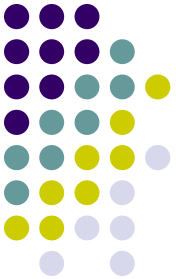
C#程序设计



第4章 数组

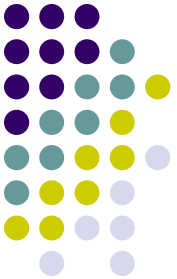
杨琦

西安交通大学

计算机教学实验中心

http://ctec.xjtu.edu.cn

授课内容



- 4.1 数组
- 4.2 程序设计举例
- 4.3 Array类
- 4.4 foreach语句
- 4.5 常用集合类

4.1、数组

迄今为止，我们使用的都是属于基本类型(整型、字符型、实型)的数据，c语言还提供了构造类型的数据，它们有：数组类型、结构体类型、共用体类型。
构造类型数据是由基本类型数据按一定规则组成的，因此有的书称它们为“导出类型”。

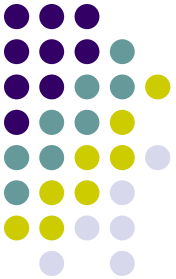
- 问题的引出：
- 实际应用的程序设计中，只用几个变量的情况是极少的；更多的情况是处理大批量的相同类型或不同类型的数据。
- 相同类型数据举例：统计交大15，000学生英语4级统考成绩；
- 不同类型数据举例：管理交大15，000学生学籍信息记录，包括：姓名、学号、出生日期、班级、各科成绩等。
- 用什么样的数据结构来描述这类应用更简洁？

1.一维数组

- 常用于处理大批量数据；
- 数据特点：存在内在联系；
- 数组——具有相同数据类型的变量集合；
- 这些变量都有相同名字，但下标不同；
- 称这些变量为数组元素；
- 只有一个下标——一维数组；
- 有两个下标——二维数组。

array[0]	array[1]	array[2]	array[3]	array[4]	array[5]	array[6]	array[7]	array[8]	array[9]

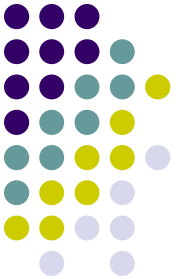
1．一维数组定义与分配



- 数据类型符 [] 数组名 = new 数据类型符[长度];
- 例如：
- int [] a= new int [10];

- 也可以写成下面两条：
- int [] a; //定义数组
- a=new int [10];//给数组分配存储空间

2．数组初始化

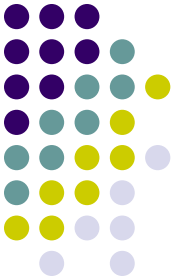


- 数据类型符 [] 数组名 ={初值列表};

- **例如：**
  - int [] a={1,2,3,4};
- **或：**
  - int [] a= new int []{1,2,3,4};

- **3．数组元素的引用 数组名[下标]**
- **例如：**
  - a数组具有4个元素a[0]、a[1]、a[2]、a[3]
  - 通常与循环配合，循环变量对应下标

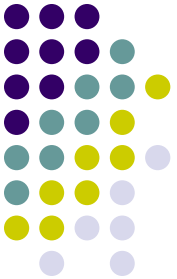
【例4-1】找出数组中的最大数



- 算法分析：
- 1、假设数组中第1个元素最大，令xmax=a[0]
- 2、将a[i]（0<= i < n）与max进行比较，
  - 若a[i] < xmax ， i=i+1，再执行2
  - 否则，令xmax=a[i]，i=i+1，再执行2
- 3、循环结束，求出最大元素并输出max。

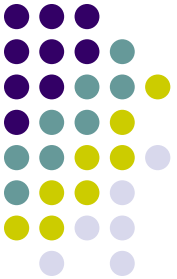
输入	2 1 7 3 12 4 9
输出	max=12

【例4-1】找出数组中的最大数



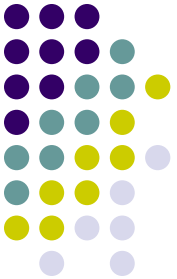
```
1.  using System;
2.  class My
3.  {
4.      static int Main()
5.      {
6.          int[] a = new int[5];
7.          Console.WriteLine("Please input an array: ");
8.          // 输入每个数组元素的值
9.          for (int i = 0; i < 5; i++)
10.             a[i] = Convert.ToInt32(Console.ReadLine());
11.         int big = a[0];
```

【例4-1】找出数组中的最大数



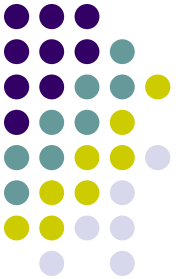
```
1.    for (int j = 1; j < 5; j++)
2.            if (a[j] > big)
3.                big = a[j];
4.        Console.WriteLine("max={0}", big);
5.        return 0;
6.      }
7.   }
```

【例4-2】求斐波那挈数列的前n项



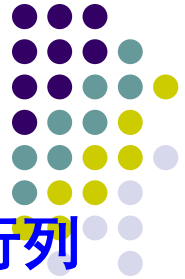
```
1.  using System;
2.  class My {
3.      static int Main()
4.      {
5.          int n;
6.          Console.WriteLine( "Please input n=? ");
7.          n=Convert.ToInt32(Console.ReadLine());
8.          int [] p = new int[n + 1];
9.          p[0] = 0;
10.         p[1] = 1;
```

