

第 8 章 数组

同字符串一样，数组也是 C#程序设计中最常使用的类型之一。数组能够按一定规律把相关的数据组织在一起，并能通过“索引”或“下标”快速的管理这些数据。本节和下一节，将深入讨论如何有效地使用数组。（以上内容属于章和节之间的过渡段，要求至少 3 行，说明本章要学习的内容，为什么要学习本章，读者能从本章中学到什么等等）（下面是说明本章的知识点，让读者知道最终学习完后能干什么）

本章主要涉及到的知识点有：

- 数组和多维数组：学会如何把数据有机的组合起来。
- 访问数组：学会查询、获取数组中的单一数据，让数据为我所用。
- 遍历数组：学会逐个浏览数组中的单一数据。
- 数组和字符串组合应用：通过本章最后的示例，演示如何组织字符串类型的数组，如何通过本章所学的知识，访问数组中的字符串。

注意：本章内容不包含动态数组。（注意技巧样式的应用。）

8.1 C#数组简介

本节首先介绍数组的基本概念，理解这些概念是学习使用 C#数组的基础。了解数组概念后，才能从数组的原理中找到学习的技巧。（节和小节之间的过渡段，至少 2 行）

8.1.1 数组的概念

首先来看什么是数组。数组即一组数据，它把一系列数据组织在一起，成为一个可操作的整体。举个简单的例子，当一个做事细心的妻子或丈夫去超市买东西时，或许会事先列出一个清单：（一定要用现实中的例子说名，注重趣味性）

- (1) 油
- (2) 盐
- (3) 酱
- (4) 醋
- (5) 毛毛熊
- (6)

可以称这个清单为“需购物品”，它规律的列出了其内部的数据，且其内部数据具有相同的性质。在 C#中，可以称这样一个清单为数组：

```
string[] myStrArr={"油", "盐", "酱", "醋", "毛毛熊"};
```

在数组中，其中的每一个元素对应排列次序下标，当使用其中的某个元素时，可以直接利用其这个次序下标，例如：

```
01 for(int i=0;i<myStrArr.Length;i++)
02 {
03     Console.WriteLine("item{0}:{1}",i,myStrArr[i]);
04 }
```

将输出数组 myStrArr 中所有的元素。

注意：同 C 语言和大部分语言一样，C#的下标也是从 0 开始，而不是从 1。**这种注意技巧两页必须有一个。**

8.1.2 多维数组

数组中的元素可以是任意的类型，如上面给出的示例：

```
string[] myStrArr={"油", "盐", "酱", "醋", "毛毛熊"};
```

其中的每个元素都是字符串类型。除此之外，元素还可以是 C#中其他的基本类型，甚至又是一个数组。如果数组的元素又是数组，那么这个数组成为多维数组，下面是一个二维数组的例子：

```
string[,] myStrArr2={{"油","盐"},{"《围城》","《晨露》"},{"毛毛熊","Snoopy"}};
```

这个例子中，数组的第一维包含了三个元素，而每一个元素又是一个数组，分别包含了两个元素。此时，称：（以下是并列样式，注意说明参数和后面的说明之间用冒号，说明完后用句号）

- 数组的秩为 2。
- 第一维的元素类型为数组，长度为 3。
- 第二维的元素类型为字符串，长度为 2。

通过这个具体的例子，请读者思考“秩”、“维”、“元素类型”，以及“长度”的概念。

同一维数组一样，多维数组中的每一个元素，也可以通过下标方式来引用，如

- myStrArr2[0]指一维数组{“油”，“盐”}
- myStrArr2[0,0]指字符串“油”，可以输出 myStrArr2 中的所有元素。

```
01 for(int i=0;i<3;i++)
02 {
03     Console.WriteLine("item{0}",i);
04     for(int j=0;j<2;j++)
05     {
06         Console.WriteLine(" item{0}{1}:{2}",i,j,myStrArr2[i,j]);
07     }
08 }
```

输出结果为：

```
item0
    item00:油
    item01:盐
```

```

item1
    item10:《围城》
    item11:《晨露》
item2
    item20:毛毛熊
    item21:Snoopy

```

8.1.3 创建 C#数组

在 C#中，使用如下语法创建一个数组：

1. 一维数组（尽量不要使用 4 级标题，如果用使用“标题 4”样式，注意编号） （不要正文一开始就是并列，至少有个过渡）

- `data_type[] arr_name = new data_type[int length]`

这种方式定义一个元素数据类型为 `data_type`，长度为 `length` 的数组 `arr_name`，例如：

```

int[] myIntArr=new int[100];           //定义一个长度为 100 的 int 数组
string[] mystringArr=new string[100];   //定义一个长度为 100 的 string 数组
object[] myObjectArr=new object[100];   //定义一个长度为 100 的 object 数组

```

其中，数据类型 `data_type` 既可以是常用数据类型（如 `int`、`float` 等），也可以是对象（如 `String`、`StringBuilder` 等）。

