



三、作业

每次课程结束后请登录 oj.shiyancang.cn ，请参照主页 OJ 使指南（教师版）

每次课程需在 OJ 上布置课程作业

作业截至时间后，

1. 请老师认真检查和评估学生的作业完成情况和完成质量，进行作业点评，根据完成情况针对性的提供辅导和调整教学；
2. 请老师手动发布学员作业成绩，作业成绩发布后，系统会自动给每一个学员家长发送成绩短信；确保每位家长及时了解学员学习情况；

3. RelPlay:

作业成绩发布后，系统会自动将已截至的作业设置为 RelPlay 模式，即为订正模式，时间为 15 天，老师需要求学员在 15 天内完成错题的 RelPlay，**四 E 班五 F 班将 RelPlay 计入升班考核。**请各位老师重视 Relplay 及 Relplay 结果。督促学员完成订正。

要求：

课前：

- 1、建好本课文件夹（如：day1）
- 2、整理好上一节课的问题。
- 3、预习本节内容，查看OJ作业

课后：

- 1、完成OJ 作业，巩固知识点
- 2、完成 replay 错题，录制 codeshow
- 3、多交流，学习更好的方法。

主要内容：



实验舱
青少年编程
走近科学 走进名校

- 1、二维数组应用
- 2、string[] 字符数组
- 3、递归调用
- 4、结构体应用
- 5、指针、链表、
- 6、高精度计算
- 7、map 、 set 应用



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（1）

zlj

2022.8

什么是数据结构？

计算机存储、组织数据的方式

程序 = 数据结构 + 算法

样例输入：

3 5

1 2 3 4 5

3 2 1 4 7

2 4 2 1 2

1、一排数据如何存储？

2 32 5 8 1 10.....

a[6]



2、若干排数据如何存储？

2 3 4 9 7 8

4 3 1 7 8 10

2 1 4 8 9 2

.....

a[6]

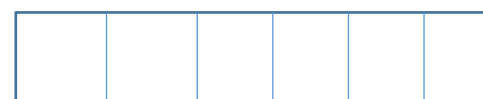
A[0]



A[0] [6]

b[6]

A[1]



A[1] [6]

c[6]

A[2]



A[2] [6]

一、二维数组

1、定义：

类型名 数组名[下标1][下标2]

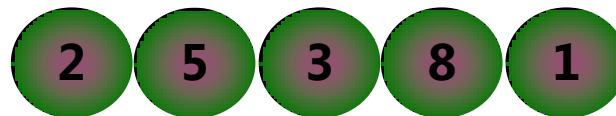
行标 列标

例：int a[4][6]