【例4-2】求斐波那挈数列的前n项



```
1.          for (int i = 1; i <= n; i++)
2.          {
3.              if (i >= 2)
4.                  p[i] = p[i - 2] + p[i - 1];
5.              Console.Write("{0} ", p[i]);
6.          }
7.          Console.WriteLine();
8.          return 0;
9.      }
10. }
```

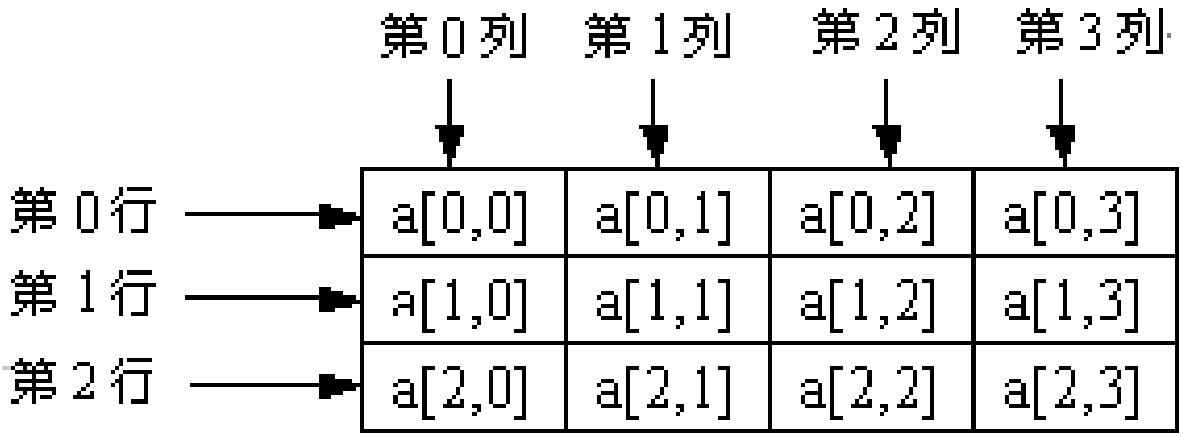
2.二维数组



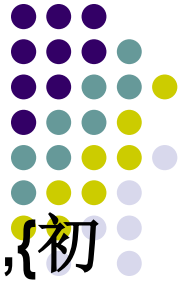
- 有两个下标的数组，适合处理如成绩报告表、矩阵等具有行列结构的数据
- C#的二维数组的每一行的元素个数可以相等，也可以不相等。
- 相等的称为方形二维数组，不同的称为参差数组。

1．方形二维数组

- （1）定义
  - 格式：数据类型符 [,] 数组名 = new 数据类型符[长度1, 长度2];
- 例如：
  - int [,] a= new int [3,4];
- 或：
  - int [,] a;
  - a=new int [3,4];



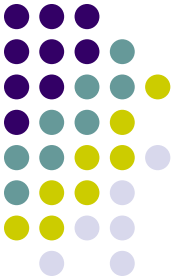
（2）赋初值（初始化）



- 数据类型符 [,] 数组名 = {{初值列表1},{初值列表2},…,{初值列表n}};

- **例如：**
  - int [,]b={{1,2,3,4},{5,6,7,8},
  - {9,10,11,12}};
- **或：**
  - int [,] a= new int[3,4]
  - {{1,2,3,4},{5,6,7,8},
  - {9,10,11,12}};

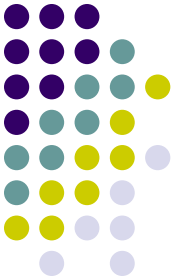
（3）元素引用



- 格式：数组名[下标1,下标2]
- 下标放在一个方括号中

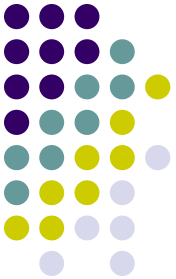
```
static void Main(string[] args)
{ int[,] a = new int[3, 4]{{1,2,3,4},{5,6,7,8},  {9,10,11,12}};
   int i, j;
   for (i = 0; i < 3; i++)
   {  for (j = 0; j < 4; j++)
            Console.Write("{0}\t",a[i,j]);
      Console.WriteLine();
   }
 }
```

【例4-3】将矩阵M置成单位阵



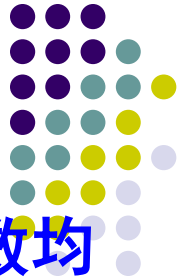
```
1.  using System;
2.  class My{
3.      static int Main()   {
4.          int[,] M = new int[5, 5] ;
5.          int i, j;
6.          // 初始化数组
7.          for (i = 0; i < 5; i++)   {
8.              for (j = 0; j < 5; j++)
9.                  M[i, j] = 0;
10.             M[i, i] = 1;
11.         }
```

【例4-3】将矩阵M置成单位阵



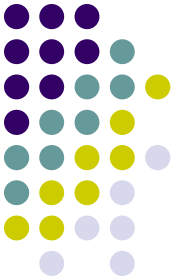
```
1.          // 输出整个数组元素
2.          for (i = 0; i < 5; i++)
3.          {
4.              for (j = 0; j < 5; j++)
5.                  Console.Write(M[i, j] + "\t");
6.              Console.WriteLine();
7.          }
8.          return 0;
9.      }
10. }
```

4.1.5 交错数组



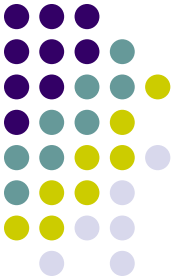
- 二维数组的每一行的长度可以不同，每一行的元素个数均可以由用户指定。
- （1）分配行
  - [格式]：数据类型符 [][] 数组名 = new 数据类型符[行数][];
  - 例如：
  - int [][] b=new int[3][];
  - 定义了一个名为b的参差数组，行数为3