- `data_type[] arr_name = new data_type[] {item1, item2, ... ,itemn}`

这种方式定义一个元素数据类型为 `data_type`，并通过“=”运算符进行赋值，其初始值为所给出的元素 `{item1, item2, ... ,itemn}` 的个数，例如：

```

int[] myIntArr2=new int[]{1,2,3};       //定义一个 int 数组，长度为 3
string[] mystringArr2=new string[]{"油","盐"}; //定义一个 string 数组，长度为 2

```

在这种定义下，可以不必给出数组的长度定义，数组的长度自动为所给出的元素 `{item1, item2, ... ,itemn}` 的个数。即下面两种定义完全相同：

- (1) `int[] myIntArr2=new int[]{1,2,3};`
- (2) `int[] myIntArr2=new int[3]{1,2,3};`

2. 多维数组

- `data_type[,...,] arr_name = new data_type[int length1,int length2,...,int lengthn]`

这种方式定义一个元素数据类型为 `data_type`，秩为 `n`，各维长度分别为 `length1`，`length2`，...`lengthn` 的数组 `arr_name`，例如：

```

int[,] myIntArr=new int[10,100];       //定义一个 10*100 的二维 int 数组
string[,] mystringArr=new string[2,2,3]; //定义一个 2*2*3 的三维 string 数组

```

- `data_type[,...,] arr_name = new data_type[,...,]`
`{`
`{item1, item2, ... ,itemn}`
`...`
`}`

例如：（这种语句太短，一定要尽量保持语句的完整性：例如下面的代码。）

```
int[,] myIntArr2= new int[,]{{1,2,3},{-1,-2,-3}}; //2*3 的二维 int 数组
string[,] mystringArr2= new string[,]{{"油","盐"},{"《围城》","《晨露》"}}; // 2*2 的二维 string 数组
```

同一维数组一样，在这种定义下，可以不必给出各维的长度定义，各维长度自动根据所给出的赋值元素确定。

3. 交错数组

C#支持各个维度的长度可以不同的多维数组，称为交错数组，也被称为“数组的数组”。交错数组的定义如下：

- `data_type[][]... arr_name = new data_type[int length1][int length2]...`

这个定义和定义多维数组非常类似，区别在于，与多维数组不同的是，必须单独初始化交错数组每一维中的元素。

例如，下面定义一个第一维长度为 3 的交错数组：

```
01 int[][] myJaggedArray = new int[3][];
02 myJaggedArray[0] = new int[5];
03 myJaggedArray[1] = new int[4];
04 myJaggedArray[2] = new int[2];
```

这个交错数组 `myJagged`，每个元素都是一个一维整数数组。第一个元素是由 5 个整数组成的数组，第二个是由 4 个整数组成的数组，而第三个是由 2 个整数组成的数组。

- `data_type[][]... arr_name = new data_type[][]...`

```
{
    new data_type[] { item1, ... , new data_type[]itemn}
    ...
}
```

这种方式在声明数组时同时进行初始化，同样可以不指定各维的长度，例如：

```
01 int[][] myJaggedArray = new int [][]
02 {
03     new int[] {1,3,5,7,9},
04     new int[] {0,2,4,6},
05     new int[] {11,22}
06 };
```

8.2 使用数组

C#中的数组是由 `System.Array` 类派生而来的引用对象，因此，可以使用 `Array` 类的方法来进行各种操作。

另外，数组常常用来实现静态的操作，即不改变其空间大小，如查找、遍历等。当然，数组也可以实现动态的操作，如插入、删除等，但不推荐使用，而是尽量使用下一章将介绍集合来代替。

8.2.1 System.Array 类

System.Array 类是 C# 中各种数组的基类，其属性和方法一览如图 8.1 所示。（注意，图编号和图都有专门的样式，要求图统一缩放为 60%，而且**必须在正文中引出“如图 8.1 所示。”**，同时图的编号在整章都是连续的）

