# 实验 3　程序流程控制(2)

**实验目的**

- 掌握 for 循环语句的使用；
- 掌握 while 循环语句的使用；
- 掌握 do…while 循环语句的使用；
- 掌握多重循环结构程序流程；
- 了解跳转语句的使用；
- 了解程序异常处理机制。

**实验内容**

**实验 3-1**　求 $n!$

**实验要求**：输入整数 $n(n\geqslant 0)$，分别利用 for 循环、while 循环、do…while 循环求 $n!$。运行效果如图 3-1 所示。


图 3-1　实验 3-1 运行效果

**操作提示**：

(1) $n!=n\times(n-1)\times(n-2)\times\cdots\times 2\times 1$。例如，$5!=5\times4\times3\times2\times1=120$。特别地，$0!=1$。

(2) 一般地，累乘的初值为 1，而累加的初值为 0。

(3) 如果输入的是负整数，则继续提示输入非负整数，直至 $n\geqslant 0$。

(4) 程序代码如图 3-2 所示。

```
sy3_1.Program                          Main(string[] args)
static void Main(string[] args)
{
    int i,n, fac=1;
    string s;
    n = -1;
    while (n< 0)
    {
        Console.Write("请输入非负整数n: ");
        s = Console.ReadLine();
        n = int.Parse(s);
    }
    // 方法一：for循环
    for (i = 1; i <= n; i++) fac *= i;
    Console.WriteLine("       for循环：{0}! = {1}", n, fac);
    // 方法二：while循环
    i = 1; fac = 1;
    while (i <= n) fac *= i++;
    Console.WriteLine("     while循环：{0}! = {1}", n, fac);
    // 方法三：do ..while循环
    i = 1; fac = 1;
    do
    {
        fac *= i; i++;
    } while (i <= n);
    Console.WriteLine("do...while循环：{0}! = {1}", n, fac);
    Console.ReadKey();
}
```

图 3-2　实验 3-1 程序代码

**实验 3-2** 显示 Fibonacii 数列

**实验要求**：显示 Fibonacii 数列：1,1,2,3,5,8,…。当 Fibonacii 值＞10 000 时停止显示。要求每行显示 5 项。运行效果如图 3-3 所示。

**操作提示**：Fibonacii 数列的生成规律为：

$$\begin{cases} F_1 = 1 & n = 1 \\ F_2 = 1 & n = 2 \\ F_n = F_{n-1} + F_{n-2} & n \geqslant 3 \end{cases}$$

程序代码如图 3-4 所示。

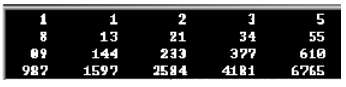



图 3-3 实验 3-2 运行效果

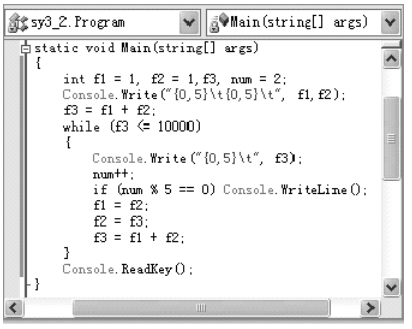
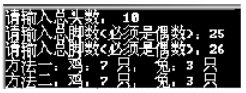


