

《数组清零》

```
2 using namespace std;
3 int n,m,a[101][101],b[101][101];
4 void find() {
5     int x,y;
6     for(int i=1; i<=n; i++)
7         for(int j=1; j<=m; j++) {
8             if(a[i][j]==0) { //a里找0
9                 x=i;
10                y=j;
11                for(int k1=1; k1<=m; k1++)//b里清0
12                    b[x][k1]=0;
13                for(int k2=1; k2<=n; k2++)
14                    b[k2][y]=0;
15            }
16        }
17 }
18 int main() {
19     for(int i=1; i<=n; i++)//输入并复制 a a是找0的数组 ,
20         for(int j=1; j<=m; j++) { //b 是操作清0数组
21             cin>>a[i][j];
22             b[i][j]=a[i][j];
23         }
24
25     find();//枚举每个位置并清0
26     for(int i=1; i<=n; i++) { //输出b数组
27         for(int j=1; j<=m; j++)
28             cout<<b[i][j]<<" ";
29         cout<<endl;
30     }
```



实验舱
青少年编程
走近科学 走进名校

实验舱蛟龙三班

二维数组 (2)

zlj

2022.8

矩阵中的 特殊位置

在矩阵中**查找数据**

二维数组在形式上是矩阵

	0	1	2	3	4	5
0	[0][0]	[0][1]	[0][2]	[0][3]	[0][4]	[0][5]
1	[1][0]	[1][1]	[1][2]	[1][3]	[1][4]	[1][5]
2	[2][0]	[2][1]	[2][2]	[2][3]	[2][4]	[2][5]
3	[3][0]	[3][1]	[3][2]	[3][3]	[3][4]	[3][5]

A [i] [j]

(1, 1)	(1, 2)	(1, 3)	(1, 4)
(2, 1)	(2, 2)	(2, 3)	(2, 4)
(3, 1)	(3, 2)	(3, 3)	(3, 4)
(4, 1)	(4, 2)	(4, 3)	(4, 4)

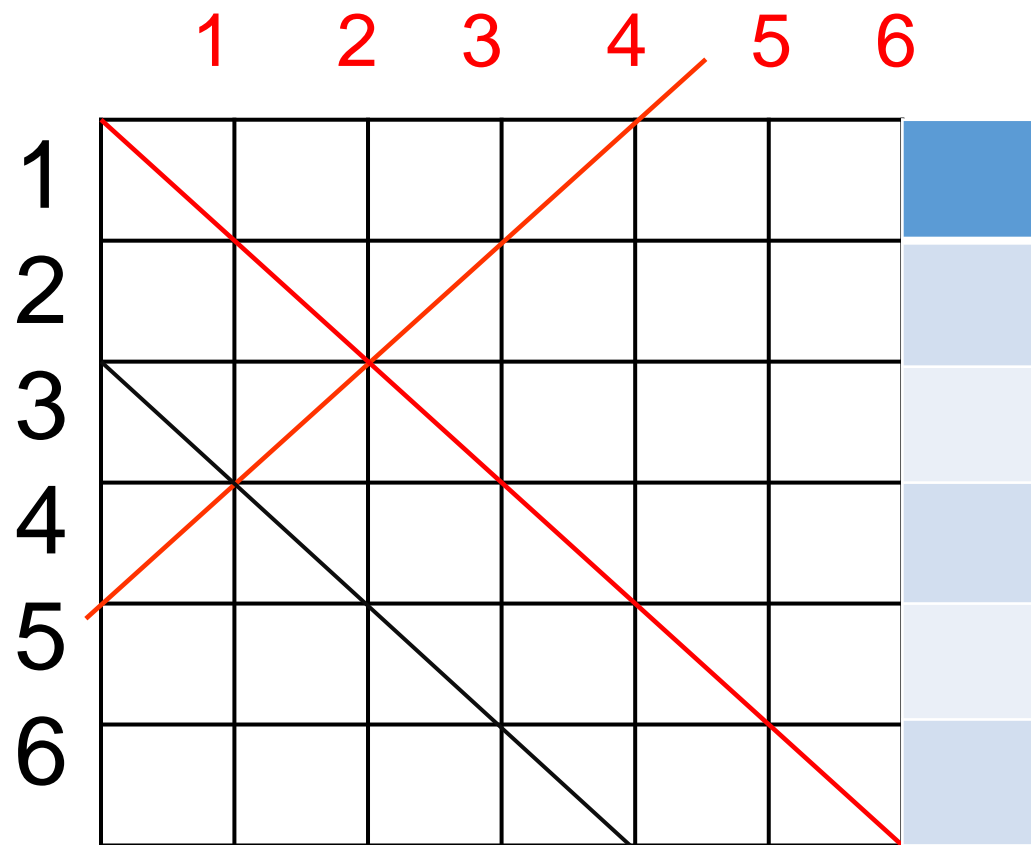
注意几点：

- 1、数组尽量开在 `main()` 外，全局变量好处多。
 - 2、数组等变量注意初始化及格式：
`memset(a,-1,sizeof(a)); a[3][2]={{1,2},{2,3}{4,5}}`
 - 3、理解 `a`, `a[0]` & `a[0][0]` 的含义
 - 4、搞清 `int a[3][4]` 有几行？几列？
 - 5、输入输出时，两个下标 (`i`, `j`) 注意不要越界
-