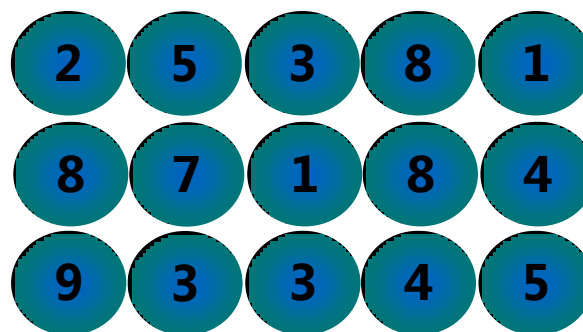
其逻辑结构是：**矩阵**

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

一维的数据：



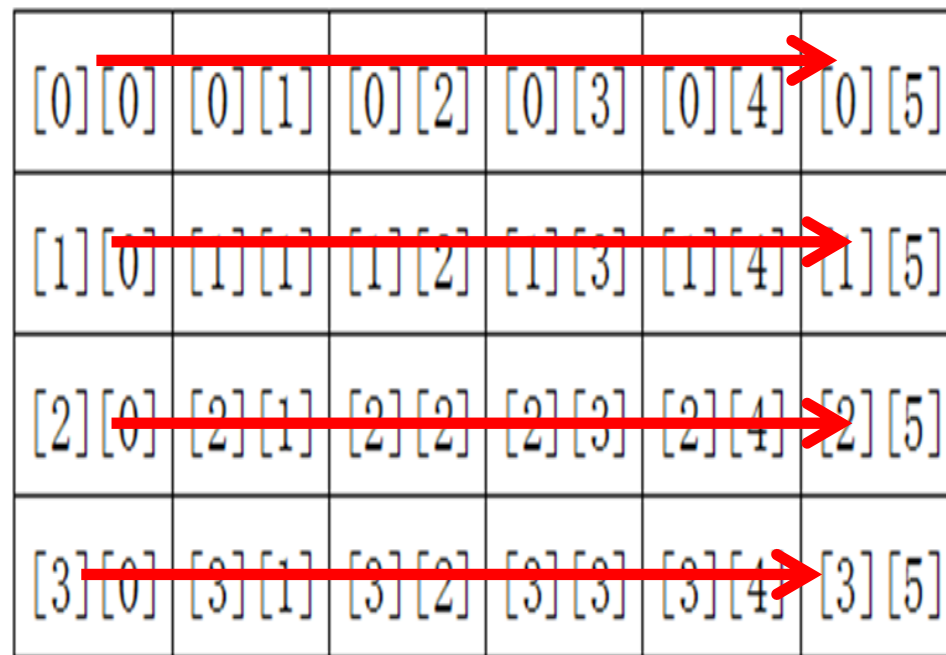
二维的数据呢：



2、二维数组的数据读入

按行顺序读取：

```
for (int i=0;i<4;i++) // 行
    for (int j=0;j<6;j++) // i行的每个数
        cin >>a[ i ][ j ];
```



The diagram illustrates a 4x6 2D array with rows indexed 0 to 3 and columns indexed 0 to 5. Red arrows indicate the row-major traversal order, starting from the top-left element [0][0] and moving horizontally across each row before jumping to the start of the next row.

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

二维数组的数据输出

1、按行顺序输出：

```
for (int i=0;i<4;i++){  
    for (int j=0;j<6;j++)  
        cout <<a[ i][ j]<<" ";  
    cout <<endl;  
}
```

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

二维数组按列 输出

2、按列顺序输出:

```
for (int i=0;i<6;i++){  
    for (int j=0;j<4;j++)  
        cout <<a[  ][  ]<<" ";  
    cout <<endl;  
}
```



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

练习：输入数据，按样例输出数据

【输入样例】

3 3

2 4 9

0 7 4

2 8 5

【输出样例】

2 8 5

0 7 4

2 4 9

思考：反序输出呢？

【输入样例】

3 3

2 4 9

0 7 4

2 8 5

【输出样例】

5 8 2

4 7 0

9 4 2

练习2: 输出矩阵1 （先列输出）

输入n行m列矩阵，（n, m均不超过100）按样例要示输出。

【输入样例】

3 4

2 4 9 1

0 7 4 6

2 8 5 8

【输出样例】

2 0 2

4 7 8

9 4 5

1 6 8

练习3: 输出矩阵2

输入n行m列矩阵，（n,m均不超过100）按样例要求输出。

【输入样例】

3 4

2 4 9 1

0 7 4 3

2 8 5 7

【输出样例】

1 3 7

9 4 5

4 7 8

2 0 2

练习4: 输出矩阵3

输入n行m列矩阵，（n,m均不超过100）按样例输出。

【输入样例】

3 3

2 4 9

0 7 4

2 8 5

【输出样例】

5 4 9

8 7 4

2 0 2

二维的读入与输出小结：

- 1、输入数据程序结构是？
- 2、行列顺序可以颠倒吗？
- 3、两个下标与平面上点的坐标有何相似处？

$A(x,y)$ 与 $A[x][y]$

3、二维数组应用注意：

1)、初始化

例：

```
int a[4][2]={{1,0},{0,1},{-1,0},{0,-1}};
```

1	0
0	1
-1	0
0	-1

给数组赋初值0

方法一

```
memset(a,0,sizeof(a))
```

方法二

在main()函数之前定义数组

2)、物理存储(内存地址)

