

第 4 章 数组

王忠民

内容简介

试想一下，如果用户要录入本次考试班级全班前 20 名同学的成绩并计算平均分数，那么就需要声明 20 个变量来存储成绩，这样岂不是麻烦，有没有更好的办法呢？

当然有：数组。

数组是一个有序的数据集合，它能够更好的存储数据。本章将详细介绍数组的知识，包括一维数组、多维数组、静态数组和动态数组的概念、声明以及用法等等。最后会以两个综合案例结束本章。

本章学习要点

- 了解数组的概念以及数组的分类
- 掌握如何声明和初始化数组
- 掌握访问数组元素和遍历数组的方法
- 熟悉静态数组常用的属性和方法
- 熟练利用 `Sort()` 方法和 `Reverse()` 方法对数组进行排序和反转操作
- 掌握静态数组中复制数组常用的 4 种方法
- 熟悉动态数组常用的属性和方法
- 掌握如何在动态数组中添加、删除和查找元素

4.1 数组概述

数组是一种常见的数据类型，且属于引用类型。在 C# 中，把一组具有同一名字、不同下标的下标变量称为数组。它是一个有序的数据集合，包含若干个相同类型的变量，这些变量都被称为数组元素。使用数组可以在很大程度上简化应用程序的代码。

数组具有以下特性：

- 数组可以是一维数组、二维数组或多维数组
- 数值数组元素的默认值为 0，而引用元素的默认值为 null
- 交错数组是数组的数组，因此，它的元素是引用类型，初始化为 null。交错数组元素的维度和大小可以不同
- 数组的索引从 0 开始，如果数组有 n 个元素，那么数组的索引是从 0 到 n-1
- 数组元素可以是任何类型，包括数组类型

4.2 一维数组

一维数组是指维度数为 1 的数组，又称作**简单数组**，它是数组最简单的形式，也是最常用的数组。这一节我们就学习如何声明和初始化一维数组。

4.2.1 声明数组

4.2.2 初始化数组

4.2.1 声明数组

要使用一个数组就必须先声明这个数组。声明一维数组的语法：

```
type[] arrayName;
```

其中 type 表示数组元素的类型，如 int、double、string 或 object 等；arrayName 表示数组的名称。

例：声明两个数组。

```
int[] intArray;  
string[] stringArray;
```

声明数组后并没有实际创建它们，还不能使用（如显示、运算等）。

在 C# 中数组是对象，必须进行实例化，以创建数组：

一维数组：

```
int[] numbers = new int[50];
```

```
string[] stringArray= new string[7];
```

多维数组：

```
string[,] names = new string[5,4];
```

这样，数组成员将自动具有该数组类型的默认初始值，是可用的了。

double ， float ， int ， long 等默认初值： 0 。

bool ： false ， char ： '\0' ， string ： null

4.2.2 初始化数组

数组在使用前必须初始化或实例化，为数组中的每一个元素进行赋值。声明数组的同时也可以初始化数组。在 C# 中提供了 3 种初始化数组的方法

```
type[] arrayName= new type[num]{value1,value2,value3,...valueN};  
type[] arrayName= new type[] {value1,value2,value3,...valueN};  
type[] arrayName= {value1,value2,value3,...valueN};
```

其中 type 表示数组的类型；arrayName 表示数组的名称，num 表示声明数组的长度；value1、value2 和 value3 等表示数组初始值列表。

```
int[] intArray3 = new int[5] { 11, 22, 33, 44, 55 };  
int[] intArray2 = new int[] { 31, 64, 97 };  
int[] intArray1 = { 95, 83, 68, 88 };
```

第一种方法，初始值列表数量必须等于声明的数组长度。

第二或第三种方法，数组长度由初始值列表的数量决定。

- 数组从零开始建立索引，即数组索引从零开始。
- 声明数组时，方括号 [] 必须跟在类型后面，而不是标识符后面。在 C# 中，将方括号放在标识符后是不合法的语法。
- 数组的大小不是其类型的一部分（而在 C 语言中它却是数组类型的一部分）。可以先声明一个数组，再指定数组长度。例：

```
int[] numbers;    // declare numbers as an int array of any size
numbers = new int[10]; // numbers is a 10-element array
numbers = new int[20]; // now it's a 20-element array
```

使用数组元素：

```
int [] num = { 12,34,67, 89};
```

```
num[0]= 33;
```

```
int i = 2 ;
```

```
num[i] = num[0] + 100 ;
```

4.3 二维数组

在某些场合一维数组已经不能满足应用的需要了。如利用数组存储若干个学生不同科目的成绩，这时候就需要使用二维数组。

二维数组是指维度数为 2 的数组，也叫矩形数组。在 C# 中，它也是比较常用的数组。上一节我们学习了如何声明和初始化一维数组，本节我们就来介绍如何声明和初始化二维数组。

