# 二维数组

主教练：党东

## ■ 二维数组

定义方式：

类型说明符　数组名[常量表达式][常量表达式];

行数　列数

int　a[3][4];

定义类型　数组名

■ 表明a数组由3×4个int型元素组成

■ 其元素分别为：a[0][0], a[0][1], a[0][2], a[0][3],
a[1][0], a[1][1], a[1][2], a[1][3],
a[2][0], a[2][1], a[2][2], a[2][3]

# 二维数组

## 二维数组

- 在定义数组的同时给数组元素赋值。

```
int a[2][3]={{1,2,3},{4,5,6}};或int a[2][3]={1,2,3,4,5,6};
b[5][5]={{1},{0,2},{0,0,3},{0,0,0,4},{0,0,0,0,5};

    1 0 0 0 0
    0 2 0 0 0
    0 0 3 0 0
    0 0 0 4 0
    0 0 0 0 5
//表示这样的一个5*5的方阵，省略的元素默认为0；
```

# 二维数组



定义：const int maxn=105;
int a[maxn][maxn];

(1)二维数组初始化：

①int a[maxn][maxn]={0};

②memset(a,0,sizeof(a));

③两重循环清为某个值。

```
for(i=0;i<maxn;i++)
    for(j=0;j<maxn;j++)a[i][j]=0;
```

(2)二维数组的读入

```
for(i=1;i<=n;i++)
    for(j=1;j<=n;j++)cin>>a[i][j];
```

(3)二维数组的输出

```
for(i=1;i<=n;i++)
{   for(j=1;j<=n;j++)cout<<a[i][j]<<" ";
    cout<<endl;
}
```

# 【试一试】方阵的最大值



**Description**

有一个n×m的矩阵，要求编程序求出其中值最大的那个元素的值，以及其所在的行号和列号。

**Input**

输入共两行；

第1行为两个整数n和m，n表示方阵的行数，m方阵的列数；

第2~n+1行，每行m个数，表示方阵每个位置的值；

**Output**

输出最大值，以及其所在的行号和列号。（数据保证最大值只有一个）

**Sample Input**

3 5
1 2 3 4 5
4 3 9 1 4
1 2 3 5 4

**Sample Output**

9 2 3

**Hint** 最大值为9，其在第2行，第3列。

# 【试一试】方阵的最大值



**【问题分析】**

先考虑解此问题的思路。从若干个数中求最大者的方法很多，我们现在采用“打擂台”算法。如果有若干人比武，先有一人站在台上，再上去一人与其交手，败者下台，胜者留台上。第三个人再上台与在台上者比，同样是败者下台，胜者留台上。如此比下去直到所有人都上台比过为止。最后留在台上的就是胜者。

程序模拟这个方法，开始时把a[0][0]的值赋给变量maxx，**maxx**就是开始时的擂主，然后让下一个元素与它比较，将二者中值大者保存在maxx中，然后再让下一个元素与新的maxx比，直到最后一个元素比完为止。maxx最后的值就是数组所有元素中的最大值。

# 【试一试】方阵的最大值



【参考代码】

```
int n,m,i,j,a[105][105],maxx=-0x7fffffff/2;
int H,L;
cin>>n>>m;
for(i=1;i<=n;i++)
        for(j=1;j<=m;j++)cin>>a[i][j];
for(i=1;i<=n;i++)
  for(j=1;j<=m;j++)
    if(a[i][j]>maxx)
                { maxx=a[i][j];
                  H=i;L=j;
            }
     cout<<maxx<<" "<<H<<" "<<L<<endl;
```

# 【例2】矩阵问题 --1072



**Description**

输入矩阵a（m行m列），将其行列互换。

**Input**

第一行为m（小于15），第二行至第m+1行为矩阵a。

**Output**

输出行列互换后的矩阵。

**Sample Input**

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

**Sample Output**

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

# 【例2】矩阵问题 --1072



**【问题分析】**

通过观察我们可以发现，整个方阵中a[i][j]的值和a[j][i]的值发生了互换。

# 【例2】矩阵问题 --1072



【参考代码】

```
int a[20][20],b[20][20],i,j,n;
cin>>n;
for(i=1;i<=n;i++)
   for(j=1;j<=n;j++)
         cin>>a[i][j];      //读入
for(i=1;i<=n;i++)
     for(j=1;j<=n;j++)
           b[j][i]=a[i][j];  //互换
for(i=1;i<=n;i++)       //输出
{
      for(j=1;j<=n;j++)cout<<b[i][j]<<" ";
      cout<<endl;
}
```