一、矩阵中斜线上的位置

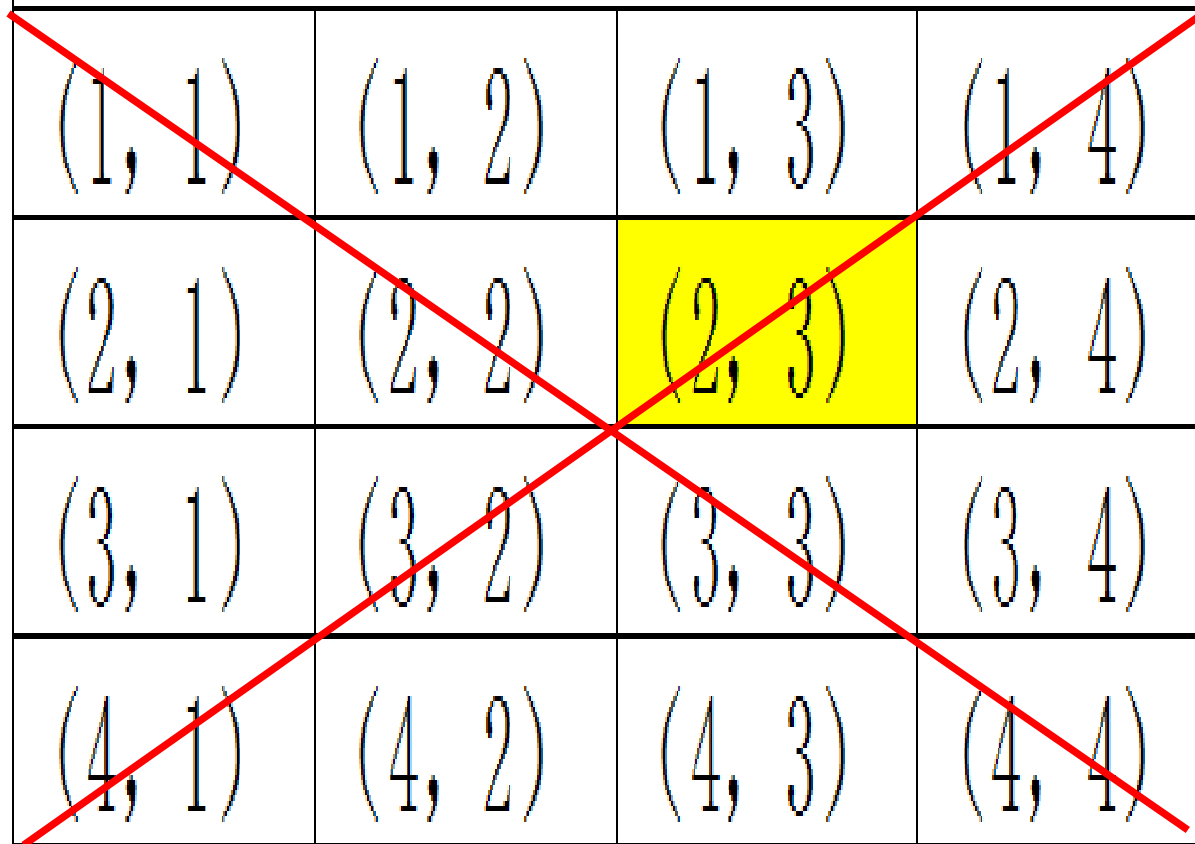
方便描述：矩阵行列都从[1][1]开始

斜线上位置与行 列的关系



- 1、 $n \times n$ 对角斜线
- 2、左右非对角斜线

1、对角上的点与 i j 的关系 ($n \times n$)

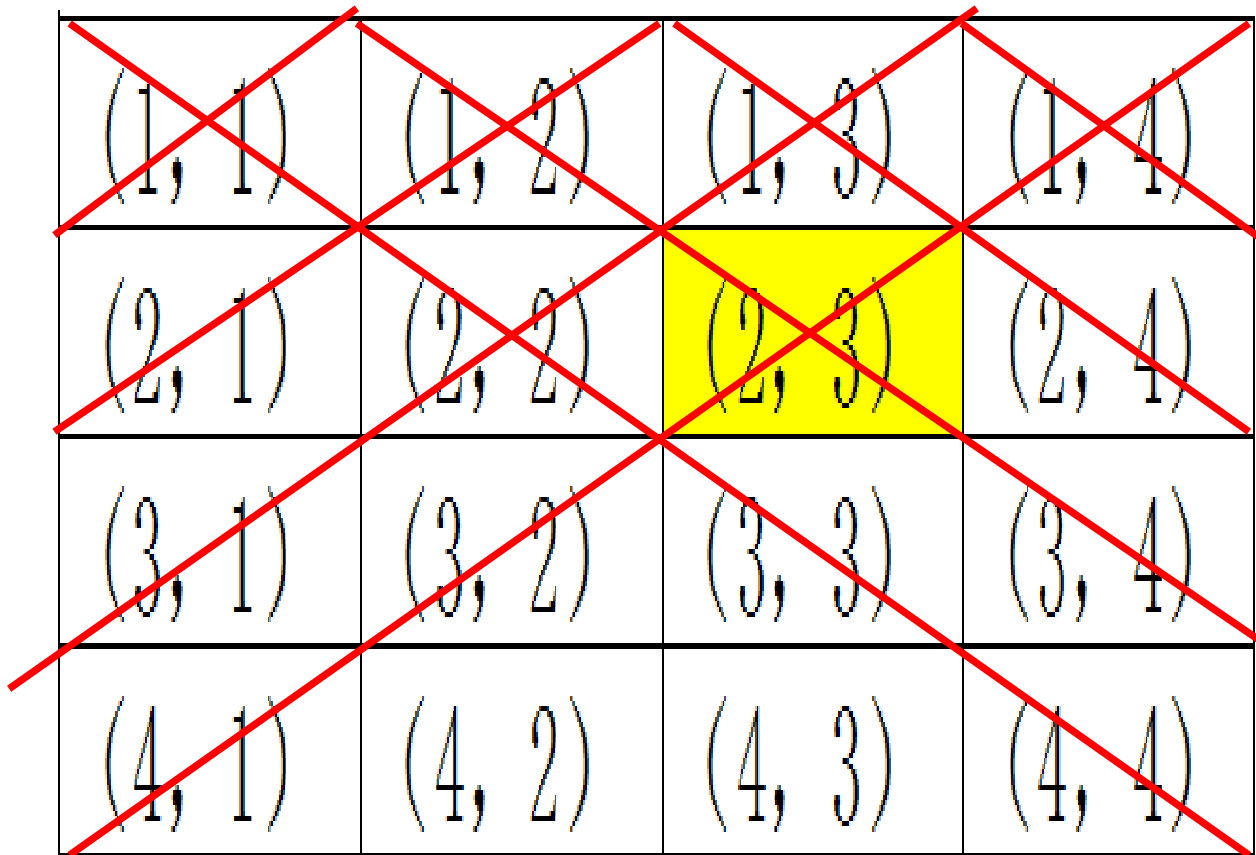


(1, 1)	(1, 2)	(1, 3)	(1, 4)
(2, 1)	(2, 2)	(2, 3)	(2, 4)
(3, 1)	(3, 2)	(3, 3)	(3, 4)
(4, 1)	(4, 2)	(4, 3)	(4, 4)

$$i = j \quad (i - j = 0)$$

$$i = n + 1 - j \quad (i + j = n + 1)$$

2、斜线与[i][j]关系 ($n \times n$)



A 4x4 grid of coordinate pairs (i, j) where i is the row index and j is the column index. Red diagonal lines are drawn across the grid. The cell containing $(2, 3)$ is highlighted in yellow.