	学号	数学	外语	物理
甲	1001	76	93	68
乙	1002	88	97	65
丙	1109	93	86	72

4.3.1

声明数组

4.3.2

初始化数组

4.3.1 声明二维数组

声明二维数组的语法如下：

```
type[,] arrayName;
```

其中 type 表示数组元素的类型，如 int、double、string 或 object 等；中括号 ([]) 中间使用一个逗号隔开，表示该数组为一个二维数组；arrayName 表示数组的名称。

例如声明两个二维数组：一个 int 类型的二维数组 numbers，不需要指明长度；另一个为 string 类型的二维数组 stuscore，指定其长度为 3。编写的代码如下：

```
double [,] numbers;  
int[,] stuscore = new int[3,4]; // 见下表
```

	学号	数学	外语	物理
甲	1001	76	93	68
乙	1002	88	97	65
丙	1109	93	86	72

4.3.2 初始化二维数组

在 C# 中提供了 3 种初始化二维数组的方法，语法如下：

```
type[,] arrayName = new type[va1length,va2length] { {value1,value2},{value3,value4},{value5,value6}
```

```
type[,] arrayName = new type[,] {{value1,value2,value3},{value4,value5,value6}};
```

```
type[,] arrayName = {{value1,value2,value3},{value4,value5,value6}};
```

		学号	数学	外语	物理
va1length	甲	1001	76	93	68
	乙	1002	88	97	65
	丙	2136	93	86	72
		va2length			

```
static void Main(string[] args)
{

    int[,] numbers = new int[2, 3] { { 23, 44, 53 }, { 1, 100, 4 } }; //2 行 3 列
    double[,] stuscore = new double[,] { { 98, 100, 65.5 }, { 75.8, 99.5, 90 },
        { 55, 100, 64 }, { 55.5, 88.5, 99.5 } }; //4 行 3 列
    string[,] bookname = { { "家", "春", "秋" }, { "雾", "雨", "电" },
        { "背影", "父亲", "童年" } }; //3 行 3 列
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
            Console.Write( bookname[i, j] );
        Console.WriteLine();
    }
    Console.Read();
}
```

4.4 多维数组

多维数组是指维度数大于 2 的数组，如三维、四维数组等。声明多维数组的语法和二维数组相似。

```
type[,...] arrayName;
```

在 C# 中也提供了 3 种初始化多维数组的方法，和二维数组一样。

```
int[,...] numsllist = new int[2, 5, 2]
{
    { { 2, 3 }, { 4, 50 }, { 1, 10 }, { 44, 53 }, { 11, 22 } },
    { { 2, 3 }, { 4, 50 }, { 1, 10 }, { 44, 53 }, { 11, 22 } }
};
```

景点

美食

天安门，长城，鸟巢	烤鸭，炸酱面，驴打滚
外滩，东方明珠，豫园	小笼包，千张，小馄饨
武侯祠，草堂，春熙路	担担面，钟水饺，麻婆豆腐
珠江，白云山，小蛮腰	虾饺，云吞面，肠粉

如何声明（类型、维度、长度）？


```

static void Main ( string [] args)
{
    string[, ] cities = new string[4, 2, 3]
        { { " 天安门 "," 长城 "," 鸟巢 "},{ " 烤鸭 "," 炸酱面 "," 驴打滚 "}},
          { { " 外滩 "," 东方明珠 "," 豫园 "},{ " 小笼包 "," 千张 "," 小馄饨 "}},
          { { " 武侯祠 "," 草堂 "," 春熙路 "},{ " 担担面 "," 钟水饺 "," 麻婆豆
腐 "}},
          { { " 珠江 "," 白云山 "," 小蛮腰 "},{ " 虾饺 "," 云吞面 "," 肠粉 "}}
        };
    foreach (string s in cities)
        Console.WriteLine(s);

    Console.ReadLine();
}

```

交错数组

是不规则的二维数组。行数固定，但每行的长度不固定。

又称为锯齿数组、数组的数组和不规则数组。

定义的语法：

```
type[][] arrayName;
```

例：各系的班级列表

物联系	物联 1	物联 2		
计算机系	计算机 1	计算机 2	计算机 3	计算机 4
电信系	电信 1	电信 2	电信 3	

交错数组的初始化：

设置第一个括号中该数组包含的行数，但第二个括号为空，因为每一行包含不同个数的元素。

```
static void Main(string[] args)
{
    string[ ][ ] arr = new string[3][ ]
        { new string [ ] {" 计算机 1 班 "," 计算机 2 班 "," 计算机 3
班 "},
          new string [2] {" 电信 1 班 "," 电信 2 班 "},
          new string [ ] {" 英语 1 班 "," 英语 2 班 "," 英语 3 班 "}
        };
    for (int i = 0; i < arr.Length;i++ )
    {
        for (int j = 0; j < arr[i].Length; j++)
            Console.Write(arr[i][j] + "__");
        Console.WriteLine();
    }
    Console.Read();
}
```