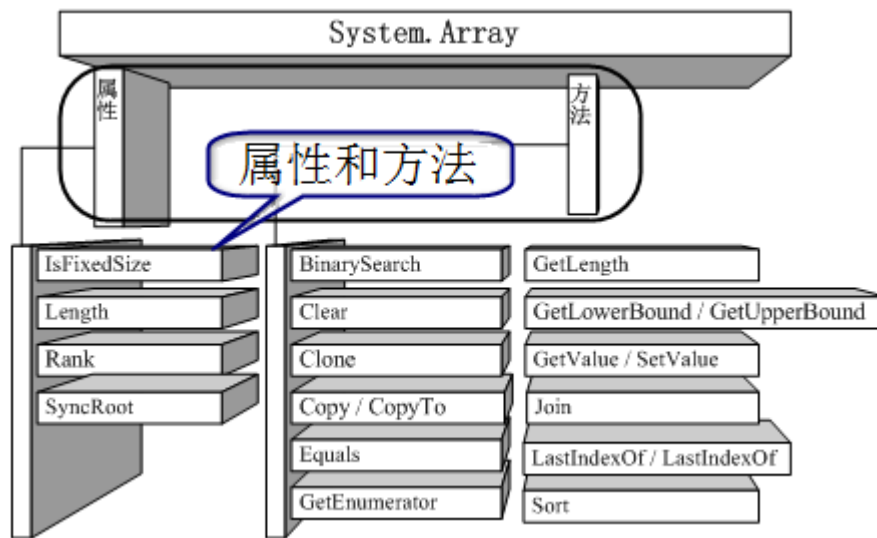


图 8.1 System.Array 类（图中的标注一定要清晰）

其常用属性和方法的简单说明如表 8.1 所示（表编号也必需在正文中引出）。先说明表的编号、表的意义，然后给出表，表编号和表内容的样式不同。

表 8.1 Array 类常用属性/方法说明（表头样式）

属性/方法	说明
IsFixedSize	指示 Array 是否具有固定大小（表格样式）
Length	获得一个 32 位整数，表示 Array 的所有维数中元素的总数
Rank	获取 Array 的秩（维数）
SyncRoot	获取可用于同步对 Array 的访问的对象
BinarySearch	使用二进制搜索算法在一维的排序 Array 中搜索值
Clone	创建 Array 的浅表副本
Copy/CopyTo	将一个 Array 的一部分复制到另一个 Array 中
GetLength	获取一个 32 位整数，表示 Array 的指定维中的元素。
GetLowerBound/GetUpperBound	获取 Array 的指定维度的下/上限
GetValue/SetValue	获取/设置 Array 中的指定元素值
IndexOf/LastIndexOf	返回一维 Array 或部分 Array 中某个值第一个/最后一个匹配项索引
Sort	对一维 Array 对象中的元素进行排序

8.2.2 访问数组元素

访问数组的元素包括读取或设置某个元素的值，最基本的方法是通过下标定位元素，

另外还可以使用 GetValue/SetValue 方法。

1. 通过下标定位元素

C#中数组对其中的元素进行排序，并从 0 开始计数，这样每一个元素都会有一个唯一的下标，通过这个下标，就可以定位唯一的一个元素。通过示例来说明：

(1) 一维数组

```
string[] myStrArr={"油", "盐", "酱", "醋", "毛毛熊"};
```

```
myStrArr[0]="油"
```

```
myStrArr[4]=" 毛毛熊"
```

如果试图访问超过下标范围的数据，则会出现如下异常：

```
System.IndexOutOfRangeException: 索引超出了数组界限。
```

(2) 多维数组

```
string[,] myStrArr2={{"油","盐"},{"《围城》","《晨露》"},{"毛毛熊","Snoopy"}};
```

```
myStrArr2[0,0]="油"
```

```
myStrArr2[4,1]=" Snoopy "
```

(3) 交错数组

```
01 int[][] myJaggedArray = new int [][]
02 {
03     new int[] {1,3,5,7,9},
04     new int[] {0,2,4,6},
05     new int[] {11,22}
06 };
```

则有：

```
01 myJaggedArray[0][0]=1
02 myJaggedArray[1][1]=2
03 myJaggedArray[2][1]=22
04 {
05     Console.WriteLine("item{0}",i);
06     for(int
07 j=myJaggedArray[i].GetLowerBound(0);j<=myJaggedArray[i].GetUpperBound(0);j++)
08     {
09         Console.WriteLine(" item{0}{1}:{2}",i,j,myJaggedArray[i][j]);
10     }
11 }
```

2. 使用 GetValue/SetValue

GetValue 方法定义如下：

- public object GetValue(params int[] indices);

其中，多个 int 型参数 indices 的含义为下标。方法返回一个 object 对象，这是 C#中所有对象的基类，使用多态性，它可以指向所有的 C#对象。

代码 8-3 使用GetValue方法，循环输出一个二维数组所有元素：

```
01 //定义二维数组
02 string[,] myStrArr2=new string[,]{{"油","盐"},{"《围城》","《晨露》"},{"毛毛熊","Snoopy"}};
```

```

03 //循环输出
04 for(int i=myStrArr2.GetLowerBound(0);i<=myStrArr2.GetUpperBound(0);i++)
05 {
06     Console.WriteLine("item{0}",i);
07     for(int j=myStrArr2.GetLowerBound(1);j<=myStrArr2.GetUpperBound(1);j++)
08     {
09         Console.WriteLine(" item{0}{1}:{2}",i,j,myStrArr2.GetValue(i,j));
10     }
11 }

