊格式，因此用户可以根据格式进行查询。

### 2. 语法说明

同时，在 Excel VBA 中，用户可以使用 **Application.FindFormat** 属性来设置或返回要查

找的单元格格式类型的搜索条件。其语法表达式是：

表达式.FindFormat

表达式代表 Application 对象的变量。

### 3. 案例说明

某公司统计了部分员工的销量，同时提供了员工所处的地区和姓名。对该数据范围设置格式查询的条件，其中原始数据如图 6.49 所示。



	A	B	C	D	E
1	编号	姓名	地区	销量	
2	114	王杰	北部	1600	
3	113	陈俊	北部	1810	
4	116	徐涛	东部	1110	
5	118	王军	东部	1124	
6	117	陈梁	东部	3420	
7	111	张梅	南部	1141	
8	110	冯涛	南部	1180	
9	112	张康	南部	1460	
10	115	王强	西部	1530	
11					

图 6.49 原始数据

### 4. 编写代码

格式查找的 VBA 代码如下：

```
Sub FindFormatCells()  
With Application.FindFormat.Font  
    .Name = "宋体"  
    .FontStyle = "Regular"  
    .Size = 10  
End With  
  
With Application.FindFormat.Font  
    MsgBox .Name & "-" & .FontStyle & "-" & .Size & _  
        " 是设置的搜索条件."  
End With  
End Sub
```

### 5. 程序结果

运行上面的程序代码，会显示程序运行的结果，如图 6.50 所示。



图 6.50 查询的结果

## 6. 程序分析

在上面的代码中，首先使用 **FindFormat** 属性设置查找的格式条件，然后通过 **Msgbox** 显示对话框，显示查找的格式条件。

## 案例 128 向前查询

### 1. 功能说明

在实际的数据分析中，查询有具体的方向。例如，在某源数据中，用户也许只需要了解到某数据之前的数据情况。这个时候，用户就需要使用到向前查询的功能。在 **Excel VBA** 中，用户可以使用 **Find** 来实现，同样可以使用 **FindPrevious** 方法来实现。在本小节中，将详细介绍如何使用 **FindPrevious** 方法。

### 2. 语法说明

**FindPrevious** 方法方法继续由 **Find** 方法开始的搜索。查找匹配相同条件的上一个单元格，并返回代表该单元格的 **Range** 对象。其语法格式如下：

**表达式.FindPrevious(After)**

参数 **After** 指定一个单元格，查找将从该单元格之前开始。此单元格对应于从用户界面搜索时的活动单元格的位置。**After** 必须是查找区域中的单个单元格。搜索是从该单元格之前开始的；直到本方法环绕到此单元格时，才检测其内容。如果未指定本参数，查找将从区域的左上角单元格之后开始。

当查找到指定查找区域的末尾时，本方法将环绕至区域的开始继续搜索。发生环绕后，为停止查找，可保存第一次找到的单元格地址，然后测试下一个查找到的单元格地址是否与其相同。

### 3. 案例说明

某公司统计了部分员的销量，同时提供了员工所处的地区和姓名。本例中，用户需要查询该数据表中的数据，其中原始数据如图 6.51 所示。



	A	B	C	D	E
1	编号	姓名	地区	销量	
2	114	王杰	北部	1600	
3	113	陈俊	北部	1810	
4	116	徐涛	东部	1110	
5	118	王军	东部	1124	
6	117	陈梁	东部	3420	
7	111	张梅	南部	1141	
8	110	冯涛	南部	1180	
9	112	张康	南部	1460	
10	115	王强	西部	1530	
11					

图 6.51 原始数据

#### 4. 编写代码

向前查找的 VBA 代码如下：

```
Sub FindPreNumbers()  
    Dim Target_Num As String  
    Dim rng As Range  
    Cells.Interior.ColorIndex = 0  
    If rng Is Nothing Then  
        Target_Num = Application.InputBox(prompt:="输入搜索的数值: ", Title:="搜索", Type:=2)  
        If Target_Num = "False" Or Target_Num = "" Then Exit Sub  
        Set rng = ActiveSheet.Cells.Find(Target_Num, , , xlWhole, xlByColumns, xlPrevious,  
False)  
    Else  
        Set rng = ActiveSheet.Cells.FindPrevious(rng)  
    End If  
    If Not rng Is Nothing Then  
        rng.Interior.ColorIndex = 3  
    End If  
End Sub
```