4.1.5 交错数组



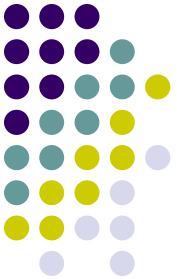
- （2）名行数组元素个数的分配
  - [格式]：数组名[i]=new 数据类型符[长度];
- 例如：
  - int [][] b=new int[3][];
  - b[0]=new int [2];
  - b[1]=new int [3];
  - b[2]=new int [4];
- （3）元素引用
  - 数组名[下标1][下标2]
  - 下标放在每个方括号中

【例4-4】 输出杨辉三角的前9行



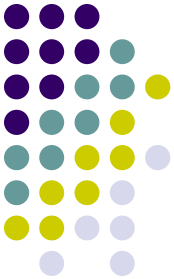
```
1.  static void Main(string[] args)
2.   {
3.      int[][] a = new int[9][];
4.      int i, j;
5.      for(i=0;i<9;i++)    a[i] = new int[i+1];
6.      for (i = 0; i < 9; i++)
7.        { a[i][i] = 1; a[i][0] = 1; }
8.      for (i = 2; i < 9; i++)
9.           for (j = 1; j < i; j++)
10.               a[i][j] = a[i - 1][j] + a[i - 1][j - 1];
```

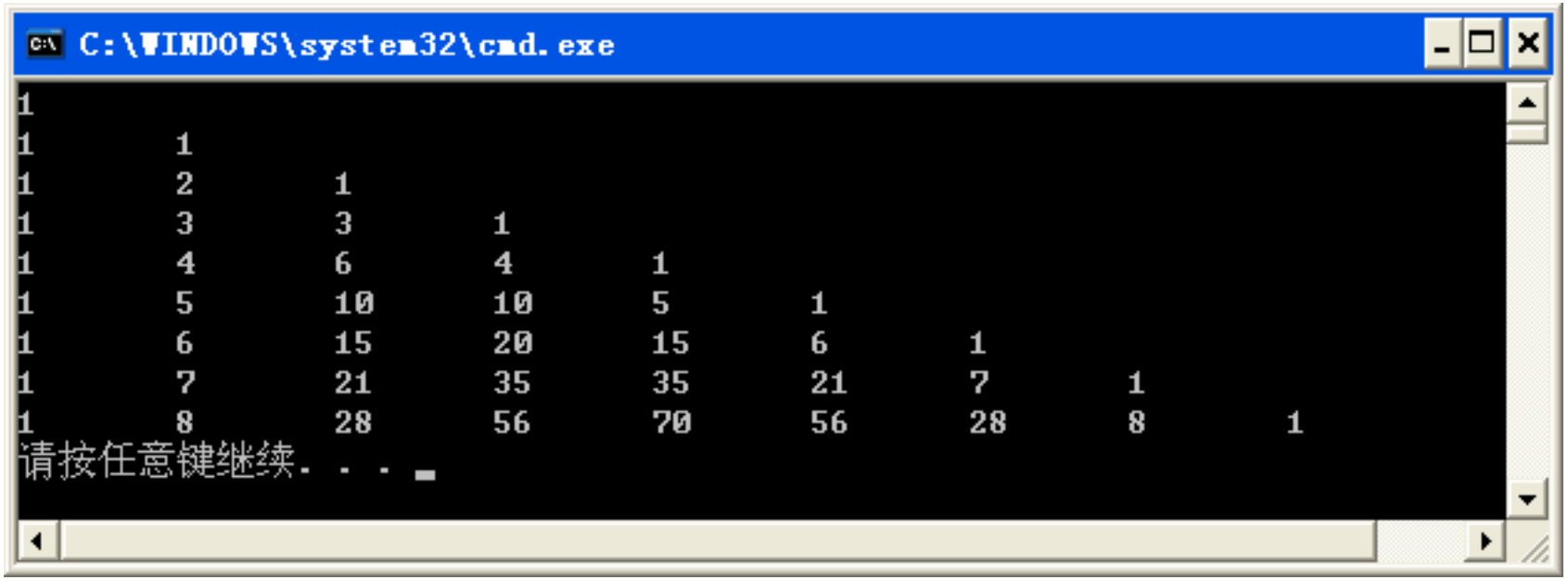
【例4-4】 输出杨辉三角的前9行



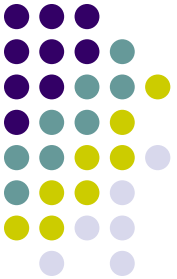
```
1.          for(i=0;i<9;i++)  {
2.              for (j = 0; j <= i; j++)
3.                  Console.Write("{0}\t",a[i][j]);
4.               Console.WriteLine();
5.          }
6.     }
```

【例4-4】 输出杨辉三角的前9行

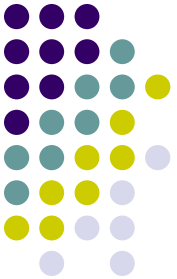




4.2 应用程序举例



【例4-4】计算如下两个矩阵之和



$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} + \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix} \quad =?$$

输入和输出

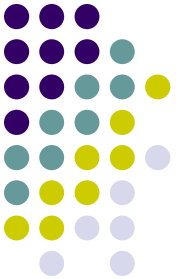
矩阵a和矩阵b的和的矩阵c为:

2 6 10 14

7 11 15 19

12 16 20 24

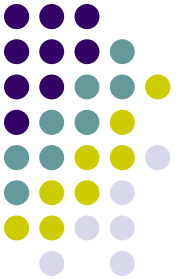
【例4-5】计算如下两个矩阵之和



- 算法说明：
- 矩阵A（MxN）和矩阵B（MxN）相加。
- A是M行、N列；B是M行、N列。

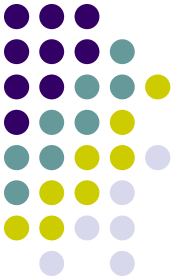
$$C_{ij} = A_{ij} + B_{ij} \quad ,i = 1,2,...,M;\ j = 1,2,...,N$$

【例4-5】计算如下两个矩阵之和



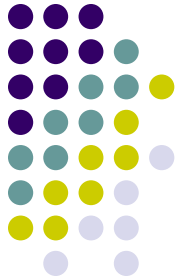
```
1.  using System;
2.  class My{
3.      static int Main()   {
4.          const int M = 3;
5.          const int N = 4;
6.          double[,] a = new double[M, N]          {
7.                  {1, 2, 3, 4},       {5, 6, 7, 8},       {9, 10, 11,12}    };
8.          double[,] b = new double[M, N]          {
9.                  {1, 4, 7,10},       {2, 5, 8, 11},      {3, 6, 9,12}      };
10.         double[,] c = new double[M, N];
11.         Console.WriteLine("矩阵a和矩阵b的和的矩阵c为:");
12.         for (int i = 0; i < M; i = i + 1)
```

【例4-5】计算如下两个矩阵之和



```
1.          {
2.              for (int j = 0; j < N; j = j + 1)
3.              {
4.                  c[i, j] = a[i, j] + b[i, j];
5.                  Console.Write(c[i, j] + "\t");
6.              }
7.              Console.WriteLine();
8.          }
9.          return 0;
10.     }
11. }
```

【例4-6】 计算50！



9 9	9 8									9	8	7	6	5	4	3	2	1	0

0	0	0	0	0	0	0	。	。	。	0	0	0	0	0	0	0	1	2	0

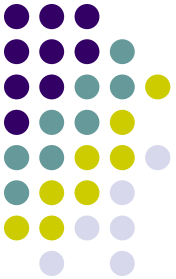
X 6

0	0	0	0	0	0	0	。	。	。	0	0	0	0	0	0	0	7	2	0

sc：进位

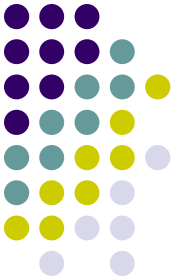
sum：和

【例4-6】计算50！



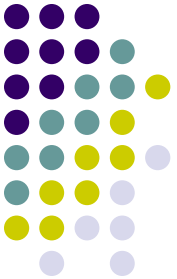
```
1.  using System;
2.  class My{
3.      static int Main()   {
4.          const int MAXSIZE = 100;
5.          int[] array = new int[MAXSIZE];
6.          int n;
7.          Console.WriteLine("n=");
8.          n = Convert.ToInt32(Console.ReadLine());
9.          int sum, sc;
10.         int i, j;
11.         for (i = 0; i < MAXSIZE; i++)
12.             array[i] = 0;
13.         array[0] = 1;
```

【例4-6】计算50！



```
1.    for (i = 2; i <= n; i++)       {
2.            sc = 0;
3.            for (j = 0; j < MAXSIZE; j++)       {
4.               sum = array[j] * i + sc;
5.               sc = sum / 10;
6.               array[j] = sum % 10;
7.            }
8.         }
9.         Console.WriteLine(n + "!=");
10.        for (i = MAXSIZE - 1; i >= 0; i--)
11.           Console.Write(array[i]);
12.        Console.WriteLine();
13.        return 0;
14.    }}
```

C# 字符串（String）



- string 类的属性
- string 类的方法

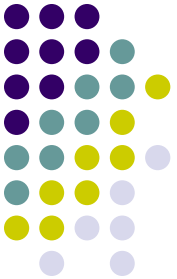
【例4-7】字符串长度

- 编写一个用来计算字符串长度的函数Length，并用主函数验证。

输入和输出

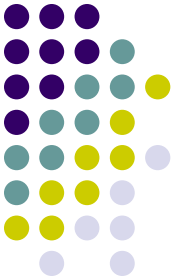
Please input a string (within 99 characters):
xi'an Jiaotong University
The length of the string is：25

【例4-7】字符串长度



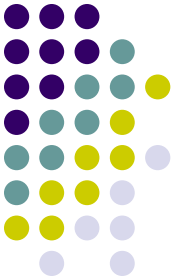
```
1.  using System;
2.  class My {
3.      static int Main()  {
4.          string strName;
5.          Console.WriteLine("Please input a string: ");
6.          strName = Console.ReadLine();
7.          Console.WriteLine("The length of the string is：  "
8.  + strName.Length);
9.          return 0;
10.     }
11. }
```

【例4-8】string类的运算符操作



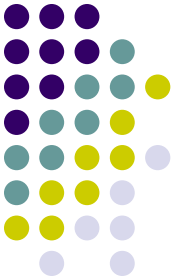
- 输入和输出
- Alpha
- Alpha
- AlphaBeta
- Alpha to Omega
- str3 > str1