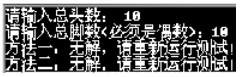

图 3-4 实验 3-2 程序代码

**实验 3-3** 鸡兔同笼问题

**实验要求**：已知在同一个笼子里总共有 $h$ 只鸡和兔，鸡和兔的总脚数为 $f$ 只，其中，$h$ 和 $f$ 由用户输入，求鸡和兔各有多少只？要求使用两种方法：一是求解方程；二是利用循环进行枚举测试。运行效果如图 3-5 所示。




(a) 合理解




(b) 无解

图 3-5 实验 3-3 运行效果

**操作提示**：

(1) 已知鸡和兔的总头数为 $h$，总脚数为 $f$。假设鸡有 $c$ 只，兔有 $r$ 只。

(2) 方法一：求解方程。由公式：

$$\begin{cases} c + r = h \\ 2c + 4r = f \end{cases}$$

解得：

$$\begin{cases} r = \dfrac{f}{2} - h \\ c = h - r \end{cases}$$

由公式推得，鸡和兔的总脚数 $f$ 必须是偶数，并且鸡和兔的只数必须是非负整数。

(3) 方法二：利用循环进行枚举测试。鸡的只数 $c$ 取值范围为：0～$h$，兔的数量 $r$ 为

$h-c$，如果满足条件 $2c+4r==f$，则求得解。

（4）程序代码如图 3-6 所示。

```
sy3_3.Program                                   Main(string[] args)
static void Main(string[] args)
{
    int c, r;                   // number of chicken & rabbit
    Console.Write("请输入总头数： ");
    String s = Console.ReadLine();
    int h = int.Parse(s);       //total heads of chicken & rabbit
    int f = 1;
    while (f % 2 != 0)
    {
        Console.Write("请输入总脚数(必须是偶数)： ");
        s = Console.ReadLine();
        f = int.Parse(s);       //total feet of chicken & rabbit
    }
    // 方法一：利用循环
    bool solution = false;      // 判断是否有解
    for (c = 0; c <= h; c++)
    {
        r = h - c;
        if (2 * c + 4 * r == f)
        {
            Console.WriteLine("方法一：鸡：{0} 只， 兔：{1} 只", c, r);
            solution = true;
        }
    }
    if (!solution) Console.WriteLine("方法一：无解，请重新运行调试！ ");
    //方法二：解方程
    r = f / 2 - h;
    c = h - r;
    solution = false;       // 判断是否有解
    if (r >= 0 && c >= 0)
    {
        Console.WriteLine("方法二：鸡：{0} 只， 兔：{1} 只", c, r);
        solution = true;
    }
    if (!solution) Console.WriteLine("方法二：无解，请重新运行调试！ ");
    Console.ReadKey();
}
```

图 3-6　实验 3-3 程序代码

**实验 3-4**　利用级数和求 π

**实验要求**：使用格利高利公式求 π 的近似值，直到最后一项的绝对值小于 $10^{-6}$ 为止。运行效果如图 3-7 所示。

$$\frac{\pi}{4} \approx 1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\cdots$$

**操作提示**：程序代码如图 3-8 所示。

```
pi = 3.141594
```

图 3-7　实验 3-4 运行效果

```
sy3_4.Program                                   Main(string[] args)
static void Main(string[] args)
{
    float n,t,pi;
    int s;
    pi=0; t=1; n=1; s=1;
    while (Math.Abs(t) >= Math.Pow(10, -6))
    {
        pi += t;
        n += 2;
        s = -s;
        t = s / n;
    }
    pi *= 4;
    Console.WriteLine("pi = {0}",pi);
    Console.ReadKey();
}
```

图 3-8　实验 3-4 程序代码

**实验 3-5** 求最大公约数和最小公倍数

**实验要求**：产生两个 0～100 之间(包含 0 和 100)的随机数 $a$ 和 $b$,求这两个整数的最大公约数和最小公倍数。运行效果如图 3-9 所示。

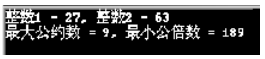



图 3-9 实验 3-5 运行效果

**操作提示**：程序代码如图 3-10 所示。

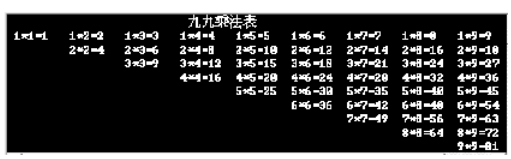
```
static void Main(string[] args)
{
    int m, n, r, m1, n1;
    Random rNum = new Random();
    m1 = rNum.Next(101);  //产生0~100之间的随机数a
    n1 = rNum.Next(101);  //产生0~100之间的随机数b
    Console.WriteLine("整数1 = {0}, 整数2 = {1}", m1, n1);
    if (m1 > n1)
    {
        m = m1; n = n1;
    }
    else
    {
        m = n1; n = m1;
    }
    do
    {
        r = m % n;
        m = n;
        n = r;
    } while (r != 0);
    Console.WriteLine("最大公约数 = {0}, 最小公倍数 = {1}", m, m1 * n1 / m);
    Console.ReadKey();
}
```