#### 5. 程序结果

选择单元格 C2，然后运行程序代码，在弹出的对话框中输入查询的数据“东部”，如图 6.52 所示。





图 6.52 输入查询的数据

单击上面对话框中的“确定”按钮，查看查询的结果，如图 6.53 所示。



图 6.53 查看查询的结果

## 6. 程序分析

在上面的代码中，首先判断模块变量 `rng` 是否为空，若为空，则打开对话框让用户输入查询条件，并使用 `Find` 方法向前查找。若模块变量 `rng` 不为空，则调用 `FindPrevious` 方法向前查找。

## 案例 129 向后查询

### 1. 功能说明

前面小节中，已经讲解到向前查询的例子。在实际应用中，向后查询也是一个十分常见的应用。例如，在某源数据中，用户也许只需要了解到某数据之后的数据情况。这个时

候，用户就需要使用到向前查询的功能。在 Excel VBA 中，用户可以使用 Find 来实现，同样可以使用 FindNext 方法来实现。在本小节中，将详细介绍如何使用 FindNext 方法。

## 2. 语法说明

FindNext 方法继续由 Find 方法开始的搜索。查找匹配相同条件的下一个单元格，并返回表示该单元格的 Range 对象。该操作不影响选定内容和活动单元格。该方法的语法格式如下：

**表达式.FindNext(After)**

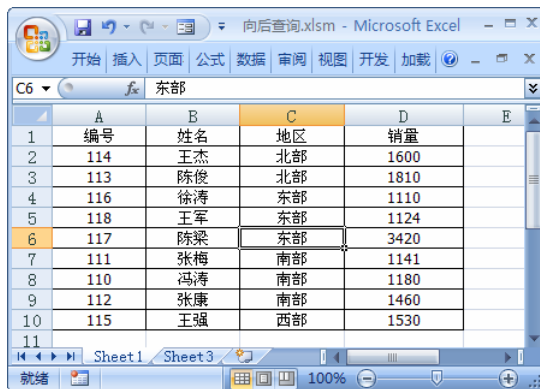
表达式表示 Range 对象的变量。

参数 After，可选，指定一个单元格，查找将从该单元格之后开始。此单元格对应于从用户界面搜索时的活动单元格位置。注意，After 必须是查找区域中的单个单元格。注意，搜索是从该单元格之后开始的；直到本方法环绕到此单元格时，才检测其内容。如果未指定本参数，查找将从区域的左上角单元格之后开始。

当查找到指定查找区域的末尾时，本方法将环绕至区域的开始继续搜索。发生环绕后，为停止查找，可保存第一次找到的单元格地址，然后测试下一个查找到的单元格地址是否与其相同。

## 3. 案例说明

某公司统计了部分员的销量，同时提供了员工所处的地区和姓名。本例中，用户需要查询该数据表中的数据，其中原始数据如图 6.54 所示。



	A	B	C	D	E
1	编号	姓名	地区	销量	
2	114	王杰	北部	1600	
3	113	陈俊	北部	1810	
4	116	徐涛	东部	1110	
5	118	王军	东部	1124	
6	117	陈梁	东部	3420	
7	111	张梅	南部	1141	
8	110	冯涛	南部	1180	
9	112	张康	南部	1460	
10	115	王强	西部	1530	
11					

图 6.54 原始数据

## 4. 编写代码

向后查找的 VBA 代码如下：

```
Sub FindNextNumbers()  
    Dim Target_Num As String  
    Dim rng As Range  
  
    Cells.Interior.ColorIndex = 0
```

```

If rng Is Nothing Then
    Target_Num = Application.InputBox(prompt:="请输入要查找的值: ", Title:="查找",
Type:=2)
    If Target_Num = "False" Or Target_Num = "" Then Exit Sub
    Set rng = ActiveSheet.Cells.Find(Target_Num, , , xlWhole, xlByColumns, xlNext, False)
Else
    Set rng = ActiveSheet.Cells.FindNext(rng)
End If
If Not rng Is Nothing Then
    rng.Interior.ColorIndex = 3
End If
End Sub

