第 5 章　集合

王忠民

内容简介

上一章我们主要学习了数组的相关知识，包括一维数组、二维数组和多维数组等，它们都是静态数组。静态数组有一个最大的弊端，就是一旦数组元素完成初始化工作，要在程序中动态添加和删除某个元素是非常困难的。为了解决这个问题。.NET 引入了各种各样的集合对象，如 Hashtable 、 Stack 以及上一章的 ArrayList 集合类等。

本章我们将详细介绍集合的相关知识，包括集合概念、 C# 中的内置集合以及如何自定义集合等内容。

本章学习要点

- 了解集合的概念和 ICollection 接口的成员
- 掌握 System.Collections 命名空间下常用的集合类
- 掌握 ArrayList 集合中添加项、删除项以及访问项等常用的操作方法
- 掌握 Hashtable 集合的概念以及操作方法
- 熟悉 Stack 集合的概念以及操作方法
- 了解 BitArray 集合的概念以及操作方法
- 掌握 SortedList 集合的概念以及操作方法
- 掌握不同集合中遍历所有元素的方法

5.1 集合概述

集合好比容器，将一系列相似的组合一起，就如同数组，用来存储和管理一组特定类型的数据对象。除了基本的数据处理功能，还直接提供了各种数据结构及算法的实现，如队列、哈希表、排序等，可以让用户轻易地完成复杂的数据操作。

在 C# 中所有集合都可以使用 GetEnumerator() 方法返回一个枚举数，遍历集合中的内容。

C# 提供的集合都在命名空间 System.Collections 中。

5.1 集合概述

集合	说明
ArrayList	可以动态增加数组和删除数组等
Hashtable	哈希表，表示键 / 值对的集合，这些键 / 值对根据键的哈希代码进行组织
Stack	堆栈，表示对象的后进先出集合
Queue	队列，表示对象的先进先出集合
BitArray	布尔集合类，管理位值的压缩数组， true 表示位是打开的（ 1 ）， false 表示位是关闭的（ 0 ）
SortedList	排序集合类，表示键 / 值对的集合，这些键值对按键排序并且可以按键和索引访问

5.2 C# 内置集合

这一节我们主要学习 C# 中 6 种内置集合的相关知识。

- 5.2.1 ArrayList 集合
- 5.2.2 Hashtable 集合
- 5.2.3 Stack 集合
- 5.2.4 Queue 集合
- 5.2.5 BitArray 集合
- 5.2.6 SortedList 集合

5.2.1 ArrayList 集合

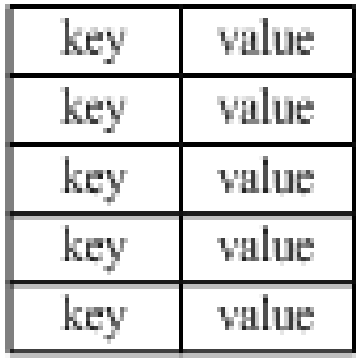
ArrayList 集合即动态数组，主要针对数组中的元素进行处理，

上一章已经介绍过它的用法，本节不再重复。

5.2.2 Hashtable 集合

Hashtable 是一种键 / 值对集合，它的数据是通过键和值来组织的。还可以被称为散列表或哈希表。

在 Hashtable 集合中每一个元素都是一个键 / 值对，而且是一一对应的关系， Hashtable 集合中不能包含重复的 key 值，通过键（ key ）就可以找到相应的值（ value ）。如果使用 Add() 方法添加一个已经存在的 key 值，程序就会抛出异常。



下表列出了 Hashtable 常用的方法。

方法	说明
Add()	将指定的键和值添加到集合。方法有两个参数：第一个参数表示键，第二个参数表示值
Remove()	根据键 (key) 删除指定的元素
Clear()	清除集合中的所有元素
ContainsKey()	判断集合中是否包含特定的 key 值
ContainsValue()	判断集合中是否包含特定的 Value 值

