

# 实验10-2

- 从键盘上为一个 $5 \times 5$ 的整型数组输入数据，并找出主对角线上元素的最大值及其所在的行号。
- 主对角线上的元素如何表示？  $a[i][i]$
- 默认 $a[0][0]$ 为主对角线上的最大值，从对角线上后一个元素依次比较，找出最大元素。

```
const int N=5;
```

```
int a[N][N];    int max,row,n,i;
```

输入数组实际长度和所有元素值

```
max=a[0][0], row=0;
```

```
i=1
```

```
i<n
```

T  $a[i][i]>max$  F

```
max=a[i][i];
```

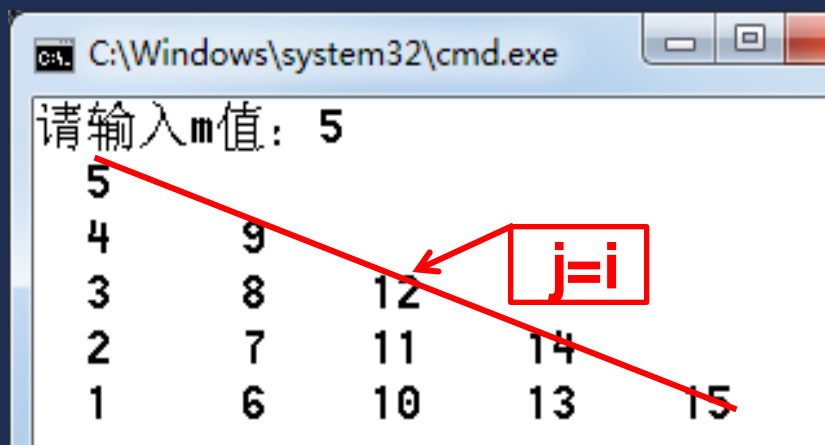
```
row=i;
```

```
i=i+1
```

输出最大值为max,行号为row

# 实验10-2

- 编程输出如下图所示三角。



**算法思路：**观察方阵的数据特点，可发现数据是从第1列倒着赋值的，赋值区域在直线 $j=i$ 的左侧 ( $j \leq i$ )，故对于该矩阵的初始化从列开始，行数递减。

```
int a[M][M], int i,j,m,s=1;
```

```
j=0 //从列开始处理
```

```
j<m
```

```
i=m-1 //倒着赋值
```

```
i>=j
```

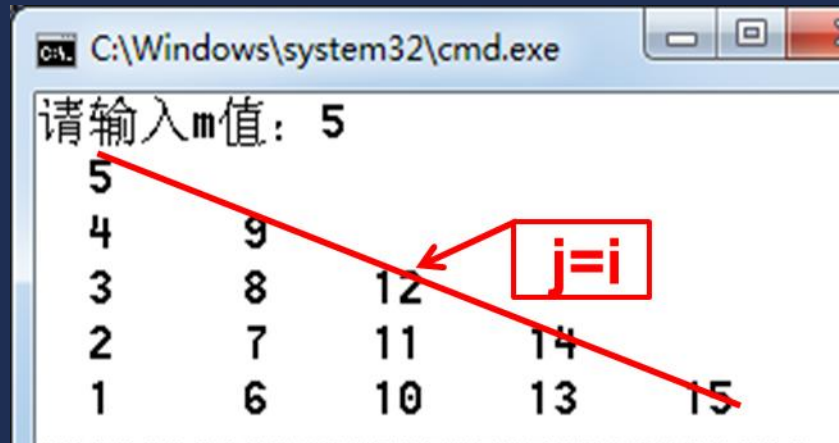
```
a[i][j]=s;
```

```
s++;
```

```
i--
```

```
j++
```

```
输出该方阵
```



C:\Windows\system32\cmd.exe

请输入m值: 5

5				
4	9			
3	8	12		
2	7	11	14	
1	6	10	13	15

A red arrow points from the value 12 to a box containing `j=i`.

```
C:\Windows\system32\cmd.exe
请输入m值: 5
5
4      9
3      8      12
2      7      11      14
1      6      10      13      15
=====
          5
        4      9
      3      8      12
    2      7      11      14
  1      6      10      13      15
请按任意键继续. . .
```

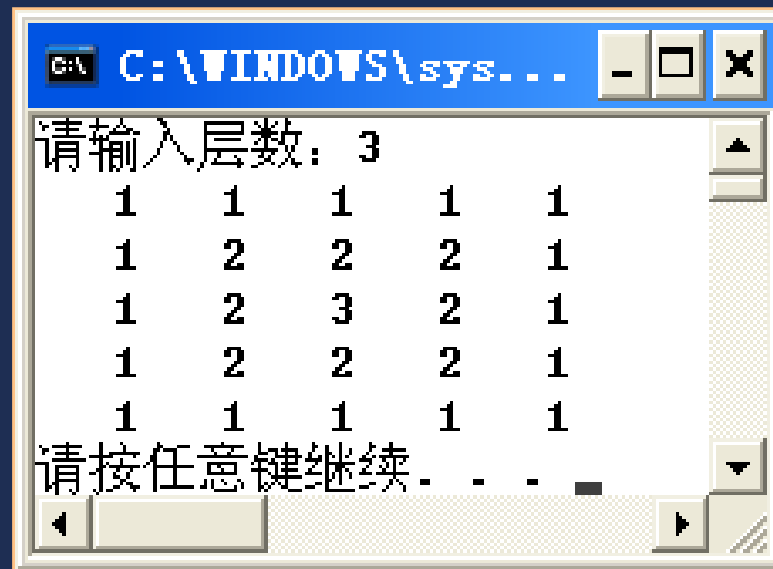
```
for(i=0;i<m;i++) //输出的行数
{
    for(j=0; j<3*(m-i);j++) //输出空格
    for(j=0; j<=i ;j++) //输出数据
        cout<<setw(6)<<a[i][j];
    cout<<endl; //每输完一行输出回车换行
}
```