4.5 数组操作

4.5.1 访问数组元素

4.5.2 使用 foreach 语句遍历数
组

4.5.3 对数组排序

4.5.1 访问数组元素

C# 中访问数组元素，通过它的下标即可访问，通常我们把下标叫做索引，它的索引从 0 开始。

以二维数组为例，在应用程序中声明并初始化二维数组 bookname，然后输出二维数组中行数索引为 1，列数索引为 2 的元素内容。代码如下：

```
string[,] bookname = {  
    { "巴金三部曲：", "家", "春", "秋" },  
    { "老舍三部曲：", "骆驼祥子", "四世同堂", "茶馆" }  
};  
Console.WriteLine(bookname[1, 2]);  
Console.ReadLine();
```

上述代码中直接使用索引 bookname[1,2] 获得第 2 行第 3 列元素的值。运行上述代码，控制台输出的结果为：四世同堂。

4.5.2 使用 foreach 语句遍历数组

遍历数组是指依次访问数组中的每一个元素，C# 中遍历数组有两种方式：
1、使用索引（或下标）遍历，如 for、while、do...while 等语句；
2、不使用索引遍历，使用 foreach 等。

以一维数组为例，在应用程序中声明并初始化数组 `authorname`，然后使用 `foreach` 语句遍历数组中元素的值，编写的代码如下：

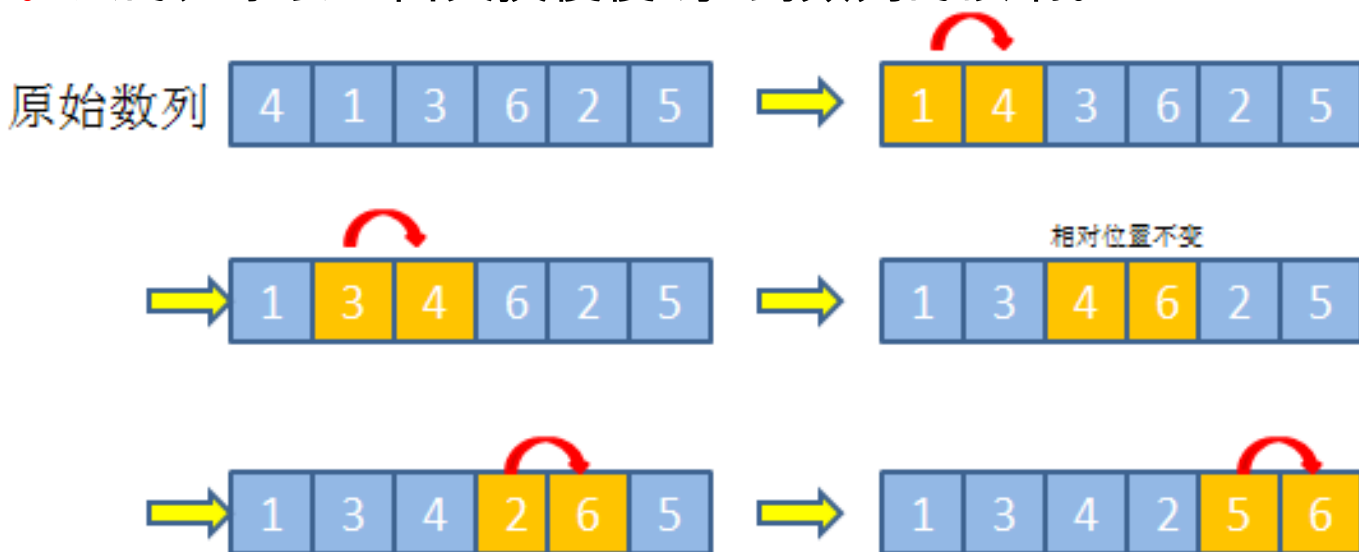
```
string[] authorname = new string[5] { "冰心", "金庸", "巴金", "老舍", "朱自清" };  
Console.WriteLine(" 循环输出的结果如下 : ");  
foreach (string item in authorname)  
{  
    Console.WriteLine(item);  
}  
Console.ReadLine();
```

4.5.3 对数组进行排序

生活中遇到的情况：刚入新班级的体育课上，老师总是会让大家站在一排，然后比较同学的个子交换位置，个子低的同学站在最前面。

程序实现：将每个同学的身高存储到数组中，用程序实现数组排序。

冒泡排序 (Bubble Sort)：从索引为 0 的元素开始，比较相邻的两个数组元素，每次比较完毕后最大的一个元素跑到本轮的末尾，实现从小到大的排序。大的元素会经由交换慢慢“浮”到数列的顶端。