```

SetValue 的功能为数组的某个元素赋值，其定义及参数表同 GetValue 相似，不作赘述。

8.2.3 遍历数组

遍历数组是指全部访问数组中的元素一次且仅一次，可以在遍历的过程中完成许多操作，如查找等。有两种方式可以遍历整个数组：

1. 使用 GetLowerBound/GetUpperBound 方法

GetLowerBound 方法可以获取数组某一维上的最低下标，而 GetUpperBound 则可获取其最高下标，利用这两个参数和 for 语句，就可以实现数组的遍历，如对于二维数组：

```

01 //定义二维数组
02 string[,] myStrArr2=new string[,]{"油","盐",{"《围城》","《晨露》"},{"毛毛熊","Snoopy"}};
03 //遍历
04 for(int i=myStrArr2.GetLowerBound(0);i<=myStrArr2.GetUpperBound(0);i++)
05 {
06     for(int j=myStrArr2.GetLowerBound(1);j<=myStrArr2.GetUpperBound(1);j++)
07     {
08         //处理每一个元素
09     }
10 }

```

2. 使用 foreach

还可以使用更为简便的方法来实现数组的遍历，那就是 foreach 关键字，这种方法对于处理高维数组尤其方便。

foreach 语句格式如下：

```

foreach (data_typt item_name in arr_name)
{
    //处理每一个元素
}

```

以下内容用来说明一些特殊技巧或声明。注意必须以“注意：说明：声明：技巧：”开头，并且使用模板提供的“注意说明技巧”样式。

注意：foreach 语句获取的元素是最深层的原子元素，因此，无论处理几维的数组，使用一层的 foreach 循环就可以了。

例如，代码 8-5 实现同样的二维数组遍历：

```
01 //定义二维数组
02 string[,] myStrArr2=new string[,]{"油","盐"},{"《围城》","《晨露》"},{"毛毛熊","Snoopy"};
03 //遍历
04 foreach(string item in myStrArr2)
05 {
06     {
07         //处理每一个元素
08     }
```

8.2.5 数组排序

对数组进行排序，是指按照一定的排序规则，如递增或递减规则，重新排序数组中的所有元素。可以使用 `Array` 类的 `Sort` 方法完成这个功能。

`Sort` 方法有多种重载方式，常用的形式如下：

- `public static void Sort(Array array);`

其中，参数 `array` 为待排序的数组。下面的示例首先定义了一个数组，含有元素 {5,4,3,2,1}，然后利用 `Sort` 方法对其排序。

```
01 /// 利用 Sort 方法进行数组排序
02 public void test1()
03 {
04     int[] myArr = { 5, 4, 3, 2, 1 }; //定义数组
05
06     //输出原始数组：原始数组:5->4->3->2->1->
07     Console.WriteLine( "原始数组:" );
08     for(int i=0;i<myArr.Length;i++)
09         Console.Write("{0}->",myArr[i]);
10
11     Array.Sort( myArr ); //对数组排序
12
13     //并输出排序后的数组：1->2->3->4->5->
14     Console.WriteLine( "排序以后数组:" );
15     for(int i=0;i<myArr.Length;i++)
16         Console.Write("{0}->",myArr[i]);
17 }
```

- `public static void Sort(Array keys, Array items);`

有时候需要进行所谓的关键字排序，例如，有两个数组 `arrSid` 和 `arrSname`，分别代表一组学生的学号和姓名，如果想要根据学号顺序输出姓名，或反之，都需要使用数组的排序操作，那么，如何把这两个数组联系在一起排序呢？这时就可以使用 `Sort` 的这种形式进行关键字排序。

其中，参数 `keys` 代表关键字数组，而 `items` 代表另一个数组。利用 `Sort`，下面的代码可实现上述需求：

```
01 /// 多个数组的关键字排序
```



```

02 public void test2()
03 {
04     //定义数组
05     int[] arrSid = { 5, 4, 3, 2, 1 };
06     string[] arrSname = { "张三", "李四", "王五", "麻子", "淘气" };
07
08     //输出原始数组：原始数组:张三(5)->李四(4)->王五(3)->麻子(2)->淘气(1)->
09     Console.WriteLine( "原始数组:" );
10     for(int i=0;i<arrSid.Length;i++)
11         Console.Write("{0}{1}->",arrSname[i],arrSid[i]);
12     Console.WriteLine();
13
14     //根据学号关键字排序
15     Array.Sort( arrSid,arrSname );
16
17     //并输出排序后的数组：淘气(1)->麻子(2)->王五(3)->李四(4)->张三(5)
18     Console.WriteLine( "排序以后数组:" );
19     for(int i=0;i<arrSid.Length;i++)
20         Console.Write("{0}{1}->",arrSname[i],arrSid[i]);
21 }

```

示例非常简单，输出已经在注释中给出，因此不作详细说明。

8.2.6 查找元素

在数组中查找元素，可以有两种解释，一是就是从整个数组中寻找与给定值相同的元素来。可以使用 `Array` 类的 `BinarySearch` 方法完成这个功能。二是判断数组中是否含有一个特定的元素，可以用 `Contains` 方法实现。