图 3-10 实验 3-5 程序代码

**实验 3-6** 打印九九乘法表

**实验要求**：利用嵌套循环打印如图 3-11 所示的呈下三角和呈上三角的九九乘法表。


(a) 呈下三角的九九乘法表




(b) 呈上三角的九九乘法表

图 3-11 实验 3-6 运行效果

**操作提示**：程序代码如图 3-12 所示。

**实验 3-7** 素数的判断

**实验要求**：分别利用 for 循环和 while 循环显示 1～100 间所有素数。要求每行显示 10 项。运行效果如图 3-13 所示。

**操作提示**：

(1) 所谓素数(或称质数),是指除了 1 和该数本身,不能被任何整数整除的正整数。判断一个正整数 $m$ 是否为素数,只要判断 $m$ 可否被 2～$\sqrt{m}$ 之中的任何一个整数整除,如果 $m$ 不能被此范围中任何一个整数整除,$m$ 即为素数,否则 $m$ 为合数。

(2) 程序代码如图 3-14 所示。

```
sy3_8.Program                                   Main(string[] args)
static void Main(string[] args)
{
    String s;
    // 下三角
    Console.WriteLine("                                九九乘法表");
    for (int i = 1; i <= 9; i++)
    {
        s = "";
        for (int j = 1; j <= i; j++)
        {
            s += (String.Format("{0}*{1}={2}", i, j, i * j)).PadRight(8);
        }
        Console.WriteLine(s);
    }
    // 上三角
    Console.WriteLine();
    Console.WriteLine("                                九九乘法表");
    for (int i = 1; i <= 9; i++)
    {
        s = "";
        s += s.PadRight(8 * (i - 1) + 1);
        for (int j = i; j <= 9; j++)
        {
            s += (String.Format("{0}*{1}={2}", i, j, i * j)).PadRight(8);
        }
        Console.WriteLine(s);
    }
    Console.ReadKey();
}
```

图 3-12　实验 3-6 程序代码

```
方法一，1~100间所有的素数为：
   2    3    5    7   11   13   17   19   23   29
  31   37   41   43   47   53   59   61   67   71
  73   79   83   89   97
方法二，1~100间所有的素数为：
   2    3    5    7   11   13   17   19   23   29
  31   37   41   43   47   53   59   61   67   71
  73   79   83   89   97
```

图 3-13　实验 3-7 运行效果

```
sy3_7.Program                                   Main(string[] args)
static void Main(string[] args)
{
    int m, k, i,num=0;
    //方法一：利用for循环和break语句
    Console.WriteLine("方法一：1~100间所有的素数为：");
    for (m = 2; m <= 100; m++)
    {
        k = (int)(Math.Sqrt(m));
        for (i = 2; i <= k; i++)
            if (m % i == 0) break;
        if (i == k + 1)
        {
            Console.Write("{0,5}", m);
            num++;
            if (num % 10 == 0) Console.WriteLine();
        }
    }
    //方法二：利用while循环和boolean变量
    Console.WriteLine("\n方法二：1~100间所有的素数为：");
    num = 0;
    for (m = 2; m <= 100; m++)
    {
        bool flag = true;           //假设整数为素数
        k = (int)(Math.Sqrt(m));
        i = 2;
        while (i <= k && flag == true)
        {
            if (m % i == 0) flag = false; //可以整除，肯定不是素数
            else i++;
        }
        if (flag == true)
        {
            Console.Write("{0,5}", m);
            num++;
            if (num % 10 == 0) Console.WriteLine();
        }
    }
    Console.ReadKey();
}
```