假设存在数组：65,81,35,100,87,67,55。

(1) 第一轮首先比较两两相邻的元素，如果左边元素大于右边元素，则交换位置。65 和 81 比较的结果就是：65 在前，81 在后；然后 81 和 35 比较的结果就是：35 在前，81 在后。以此类推，第一轮比较后的结果是：65,35,81,87,67,55,100。

(2) 经过第一轮的比较，最大的元素跑到了最后一个。所以第二轮比较，最后一个元素就不需要进行比较了。

(3) 第二轮还是从索引 0 和 1 开始比较，比较方法和前两步相同。第二轮比较后的结果是：35,65,81,67,55,87,100。

(4) 第三轮和第四轮等以此类推。排序后的结果就是：35,55,65,67,81,87,100。

例：老师要录入总分排名前 10 名同学的数学成绩，并且进行排序。编写的代码如下：


```
int[] score = new int[10];           // 声明数组，存放 10 名同学的成绩
int i, j, temp;                       // 变量 i，j 用于循环数据，temp 变量交换
for (i = 0; i < score.Length; i++)    // 循环录入 10 名同学的数学成绩
{
    Console.WriteLine(" 请输入第 {0} 名学生的数学成绩：", i + 1);
    score[i] = int.Parse(Console.ReadLine());
}
for (i = 0; i < score.Length - 1; i++) // 开始排序，需交换 score.Length-1 次
{
    for (j = 0; j < score.Length - 1 - i; j++)
    {
        if (score[j] > score[j + 1])
        {
            temp = score[j];
            score[j] = score[j + 1];
            score[j + 1] = temp;
        }
    }
}
Console.WriteLine(" 对成绩进行排序，从小到大依次为：");
foreach (int stu in score) // 重新输出排序后的结果
{
    Console.Write(stu + "\t");
}
Console.ReadLine();
```

上面程序是按升序排列。如何按降序排列？

4.6 静态数组

.NET 框架提供了两种数组：静态数组和动态数组。静态数组是指从建立数组开始到运行结束时，数组的维度和大小是不能更改的。静态数组由 System.Array 类实现，此前所介绍的数组都属于静态数组。

4.6.1 属性

4.6.2 方法

4.6.3 获取数组长度和数组元素的值

4.6.4 数组排序

4.6.5 复制数组

4.6.1 属性

System.Array 类包含了 7 个属性，常用的是：

属性	描述
Length	数组的长度，即数组所有维度中元素的总数。为 32 位整数
LongLength	数组的长度，即数组所有维度中元素的总数。为 64 位整数
Rank	数组的秩，即数组的维度数

我们经常使用 Length 获得数组的长度，如果数组的长度大于 32 位，则只能通过 LongLength 属性获得数组长度。

如果是二维数组或多维数组，可以使用 Rank 属性获得数组的维度数。

4.6.2 方法

System.Array 包含多个方法：

方法	描述
GetValue()	获取指定元素的值
SetValue()	设置指定元素的值。
Clear()	清除数组中的所有元素。
IndexOf()	获取匹配的元素的索引
LastIndexOf()	获取匹配的最后一个元素的索引
Sort()	对一维数组中的元素排序
Reverse()	反转一维数组中元素的顺序
GetLength()	获取指定维度数组的元素数量。该值为 32 位整数
GetLongLength()	获取指定维度数组的元素数量。该值为 64 位整数
FindIndex()	搜索指定元素，并获取第一个匹配元素的索引
FindLastIndex()	搜索指定元素，并获取最后一个匹配元素的索引
Copy()	将一个数组中的一部分元素复制到另一个数组
CopyTo()	将一维数组中的所有元素复制到另外一个一维数组
Clone()	复制数组
GetUpperBound()	获取数组中指定维度的上限

- ◆ **GetValue**(int index) 返回一维数组第 index 个元素的值；
- ◆ **GetValue**(int idx1, int idx2) 获取 2 维数组的某个元素的值；
- ◆ **GetValue**(int idx1, int idx2,int idx3) 3 维数组元素的值；
- ◆ **SetValue** (value , idx1) 或 **SetValue** (value,idx1,idx2) ；
- ◆ **GetLength**(int dim) 返回多维数组第 dim 维的元素的个数；
- ◆ **CopyTo**(Array dest, int start) 将调用此方法的数组中元素复制到 dest 中，将 dest 中从 start 位置开始往后的内容覆盖；
- ◆ **Array.Sort**(数组名字)：对指定的数组中的元素排序（升序）；
- ◆ **Array.Reverse**(数组名字)：将指定的数组中的元素顺序反转；
- ◆ **Array.Copy**(Array src, *long idx1*, Array dest, *long idx2*, *long length*)
- ◆ **Array.Clear**(数组名字，起始位置，长度)

```
static void Main(string[] args)
{
    string[] stringArr = new string[] { "good", "nice", "fine" };
    int[,] intArr = { {2, 3, 4},{2,2,2} };

    stringArr.SetValue("perfect", 2);
    foreach (string s in stringArr)
        Console.WriteLine(s);

    intArr.SetValue(999, 0, 1);
    foreach (int i in intArr)
        Console.WriteLine(i);

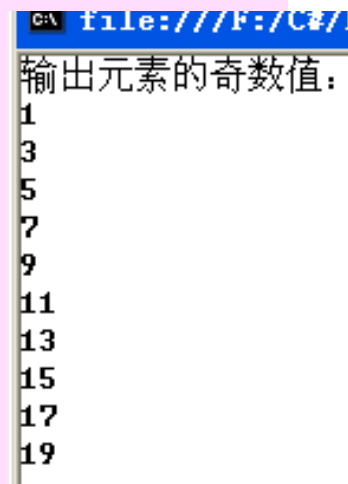
    Console.Read();
}
```