1. BinarySearch 方法

`BinarySearch` 使用二进制搜索算法在一维的排序 `Array` 中搜索值，注意必须是已经排序的数组，如果找到给定的值，则返回其下标；否则，返回一个负整数。其常用形式如下：

- `public static int BinarySearch(Array array,object value);`

其中，参数 `array` 为待搜索的数组，`value` 为待寻找的元素值。下面的示例首先定义了一个数组，含有元素 {5,4,3,2,1}，然后利用 `BinarySearch` 方法返回其中的元素 3 的下标 (2)。

代码 8-9 利用 `BinarySearch` 搜索数组元素示例：Class1.cs

```

01 /// 利用 BinarySearch 方法搜索元素
02 public void test1()
03 {
04     //定义数组
05     int[] myArr = { 5, 4, 3, 2, 1 };
06
07     //对数组排序

```

```

08     Array.Sort( myArr );
09
10     //搜索
11     int target=3;
12     int result=Array.BinarySearch(myArr,target);    //2
13     Console.WriteLine("{0}的下标为{1}",target,result);    //2
14 }

```

2. Contains 方法

望文知义，Contains 方法可以确定某个特定值是否包含在数组中，返回一个 bool 值。Array 类的这个方法实际上是对 IList 接口中方法的实现，其常用形式为：

- bool IList.Contains(object value);

其中，参数 value 代表所要验证的元素值，下面的示例判断学生数组 arrSname 中是否包含“王五”：

代码 8-10 利用 Contains 判断数组是否包含某个元素示例：Class1.cs

```

01  /// 判断是否包含某个值
02  public void test2()
03  {
04      //定义数组
05      string[] arrSname = { "张三", "李四", "王五", "麻子", "淘气" };
06
07      //判断是否含有某值
08      string target="王五";
09      bool result=((System.Collections.IList)arrSname).Contains(target);
10      Console.WriteLine("包含{0}?{1}",target,result); //true
11  }

```

可以看到，在使用 Contains 方法时，需要首先将数组转换为 IList（队列集合）对象。这是因为，本质上，数组是一种特殊的集合对象，因此可以把它转换为一个集合对象。对于集合，将在下一章中对其进行详细的讨论。

8.2.7 反转数组

反转数组是指，将一维数组中的全部或者部分元素的顺序，按照其逆序重新排列。可以使用 Array 类的 Reverse 静态方法完成这个功能。其常用的形式为：

- public static int Reverse(Array array);

这个重载形式可以反转整个数组元素，参数 array 为待反转的数组。下面的示例首先定义了一个数组，含有元素 {5,4,3,2,1}，然后利用 Reverse 方法进行反转。

代码 8-11 利用 Reverse 反转数组示例：Class1.cs

```

1  /// 利用 Reverse 方法反转数组
2  public void test1()
3  {
4      //定义数组
5      int[] myArr = { 5, 4, 3, 2, 1 };

```

```

6
7      //输出原始数组：原始数组:5->4->3->2->1->
8      Console.WriteLine( "原始数组:" );
9      for(int i=0;i<myArr.Length;i++)
10         Console.Write("{0}->",myArr[i]);
11      Console.WriteLine();
12
13      //对数组反转
14      Array.Reverse( myArr );
15
16      //并输出反转后的数组：1->2->3->4->5->
17      Console.WriteLine( "反转以后数组:" );
18      for(int i=0;i<myArr.Length;i++)
19         Console.Write("{0}->",myArr[i]);
20 }

```

另外，还可以反转数组的一部分元素，这时的 Reverse 重载形式为：

- `public static int Reverse(Array array,int index, int length);`

其中，参数 index 指定所要反转元素的其实下标，而 length 指定所要反转的元素个数。

8.2.8 复制数组

复制数组可以得到一个和原数组完全一样的新的数组，可以用 Array 的 Copy 或 CopyTo 方法来实现。

1. Copy 方法

在使用 Copy 方法进行数组复制操作之前，必须首先为新的数组分配空间，然后再通过复制操作向新的数组空间中填入元素值。

静态方法 Copy 的常用重载形式为：

- `public static void Copy(Array sourceArray,Array destinationArray,int length);`

其中，参数 sourceArray 为源数组，destinationArray 为目标数组，而 length 为要复制的元素数目，默认的复制操作从第一个元素开始。

下面的示例首先定义了一个数组，含有元素{5,4,3,2,1}，然后利用 Copy 方法获取一个新的数组，包含了源数组的前三项。

代码 8-12 利用 Copy 复制数组示例：Class1.cs

```

1  /// 利用 Copy 静态方法复制数组
2  public void test1()
3  {
4      //定义数组
5      int[] myArr = { 5, 4, 3, 2, 1 };
6
7      //输出原始数组：原始数组:5->4->3->2->1->
8      Console.WriteLine( "原始数组:" );
9      for(int i=0;i<myArr.Length;i++)