$(1, 1)$	$(1, 2)$	$(1, 3)$	$(1, 4)$
$(2, 1)$	$(2, 2)$	$(2, 3)$	$(2, 4)$
$(3, 1)$	$(3, 2)$	$(3, 3)$	$(3, 4)$
$(4, 1)$	$(4, 2)$	$(4, 3)$	$(4, 4)$

$$i - j = k \quad (-n < k < n)$$

$$i + j = k \quad (2 \leq k \leq 2 \times n)$$

线与[i][j]关系

结论:

- 1、同一行时: i 相等 同一列时: j 相等
 - 2、 $N \times N$ 矩阵时, 同一对角线, $i=j$ 或 $i+j=n+1$ ($i=n+1-j$)
 - 3、 $N \times M$ 矩阵中, 在同一斜线的位置:
 $i-j$ 相等时在同一斜线上 (左上到右下) 每一斜线值等于多少?
 $i+j$ 相等时在同一斜线 (左下到右上)
-

例1：矩阵求值

输入一个 $n \times n$ 的矩阵，找出两条对角线上的元素的最大值并输出。

输入格式

第1行一个整数 n ，以下 n 行，每行 n 个正整数。

输出格式

一行，一个整数，表示两条对角线上的数的最大值。

样例输入

4

1 2 3 4

2 3 2 1

4 5 6 2

2 3 4 1

样例输出

6

- 1、输入数据
- 2、**枚举**对角线上的每个元素 并求最值
- 3、输出ans

算法1：同时枚举 i,j

```
3  int n,a[100][100];
4  int main() {
5      cin>>n;
6      for(int i=1; i<=n; i++)
7          for(int j=1; j<=n; j++)
8              cin>>a[i][j];
9      int ans=(1);
10     for(int i=1; i<=n; i++) {
11         for(int j=1; j<=n; j++)
12             if((i==j)|| (2)) { // 对角线的条件
13                 ans=max(ans,a[i][j]);
14             }
15     }
16     cout<<ans<<endl;
17     return 0;
```

算法二：枚举 i , 计算 j .

2、根据对角线上 i, j 的关系 (枚举行或列均可)。

```
for (int i=1; i<=n; i++){  
     $j1 = i$  ,  $j2 = n+1-i$  ;  
    if( $j2$  不越界)  
         $ans = \max(ans, a[i][j1])$ ;  
         $ans = \max(ans, a[i][j2])$   
}
```

(1, 1)	(1, 2)	(1, 3)	(1, 4)
(2, 1)	(2, 2)	(2, 3)	(2, 4)
(3, 1)	(3, 2)	(3, 3)	(3, 4)
(4, 1)	(4, 2)	(4, 3)	(4, 4)

填空：

```
3  int a[101][101],n;//矩阵对角线最大值
4  int main() {
5      cin>>n;
6      for(int i=1; i<=n; i++)//输入
7          for(int j=1; j<=n; j++)
8              cin>>a[i][j];
9      int ans=a(1);//初始值
10     for(int i=1; i<=(2); i++) { //枚举行
11         int j=(3); //计算对角线列标
12         ans=max(ans,max(a[i][i],a[i][j]));
13     } //擂台找到最值
14     cout<<ans<<endl;
15     return 0;
```

总结：

枚举 每个位置，可以有两种方式：

- 1、直接用二维枚举 i, j (时间复杂度 $O(N^2)$)
- 2、用一维枚举行 i , 或 j , 用它们的关系算出 j , 或 i , 并用约束条件判断 i , 或 j , 是否合法（行列数）

(1, 1)	(1, 2)	(1, 3)	(1, 4)
(2, 1)	(2, 2)	(2, 3)	(2, 4)
(3, 1)	(3, 2)	(3, 3)	(3, 4)
(4, 1)	(4, 2)	(4, 3)	(4, 4)

例2：同行列对角线的格子

输入三个自然数N，i，j（ $1 \leq i \leq N$ ， $1 \leq j \leq N$ ），输出在一个N*N格的棋盘（行列均从1开始编号），与格子（i，j）同行、同列、同一对角线的所有格子的位置。

如：n=4，i=2，j=3表示了棋盘中的第二行第三列的格子
当n=4，i=2，j=3时，输出的结果是：

- (2,1) (2,2) (2,3) (2,4) 同一行上格子的位置
- (1,3) (2,3) (3,3) (4,3) 同一列上格子的位置
- (1,2) (2,3) (3,4) 左上到右下对角线上的格子的位置
- (4,1) (3,2) (2,3) (1,4) 左下到右上对角线上的格子的位置

第一列	第二列	第三列	第四列	
				第一行
		(2,3)		第二行
				第三行
				第四行

(1, 1)	(1, 2)	(1, 3)	(1, 4)
(2, 1)	(2, 2)	(2, 3)	(2, 4)
(3, 1)	(3, 2)	(3, 3)	(3, 4)
(4, 1)	(4, 2)	(4, 3)	(4, 4)