4.6.3 获取数组长度和数组元素的值

例：声明一个长度为 20 的一维数组 stuid。stuid.Length 获得数组长度，SetValue() 方法为数组元素赋值，然后使用 GetValue() 方法循环取出数组元素的值，判断元素的值是否为奇数，如果是则输出元素的值。

```
static void Main(string[] args)
{
    int[] stuid = new int[20];
    for (int i = 0; i < stuid.Length; i++)
        stuid.SetValue(i + 1, i);           // 或 stuid[i]= i+1;

    for (int i = 0; i < stuid.Length; i++)
    {
        int number = (int) stuid.GetValue(i); // object 转成 int
        if (number % 2 == 1) Console.WriteLine(number);
    }
    Console.Read();
}
```



```
C:\ file:///F:/C#/
输出元素的奇数值:
1
3
5
7
9
11
13
15
17
19
```


作业：

对于一个已经建立的数组，利用 Rank、GetLength 和 GetUpperBound，逐行显示出每个维度的元素个数、最大上标值。

4.6.4 数组排序

Array 类的 Sort() 方法实现数组排序的功能。

例：string 类型的一维数组 myHobby 存储个人爱好，循环输出各个元素的值。然后使用 Sort() 方法对数组进行排序（升序），再 Reverse 倒序

```
static void Main(string[] args)
{
    string[] myHobby = new string[4] { "爬山", "游泳", "足球", "唱歌" };

    Console.WriteLine("排序前的数组元素是：");
    foreach (string s in myHobby)
        Console.WriteLine(s);
    Array.Sort(myHobby);
    Console.WriteLine("排序后的元素是：");
    foreach (string s in myHobby)
        Console.WriteLine(s);
    Array.Reverse(myHobby);
    Console.WriteLine("倒序后的元素是：");
    foreach (string s in myHobby)
        Console.WriteLine(s);
    Console.Read();
}
```

4.6.5 复制数组

C# 中复制数组常用 4 种方法：for 语句、Copy()、CopyTo() 以及 Clone()。

1.for 语句复制数组

例：string 类型的二维数组 book 和 copybook，利用 for 语句循环，使用 SetValue 方法将 book 数组中的元素全部复制到 copybook 数组中

```
static void Main(string[] args)
{
    string[,] book = new string[2, 4]
        { {"西游记", "水浒传", "红楼梦", "三国演义"},
          {"荷马史诗", "神曲", "哈姆雷特", "浮士德"} };
    string[,] bookCopy = new string[book.GetLength(0), book.GetLength(1)];
    for (int i = 0; i <= book.GetUpperBound(0); i++)
        for (int j = 0; j <= book.GetUpperBound(1); j++)
            bookCopy.SetValue(book[i, j], i, j);
    foreach (string s in bookCopy) Console.WriteLine(s);
    Console.Read();
}
```