【例4-8】string类的运算符操作



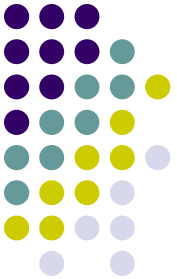
```
1.  string str1 = "Alpha";       string str2 = "Beta";
2.      string str3 = "Omega";       string str4;
3.      str4 = str1;                 // 字符串赋值
4.      Console.WriteLine(str1 + "\n" + str4);
5.      str4 = str1 + str2;          // 字符串连接
6.      Console.WriteLine(str4);
7.      str4 = str1 + " to " + str3;
8.      Console.WriteLine(str4);
9.      // 字符串比较
10.     if (str3.CompareTo(str1) > 0) Console.WriteLine("str3 > str1");
11.     if (str3 == str1 + str2)
12.         Console.WriteLine("str3 == str1+str2");
```

【例4-9】小写转换为大写字母



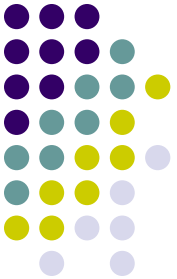
- 输入和输出
- The original string is: This is a sample
- After transform: THIS IS A SAMPLE

【例4-9】小写转换为大写字母



```
1.  using System;
2.  class My {
3.      static int Main()   {
4.          string str="This is a sample";
5.          string dest;
6.          Console.WriteLine( "The original string is: "+str );
7.          dest=str.ToUpper();
8.          Console.WriteLine( "After transform: "+dest );
9.          return 0;
10.     }
11. }
```

【例4-10】编写一个字符串拆分程序



"This is a list of words."

输入和输出

This

is

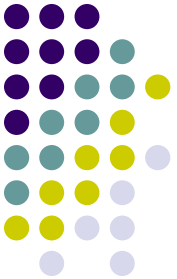
a

list

of

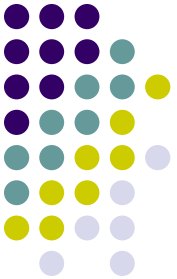
words

【例4-10】编写一个字符串拆分程序

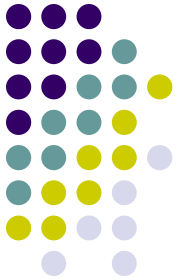


```
1.  using System;
2.  class My{
3.      static int Main()   {
4.          string source = "This is a list of words.";
5.          string[] split=source.Split(new char []{' ',',',',','.'} );
6.          foreach(string s in split)      {
7.              if(s.Trim()!="" )
8.              Console.WriteLine(s);
9.          }
10.         return 0;
11.     }
12. }
```

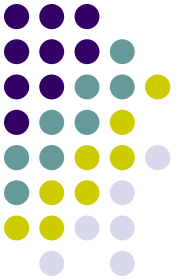
【例4-11】 替换加密(恺撒加密法)



```
1.  using System;
2.  class My{
3.      static int Main()   {
4.          string s;
5.          s = Console.ReadLine();
6.          char[] p = s.ToCharArray();
7.          for (int i = 0; i < p.Length; i++)
8.              if (p[i] >= 'a' && p[i] <= 'z')
9.                  p[i] = Convert.ToChar( (p[i] - 'a' + 3) % 26 + 'a');
10.             else if(p[i] >= 'A' && p[i] <= 'Z')
11.                 p[i] = Convert.ToChar((p[i] - 'A' + 3) % 26 + 'A');
12.         Console.WriteLine(p);
13.         return 0;
14.     }
```

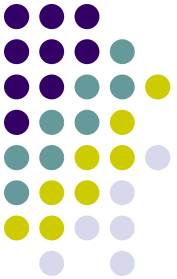


4.3 Array 类



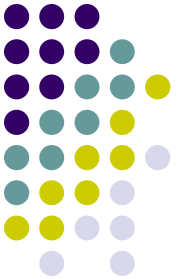
- Array 类的属性
- Array 类的方法

【例4-12】实现字符串的反转



```
1.  using System;
2.  class My{
3.      static int Main()   {
4.          string str1 = "China";
5.          char[] str2 = str1.ToCharArray();
6.          Array.Reverse(str2);
7.          string str3 = new string(str2);
8.          Console.WriteLine("The result is: {0}", str3);
9.          return 0;
10.     }
11. }
```

【例4-13】实现数据元素的排序和翻转

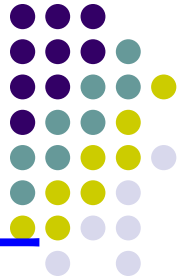


- 输入和输出
-
- 排序后的数组：
- 61 87 170 275 503 512 897 908

- 翻转后的数组：
- 908 897 512 503 275 170 87 61

```
1.  class My {
2.      static int Main()   {
3.          int[] list = new int[8] { 503, 87, 512, 61, 908, 170, 897, 275 };
4.          int Count = list.Length;
5.          Array.Sort(list);
6.          Console.WriteLine("排序后的数组： ");
7.          for (int k = 0; k < Count; k++)
8.              Console.Write("{0} ", list[k]);
9.          Console.WriteLine();
10.         Array.Reverse(list);
11.         Console.WriteLine("翻转后的数组： ");
12.         for (int k = 0; k < Count; k++)
13.             Console.Write("{0} ", list[k]);
14.         Console.WriteLine();
15.         return 0;
```

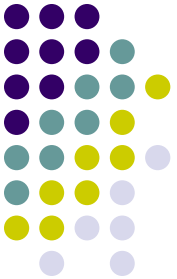
4.4 foreach语句



- foreach语句是专用于对数组、集合等数据结构中的每一个元素进行循环操作的语句，通过它可以列举数组、集合中的每一个元素，并且通过执行循环体对每一个元素进行需要的操作。