【问题分析】

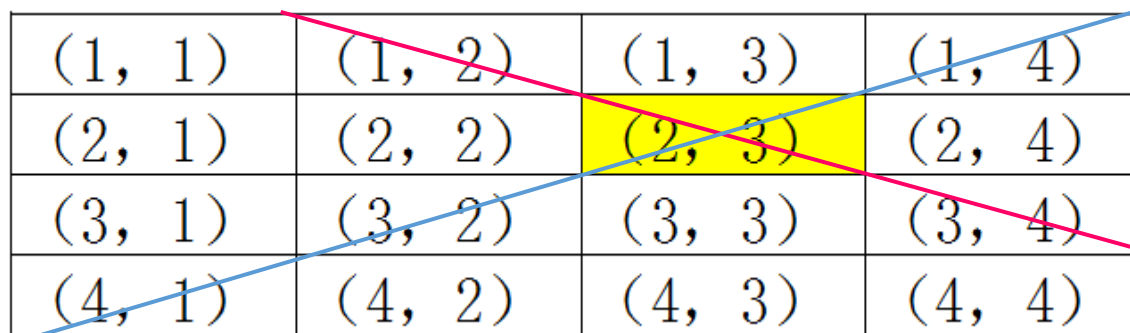
例： $n=4$ ， $(x,y)=(2,3)$

输出行： $a[x][i]$

输出列： $a[i][y]$

输出左上到右下对角线： $y-x=j-i$

输出左下到右上对角线： $x+y=i+j$



(1, 1)	(1, 2)	(1, 3)	(1, 4)
(2, 1)	(2, 2)	(2, 3)	(2, 4)
(3, 1)	(3, 2)	(3, 3)	(3, 4)
(4, 1)	(4, 2)	(4, 3)	(4, 4)

代码参考：

```
3  int a[201][201],n,x,y;
4  int main() {
5      cin>>n>>x>>y;
6      for(int i=1;i<=n;i++)//同一 x 行
7          cout<<'('<<(1)<<','<<i<<')'<<" ";
8      cout<<endl;
9      for(int i=1;i<=n;i++)//同一 y 列
10         cout<<'('<<i<<','<<(2)<<')'<<" ";
11     cout<<endl;
12     for(int i=1;i<=n;i++)
13         for(int j=1;j<=n;j++)//同一对角线：左上右下
14             if((3)==y-x)cout<<'('<<i<<','<<j<<')'<<" ";
15     cout<<endl;
16     for(int i=n;i>=1;i--)
17         for(int j=1;j<=n;j++)//同一对角线 右下到左上
18             if((4)==x+y)cout<<'('<<i<<','<<j<<')'<<" ";
19     cout<<endl;
20     return 0;
```

练习：用一维枚举

对角线上的位置枚举：

i, j 和为定值或差为定值,从一个点知道了和与差，我们可以用一循环枚举 i , 算出 $j=s-i$, 或 $j=s+l$, j 满足约束条件即可。一层循环完成枚举

练习：小Z找最值

输入一个 $N \times M$ 的二维矩阵，（ $1 \leq n, m \leq 100$ ），以及矩阵中某一个点的位置，小Z想编程快速找出跟这个点同行，同列、同斜线上的数的最大值并输出。

输入格式：

第一行两个整数：n,m分别表示矩阵的行与列，以下n行，m列的若干个整数。

最后一行，两个整数x,y分别表示矩阵中某个点的行与列。

输出格式：

1行一个整数，表示与x,y同行、同列、同斜线上数据的最大值。

输入样例：

4 4

1 2 3 4

2 3 2 3

8 7 3 2

1 2 3 4

2 3

输出样例：

7

样例说明：与第2行第3列，同行，同列、同斜线（两个对角线哦）的最大值的数是7。

填空

```
2 using namespace std;
3 int n,m,a[101][101],x,y;
4 int main() {
5     cin>>n>>m;//读入数据
6     for(int i=1; i<=n; i++)
7         for(int j=1; j<=m; j++)
8             cin>>a[i][j];
9     cin>>x>>y;//读入定点
10    int ans=(1);
11    for(int i=1; i<=n; i++)
12        for(int j=1; j<=m; j++) { //同行同列同斜线
13            if(i==x || (2) || (3) || (4))
14                ans=max(ans,a[i][j]);
15        }
16    cout<<ans<<endl;
```

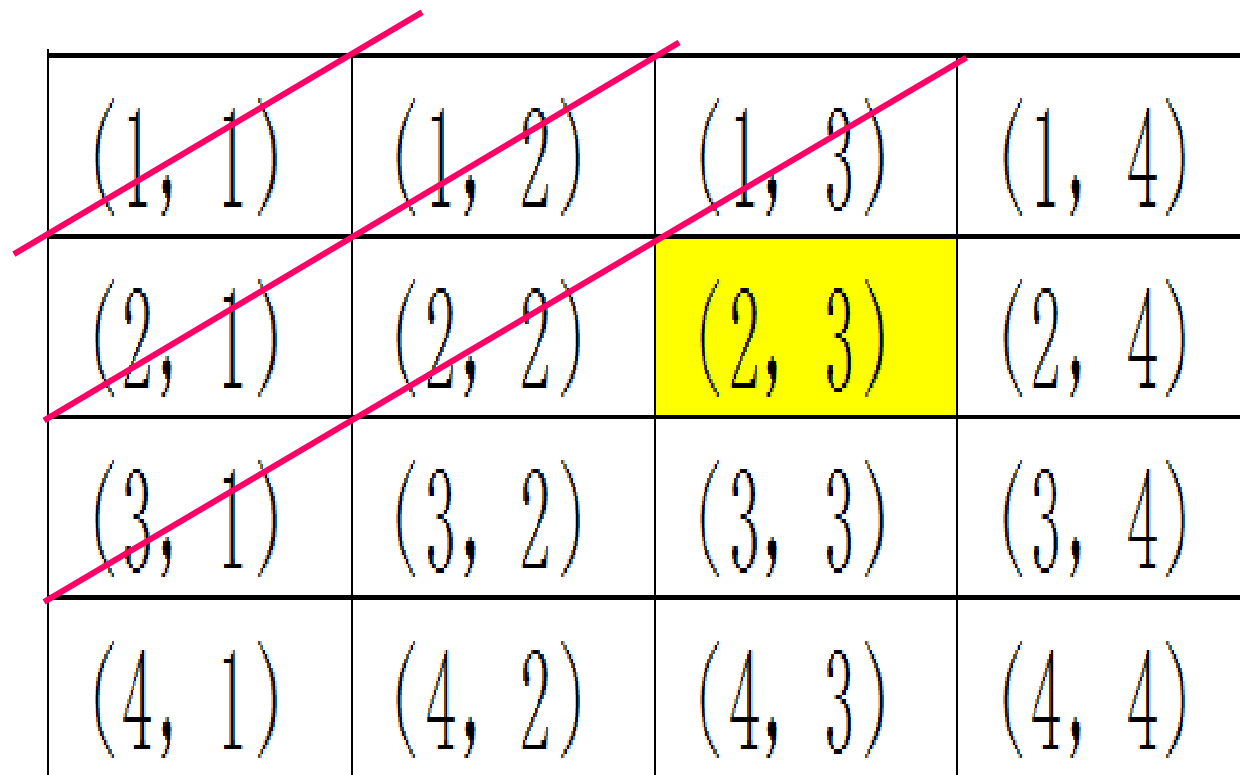
根据行列约束条件来确定位置：

```
10  int ans=a[x][y];
11  for(int i=1; i<=m; i++) //同行位置
12      ans=max(ans,a[x][i]);
13  for(int i=1; i<=n; i++) //同列位置
14      ans=max(ans,a[i][y]);
15  for(int i=1; i<=n; i++) { //枚举对角线位置
16      int j1= (    ), j2= (    ) ;
17      if(      ) //约束条件
18          ans=max(ans,a[i][j1]);
19      if(      )
20          ans=max(ans,a[i][j2]);
21  }
22  cout<<ans<<endl;
```