```

```

10         Console.Write("{0}->",myArr[i]);
11         Console.WriteLine();
12
13         //复制数组
14         int[] newArr=new int[3];
15         Array.Copy(myArr,newArr,3);
16
17         //并输出反复制的数组：5->4->3->
18         Console.WriteLine( "复制数组:" );
19         for(int i=0;i<newArr.Length;i++)
20             Console.Write("{0}->",newArr[i]);
21     }

```

注意：在使用 Copy 进行数组之前，必须要首先定义一个新的数组，并对其分配空间。另外，还需保证所分配的空间足够容纳所要复制的元素，否则，在复制时将出现异常：“System.ArgumentException: 目标数组的长度不够。”。

2. CopyTo 方法

CopyTo 和 Copy 方法的功能类似，但它是一个实例方法，即需要在数组对象上引用。另外的区别是，它只能复制源数组所有的元素，并可以空值元素在目标数组中存放的起始位置。其常用重载形式为：

- public virtual void CopyTo(Array array, int index);

其中，参数 array 代表所要复制的目标数组，index 则表示开始复制的元素下标。下面的代码实现对源数组的复制，并从第三个位置开始存放。

代码 8-13 利用 CopyTo 复制数组示例：Class1.cs

```

1  /// 利用 CopyTo 实例方法复制数组
2  public void test2()
3  {
4      //定义数组
5      int[] myArr = { 5, 4, 3, 2, 1 };
6
7      //输出原始数组：原始数组:5->4->3->2->1->
8      Console.WriteLine( "原始数组:" );
9      for(int i=0;i<myArr.Length;i++)
10         Console.Write("{0}->",myArr[i]);
11         Console.WriteLine();
12
13         //复制数组
14         int[] newArr=new int[7];
15         myArr.CopyTo(newArr,2);
16
17         //并输出反复制的数组：0->0->5->4->3->2->1->
18         Console.WriteLine( "复制数组:" );
19         for(int i=0;i<newArr.Length;i++)
20             Console.Write("{0}->",newArr[i]);
21     }

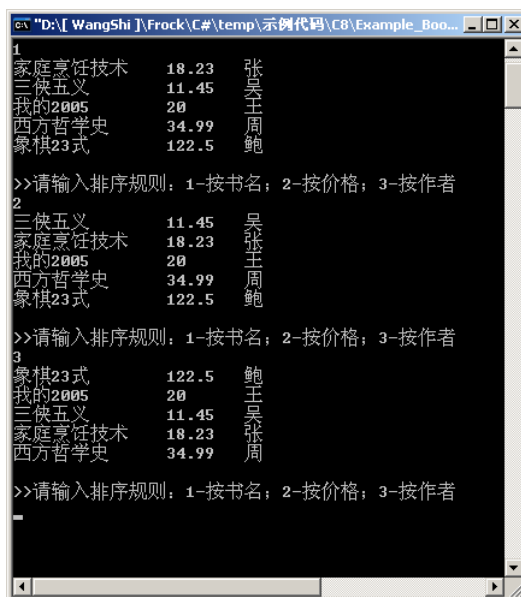
```

另外，从输出结果还可以看到，在初始化 `int` 型数组时，其默认值为 0。那么对于字符串数组，默认值是什么呢？请读者自己试验。

8.3 综合示例

【本节示例参考：`\源代码\C8\Example_BookList`】

本节将利用上一章和本章所介绍的字符串和数组操作技术，实现一个完整的示例：我的书架之图书排序。其功能为按照一定的排列顺序显示一系列图书信息，最终实现结果如图 8.2 所示。



```
1
家庭烹饪技术 18.23 张
三侠五义 11.45 吴
我的2005 20 王
西方哲学史 34.99 周
象棋23式 122.5 鲍

>>请输入排序规则：1-按书名；2-按价格；3-按作者
2
三侠五义 11.45 吴
家庭烹饪技术 18.23 张
我的2005 20 王
西方哲学史 34.99 周
象棋23式 122.5 鲍

>>请输入排序规则：1-按书名；2-按价格；3-按作者
3
象棋23式 122.5 鲍
我的2005 20 王
三侠五义 11.45 吴
家庭烹饪技术 18.23 张
西方哲学史 34.99 周

>>请输入排序规则：1-按书名；2-按价格；3-按作者
```

图 8.2 图书列表示例最后结果

整体的思路是：首先，每本图书的信息用一个图书对象来表示，包括书名、价格、作者三项，然后所有的图书信息放在一个数组 `arrBooks` 中，最后通过一个 `BookList` 类来完成图书显示。如果实例比较大，必须先给出一个实例的实现流程，最好用 Visio 画，编辑可以直接修改。