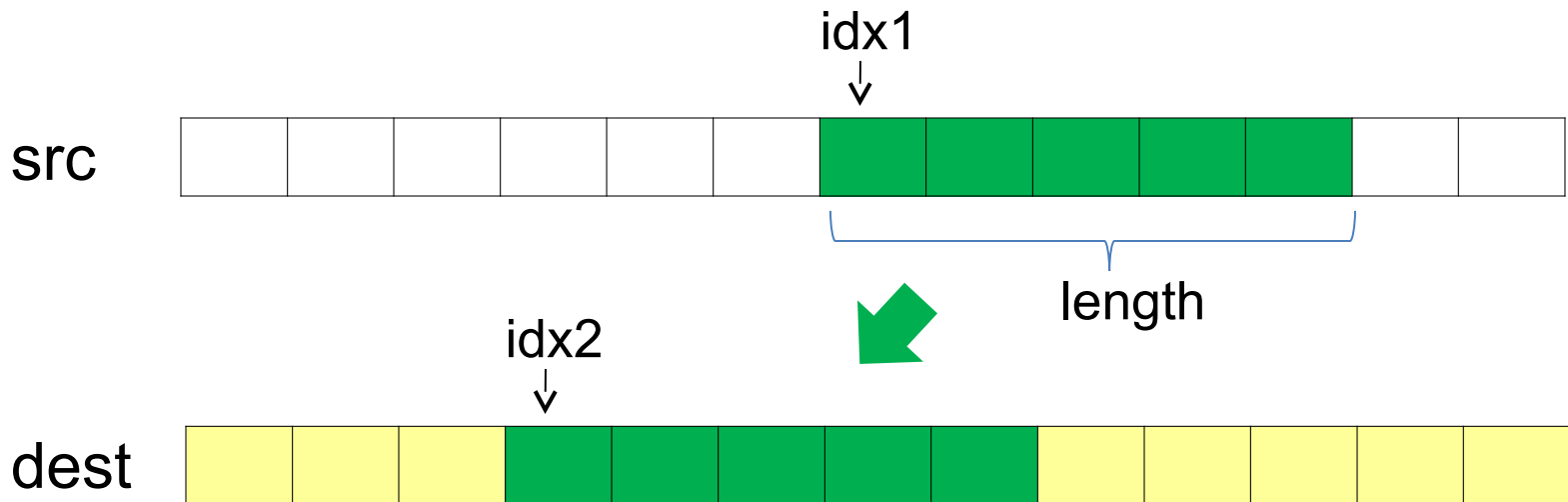
4.6.5 复制数组

2.Copy() 方法复制数组

Copy() 方法用来复制数组，将数组中的部分元素复制到另一个数组。

Array.Copy(**Array** src, *int* idx1, **Array** dest, *int* idx2, *int* length)

参数可以是 5 个，也可以是 3 个（同时省略掉 *idx1* 和 *idx2*，按 0 处理）。



```
static void Main(string[] args)
{
    string[,] book = new string[2, 4]
        { {"西游记", "水浒传", "红楼梦", "三国演义"},
          {"荷马史诗", "神曲", "哈姆雷特", "浮士德"} };
    string[,] bookCopy = new string[book.GetLength(0), book.GetLength(1)];

    Array.Copy(book, bookCopy, book.Length); // 从 0 号索引全部复制

    foreach (string s in bookCopy)
        Console.WriteLine(s);
    Console.Read();
}
```

4.6.5 复制数组

3.CopyTo() 方法复制数组

CopyTo() 方法是将一维数组中的所有元素全部复制到另外一个数组中。
此方法仅仅适用于一维数组。

CopyTo(Array dest, long start) 将调用此方法的数组中的元素从 0 号索引开始全部复制到 dest 中，覆盖 dest 中从 start 位置开始往后的内容。

```
static void Main(string[] args)
{
    int[] arr = new int[] { 160, 180, 165 };
    int[] arrCopy = new int[5];
    arr.CopyTo(arrCopy, 1);
    foreach (int item in arrCopy)
    {
        Console.WriteLine(item + " ");
    }
    Console.Read();
}
```

注意：数
组索引从
0 开始

4.6.5 复制数组

4.Clone () 方法复制数组

Clone() 方法用来复制数组的值，可以是一维数组、二维数组、多维数组。Clone() 方法返回的是一个对象，所以使用时要强制转换成适当的类型。

```
static void Main(string[] args)
{
    int[] arr = new int[] { 160, 180, 165, 170, 155 };
    int[] theCopy = (int[]) arr.Clone();

    foreach (int item in theCopy)
    {
        Console.WriteLine(item );
    }
    Console.Read();
}
```

4.7 动态数组

与静态数组相反的就是**动态数组**，又称为**可变数组**。动态数组是指在**程序执行过程中**可以**改变数组的大小**，或者释放数组所占用的空间。动态数组由 `System.ArrayList` 类实现，本节我们主要介绍动态数组的相关知识，如动态数组的属性、方法以及如何增加元素等内容。

4.7.1 属性

4.7.2 方法

4.7.3 增加元素

4.7.4 删除元素

4.7.1 属性

```
using System.Collections;
```

声明动态数组：

```
ArrayList arrayListName = new ArrayList();
```

System.ArrayList 类包含了 6 个属性，常用的有：

属性	描述
Capacity	数组的容量。可以获取或设置动态数组能够容纳的最大数量，它的值可以动态修改。
Count	数组元素的数量。可以获取动态数组中实际元素的数量。
IsFixedSize	表示数组的大小是否固定
IsReadOnly	表示数组是否为只读

```
using System.Collections;
```

```
//=====
```

```
static void Main(string[] args)
```

```
{
```

```
    ArrayList arr = new ArrayList();
```

```
    Console.WriteLine("Capacity is {0}, Count is {1}",arr.Capacity,arr.Count);
```

```
    Console.WriteLine("IsFixedSize is {0}, IsReadOnly is {1}", arr.IsFixedSize,  
                      arr.IsReadOnly);
```

```
    Console.Read();
```

```
}
```