- [格式]：
- foreach(数据类型符 变量名 in 数组或集合)
- 循环体;

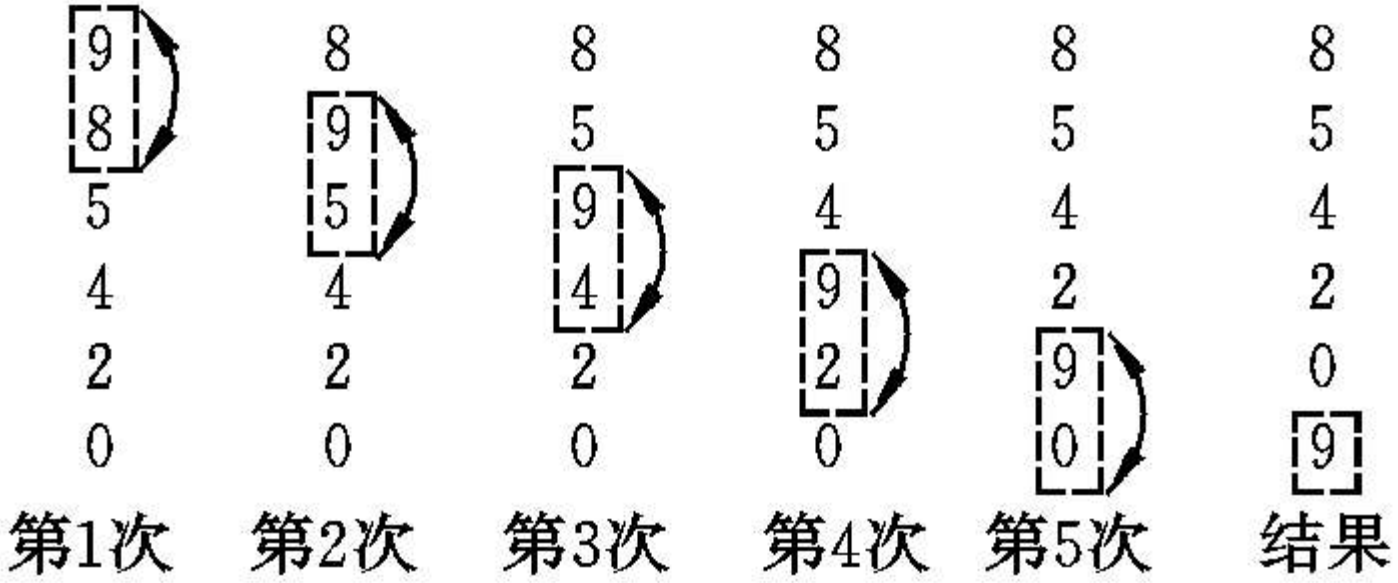
例：使用foreach求二维数组最小值



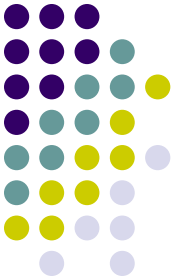
```
1.  using System;
2.  class My{
3.      static void Main(string[] args)
4.      {
5.          int[,] a = new int[3, 4]{{1,2,3,4},{5,11,7,8},{9,10,-11,7}};
6.          int min = a[0, 0];
7.          foreach (int i in a)
8.              if (i < min) min = i;
9.          Console.WriteLine("最小值={0}", min);
10.     }
11. }
```

【例4-14】冒泡排序算法分析

- (1)两两比较相邻元素A(I)和A(I+1)(I=1,2,…N-1),如果A(I)>A(I+1),则交换它们的位置 A(I)↔ A(I+1);
- (2)对剩下的N-1个元素,再两两进行比较,按同样规则交换它们的位置,经过N-2次比较,将次最大值交换到A(N-1)的位置;
- (3)如法炮制,经过N-1趟的“冒泡处理”,每趟进行N-i次的比较,全部数列有序。

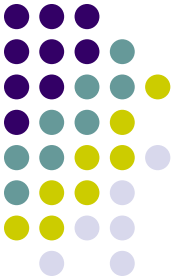


【例4-14】冒泡排序算法分析



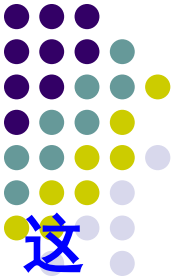
```
1.  using System;
2.  class My{
3.      static int Main()   {
4.          int[] list =        {503, 87, 512, 61, 908, 170, 897, 275,
5.         653, 426, 154, 509, 612, 677, 765, 703   };
6.          int Count = list.Length;
7.          for (int i = 0; i < Count; i++)
8.              for (int j = Count - 1; j > i; j--)
9.                  if (list[j - 1] > list[j])     {
10.                     int tmp = list[j - 1];    list[j - 1] = list[j];
11.                     list[j] = tmp;
12.                 }
```

【例4-14】冒泡排序算法分析



```
1.          Console.WriteLine("The result is :");
2.          for (int k = 0; k < Count; k++)
3.              Console.Write("{0} ", list[k]);
4.          Console.WriteLine();
5.          return 0;
6.      }
7.  }
```

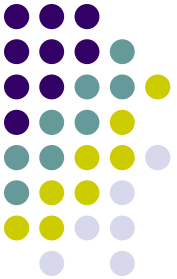
4.5 常用集合类（Collection）



- 集合（Collection）类是专门用于数据存储和检索的类。这些类提供了对栈（stack）、队列（queue）、列表（list）和哈希表（hash table）的支持。大多数集合类实现了相同的接口。