属性： Count 获取包含在 Hashtable 中的键 / 值对的数目。

- 使用 Hashtable 时必须引用 System.Collections 命名空间。
- 根据 key 键访问 value 值： hashTableName[key]
- 不能通过索引值访问其中元素，所以不能通过 for 语句进行遍历。
- 单个访问或遍历时，必须注意数据类型及其转换。

```
using System.Collections;
public static void Main()
  {  Hashtable  hs = new  Hashtable();
     hs.Add(001, " 周杰伦 ");
     hs.Add(002, " 汪峰 ");
     hs.Add( "003“ , " 林俊杰 ");
     Console.WriteLine(hs[02]);
     Console.WriteLine(hs[003]);
     Console.WriteLine(hs["003"]);
     hs.Remove(003);
     Console.WriteLine("Revome(003) 以后： "+hs["003"]);
     hs.Remove("003");
     Console.WriteLine("Revome(\"003\") 以后： " + hs["003"]);
     Console.ReadLine();       }
```

□可采用 foreach 语句，遍历哈希表元素。 例：

①foreach (int i in table1.keys)

②foreach (string s in table1.values)

```
public static void Main()
  {   Hashtable hstl = new Hashtable();
      hstl.Add(103, " 周杰伦 ");
      hstl.Add(102, " 汪峰 ");
      hstl.Add(101, " 林俊杰 ");
      foreach (string s in hstl.Values)
          Console.WriteLine(s);
      foreach (int i in hstl.Keys)
          Console.WriteLine("{0},{1}", i, hstl[i]);

      Console.ReadLine();
  }
```

③foreach (DictionaryEntry item in table1)

```
 public static void Main()
{  Hashtable  hstl = new  Hashtable();
  hstl.Add(103, " 周杰伦 ");
  hstl.Add(102, " 汪峰 ");
  hstl.Add(101, " 林俊杰 ");
  foreach (DictionaryEntry  dr  in  hstl)   // 通过 DictionaryEntry 遍历 hashtable

   {     int   key  =  (int)  dr.Key;          //  必须将 object 类型显式转换为 int
      string  value = (string) dr.Value;   // //  必须显式转换
      Console.WriteLine(" 编号为 "+key+"  的号码为 :  "+value);
   }

  if (hstl.ContainsKey(103))
      Console.WriteLine(" 存在 key 为 103 的记录 ");
  if (hstl.ContainsValue(" 林俊杰 "))
     Console.WriteLine(" 存在 Value 为 \" 林俊杰 \" 的记录 ");

  Console.ReadLine();
}
```

5.2.3 Stack 集合

Stack（堆栈）集合用于实现一个后进先出（Last In First Out，LIFO）的机制，最后进去的元素最先离开集合，第一个元素最后离开集合。

方法	说明
Push()	将指定元素插入到 Stack 集合的顶部
Pop()	返回并删除 Stack 集合顶部的元素
Peek()	返回 Stack 集合顶部的元素，但不删除该元素
Clear()	从 Stack 中移除所有对象
Contains(object obj)	确定某元素是否在 Stack 中

Count 属性是 Stack 集合中最常用的属性，用来存储集合的实际元素数（容量）。向 Stack 添加元素时，将通过重新分配来根据需要自动增大容量。

使用 Push 、 Pop 、 for 、 foreach 访问堆栈

```
using System.Collections;
public static void Main()
    {   Stack  myStack = new  Stack();
        myStack.Push(" 一楼 ");    myStack.Push(" 二楼 ");
        myStack.Push(" 三楼 ");    myStack.Push(" 四楼 ");
        foreach (Object  obj   in  myStack)
            Console.WriteLine(obj);
        Console.WriteLine("============");
        string  s = (string)myStack.Pop();
        myStack.Push("hello");

        for (int  i = 0;  i < myStack.Count  ;  i++ )
        {
            s = (string)myStack.Pop();
            Console.WriteLine(s);
        }
        Console.ReadLine();
    }
```