4.7.2 方法

System.ArrayList 类型的常用方法如下：

方法	描述
Add()	添加元素到数组的末尾
AddRange()	将集中的元素添加到数组的末尾
BinarySearch()	使用二分查找算法搜索已排序的数组中的指定元素
Clear()	清空数组中的所有元素
Contains()	判断数组中是否存在指定的元素
Insert()	插入元素到数组的指定位置
InsertRange()	将集中的某个元素插入到数组的指定位置
Remove()	从数组移除指定的元素
RemoveRange()	从数组中移除一定范围的元素
RemoveAt()	从数组中移除指定位置的元素
GetRange()	获取数组的一个子集
SetRange()	将集中的元素复制到数组中一定范围的元素上
IndexOf()	获取指定元素在数组中的第一个索引位置，从数组的开头处开始查找
LastIndexOf()	获取指定元素在数组中的第一个索引位置，从数组的结尾处开始查找
Sort()	对数组或一部分数组元素进行排序
Reverse()	将数组或其一部分中的元素进行反转
CopyTo()	将数组或其一部分中的元素复制到数组中
ToArray()	将数组的元素复制到新数组中

4.7.3 增加、插入、排序、包含

Add() 和 Insert() 方法都可以向动态数组中添加元素：

- Add() 方法可以将新元素添加到数组末尾处。 arrayName.Add(元素值)
- Insert() 方法可以向指定位置插入数组元素。 arrayName.Insert(int idx, 元素值)

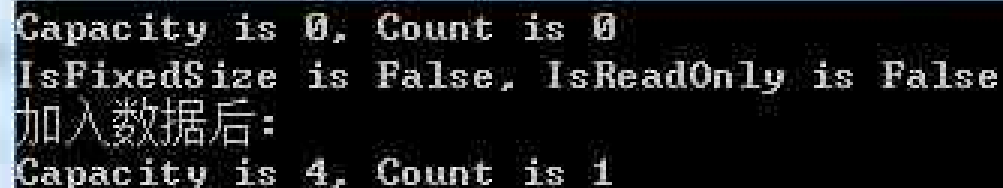
```
static void Main(string[] args)
{
    ArrayList arr = new ArrayList();
    arr.Add(" 欧也妮·葛朗台 "); arr.Add(" 巴黎圣母院 ");
    arr.Insert(0, " 茶花女 "); arr.Insert(2, " 基督山伯爵 ");
    arr.Sort();
    foreach (string item in arr)
        Console.WriteLine(item);
    Console.WriteLine(" 数组中是否包含 \" 茶花女 \"" + arr.Contains(" 茶花女 "));
    Console.WriteLine(" 数组中是否包含 \" 茶花 \"" + arr.Contains(" 茶花 "));

    Console.ReadLine();
}
```

随着动态数组中元素的增加，其 Count 增加，Capacity 也自动增加。

```
static void Main(string[] args)
{
    ArrayList arr = new ArrayList();
    Console.WriteLine("Capacity is {0}, Count is {1}",arr.Capacity,arr.Count);
    arr.Add(" 数据库原理 ");
    Console.WriteLine(" 加入数据后 :\nCapacity is {0}, Count is {1}",
                      arr.Capacity, arr.Count);

    Console.Read();
}
```



```
Capacity is 0, Count is 0
IsFixedSize is False, IsReadOnly is False
加入数据后:
Capacity is 4, Count is 1
```

AddRange 方法把某个集合的元素值都添加到数组中来。

```
static void Main(string[] args)
{
    ArrayList arr = new ArrayList();
    ArrayList arr2 = new ArrayList();
    for (int i = 0; i < 10; i++)
        arr.Add(i + 10);
    arr2.AddRange(arr);
    foreach (int k in arr2)
        Console.WriteLine(k);

    Console.Read();
}
```

4.7.4 删除元素

可以使用 Remove()、RemoveAt() 和 RemoveRange() 方法删除数组元素。

(1)、 **Remove()** 从数组中移除第一个找到的元素值。

arrayName.Remove(要删除的元素值) ;

~~而删除的第二个元素则不会被删除~~

```
static void Main(string[] args)
{
    ArrayList arr = new ArrayList();
    arr.Add(12); arr.Add(34);
    arr.Add(12); arr.Add(34);
    arr.Remove(34); // 删除掉第一个 34
    arr.Remove("12");// 本句无效果

    foreach (int i in arr)
        Console.WriteLine(i);
    Console.ReadLine();
}
```

4.7.4 删除元素

(2)、 **RemoveAt()** 从数组中移除指定位置的元素

arrayName.RemoveAt(int idx) ;

idx 是要删除的元素的索引号。

`arr.RemoveAt(1);`

(3)、 **RemoveRange()** 从数组中移除指定范围内的元素。

arrayName.RemoveRange(int start, int count);