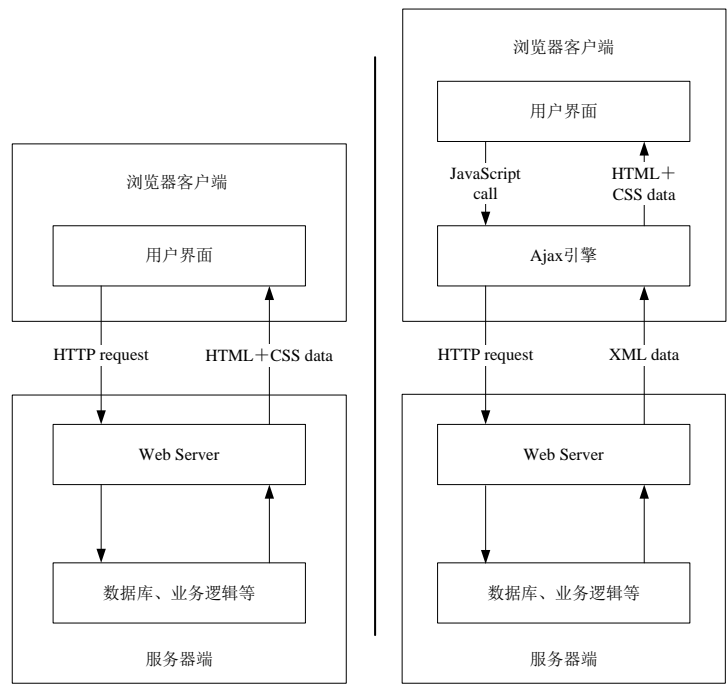


图 8.3 本例的实现流程

1. Book 类

Book 类如图 8.3 所示。

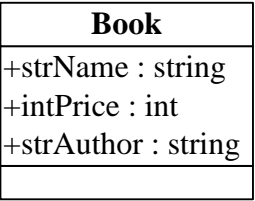


图 8.3 Book 类图

属性介绍：

- strName: 图书名
- intPrice: 图书价格
- strAuthor: 图书作者

图书类 Book 的实现代码参考如下。

代码 8-15 图书类 Book 的实现代码：Class1.cs

```
1.  /// <summary>
2.  /// 图书类
3.  /// </summary>
4.  class Book
5.  {
6.      public string strName;        //图书名
```

```

7.      public double dblPrice;          //图书价格
8.      public string strAuthor;         //图书作者
9.
10.     /// <summary>
11.     /// 构造函数
12.     /// </summary>
13.     /// <param name="_strName">图书名</param>
14.     /// <param name="_dblPrice">图书价格</param>
15.     /// <param name="_strAuthor">图书作者</param>
16.     public Book(string _strName,double _dblPrice,string _strAuthor)
17.     {
18.         this.strName=_strName;
19.         this.dblPrice=_dblPrice;
20.         this.strAuthor=_strAuthor;
21.     }
22. }

```

图书类的实现非常简单，它有三个公共属性（第 6-9 行），还有一个构造函数（第 10-21 行）。

2. BookList 类

BookList 类，用于按照不同的排序规则显示图书列表，它有三个静态方法，输入参数都是一个图书数组，功能为把所有书目进行排序并显示出来。BookList 类如图 8.4 所示。

BookList
+DisplayByName() +DisplayByPrice() +DisplayByAuthor()

图 8.4 BookList 类

这三个排序显示的静态方法为：

- DisplayByName：根据图书名顺序显示多本图书
- DisplayBy：根据图书价格顺序显示图书
- DisplayBy：根据图书作者顺序显示图书

下面单独来看 DisplayByName 的实现，代码如下。

代码 8-16 按图书名显示图书实现代码：Class1.cs

```

1.     /// <summary>
2.     /// 按照图书名显示一个图书数组中的多本图书
3.     /// </summary>
4.     /// <param name="arrBooks">图书数组</param>
5.     public static void DisplayByName(Book[] arrBooks)
6.     {
7.         //获取图书数目，用户动态建立“书名”数组
8.         int bookNumber=arrBooks.GetUpperBound(0)-arrBooks.GetLowerBound(0)+1;
9.

```

```

10.    //使用 CreateInstance 方法，动态建立“书名”数组
11.    int[] lengths=new int[]{bookNumber};
12.    int[] lowerBounds=new int[]{0};
13.    Array arrNames=
        Array.CreateInstance(Type.GetType("System.String"),lengths,lowerBounds);
14.
15.    //为“书名”数组赋值
16.    for(int i=arrBooks.GetLowerBound(0);i<=arrBooks.GetUpperBound(0);i++)
17.        arrNames.SetValue(arrBooks[i].strName,i);
18.
19.    //利用 Sort 方法，以“书名”为键，将图书排序
20.    Array.Sort(arrNames,arrBooks);
21.
22.    //显示排序后的图书列表
23.    foreach(Book item in arrBooks)
24.    {
25.        Console.WriteLine("{0} {1} {2}",item.strName,item.dblPrice,item.strAuthor);
26.    }
27. }