四楼
三楼
二楼
一楼

结果？？？

利用 IEnumerator 访问堆栈中的元素

IEnumerator　枚举数可用于读取各种集合中的数据。
最初枚举数被定位于集合中第一个元素的前面，在创建枚举数之后或在调用 Reset 方法之后，必须先调用 MoveNext 方法使枚举数前进到集合的第一个元素，然后才能读取 Current 的值；否则 Current 未定义。

```
public static void Main()
{  Stack  myStack = new  Stack();
   myStack.Push(" 一楼 ");
   myStack.Push(" 二楼 ");
   myStack.Push(" 三楼 ");
   IEnumerator  items = myStack.GetEnumerator();
   for (int  i = 1; i <= myStack.Count;i++ )
    {
       items.MoveNext();
       Console.WriteLine(items.Current);
    }
   Console.ReadLine();
}
```

GetEnumerator()　**可以访问各种集合**

GetEnumerator()　// 静态数组、堆栈、队列、位数组等

GetEnumerator(int startindex , int count)　// 动态数组 ArrayList

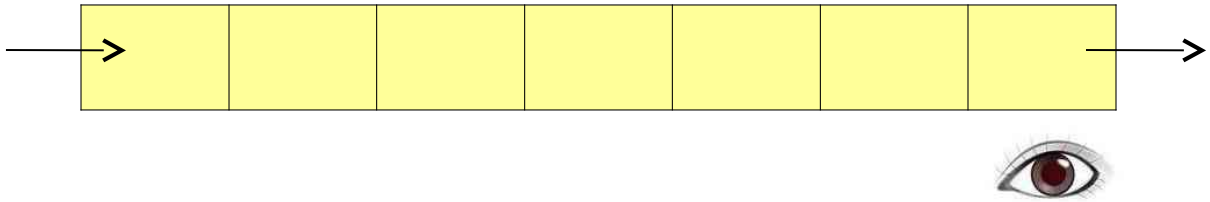
MoveNext()　返回 bool 型，下移是否成功

```
public static void Main()
  {
     ArrayList  arr = new  ArrayList();
     for (int  i = 0; i < 8; i++)
         arr.Add(i*10);
     IEnumerator  items = arr.GetEnumerator(3, 3);  // 从索引 3 号开始取 3 个元素
     while (items.MoveNext())
          Console.WriteLine(items.Current);
     Console.ReadLine();
  }
```

5.2.4 Queue 集合

Queue（队列）集合用于实现一个先进先出（ First In First Out ， LIFO ）的机制。例如，排队买票，站在队前的人先拿到票离开；游玩时坐摩天轮，也是排在队列前的人先坐。 Queue 集合常用的方法：

方法	说明
Enqueue()	将对象添加到 Queue 集合的结尾处
Dequeue()	移除并返回位于 Queue 集合开始处的对象
Peek()	返回位于 Queue 集合开始处的对象但不将其移除
Clear()	从 Queue 中移除所有对象。



属性： Count ，获取 Queue 中包含的元素数。

using System.Collections;

```
public static void Main()
{
Queue  myQ = new  Queue();
myQ.Enqueue("The");
myQ.Enqueue("quick");
myQ.Enqueue("brown");
myQ.Enqueue("fox");
Console.WriteLine("Count: {0}", myQ.Count);
Console.WriteLine(myQ.Peek());
Console.WriteLine(myQ.Dequeue());
Console.WriteLine("==========");
foreach (object   obj   in  myQ)
    Console.WriteLine( Convert.ToString(obj));
Console.ReadLine();
}
```

```
IEnumerator   items = myQ.GetEnumerator();
while (items.MoveNext())
   Console.WriteLine(items.Current);
```