# 【思考】回型方阵



**Description**

输入一个正整数 n，输出 n×n 的回型方阵。例如，n=5 时，输出：

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

**Input**

一行一个正整数 n，2≤n≤9。

**Output**

共 n 行，每行包含 n 个正整数，之间用一个空格隔开。

**Sample Input**

5

**Sample Output**

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

# 【思考】回型方阵



**【问题分析】**

**定义一个二维数组 a[n][n] 存储回型方阵。**

**方法1**：通过“**一圈一圈**”赋值的方法做，先给 a[1][1] ~ a[n][n] 全部赋值 1，然后给 a[2][2] ~ a[n-1][n-1] 全部赋值 2，……共 n/2 圈(如果 n 是奇数，则最后一圈就是一个数)。

**方法2**：先给左上角的 a[n/2][n/2] 赋值，a[i][j] = min(i，j)，右上角、左下角、右下角三部分，通过下标的对称性复制过去即可。

例如，a[i][n+1-j] = a[n+1-i][j] = a[n+1-i] [n+1-j] = a[i][j]。

# 【思考】回型方阵



【方法1参考代码】

```
int n,i,j,k,a[10][10];
int main(){
    cin >> n;
    for(k = 1; k <= (n+1)/2; k++)    //确定第几圈
        for(i = k; i <= n+1-k; i++)   //双重循环为(k,k)~(n+1-k,n+1-k)大小的方阵赋值
            for(j = k; j <= n+1-k; j++)
                a[i][j] = k;
    for(i = 1; i <= n; i++){
        for(j = 1; j < n; j++)
            cout << a[i][j] << " " ;
        cout << a[i][n] << endl;
    }
    return 0;
}
```

# 【练习1】求平均分 --1074



**Description**

竞赛小组共有n（n <= 20）位同学，这学期每位同学共参与了三项比赛，请统计每位同学的平均分。

**Input**

输入第一行为n（n <= 20）位同学，第2行至第n+1行分别为这n个同学的三项比赛成绩(成绩均小于等于100)

**Output**

输出只有一行为这n个同学的平均分（保留2位小数），用空格分开

**Sample Input**

```
3
50 55 60
60 65 70
70 75 80
```

**Sample Output**

```
55.00 65.00 75.00
```

# 【练习1】求平均分 --1074



【问题分析】

```
double t[65][5];  //第1-2-3列分别代表三项成绩，第4列代表平均分。
int main()
{
        int n,i;
        cin>>n;
        for(i=1;i<=n;i++)
        {
                cin>>t[i][1]>>t[i][2]>>t[i][3];
                t[i][4]=(t[i][1]+t[i][2]+t[i][3])/3;
        }
        for(i=1;i<=n;i++)
                cout<<fixed<<setprecision(2)<<t[i][4]<<" ";
        return 0;
}
```

# 【练习2】杨辉三角 --1092



**Description**

输出杨辉三角的前N行(N<20)

**Input**

输入只有一行，包括1个整数N(N<20)

**Output**

输出只有N行

**Sample Input**

5

**Sample Output**

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

# 【练习2】杨辉三角 --1092



**【问题分析】**

通过观察可以发现图形的以下特点：

**1.每一行的数字个数与行号相同。**

**2.位置为（i , j ）的值a[i][j]等于上方a[i-1][j]和左上方a[i-1][j-1]之和。**

**注意：当上方a[i-1][j]没有数值时，默认为0，也满足该等式。**

# 【练习2】杨辉三角 --1092



【参考代码】

```
int a[10][10];
int main()
{       int n;
        cin>>n;
        a[0][0]=1;
        for(int i=1;i<=n;i++)
        {       for(int j=1;j<=i;j++)
                {       a[i][j]=a[i-1][j-1]+a[i-1][j];
                        cout<<a[i][j]<<" ";
                }
                cout<<endl;
        }
        return 0;
}
```

# 【练习3】马鞍数 --1073



**Description**

求一个nxn数矩阵中的马鞍数，所谓的马鞍数是指在行上最小而在列上是最大的数。

**Input**

第一行为一个数n（n<20），第二行至第n+1行为矩阵。

**Output**

输出马鞍数的坐标即(行号,列号)，多个马鞍数的话每行一个。

**Sample Input**

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

**Sample Output**

```
(1,1)
```

# 【练习3】马鞍数 --1101



【核心代码】

```
for(i=1;i<=n;i++)            //一行行尝试
{
        maxx=a[i][1];b=1;
        for(j=2;j<=n;j++)                    //找到每行中的最小
                if(a[i][j]<maxx)
                {maxx=a[i][j];b=j;}
        for(j=1;j<=n;j++)
                if(maxx<a[j][b])  break;     //判断是否每列最大
        if(j>n)cout<<"("<<i<<","<<b<<")"<<endl;
}
```