```

第 8 行，方法首先获取了输入参数 `arrBooks` 的图书数目，这是为了后面动态建立书名数组而准备。

接下来，第 10—13 行利用 `Array` 的 `CreateInstance` 静态方法，动态建立了一个一维数组 `arrNames`，下标从 0 开始，用于存储图书的书名信息，这是为了后面利用 `Sort` 方法进行排序作准备。

第 15—17 行对书名数组进行赋值。

第 20 行进行了排序，这是一个键值排序，以书名为键，排序所有的图书。

第 22—26 行顺序显示排序好后的图书信息。

另外的两个方法 `DisplayByPrice` 和 `DisplayByAuthor` 与上面所介绍的 `DisplayByName` 非常类似，此处不再进行详细的说明。

而整个 `BookList` 类的实现如下。

代码 8-17 `BookList` 类完整实现代码：Class1.cs

```

1.  class BookList
2.  {
3.      /// <summary>
4.      /// 按照图书名显示一个图书数组中的多本图书
5.      /// </summary>
6.      /// <param name="arrBooks">图书数组</param>
7.      public static void DisplayByName(Book[] arrBooks)
8.      {
9.          //...
10.     }
11.
12.     /// <summary>
13.     /// 按照图书价格显示一个图书数组中的多本图书

```



```

14.    /// </summary>
15.    /// <param name="arrBooks">图书数组</param>
16.    public static void DisplayByPrice(Book[] arrBooks)
17.    {
18.        //...
19.    }
20.
21.    /// <summary>
22.    /// 按照图书名显示一个图书数组中的多本图书
23.    /// </summary>
24.    /// <param name="arrBooks">图书数组</param>
25.    public static void DisplayByAuthor(Book[] arrBooks)
26.    {
27.        //...
28.    }
29. }

```

类分别实现了上面介绍的三个函数，除此之外，没有其他更多的内容。

3. 主函数

主测试函数首先使用 `Book` 类，定义了 5 本数，并把这 5 本书放在一个数组内，然后通过 `BookList` 类的静态方法进行排序输出。实现代码如下所示。

代码 8-18 Main 函数: Class1.cs

```

1.    /// <summary>
2.    /// 应用程序的主入口点。
3.    /// </summary>
4.    static void Main(string[] args)
5.    {
6.        Book b1=new Book("我的 2005",20,"王");
7.        Book b2=new Book("家庭烹饪技术",18.23,"张");
8.        Book b3=new Book("西方哲学史",34.99,"周");
9.        Book b4=new Book("三侠五义",11.45,"吴");
10.       Book b5=new Book("象棋 23 式",122.50,"鲍");
11.
12.       Book[] myBooksArr=new Book[]{b1,b2,b3,b4,b5};
13.
14.       Console.WriteLine("\n>>请输入排序规则：1-按书名；2-按价格；3-按作者");
15.       string type=Console.ReadLine();
16.       switch(Convert.ToInt32(type))
17.       {
18.           case 1:
19.               BookList.DisplayByName(myBooksArr);
20.               break;
21.           case 2:
22.               BookList.DisplayByPrice(myBooksArr);
23.               break;
24.           case 3:

```

```
25.         BookList.DisplayByAuthor(myBooksArr);
26.         break;
27.     default:
28.         break;
29. }
30. }
```

示例使用了 switch 语句，来判断用户的输入，然后根据用户的选择，调用 BookList 不同的方法。

4. 扩展

读者可以对从以下方面这个应用进行进一步的扩展：

- (1) 添加一本新图书到图书列表中；
- (2) 显示图书价格大于参数 min 的所有图书（参考 8.2.2、8.2.3 内容）
- (3) 根据书名、作者，或者价格，查找某本图书（参考 8.2.6）

8.4 温故知新

● 学完本章中，读者需要回答：

1. 什么是数组？
2. .NET 基础类库中的 System.Array 和数组是什么关系？
3. 能够使用数组完成（1）遍历（2）同步（3）排序（4）查找（5）反转（6）复制操作吗？

4. 如何根据不同的情况动态生成数组？

● 在下一章中，读者会了解到：

1. C#中的集合命名空间 System.Collections；
2. 使用 ArrayList 类完成（1）添加元素（2）插入元素（3）删除元素（4）排序元素（5）查找元素（6）遍历列表操作；
3. Queue 类和 Stack 类的作为功能受限的列表，其典型操作的实现。
4. 哈希表的概念，以及使用 Hashtable 和 SortedList 类实现普通的哈希表和排序的哈希表。

8.5 习题

通过下面的习题来检验本章的学习，习题答案参考光盘。

【本章习题答案在光盘中 \源代码\C8\习题】

1. 创建一个学生数组，并从控制台程序中输出这些学生。
2. 有以下几个数值数组，从控制台程序中输出大于 25 的数。