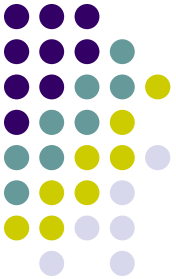
- 集合（Collection）类服务于不同的目的，如为元素动态分配内存，基于索引访问列表项等等。这些类创建 Object 类的对象的集合。在 C# 中，Object 类是所有数据类型的基类。

各种集合类和它们的用法



- 动态数组（ArrayList）
- 哈希表（Hashtable）
- 排序列表（SortedList）
- 堆栈（Stack）
- 队列（Queue）
- 点阵列（BitArray）

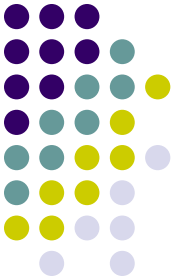
【例4-15】用ArrayList类计算数据的中间值



- 输入和输出
-
- 排序后的数组：
- 61 87 503 512 908

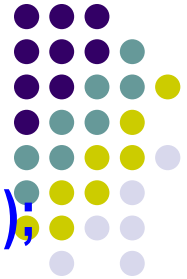
- 数组的中间值为503
- 数组的元素个数为5
- 数组的元素容量为5

【例4-15】用ArrayList类计算数据的中间值



```
1.  using System;
2.  using System.Collections;
3.  class My{
4.      static int Main()   {
5.          int[] mylist = new int[5]{503, 87, 512, 61, 908};
6.          ArrayList list = new ArrayList(mylist);
7.          list.Sort();
8.          Console.WriteLine("排序后的数组： ");
9.          int Count = list.Count;
10.         for (int k = 0; k < Count; k++)
11.             Console.Write("{0} ", list[k]);
12.         Console.WriteLine();
```

【例4-15】用ArrayList类计算数据的中间值

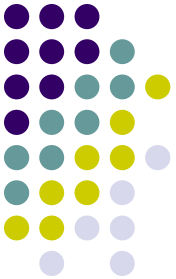


```
1.          Console.WriteLine("数组的中间值为{0} ", list[Count/2]);
2.          Console.WriteLine("数组的元素个数为{0} ", list.Count);
3.          Console.WriteLine("数组的元素容量为{0} ", list.Capacity);
4.          return 0;
5.      }
6.  }
```

【例4-16】利用Queue类，实现数据插入和删除。

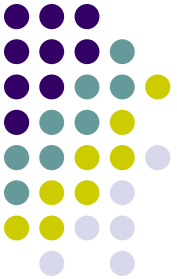
```
1.  class My  {
2.      static int Main()   {
3.          Queue myque = new Queue(7);
4.          myque.Enqueue( "Sunday");
5.          myque.Enqueue("Monday");
6.          myque.Enqueue("Tuesday");
7.          myque.Enqueue("Wednesday");
8.          myque.Enqueue("Thursday");
9.          myque.Enqueue("Friday");
10.         myque.Enqueue("Saturday");
11.         Console.WriteLine("我的队列包含：  ");
12.         for (int i = 0; myque.Count>0; i++)
13.             Console.Write(myque.Dequeue()+"  ");
```

【例4-17】用集合类Stack，是否是回文数



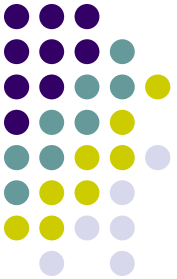
```
1.  using System;
2.  using System.Collections;
3.  class My{
4.      static int Main()
5.      {
6.          string a;
7.          int i;
8.          Stack x = new Stack();
9.          Console.Write("请输入一个整数： ");
10.         a = Console.ReadLine();
11.         for (i = 0; i < a.Length; i++)
12.             x.Push(a[i]);
13.         char[] b = new char[a.Length];
```

【例4-16】用集合类Stack，是否是回文数



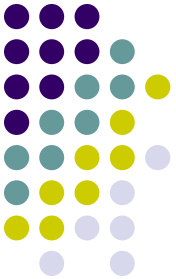
```
1.    for (i = 0; x.Count > 0; i++)
2.          {
3.              b[i] = (char)x.Pop();
4.          }
5.          string c = new string(b);
6.          if (a == c)
7.              Console.WriteLine(a + "是回文数");
8.          else
9.              Console.WriteLine(a + "不是回文数");
10.         return 0;
11.     }
12. }
```