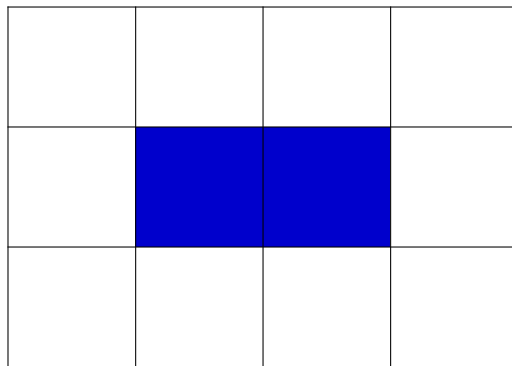
```
static void Main(string[] args)
{
    ArrayList arr = new ArrayList();
    for (int i = 0; i <= 6; i++)
        arr.Add(i * 10);

    arr.RemoveRange(3, 2);
    foreach (int i in arr)
        Console.WriteLine(i);
    Console.ReadLine();
}
```


4.8 求 3*4 矩阵的所有外侧元素的和

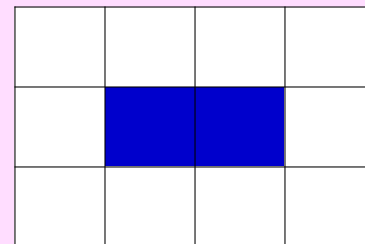
本节我们就综合本章所讲解的内容，实现求 3*4 矩阵的所有外侧元素和的功能。

例：求 3*4 矩阵的外侧元素之和。在应用程序中声明并初始化一个 int 类型的二维数组 number，循环输出二维数组中所有元素的值，if 语句判断是是否为 3*4 矩形的外侧值。



4.8 求 3*4 矩阵的所有外侧元素的和

```
public static void Main(string[] args)
{
    int[,] number = new int[,] { { 1, 2, 3, 4 }, { 5, 6, 7, 8 }, { 9, 10, 11,
12 } };
    int sum = 0;
    for (int i = 0; i < number.GetLength(0); i++) // 外层需要循环的次数
    {
        for (int j = 0; j < number.GetLength(1); j++) // 输出二维度数组的值
        {
            if (i == 0 || i == 2 || j == 0 || j == 3) // 是否为矩形的外侧值
                sum += number[i, j];
            Console.Write("{0}\t", number[i, j]);
            if (j == 3)
                Console.WriteLine();
        }
    }
    Console.WriteLine("3*4 矩形所有外侧的和是 : "+sum);
    Console.ReadLine();
}
```



4.9 输出学生的成绩

例

某班考试结束后老师需要录入学生的成绩。

使用二维数组存储若干个学生不同科目的成绩，实现所需的功能：

- ① 利用一个 int 类型的二维数组 stuscore 存储 5 名学生的语文成绩和数学成绩。
- ② 提示用户输入学生编号、语文成绩和数学成绩。
- ③ 将录入的成绩显示给用户。
- ④ 将语文不及格的学生学号显示出来，如果不存在语文不及格的学生则显示“语文考试全部通过”。

public static void Main(string[] args)		
{ int[,] stuscore = new int[5, 3]; int i;		
for (i = 0; i < stuscore.GetLength(0); i++) // 输入学生编号、语文和数学成绩		
{ Console.Write(" 请输入第 {0} 位学生的学生编号 : ", i + 1);	1001	83
stuscore.SetValue(Convert.ToInt32(Console.ReadLine()), i, 0);	1012	78
Console.Write(" 请输入此学生的语文成绩 : ");	2103	98
stuscore.SetValue(Convert.ToInt32(Console.ReadLine()), i, 1);	2115	87
Console.Write(" 请输入此学生的数学成绩 : ");	2326	56
stuscore.SetValue(Convert.ToInt32(Console.ReadLine()), i, 2);	92	79
Console.WriteLine("*****");	65	81
}	77	
for (i = 0; i < stuscore.GetLength(0); i++) // 输出学生编号、语文和数学成绩		
Console.WriteLine(" 第 {0} 位学生的学生编号 : {1} , 语文成绩 :{2}, 数学成绩 : {3}", i + 1,		
stuscore.GetValue(i, 0), stuscore.GetValue(i, 1), stuscore.GetValue(i, 2));		
Console.WriteLine("*****");		
bool allPass = true; i = 0;		
while (i < stuscore.GetLength(0))		
{ if (stuscore[i, 1] < 60) // 判断学生语文成绩是否及格 , 如果不及格		
{		
Console.WriteLine(" 语文不及格的学号为 : {0}", stuscore[i, 0]);		
allPass= false;		
}		
i++;		
}		
if (allPass == true)		
Console.WriteLine(" 语文考试全部通过 ! ");		
Console.ReadLine();		
}		