## 图案输出方法：

- 1、用外层循环控制输出的行数，循环体中用两个顺序的循环，一个控制每行空格的输出(图案的左边线)，一个控制每行数据的输出(图案的右边线)。
- 2、用外层循环控制输出的行数，在循环体中，用一个循环(内层)控制每行内容的输出 (图案的右边线)，在什么时候该输出空格，什么时候该输出数据，可以一个选择结构(图案的左边线)确定。

# 实验10-3

- 编写程序，输入数字 $m$  ( $m < 10$ )，输出 $m$ 层回形方阵，方阵最外层是第一层，依次往内，每层用的数字和层数相同。



请输入层数: 3 (m)

1	1	1	1	1
1	2	2	2	1
1	2	3	2	1
1	2	2	2	1
1	1	1	1	1

请按任意键继续. . .

请输入正方形层数: 2

1	1	1
1	2	1
1	1	1

请按任意键继续. . .

第3层(k=3)      for(k=1; k<=m; k++)    //外层循环, 控制层数

	0	1	2	3	4
0	1	1	1	1	1
1	1	2	2	2	1
2	1	2	3	2	1
3	1	2	2	2	1
4	1	1	1	1	1

```

cout<<"请输入层数. ";
cin>>m;
for(k=1;k<=m;k++) //k控制输出层数
{
    for( i=k-1 ; i<2*m-k ;i++)
        for( j=k-1 ; j<2*m-k ;j++)
            a[i][j]=k;
    //每一层填充相应数据
}

```

寻找i、j与m(k)之间的关系

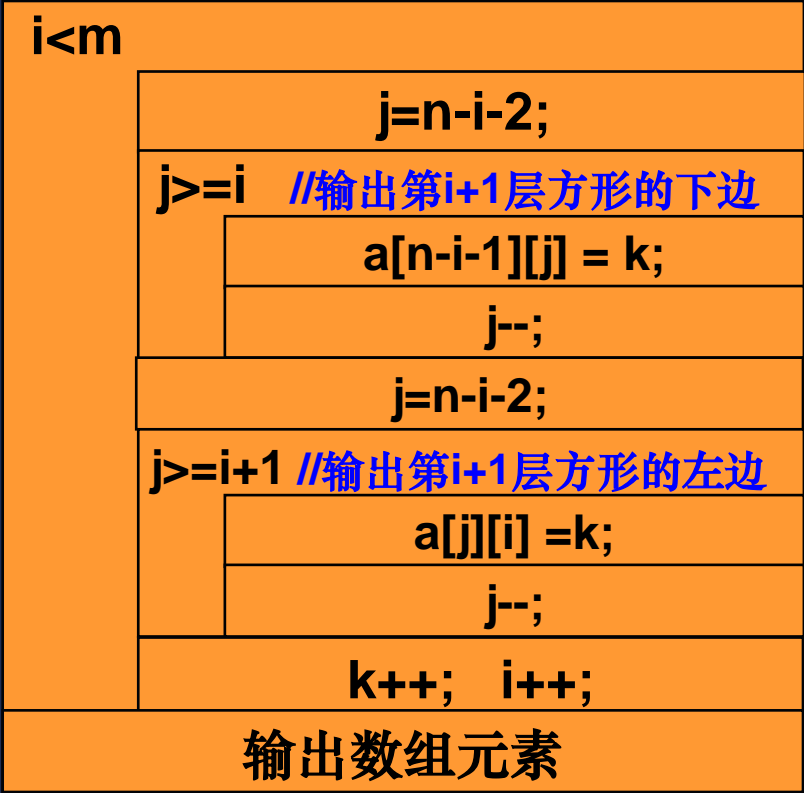


请输入层数: 3

1	1	1	1	1
1	2	2	2	1
1	2	3	2	1
1	2	2	2	1
1	1	1	1	1

请输入层数: 2

1	1	1	1
1	2	2	1
1	2	2	1
1	1	1	1



# 实验10-4

- 编写程序，要求输入某班N名同学的序号及3门课成绩，计算每位同学的平均成绩以及每门课的平均成绩（学号为3位整数，成绩也为整数，课程名称为：科目1、科目2、科目3）。
- 算法思路：用一个二维数组( $a[N][3]$ )保存学生的学号和三门成绩。对每一行求平均值得到的就是该生的平均成绩；对每一列求平均值则得到的是课程的平均成绩。



```
const int N=20;
```

```
int a[N][4], int i,j,n,sum,avg;
```

```
cin>>n; //实际行数(人数)
```

```
i=0
```

```
i<n
```

```
sum=0;
```

```
j=1 //成绩数据从第2列开始
```

```
j<4
```

```
sum=sum+a[i][j];
```

```
j++;
```

```
avg=sum/3;
```

```
cout<<"第"<<i+1<<  
"同学平均成绩"<<ave<<endl;
```

```
i++;
```

```
j=1 //成绩数据从第2列开始
```

```
j<4
```

```
sum=0;
```

```
i=0
```

```
i<n
```

```
sum=sum+a[i][j];
```

```
i++;
```

```
avg=sum/n;
```

```
cout<<"第"<<j+1<<  
"门功课的平均成绩"<<ave<<endl;
```

```
j++;
```