图 3-14　实验 3-7 程序代码

**实验 3-8** 异常处理

**实验要求**：输入任意两个整数，求两者的商。使用异常处理机制捕捉零除异常和参数格式异常。运行效果如图 3-15 所示。


(a) 输入格式异常　(b) 零除异常　(c) 正确的商

图 3-15　实验 3-8 运行效果

**操作提示**：

（1）当试图用零除整数值或十进制数值时，将引发 DivideByZeroException 异常。

（2）当方法调用中参数的格式不符合对应的形参类型的格式时，将引发 FormatException 异常。

（3）程序代码如图 3-16 所示。

```
sy3_8.Program                          Main(string[] args)
static void Main(string[] args)
{
    int i,j,k;
    Console.Write("请输入被除数：");
    try
    {
        String s = Console.ReadLine();
        i = int.Parse(s);          //被除数
        Console.Write("请输入除数：");
        s = Console.ReadLine();
        j = int.Parse(s);          //除数
        k = i / j;
        Console.WriteLine("被除数/除数 = {0}",  k);
    }
    catch (FormatException e1)
    {
        Console.WriteLine("输入格式不正确！");
    }
    catch (DivideByZeroException e2)
    {
        Console.WriteLine("除数不能为0！");
    }

    Console.ReadKey();
}
```

图 3-16　实验 3-8 程序代码

# 实验 4　数组和指针

**实验目的**

- 掌握数组的声明、实例化和初始化；
- 掌握数组元素的引用；
- 掌握一维数组的操作；
- 掌握二维数组的操作；
- 了解 System. Array 类常用方法和属性的使用；
- 了解指针的基本操作。

**实验内容**

**实验 4-1**　求若干学生的平均身高、最高身高、最低身高以及高于平均身高的人数

**实验要求**：已知 10 个学生的身高为 156,150,167,178,180,176,173,154,155,158,求平均身高、最高身高、最低身高,并统计高于平均身高的人数。运行效果如图 4-1 所示。


图 4-1　实验 4-1 运行效果

**操作提示**：

(1) 求一维数组中各元素的平均值,先利用循环对每个元素的值进行累加,数组各元素之和除以数组长度,即为数组各元素之平均值。

(2) 求若干数中的最小值的方法一般如下：

① 将最小值的初值设为一个比较大的数,或者取第一个数为最小值的初值。

② 利用循环,将每个数与最小值比较,若此数小于最小值,则将此数设置为最小值。

(3) 求若干数中的最大值的方法一般如下：

① 将最大值的初值设为一个比较小的数,或者取第一个数为最大值的初值。

② 利用循环,将每个数与最大值比较,若此数大于最大值,则将此数设置为最大值。

(4) 程序代码如图 4-2 所示。

**实验 4-2**　统计各分数段学生的人数和百分比

**实验要求**：已知某班 10 个学生的英语考试成绩为 80,90,67,89,78,85,45,69,77,95,统计优良中差各分数段的人数和所占百分比。假设成绩 90～100 为优,80～89 为良,60～79 为中,0～59 为差。运行效果如图 4-3 所示。