【例4-18】用集合类SortedList，实现查找



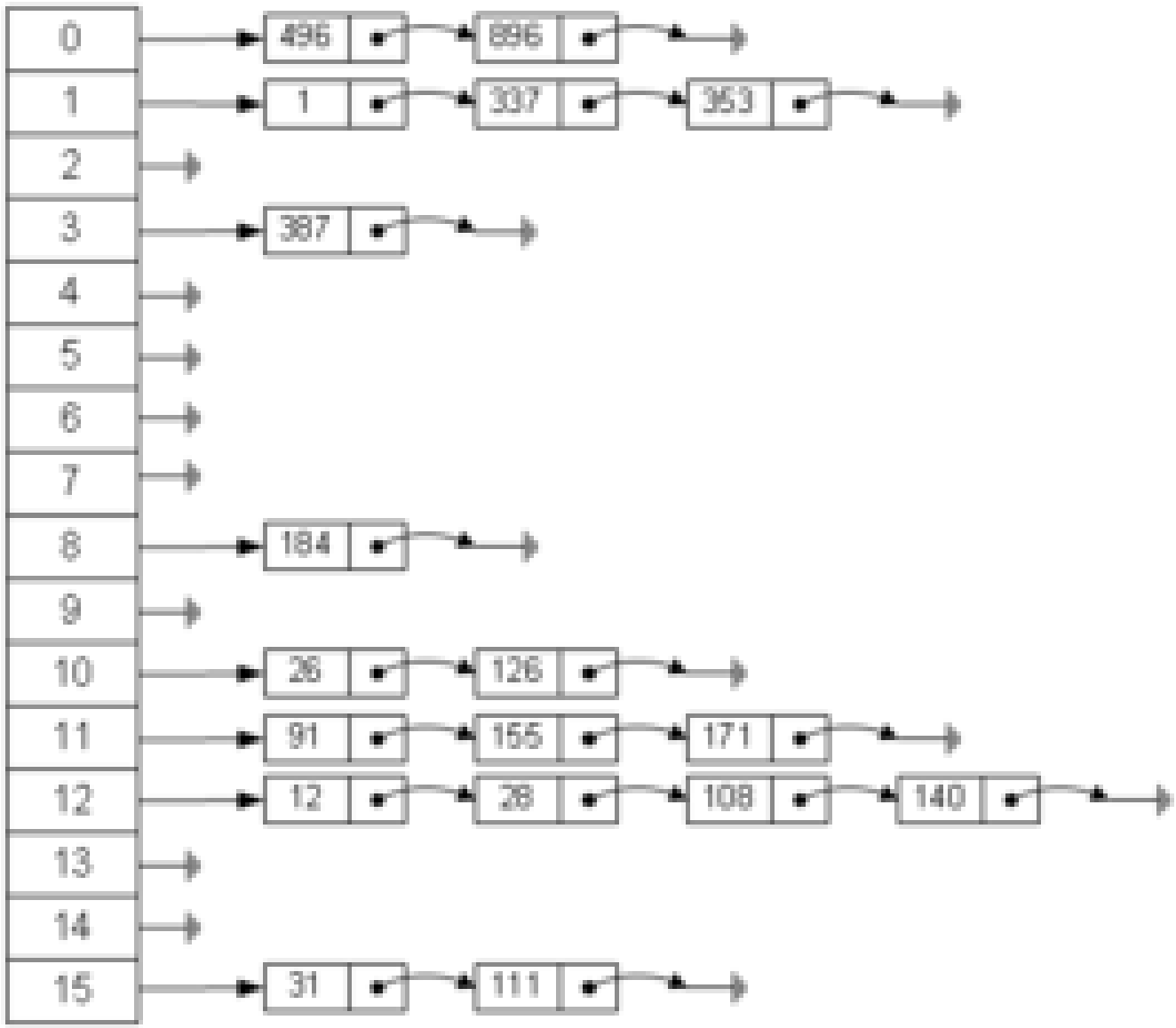
```
1.  using System;
2.  using System.Collections;
3.  class My{
4.      static int Main()   {
5.          SortedList mydict = new SortedList();
6.          mydict.Add("Sunday", "星期日");
7.          mydict.Add("Monday", "星期一");
8.          mydict.Add("Tuesday", "星期二");
9.          mydict.Add("Wednesday", "星期三");
10.         mydict.Add("Thursday", "星期四");
11.         mydict.Add("Friday", "星期五");
12.         mydict.Add("Saturday", "星期六");
13.
```

【例4-18】用集合类SortedList，实现查找

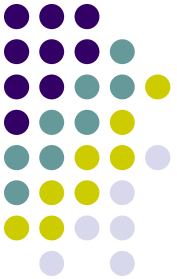


```
1.          Console.Write("请输入一个单词：");
2.          string word;
3.          word = Console.ReadLine();
4.          int p = mydict.IndexOfKey(word);
5.          if (p < 0)
6.              Console.WriteLine("没找到");
7.          else
8.              Console.WriteLine("单词：{0}\n解释：{1}",
9.      mydict.GetKey(p), mydict.GetByIndex(p));
10.         return 0;
11.     }
12. }
```

【例4-19】用集合类Hashtable，实现查找

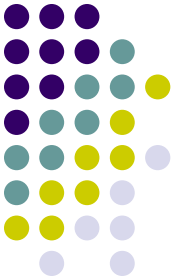


【例4-19】用集合类Hashtable，实现查找



```
1.  using System;
2.  using System.Collections;
3.  class My{
4.      static int Main()   {
5.          Hashtable mydict = new Hashtable();
6.          mydict.Add("Sunday", "星期日");
7.          mydict.Add("Monday", "星期一");
8.          mydict.Add("Tuesday", "星期二");
9.          mydict.Add("Wednesday", "星期三");
10.         mydict.Add("Thursday", "星期四");
11.         mydict.Add("Friday", "星期五");
12.         mydict.Add("Saturday", "星期六");
```

【例4-19】用集合类Hashtable，实现查找



```
1.          Console.Write("请输入一个单词：");
2.          string word;
3.          word = Console.ReadLine();
4.          if (mydict.Contains(word))
5.              Console.WriteLine("单词：{0}\n解释：{1}",word, mydict[wor
6.          else
7.              Console.WriteLine("没找到");
8.          mydict.Clear();
9.          return 0;
10.     }
11. }
```

结　束　语

- 学好程序设计语言的唯一途径是

- 你的编程能力与你在计算机上投入的时间成

正比