5.2.5 BitArray 集合

BitArray 是布尔集合类，又可以被称为位数组。它的值只有 true 和 false 两个值，其中 true 表示 1 ， false 表示 0 。

方法	说明
And(BitArray)	当前 BitArray 中元素和指定 BitArray 中相应元素按位“与”
Or()	当前 BitArray 中元素和指定 BitArray 中相应元素按位“或”
Xor()	当前 BitArray 中元素和指定 BitArray 中元素按位“异或”
Not()	反转当前 BitArray 中的所有位值
Get(int idx)	获取 BitArray 中特定位置处的位的值
Set(int idx, bool value)	将 BitArray 中特定位置处的位设置为指定值
SetAll (bool value)	将 BitArray 中的所有位设置为指定值

常用属性： Length ，表示该对象包含的元素个数。

```
public static void Main()     // using  System.Collections;
{   BitArray  bitList1 = new  BitArray(10);  // 10 是长度
    BitArray  bitList2 = new  BitArray(10);
    Random  rand= new  Random();
    for (int  i = 0; i < bitList1.Length;i++ )
     { bitList1.Set(i, rand.Next(1, 100) % 2 == 0); //1 到 100 间的随机数
       bitList2.Set(i, rand.Next(1, 100) % 2 == 0);
     }
    ShowIt(bitList1); ShowIt(bitList2);
    bitList1 = bitList1.And(bitList2);
    ShowIt(bitList1);
    Console.WriteLine(bitList1.Get(3) );
    bitList1.SetAll(true); ShowIt(bitList1);
    Console.ReadLine();
}

protected static  void  ShowIt(BitArray  bitlist)
{   IEnumerator  items = bitlist.GetEnumerator();
    while (items.MoveNext())
        Console.Write( items.Current +"\t");
    Console.WriteLine();
}
```

5.2.6 SortedList 集合

SortedList 集合也是表示键 / 值的集合，它兼顾了 ArrayList 和 Hashtable 优点，既可以按照索引访问元素，也可以通过键名访问元素。

键不允许重复，不能为 null ，但值可以。

元素是经过排序存放的，所以又被称作排序列表。每个元素都是一个可作为 DictionaryEntry 对象进行访问的键 / 值对。

index	Key	value
0	k1	v1
1	k2	v2
2	k3	v3
3	k4	v4
4	k5	v5

SortedList 常用方法：

方法	说明
Add(key,value)	将指定的键和值的元素添加到集合中
IndexOfKey()	返回集合中指定键的索引
IndexOfValue()	返回指定的值在集合中第一个匹配项的索引
SetByIndex()	替换集合中指定索引处的值
GetByIndex()	获取集合中指定索引处的值
GetKey()	获取集合中指定索引处的键
GetKeyList()	获取集合中的键
GetValueList()	获取集合中的值
Remove()	从集合中移除带有指定键的元素
RemoveAt()	移除集合中的指定索引处的元素

```
public static void Main()     //  using System.Collections;
{   SortedList  list = new  SortedList();
    list.Add(" 豫 C", " 洛阳 ");list.Add(" 豫 A", " 郑州 ");
    list.Add(" 豫 D", " 平顶山 ");list.Add(" 豫 B", " 开封 ");
    list.Add(" 豫 XX", " 开封 ");
    ShowIt(list);
    Console.WriteLine(" 键 {0} 的索引号是 {1}", " 豫 C", list.IndexOfKey(" 豫 C"));
    Console.WriteLine(" 值 {0} 的索引号是 {1}", " 开封 ", list.IndexOfValue(" 开
封 "));
    Console.WriteLine(" 索引号 {0} 的键是 {1}", 4, list.GetKey(4));
    Console.WriteLine("------ Remove, SetByIndex :");
    list.Remove(" 豫 XX"); list.SetByIndex(0, " 省会 ");
    ShowIt(list);
    Console.ReadLine();
}

protected  static  void  ShowIt  ( SortedList   list )
{    IDictionaryEnumerator   items  =  list.GetEnumerator();
    while (items.MoveNext())
        Console.WriteLine(items.Key+","+items.Value);
}
```