```

## 5. 程序结果

选择单元格 C4，然后运行程序代码，在弹出的对话框中输入查询的数据“东部”，如图 6.55 所示。

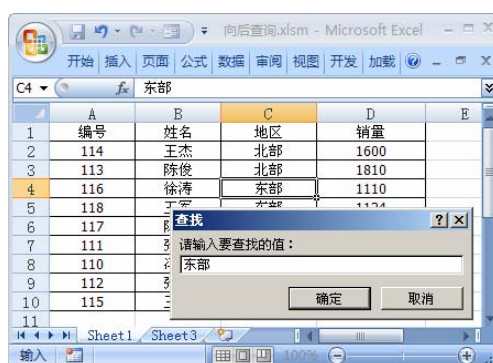


图 6.55 输入查询条件

单击上面对话框中的“确定”按钮，查看查询的结果，如图 6.56 所示。



图 6.56 查询结果

## 6. 程序分析

在上面的代码中，首先判断模块变量 `rng` 是否为空，若为空，则打开对话框让用户输入查询条件，并使用 `Find` 方法向前查找。若模块变量 `rng` 不为空，则调用 `FindNext` 方法向前查找。

## 案例 130 模糊查询

### 1. 功能说明

在前面章节讲解数据筛选的时候，曾经介绍过模糊查询的方法。在 Excel 中，实际上还可以通过 `Like` 运算符来进行模糊查询。在本小节中，将详细讲解如何使用 `Like` 运算符来进行模糊查询。

### 2. 语法说明

在 Excel 中，`Like` 运算符可用来比较两个字符串。其使用方法如下：

```
result = string Like pattern
```

`Like` 运算符的语法具有以下几个部分：

- `result`：运算的结果。
- `string`：被查询的字符串。
- `pattern`：查询字符串，该字符串可建立模式匹配。

如果 `string` 与 `pattern` 匹配，则 `result` 为 `True`；如果不匹配，则 `result` 为 `False`。但是如果 `string` 或 `pattern` 中有一个为 `Null`，则 `result` 为 `Null`。

`pattern` 中的字符可使用以下匹配模式：

- `?`：可为任何单一字符。
- `*`：零个或多个字符。
- `#`：任何一个数字(0 - 9)。
- `[charlist]`：`charlist` 中的任何单一字符。
- `[!charlist]`：不在 `charlist` 中的任何单一字符。

在中括号(`[ ]`)中，可以用由一个或多个字符(`charlist`)组成的组与 `string` 中的任一字符进行匹配，这个组几乎包括任何一个字符代码以及数字。

例如：

```
MyCheck = "王三" Like "王*"      ' 返回 True。
MyCheck = "D" Like "[A-Z]"        ' 返回 True。
MyCheck = "D" Like "[!A-Z]"       ' 返回 False。
MyCheck = "b3b" Like "b#b"        ' 返回 True。
```

### 3. 案例说明

某公司统计了部分员的销量，同时提供了员工所处的地区和姓名。本例中，用户需要查询该数据表中的数据，其中原始数据如图 6.57 所示。



	A	B	C	D	E
1	编号	姓名	地区	销量	
2	114	王杰	北部	1600	
3	113	陈俊	北部	1810	
4	116	徐涛	东部	1110	
5	118	王军	东部	1124	
6	117	陈梁	东部	3420	
7	111	张梅	南部	1141	
8	110	冯涛	南部	1180	
9	112	张康	南部	1460	
10	115	王强	西部	1530	
11					

图 6.57 原始数据

#### 4. 编写代码

模糊查询的 VBA 代码如下：

```
Sub FluSearch()  
    Dim Target_Num As String  
    Dim str As String  
    Dim SearchRng As Range  
    Dim rng As Range  
  
    Target_Num = Application.InputBox(prompt:="输入查询的数值: ", _  
        Title:="模糊查询", Type:=2)  
    If Target_Num = "False" Or Target_Num = "" Then Exit Sub  
  
    Application.ScreenUpdating = False  
    Application.DisplayAlerts = False  
  
    Set rng = ActiveSheet.Range("A1").CurrentRegion  
    str = "*" & Target_Num & "*"  
  
    For Each SearchRng In rng.Cells  
        If SearchRng.Value Like str Then  
            SearchRng.Interior.ColorIndex = 3  
        End If  
    Next  
  
    Application.ScreenUpdating = True  
    Application.DisplayAlerts = True  
End Sub
```