**操作提示**：程序代码如图 4-4 所示。

**实验 4-3**　冒泡法排序

**实验要求**：随机生成 10 个学生的成绩(取值为 0～100 的整数,包含 0 和 100),置于一维数组中,并利用冒泡法对成绩数组元素递减排序。运行效果如图 4-5 所示。

```
sy4_1.Program                                   Main(string[] args)
static void Main(string[] args)
{   // 声明并初始化有10个整数的数组
    int[] height = new int[10] { 158, 150, 167, 178, 180, 173, 173, 154, 155, 158 };
    int i, sumHeight = 0, avgHeight, maxHeight=0, minHeight=500, overAvg = 0;
    // 显示数组内容
    Console.WriteLine("学生身高如下：");
    for (i = 0; i < 10; i++) Console.Write("{0,4}", height[i]);
    // 求身高总和、最高身高、最低身高
    for (i = 0; i < 10; i++)
    {
        sumHeight += height[i];
        if (height[i] > maxHeight) maxHeight = height[i];
        if (height[i] < minHeight) minHeight = height[i];
    }
    avgHeight = sumHeight / 10;      //求平均身高
    for (i = 0; i < 10; i++)   // 统计高于平均身高的学生人数
    {
        if (height[i] > avgHeight) overAvg++;
    }
    Console.WriteLine("\n平均身高={0}，最高身高={1}，最低身高={2}", avgHeight, maxHeight, minHeight);
    Console.WriteLine("高于平均身高的学生人数={0}", overAvg);
    Console.ReadKey();
}
```

图 4-2　实验 4-1 程序代码

```
学生成绩如下：
  80  70  67  89  78  85  45  69  77  95
优(90~100)分数段的学生人数=2, 所占百分比=20%
良(80~ 89)分数段的学生人数=3, 所占百分比=30%
中(60~ 79)分数段的学生人数=4, 所占百分比=40%
差( 0~ 59)分数段的学生人数=1, 所占百分比=10%
```

图 4-3　实验 4-2 运行效果

```
sy4_2.Program                                   Main(string[] args)
static void Main(string[] args)
{
    // 声明并初始化有10个整数(0~100)的数组
    int[] score = new int[10] { 80, 90, 67, 89, 78, 85, 45, 69, 77, 95 };
    int i, ANum = 0, BNum = 0, CNum = 0, DNum = 0;
    // 显示数组内容
    Console.WriteLine("学生成绩如下：");
    for (i = 0; i < 10; i++) Console.Write("{0,4}", score[i]);
    // 求成绩总和
    for (i = 0; i < 10; i++)
    {
        switch (score[i] / 10)
        {
            case 10:
            case 9:
                ANum++;
                break;
            case 8:
                BNum++;
                break;
            case 7:
            case 6:
                CNum++;
                break;
            default:
                DNum++;
                break;
        }
    }
    Console.WriteLine("\n优(90~100)分数段的学生人数={0}，所占百分比={1:#0.##%}", ANum, ANum / 10.0);
    Console.WriteLine("良(80~ 89)分数段的学生人数={0}，所占百分比={1:#0.##%}", BNum, BNum / 10.0);
    Console.WriteLine("中(60~ 79)分数段的学生人数={0}，所占百分比={1:#0.##%}", CNum, CNum / 10.0);
    Console.WriteLine("差( 0~ 59)分数段的学生人数={0}，所占百分比={1:#0.##%}", DNum, DNum / 10.0);
    Console.ReadKey();
}
```

图 4-4　实验 4-2 程序代码

```
原始数组:
  74  56  20  85   5  90  44 100  75  93
降序数组:
 100  93  90  85  75  74  56  44  20   5
```

图 4-5　实验 4-3 运行效果

**操作提示：**

(1) 对于包含 $N$ 个元素的一维数组 A，按递减顺序排序的冒泡法的算法为：

① 第 1 轮比较：从第一个元素开始，对数组中所有 $N$ 个元素进行两两大小比较，如果不满足降序关系，则交换。即 A[0]与 A[1]比较，若 A[0]＜A[1]，则 A[0]与 A[1]交换；然后 A[1]与 A[2]比较，若 A[1]＜A[2]，则 A[1]与 A[2]交换；……直至最后 A[$N$－2]与 A[$N$－1]比较，若 A[$N$－2]＜A[$N$－1]，则 A[$N$－2]与 A[$N$－1]交换。第一轮比较完成后，数组元素中最小的数排到数组最后。