二、按斜线输出数据

方便描述：矩阵行列都从[1][1]开始

按斜线输出



A 4x4 grid of coordinate pairs (i, j) where i is the row index and j is the column index. Red diagonal lines are drawn from the top-left to the bottom-right, passing through the cells $(1,1)$, $(2,2)$, and $(3,3)$. The cell $(2,3)$ is highlighted in yellow.

$(1, 1)$	$(1, 2)$	$(1, 3)$	$(1, 4)$
$(2, 1)$	$(2, 2)$	$(2, 3)$	$(2, 4)$
$(3, 1)$	$(3, 2)$	$(3, 3)$	$(3, 4)$
$(4, 1)$	$(4, 2)$	$(4, 3)$	$(4, 4)$

例4：斜线处理2

输入n行m列数据，斜线输出。（按从右上往左下斜方向）

【输入说明】

第一行是n，以下是n行n列数字矩阵，以上所有数字均为整数。

【输出说明】

一行 $n \times n$ 个整数，每个整数后面有一个空格。

【输入样例】

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

【输出样例】

```
2 4 0 9 7 2 4 8 5
```

斜线输出

分析：

第一列	第二列	第三列	第四列	
				第一行
		(2,3)		第二行
				第三行
				第四行

- 1、有几条斜线？数量有无规律？
- 2、每个斜线的特点是？ $i+j=2$ $i+j=3$ $i+j=8$
- 3、注意斜线的方向？

算法1：枚举每一个位置 (i,j) if (i+j==s)

```
2  using namespace std; // 斜线输出2
3  int n, a[101][101];
4  int main() {
5      cin >> n;
6      for (int i = 1; i <= n; i++) // 读入矩阵
7          for (int j = 1; j <= n; j++) cin >> a[i][j];
8
9      for (int s = 2; s <= (1); s++) { // 枚举斜线
10         for (int i = 1; i <= n; i++) // 枚举行
11             for (int j = 1; j <= n; j++) // 枚举列
12                 if ((2)) cout << a[i][j] << " ";
13     }
14     cout << endl;
15     return 0;
}
```

思考：10行与11行能颠倒吗？如果两行换一下，会输出什么？为什么？

算法2:

$i+j=\text{定值}$

可以穷举每一定值（斜线），再一重循环枚举 i , $j=\text{定值}-i$ 即可。

注意：用一个条件来约束 j （出界）

算法2填空：

```
for(int i=1; i<=n; i++) {  
    for(int j=1; j<=n; j++) {  
        cin>>a[i][j];  
    }  
}  
for(int k=2; k<=    ; k++) { // 枚举斜线  
    for(int i=1; i<=n; i++) { // 枚举 i  
        int j=    ; // 计算 j  
        if(j>=1&&j<=n) { // j的范围  
            if(i+j==k) cout<<a[i][j]<<' ';  
        }  
    }  
}
```

例5：斜线处理1

输入 n 行 n 列数据，斜线输出。

【输入说明】

第一行是 n ，以下是 n 行 n 列数字矩阵，以上所有数字均为整数。（ $n \leq 20$ ）

【输出说明】

一行 $n \times n$ 个整数，每个整数后面有一个空格。

【输入样例】

3

2 4 9

0 7 4

2 8 5

【输出样例】

2 0 8 2 7 5 4 4 9

分析：

- 1、与上一题大致相同，几条斜线？
- 2、斜线的方向？
- 3、斜线的行列特点？

第一列	第二列	第三列	第四列	
				第一行
		(2,3)		第二行
				第三行
				第四行


```

3  int n,a[101][101];
4  int main() {
5      cin>>n;
6      for(int i=1; i<=n; i++)//输入数据
7          for(int j=1; j<=n; j++)cin>>a[i][j];
8
9      for(int s=(1); (2); s--) { //要输几条斜线?
10                                     //每条斜线的特点?
11          for(int i=1; i<=n; i++)
12              for(int j=1; j<=n; j++)
13                  if((3))cout<<a[i][j]<<" ";
14          //根据i,j判断在哪条斜线
15      }
16      cout<<endl;
17      return 0;

```

思考：1、for(int i= for(int j= 枚举行列的顺序能变吗？

2、for(int i=1;i<=n 能改成 for(int i=n;i>=1吗，改后会输出什么？

算法2：

```
3  int a[101][101],n;
4  int main() {
5      cin>>n;    //读入矩阵
6      for(int i=1; i<=n; i++)
7          for(int j=1; j<=n; j++)
8              cin>>a[i][j];
9      for(int k=(1); k(2); k--) { //定值
10         for(int i=1; i<=n; i++) { //枚举行
11             int j=(3);    //计算列
12             if(j>=1&& j<=n) //约束条件
13                 cout<<a[i][j]<<" ";
14         }
15     }
16     cout<<endl;
17     return 0;
```