#### 5. 程序结果

运行程序代码，在弹出的对话框中输入查询的数据“东部”，如图 6.58 所示。



图 6.58 输入查询的数据

单击上面对话框中的“确定”按钮，查看查询的结果，如图 6.59 所示。

	A	B	C	D	E
1	编号	姓名	地区	销量	
2	114	王杰	北部	1600	
3	113	陈俊	北部	1810	
4	116	徐涛	东部	1110	
5	118	王军	东部	1124	
6	117	陈梁	东部	3420	
7	111	张梅	南部	1141	
8	110	冯涛	南部	1180	
9	112	张康	南部	1460	
10	115	王强	西部	1530	
11					

图 6.59 模糊查询的结果

## 6. 程序分析

在上面的代码中，首先让用户输入查询条件，接着使用循环对单元格进行比较，在比较时使用 Like 进行模糊查询，如果单元格中包含指定条件的值，则设置单元格的底色。

## 6.4 处理公式

公式是 Excel 中的特殊对象，在单元格中如果包含公式，则该单元格对象和其他单元格在处理起来会有不同。因此，在对 Excel 进行数据分析的时候，常常需要首先判断单元格中的公式信息是十分重要的。在本小节中，将详细讲解如何使用 VBA 处理公式信息。

### 案例 131 判断单元格是否包含公式

#### 1. 功能说明

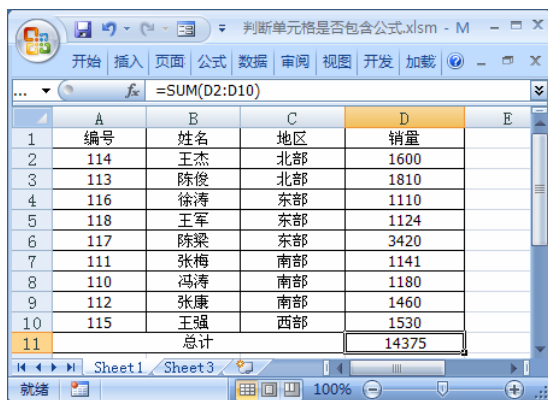
在 Excel 分析数据的时候，常常需要首先判断单元格中是否包含了公式。为此，Excel VBA 中特定提供了单元格的属性，本小节将详细讲解。

## 2. 语法说明

本例使用 Range 对象的 HasFormula 属性来判断指定单元格是否包含公式，如果区域中所有单元格均包含公式，则该属性值为 True；如果所有单元格均不包含公式，则该属性值为 False；其他情况下为 null。本例对当前单元格区域中的单元格逐个进行判断，并显示出具有公式的单元格。

## 3. 案例说明

某公司统计了部分员的销量，同时提供了员工所处的地区和姓名。同时为了统计销量的信息，公司添加了“总计”项目，计算销量的总和，其中原始数据如图 6.60 所示。



	A	B	C	D	E
1	编号	姓名	地区	销量	
2	114	王杰	北部	1600	
3	113	陈俊	北部	1810	
4	116	徐涛	东部	1110	
5	118	王军	东部	1124	
6	117	陈梁	东部	3420	
7	111	张梅	南部	1141	
8	110	冯涛	南部	1180	
9	112	张康	南部	1460	
10	115	王强	西部	1530	
11		总计		14375	

图 6.60 原始数据

## 4. 编写代码

判断单元格中包含公式的 VBA 代码如下：

```
Sub GetFormula()  
    Dim rng As Range  
    Dim FormlaRng As Range  
  
    Set rng = ActiveSheet.Range("A1").CurrentRegion  
  
    For Each FormlaRng In rng.Cells  
        If FormlaRng.HasFormula Then  
            MsgBox "单元格" & FormlaRng.Address & " 包含公式！"  
        End If  
    Next  
  
End Sub
```