② 第 2 轮比较：从第一个元素开始，对数组中前 $N$－1 个元素(第 $N$ 个元素，即 A[$N$－1]已经最小，无需参加排序)继续两两大小比较，如果不满足降序关系，则交换。第二轮比较完成后，数组元素中次小的数排到最后，即 A[$N$－2]为数组元素中次小的数。

③ 依此类推，进行第 $N$－1 轮比较后，数组中所有元素均按递减顺序排好序。

(2) 程序代码如图 4-6 所示。

```
sy4_3.Program                          Main(string[] args)
public static void DisplayMatrix(int[] A)
{  //打印矩阵
    foreach (int i in A) Console.Write("{0,4} ", i);
    Console.WriteLine();
}
static void Main(string[] args)
{
    int i, t;
    int[] A = new int[10];
    Random rNum = new Random();
    for (i = 0; i < A.Length; i++) A[i] = rNum.Next(101);
    //数组A赋值(0~100之间的随机数)
    Console.WriteLine("原始数组: ");
    DisplayMatrix(A);
    //冒泡排序
    int N = A.Length;                    //获取数组A的长度N
    for (int loop = 1; loop <= N - 1; loop++)//外循环进行N-1轮比较
    {
        for (i = 0; i <= N - 1 - loop; i++)  //内循环两两比较，小数上浮
            if (A[i] < A[i + 1])
            {
                t = A[i];
                A[i] = A[i + 1];
                A[i + 1] = t;
            }
    }
    Console.WriteLine("降序数组: ");
    DisplayMatrix(A);  // 打印矩阵
    Console.ReadKey();
}
```

图 4-6　实验 4-3 程序代码

**实验 4-4**　选择法排序

**实验要求：**随机生成 10 个学生的成绩(取值为 0～100 的整数，包含 0 和 100)，置于一维数组中，并利用选择法对成绩数组元素递减排序。运行效果如图 4-7 所示。

```
原始数组:
  58   33   43   72    9   44    2    3   24   95
降序数组:
  95   72   58   44   43   33   24    9    3    2
```

图 4-7　实验 4-4 运行效果

**操作提示：**

(1) 对于包含 $N$ 个元素的一维数组 A，按递减顺序排序的选择法的基本思想是：每次在若干无序数据中查找最大数，并放在无序数据中的首位。其算法为：

① 从 $N$ 个元素的一维数组中找最大值及其下标，最大值与数组的第 1 个元素交换。

② 从数组的第 2 个元素开始的 $N$－1 个元素中再找最大值及其下标，该最大值(即整个数组元素的次大值)与数组第 2 个元素交换。

③ 依此类推，进行第 $N-1$ 轮选择和交换后，数组中所有元素均按递减顺序排好序。

（2）程序代码如图 4-8 所示。

```
sy4_4.Program                          Main(string[] args)
public static void DisplayMatrix(int[] A)
{
    foreach (int i in A) Console.Write("{0,4} ", i);
    Console.WriteLine();
}
static void Main(string[] args)
{
    int i, t, MaxI;
    int[] A = new int[10];
    Random rNum = new Random();
    //数组A赋值（0~100之间的随机数）
    for (i = 0; i < A.Length; i++) A[i] = rNum.Next(101);
    Console.WriteLine("原始数组: ");
    DisplayMatrix(A);
    //选择法递减排序
    int N = A.Length; //获取数组A的长度N
    for (int loop = 0; loop <= N - 2; loop++)//外循环进行N-1轮比较
    {
        MaxI = loop;
        for (i = loop; i <= N - 1; i++)    //内循环中在无序数中找最大值
            if (A[i] > A[MaxI]) MaxI = i;
        //最大值与无序数中的第一个元素交换
        t = A[loop];
        A[loop] = A[MaxI];
        A[MaxI] = t;
    }
    Console.WriteLine("降序数组: ");
    DisplayMatrix(A);
    Console.ReadLine();
}
```