如果这样输出如何修改程序？

3
1 2 3
4 5 6
7 8 9

输出：
7 8 4 9 5 1 6 2 3

例6：蛇形填数

输入n，按蛇形输出1—N*N的矩阵。

5

输出：

1 2 6 7 15

3 5 8 14 16

4 9 13 17 22

10 12 18 21 23

11 19 20 24 25

(1, 1)	(1, 2)	(1, 3)	(1, 4)
(2, 1)	(2, 2)	(2, 3)	(2, 4)
(3, 1)	(3, 2)	(3, 3)	(3, 4)
(4, 1)	(4, 2)	(4, 3)	(4, 4)

分析：

解决哪些问题？

1、几条斜线上填数，斜线特点？

2、斜线有2个方向，怎么处理？

3、依次填数

1 2 6 7 15

3 5 8 14 16

4 9 13 17 22

10 12 18 21 23

11 19 20 24 25

(1, 1)	(1, 2)	(1, 3)	(1, 4)
(2, 1)	(2, 2)	(2, 3)	(2, 4)
(3, 1)	(3, 2)	(3, 3)	(3, 4)
(4, 1)	(4, 2)	(4, 3)	(4, 4)

```

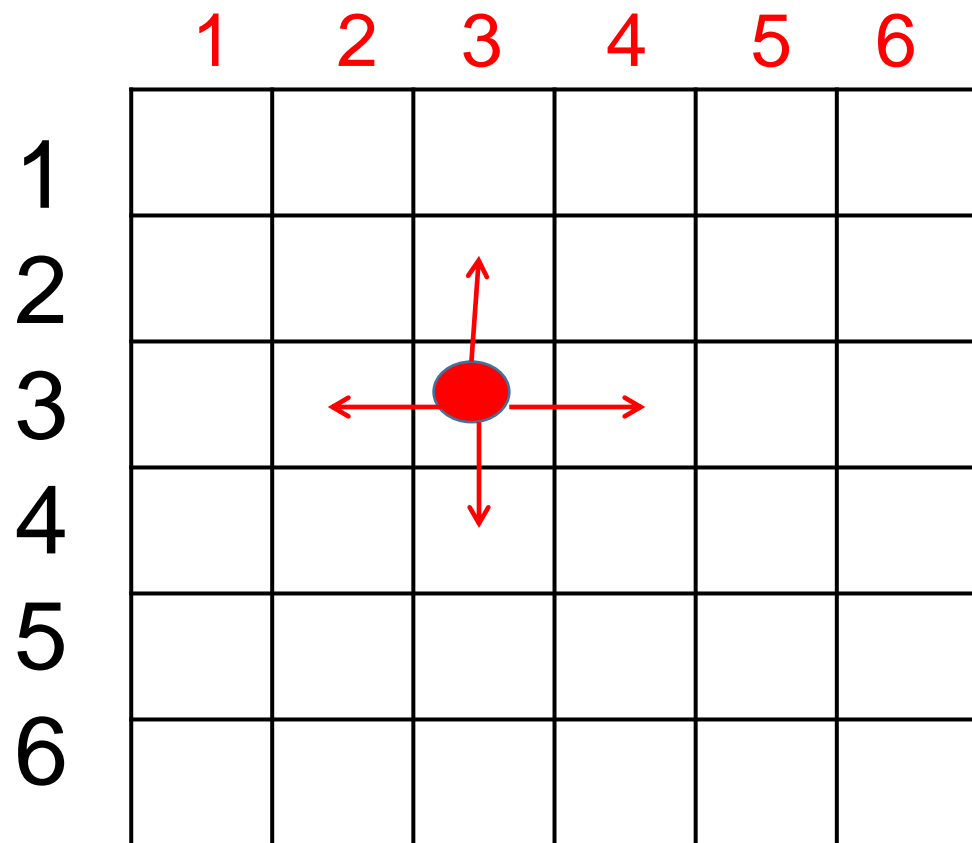
4 int main() {
5     cin>>n;
6     int t=0;
7     for(k=2; k<=(1); k++) { //枚举定值
8         if(k%2) //判断方向
9             for(i=1; i<=n; i++) { //枚举行i
10                 j=(2); //计算列 j
11                 if(j>=1&&j<=n) //约束条件
12                     a[i][j]=++t;
13             }
14         else
15             for(i=(3); i>=1; i--) {
16                 j=k-i;
17                 if(j>=1&&j<=n)
18                     a[i][j]=++t;
19             }
20     }
21     for(i=1; i<=n; i++) { //输出
22         for(j=1; j<=n; j++)
23             cout<<a[i][j]<<" ";
24         cout<<endl;
25     }
}