## 5. 程序结果

运行程序代码，得到的结果如图 6.61 所示。



图 6.61 判断结果

6. 程序分析

在上面代码中，首先选定查看的循环区域，然后通过 HasFormula 属性来判断是否有单元格包括公式，最后显示搜索的结果。

案例 132 自动填充公式

1. 功能说明

在 Excel 的基本操作中，公式的自动填充是一种十分常用的功能。利用公式的相对引用和自动填充功能，在输入类似公式的时候，用户可以不用单独的每个具体输入，而只需要输入其中的一个，然后自动填充就可以。同样的功能，也可以使用 Excel VBA 代码来实现上述功能

2. 语法说明

本例使用 Range 对象的 AutoFill 方法，对指定区域中的单元格执行自动填充。该方法的语法格式如下：

表达式.AutoFill(Destination, Type)

该方法有两个参数，其含义如下：

- Destination：要填充的单元格。目标区域必须包括源区域。
- Type：指定填充类型。填充类型可使用 xlAutoFillType 枚举类型，其值如表 6.4 所示。

表 6.4 xlAutoFillType枚举值

名称	值	描述
xlFillCopy	1	将源区域的值和格式复制到目标区域，如有必要可重复执行。
xlFillDays	5	将星期中每天的名称从源区域扩展到目标区域中。格式从源区域复制到目标区域，如有必要可重复执行。
xlFillDefault	0	Excel确定用于填充目标区域的值和格式。
xlFillFormats	3	只将源区域的格式复制到目标区域，如有必要可重复执行。
xlFillMonths	7	将月名称从源区域扩展到目标区域中。格式从源区域复制到目标区域，如有必要可重复执行。
xlFillSeries	2	将源区域中的值扩展到目标区域中，形式为系列（如，“1, 2”扩展为“3, 4, 5”）。格式从源区域复制到目标区域，如有必要可重复执行。
xlFillValues	4	只将源区域的值复制到目标区域，如有必要可重复执行。
xlFillWeekdays	6	将工作周每天的名称从源区域扩展到目标区域中。格式从源区域复制到目标区域，如有必要可重复执行。



xlFillYears	8	将年从源区域扩展到目标区域中。格式从源区域复制到目标区域，如有必要可重复执行。
xlGrowthTrend	10	将数值从源区域扩展到目标区域中，假定源区域的数字之间是乘法关系（如，“1, 2,” 扩展为 “4, 8, 16”，假定每个数字都是前一个数字乘以某个值的结果）。格式从源区域复制到目标区域，如有必要可重复执行。
xlLinearTrend	9	将数值从源区域扩展到目标区域中，假定数字之间是加法关系（如，“1, 2,” 扩展为 “3, 4, 5”，假定每个数字都是前一个数字加上某个值的结果）。格式从源区域复制到目标区域，如有必要可重复执行。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区。并且对第一名员工的上下半年销量进行的总计。基础数据如图 6.62 所示。

	A	B	C	D	E	F
	编号	地区	上半年销量	下半年销量	总计	
2	114	北部	1600	1300	2900	
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141	1375		
8	110	南部	1180	1300		
9	112	南部	1460	1300		
10	115	西部	1530	1270		
11						

图 6.62 原始数据

### 4. 编写代码

填充公式的 VBA 代码如下：

```
Sub FillFormlua()
    Dim i As Long
    Dim rng As Range

    Set rng = Range("A1").CurrentRegion
    i = rng.Rows.Count

    Range("E2").AutoFill _
        Destination:=Range(Cells(2, 5), Cells(i, 5))
End Sub
```

### 5. 程序结果

运行程序代码，得到的结果如图 6.63 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量	总计	
2	114	北部	1600	1300	2900	
3	113	北部	1810	1330	3140	
4	116	东部	1110	1300	2410	
5	118	东部	1124	1375	2499	
6	117	东部	3420	1390	4810	
7	111	南部	1141	1375	2516	
8	110	南部	1180	1300	2480	
9	112	南部	1460	1300	2760	
10	115	西部	1530	1270	2800	
11						