图 4-8　实验 4-4 程序代码

**实验 4-5**　两个矩阵的相加和相减

**实验要求**：利用随机数形成并显示如图 4-9 所示的 4 行 4 列的二维矩形数组 A（数值元素取值为 50～100 的整数，包含 50 和 100）和数组 B（数值元素取值为 0～10 的整数，包含 0 和 10）。

```
数组A的内容:
 88  59  49  35
 46  50  67  95
 74  34  84  73
 63  34  29  10
```

(a) 数组A的内容

```
数组B的内容:
 2  7  2  6
 3  1  4  2
 5  0  2  7
 0  2  9  9
```

(b) 数组B的内容

图 4-9　数组的内容

（1）分别以上三角形式显示数组 A 的内容、下三角形式显示数组 B 的内容，如图 4-10 所示。

（2）将两个矩阵相加，结果放入矩阵 C 中；将两个矩阵相减，结果放入矩阵 D 中，如图 4-11 所示。

```
上三角形式显示数组A的内容:
 88  59  49  35
     50  67  95
         84  73
             10
```

(a) 以上三角形式显示数组A

```
下三角形式显示数组B的内容:
 2
 3  1
 5  0  2
 0  2  9  9
```

(b) 以下三角形式显示数组B

图 4-10　显示数组的内容

```
数组A和B相加之和:
 90  66  51  41
 49  51  71  97
 79  34  86  80
 63  36  38  19
```

(a) 矩阵相加

```
数组A和B相减之差:
 86  52  47  29
 43  49  63  93
 69  34  82  66
 63  32  20   1
```