```

三、矩阵中点与四周的关系

方便描述：矩阵行列都从[1][1]开始

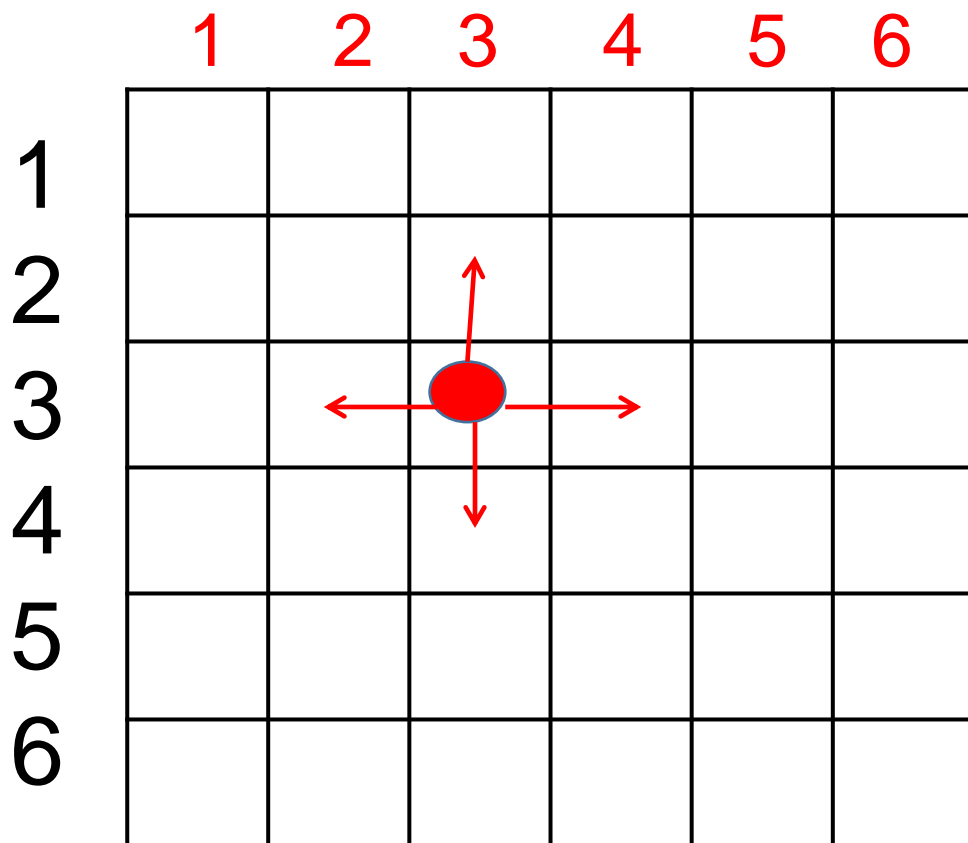
1、四个方向（上下左右）



$A[3][3]$

$A[x][y]$

确定点与 4 个位置关系



一维表示法:

$Dx[4] = \{-1, 1, 0, 0\}$

$Dy[4] = \{0, 0, -1, 1\}$ 对应: 上下左右

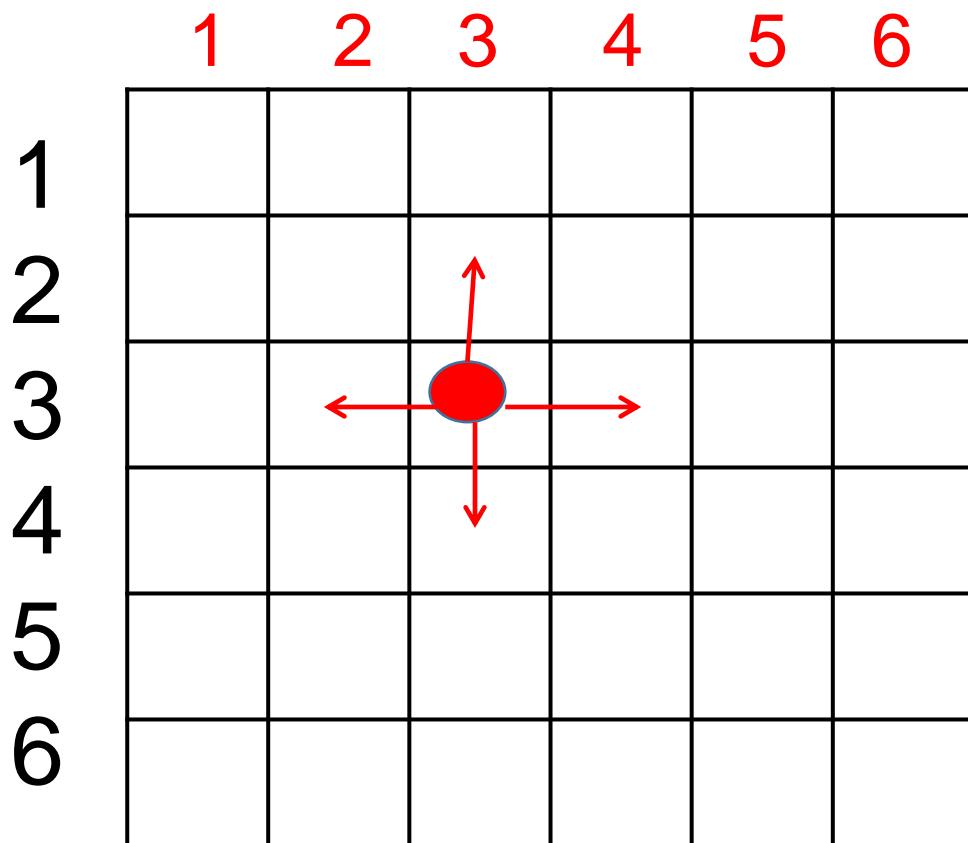
0 1 2 3

For (int i=0; i<4; i++) { //枚举4个方向

$x = x + dx[i]; y = y + dy[i];$

}

确定点与 4 个位置关系：



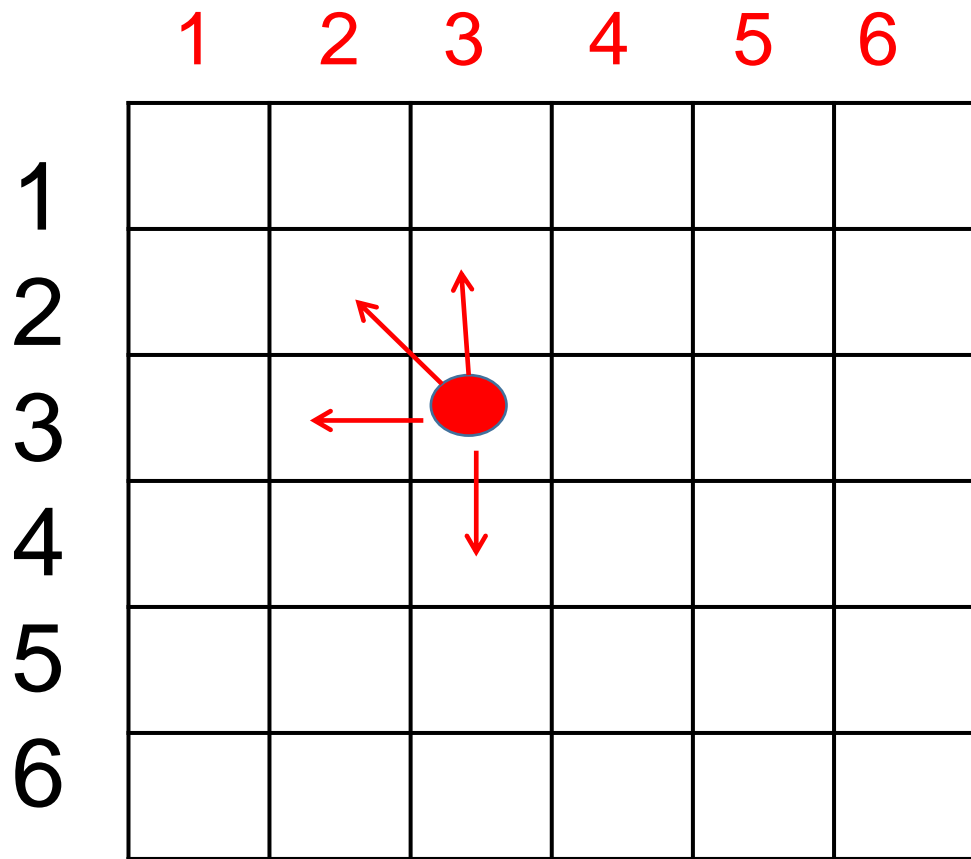
二维表示法：

`dir[][2]={ {0,-1},{0,1},{-1,0},{1,0} };`

```
For (int i=0; i<4; i++) { //枚举4个方向
    x=x+dir[i][0]; y=y+dir[i][1];
}
```

```
int dir[4][2]={{-1,0}, {1,0}, {0,-1}, {0,1} };
int a[100][100];
int main() {
    int n;
    cin>>n;
    for(int i=1;i<=n;i++){ ... }
    int x,y; // 2,3
    cin>>x>>y;
    for(int i=0;i<4;i++){ //四个方向
        int dx=x+dir[i][0];
        int dy=y+dir[i][1];
        cout<<a[dx][dy]<<" ";
    }
}
```

2、八个方向



$A[x][y]$ 8方向二维表示法:

$dir[][2] = \{ \{1,0\}, \{0,1\}, \{0,-1\}, \{-1,0\}, \{1,1\}, \{-1,-1\}, \{-1,1\}, \{1,-1\} \};$

```
For (int i=0; i<8; i++) { //枚举8个方向
```

```
    x=x+dir[i][0]; y=y+dir[i][1];
```

```
}
```

例6：图像模糊处理

给定 n 行 m 列的图像各像素点的灰度值，要求用如下方法对其进行模糊化处理：

- 1、四周最外侧的像素点灰度值不变；
- 2、中间各像素点新灰度值为该像素点及其上下左右相邻四个像素点原灰度值的平均（舍入到最接近的整数）。

输入：第一行包含两个整数 n 和 m ，表示图像包含像素点的行数和列数。 $1 \leq n \leq 100$ ， $1 \leq m \leq 100$ 。

接下来 n 行，每行 m 个整数，表示图像的每个像素点灰度。相邻两个整数之间用单个空格隔开，每个元素均在 $0 \sim 255$ 之间。

输出： n 行，每行 m 个整数，为模糊处理后的图像。相邻两个整数之间用单个空格隔开。

样例输入

```
4 5
100 0 100 0 50
50 100 200 0 0
50 50 100 100 200
100 100 50 50 100
```

样例输出

```
100 0 100 0 50
50 80 100 60 0
50 80 100 90 200
100 100 50 50 100
```

分析：

- 1、新的图像根据原图上下左右的值变化而来。
- 2、如果用1个数组，变化后会覆盖一些值，怎么办？用2个数组？
一个原数据，一个变化后的数据
- 3、输出

参考：

- 1、读入原数组数据
- 2、处理：读一个位置，判断是否灰度化.... 并在另一数组标记处理后的值。
- 3、输出

填空：

```
3  int a[101][101],b[101][101],n,m;
4  int dx[4]= {-1,1,0,0},dy[4]= {0,0,-1,1};//预处理4个方向
5  int main() {
6      cin>>n>>m;
7      for(int i=1; i<=n; i++)//输入数据
8          for(int j=1; j<=m; j++)cin>>a[i][j];
9      for(int i=1; i<=n; i++)//构造b数据
10         for(int j=1; j<=m; j++) {
11             if((i==1)|| (1)) {//四周的数不变
12                 b[i][j]=a[i][j];
13             } else {
14                 int s=(2);//加上原位置值
15                 for(int k=0; (3); k++)
16                     s+=a[(4)][(5)];//加上上下左右的值
17                 b[i][j]=round(s/5.0);
18             }
19             //也可以用下式
20             // b[i][j]=round((a[i][j]+a[i-1][j]+a[i+1][j]+a[i][j-1]+a[i][j+1])/5.0);
21         }
22         for(int i=1; i<=n; i++) {
23             for(int j=1; j<=m; j++)
24                 cout<< (6) <<" ";//输出
```

例7：杨辉三角

输入一个数 n

输出杨辉三角的前 n 行

输入

3

输出：

1

1 1

1 2 1

例8：扫雷地图

输入格式：

两个正整数 n, m ($3 \leq n < m \leq 40$) n 表示矩阵行列, m 表示雷的数量。

以下 m 行, 每行两个整数, 表示有地雷的位置 x, y

输出格式：

$n \times n$ 的矩阵, 每个数字用空格分隔;

样例输入：

```
3 2
```

```
2 1
```

```
3 3
```

样例输出：

```
1 1 0
```

```
- 1 2 1
```

```
1 2 - 1
```

例9：规则矩阵

$N \times M$ 的矩阵，如何确定有几条从左至右的斜线？

$$k1=1-m \quad k2=n-1$$

本课小结：

- 1、对角线 位置上 i,j 的关系
 - 2、矩阵斜线上的点与 i,j 的关系
 - 3、矩阵中的点与其他相邻位置的关系
-