图 6.63 填充公式的结果

## 6. 程序分析

在上面的代码中，首先获取当前区域的行，接着使用 `AutoFill` 方法在垂直方向填充相应的公式，用户可以检测填充公式后的情况，如图 6.64 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量	总计	
2	114	北部	1600	1300	2900	
3	113	北部	1810	1330	3140	
4	116	东部	1110	1300	2410	
5	118	东部	1124	1375	2499	
6	117	东部	3420	1390	4810	
7	111	南部	1141	1375	2516	
8	110	南部	1180	1300	2480	
9	112	南部	1460	1300	2760	
10	115	西部	1530	1270	2800	
11						

图 6.64 检测填充效果

## 案例 133 锁定公式

### 1. 功能说明

在 Excel 表格中，默认情况下工作表单元格中的公式是可以编辑的。但是，对于一些特定的运算过程和结果，用户则希望使用者不能编辑或者修改公式。这个时候，用户需要锁定公式。在本小节中，将详细介绍如何 Excel VBA 锁定公式。

### 2. 语法说明

要锁定公式，可通过 `Range` 对象的 `Locked` 属性来进行设置。`Locked` 属性：指明对象

是否已被锁定。当设置属性为 **True** 时，对指定区域锁定。但要真正锁定单元格，必须使用 **Protect** 方法对工作表进行保护。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区。并且对员工的上下半年销量计算了对应的总计。由于“总计”数据列中包含有公式，为了防止使用者编辑或者修改该公式，需要对该列数据锁定，基础数据如图 6.65 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量	总计	
2	114	北部	1600	1300	2900	
3	113	北部	1810	1330	3140	
4	116	东部	1110	1300	2410	
5	118	东部	1124	1375	2499	
6	117	东部	3420	1390	4810	
7	111	南部	1141	1375	2516	
8	110	南部	1180	1300	2480	
9	112	南部	1460	1300	2760	
10	115	西部	1530	1270	2800	
11						

图 6.65 源数据

### 4. 编写代码

锁定公式的 VBA 代码如下：

```
Sub LockFormula()  
  
    Selection.SpecialCells(xlCellTypeFormulas).Select  
    Selection.Locked = True  
  
    Worksheets("Sheet1").Protect DrawingObjects:=True, Contents:=True, Scenarios:=True  
    Worksheets("Sheet1").EnableSelection = xlNoRestrictions  
End Sub
```

### 5. 程序结果

运行程序代码，得到的结果如图 6.66 所示。



图 6.66 锁定的结果

## 6. 程序分析

在上面的代码中，首先通过 `SpecialCells(xlCellTypeFormulas)` 确定包含公式的单元格区域，然后将其锁定。

## 案例 134 隐藏公式

### 1. 功能说明

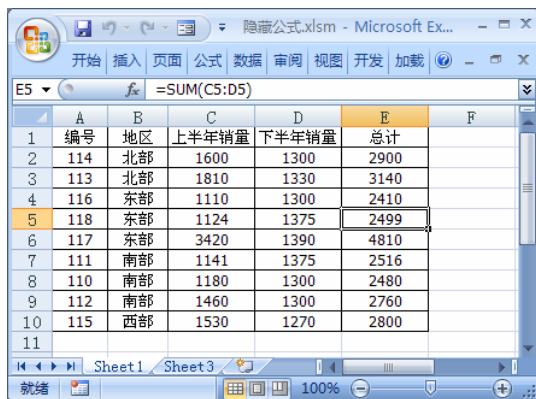
在使用 Excel 处理数据的时候，常常需要隐藏公式。因为公式显示了相应的计算过程，所以，为了让使用不了解计算过程，有时需要隐藏对应的公式。在本小节中，将详细介绍如何使用 Excel VBA 隐藏单元格中的公式。

### 2. 语法说明

要隐藏公式，可通过 `Range` 对象的 `FormulaHidden` 属性来进行设置。`FormulaHidden` 属性指明在工作表处于保护状态时是否隐藏公式。当设置属性为 `True` 时，对指定区域的公式隐藏。但要真正隐藏单元格，必须使用 `Protect` 方法对工作表进行保护。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区。并且对第员工的上下半年销量计算了对应的总计。由于“总计”数据列中包含有公式，为了防止使用者编辑或者修改该公式，需要对该列数据的公式隐藏，基础数据如图 6.67 所示。