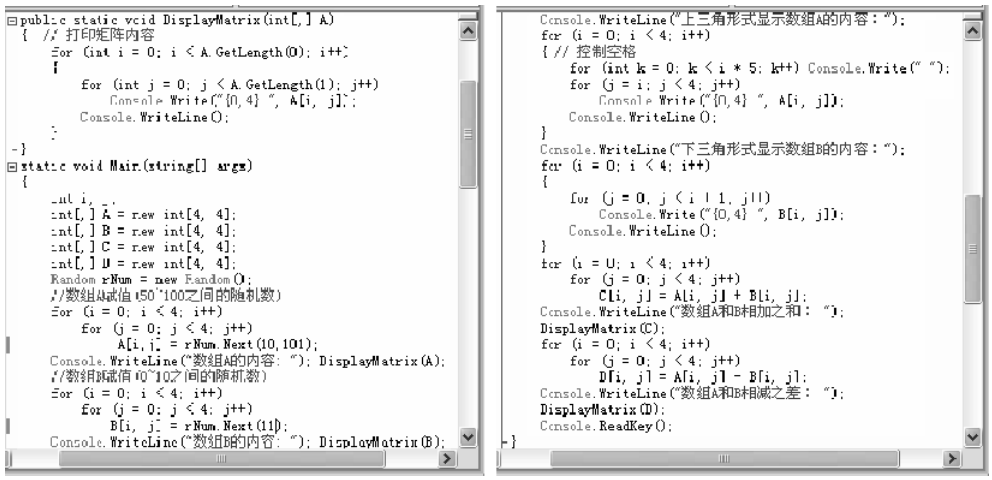
(b) 矩阵相减

图 4-11　两个矩阵之和与之差

**操作提示**：程序代码如图 4-12 所示。

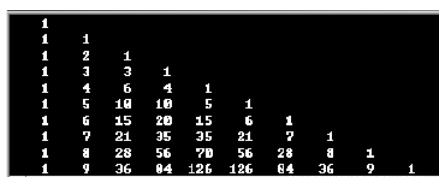
**实验 4-6**　打印杨辉三角

**实验要求**：杨辉三角即二项式定理的系数表。要求打印出 10 行，运行效果如图 4-13 所示。



```
public static void DisplayMatrix(int[,] A)
{  // 打印矩阵内容
    for (int i = 0; i < A.GetLength(0); i++)
    {
        for (int j = 0; j < A.GetLength(1); j++)
            Console.Write("{0,4} ", A[i, j]);
        Console.WriteLine();
    }
}
static void Main(string[] args)
{
    int i, j;
    int[,] A = new int[4, 4];
    int[,] B = new int[4, 4];
    int[,] C = new int[4, 4];
    int[,] D = new int[4, 4];
    Random rNum = new Random();
    //数组A赋值(50~100之间的随机数)
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            A[i,j] = rNum.Next(10,101);
    Console.WriteLine("数组A的内容: "); DisplayMatrix(A);
    //数组B赋值(0~10之间的随机数)
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            B[i, j] = rNum.Next(11);
    Console.WriteLine("数组B的内容: "); DisplayMatrix(B);
    Console.WriteLine("上三角形式显示数组A的内容: ");
    for (i = 0; i < 4; i++)
    { // 控制空格
        for (int k = 0; k < i * 5; k++) Console.Write(" ");
        for (j = i; j < 4; j++)
            Console.Write("{0,4} ", A[i, j]);
        Console.WriteLine();
    }
    Console.WriteLine("下三角形式显示数组B的内容: ");
    for (i = 0; i < 4; i++)
    {
        for (j = 0; j < i + 1; j++)
            Console.Write("{0,4} ", B[i, j]);
        Console.WriteLine();
    }
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            C[i, j] = A[i, j] + B[i, j];
    Console.WriteLine("数组A和B相加之和: ");
    DisplayMatrix(C);
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            D[i, j] = A[i, j] - B[i, j];
    Console.WriteLine("数组A和B相减之差: ");
    DisplayMatrix(D);
    Console.ReadKey();
}
```

图 4-12　实验 4-5 程序代码



```
   1
   1   1
   1   2   1
   1   3   3   1
   1   4   6   4   1
   1   5  10  10   5   1
   1   6  15  20  15   6   1
   1   7  21  35  35  21   7   1
   1   8  28  56  70  56  28   8   1
   1   9  36  84 126 126  84  36   9   1
```

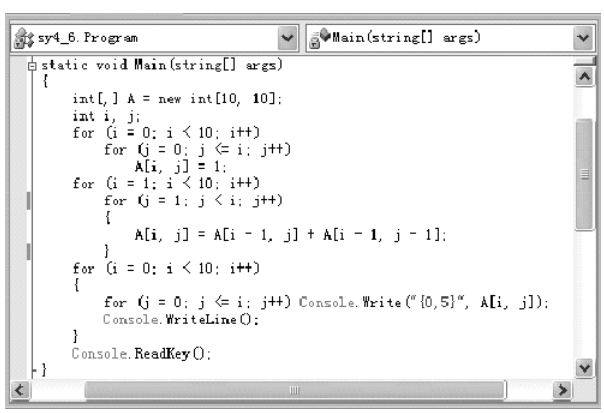
图 4-13　实验 4-6 运行效果

**操作提示：**

(1) 定义一个二维数组，杨辉三角只需处理(赋值和输出)下三角各元素即可。所有下三角各元素初始化为 1。

(2) 杨辉三角下三角各元素满足如下条件：第一列及对角线上的元素均为 1；其余每个元素等于它上一行同一列元素与上一行前一列元素之和，即：A[i,j]= A[i−1,j]+A[i−1,j−1]。

(3) 程序代码如图 4-14 所示。



```
sy4_6.Program                    Main(string[] args)
static void Main(string[] args)
{
    int[,] A = new int[10, 10];
    int i, j;
    for (i = 0; i < 10; i++)
        for (j = 0; j <= i; j++)
            A[i, j] = 1;
    for (i = 1; i < 10; i++)
        for (j = 1; j < i; j++)
        {
            A[i, j] = A[i - 1, j] + A[i - 1, j - 1];
        }
    for (i = 0; i < 10; i++)
    {
        for (j = 0; j <= i; j++) Console.Write("{0,5}", A[i, j]);
        Console.WriteLine();
    }
    Console.ReadKey();
}
```

图 4-14　实验 4-6 程序代码