	A	B	C	D	E	F
	编号	地区	上半年销量	下半年销量	总计	
1	114	北部	1600	1300	2900	
2	113	北部	1810	1330	3140	
3	116	东部	1110	1300	2410	
4	118	东部	1124	1375	2499	
5	117	东部	3420	1390	4810	
6	111	南部	1141	1375	2516	
7	110	南部	1180	1300	2480	
8	112	南部	1460	1300	2760	
9	115	西部	1530	1270	2800	
10						
11						

图 6.67 源数据

### 4. 编写代码

隐藏公式的 VBA 代码如下：

```
Sub HideFormula()  
  
    Selection.SpecialCells(xlCellTypeFormulas).Select
```

```
Selection.FormulaHidden = True
```

```
Worksheets("Sheet1").Protect DrawingObjects:=True, Contents:=True, Scenarios:=True  
Worksheets("Sheet1").EnableSelection = xlNoRestrictions
```

```
End Sub
```

## 5. 程序结果

运行程序代码，得到的结果如图 6.68 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量	总计	
2	114	北部	1600	1300	2900	
3	113	北部	1810	1330	3140	
4	116	东部	1110	1300	2410	
5	118	东部	1124	1375	2499	
6	117	东部	3420	1390	4810	
7	111	南部	1141	1375	2516	
8	110	南部	1180	1300	2480	
9	112	南部	1460	1300	2760	
10	115	西部	1530	1270	2800	
11						

图 6.68 隐藏公式的结果

## 6. 程序分析

用户可以通过基本的 Excel 操作来取消隐藏公式。具体操作是：选择“审阅”|“更改”|“撤销工作表保护”选项，如图 6.69 所示。



图 6.69 撤销工作表的保护

当用户单击对应的选项后，可以查看隐藏的公式，如图 6.70 所示。



图 6.70 查看隐藏的公式

### 案例 135 将公式转换为数值

#### 1. 功能说明

前面小节已经介绍过，在 Excel 中，公式和数值是两个不同的对象。在很多情况下，只是借助公式进行数值的计算，得到对应的结果。公式只是计算过程，因此，这个时候需要将公式转换为数值。

#### 2. 语法说明

将单元格公式转算为计算结果的表示方法很简单，只需通过以下的赋值运算即要：

```
rng.Value = rng.Value
```

以上赋值语句中，`rng` 表示 `Range` 对象，该语句首先通过右侧的表达式 `rng.Value` 获取指定单元格的值（如果是公式，则获取公式的计算结果），再将该值赋值给单元格的 `Value` 变量，从而取代单元格原有的内容（公式）。

#### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区。并且对第员工的上下半年销量计算了对应的总计。由于“总计”数据列中包含有公式，为了防止使用者编辑或者修改该公式，需要对该列数据的公式转换为数值，基础数据如图 6.71 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量	总计	
2	114	北部	1600	1300	2900	
3	113	北部	1810	1330	3140	
4	116	东部	1110	1300	2410	
5	118	东部	1124	1375	2499	
6	117	东部	3420	1390	4810	
7	111	南部	1141	1375	2516	
8	110	南部	1180	1300	2480	
9	112	南部	1460	1300	2760	
10	115	西部	1530	1270	2800	
11						
12						

图 6.71 原数据表

#### 4. 编写代码

公式转为数值的 VBA 代码如下：

```
Sub ChangeNumbers()
    Dim rng As Range
    Dim FormulaRng As Range

    Set rng = ActiveSheet.Range("A1").CurrentRegion
    For Each FormulaRng In rng.Cells
        If FormulaRng.HasFormula Then
            FormulaRng.Value = FormulaRng.Value
        End If
    Next
End Sub
```

#### 5. 程序结果

运行程序代码，得到的结果如图 6.72 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量	总计	
2	114	北部	1600	1300	2900	
3	113	北部	1810	1330	3140	
4	116	东部	1110	1300	2410	
5	118	东部	1124	1375	2499	
6	117	东部	3420	1390	4810	
7	111	南部	1141	1375	2516	
8	110	南部	1180	1300	2480	
9	112	南部	1460	1300	2760	
10	115	西部	1530	1270	2800	
11						
12						

图 6.72 转换为数值

## 6. 程序分析

在上面的代码中，首先获取工作表的当前区域，再逐个单元格判断，如果单元格有公式，则进行转换。