存储器其实没有二维的概念，逻辑上只有一维，各储存单元按顺序排下去。二维数组在存储时根据从上到下，从左到右的顺序存放。如右图所示：

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]

a代表整个
二维数组的
首地址

a	0	a[0][0]
	1	a[0][1]
	2	a[0][2]
	3	a[0][3]
	4	a[1][0]
	5	a[1][1]
	6	a[1][2]
	7	a[1][3]
	8	a[2][0]
	9	a[2][1]
	10	a[2][2]
	11	a[2][3]

思考：你现在知道二维数组声明时二维长度为啥不能省略吗？

二维数组可以看作是特殊的一维数组，它的元素又是一维数组
如二维数组 `int a[3][4];` 可分解为三个一维数组，其数组名分别为：

a[0]				
a[1]				
a[2]				

对这三个一维数组不需另作说明，直接使用。这三个一维数组都有4个元素，例如：一维数组 `a[0]` 的元素为：
`a[0][0]`, `a[0][1]`, `a[0][2]`, `a[0][3]`。

思考 : int a[3][4];

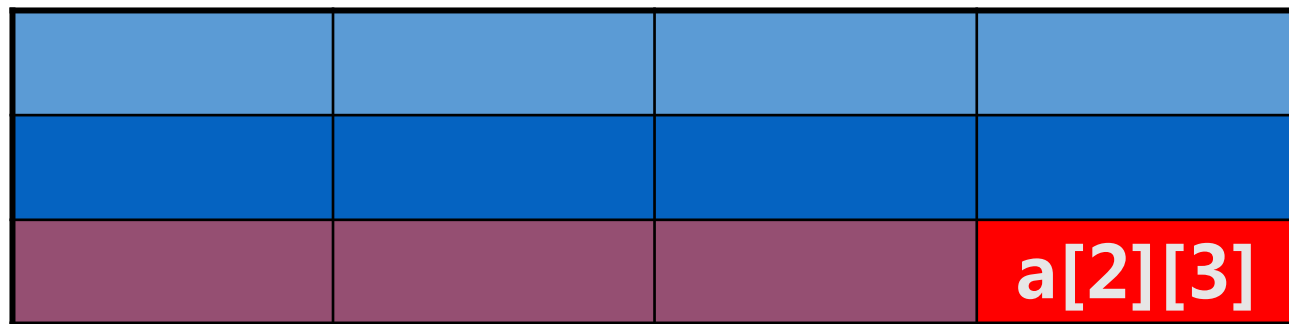
cout<<sizeof(a)<<" "<<sizeof(a[0])<<endl;输出值是多少？

48 16

因为a是二维数组所以sizeof(a)输出为48，从另外的角度看，它是一个3个一维数组，每个元素占16个字节，所以输出48.

a[0]是有4个元素的一维数组，每个元素占4个字节，所以sizeof(a[0])输出16

思考：编译器是如何计算int a[3][4]中a[2][3] 地址的？



先计算a[2]的地址： $a + 2 * (\text{sizeof}(\text{int}) * 4)$ ；

然后再计算a[2][3]的地址： $a[2] + 3 * \text{sizeof}(\text{int})$;

即： $a + 2 * (\text{sizeof}(\text{int}) * 4) + 3 * \text{sizeof}(\text{int})$;

思考：二维数组a[3][4]中，

地址：a a[0] &a[0][0] 代表的含义你知道吗？

			a[2][3]

二、二维数组实战应用

1、二维数组的**顺序**遍历、查找

按行优先原则

```
for(int i=0;i<n;i++)//先从行开始  
    for(int j=0;j<n;j++) //再从列依次读取/输出  
        cin>>a[i][j];
```

例1：《寻找最大值》

从 $n*n$ 的整数矩阵中寻找一个最大值，输出它的**位置**及值。

如：

5

1 2 2 1 2

5 6 7 8 3

9 3 0 5 3

7 2 1 4 6

3 0 8 2 4

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

输出：Max=9

3 1

（输出解释：最大值是9，第3行1列）

补全程序：

```
3  int a[101][101],n;
4  int main() {
5      cin>>n;
6      for(int i=1; i<=n; i++) //读入数据
7          for(int (1) j++)
8              cin>> (2) ;
9      int Max=(3),x=1,y=1;
10     for(int i=1; i<=n; i++) //打擂台找最值
11         for(int j=1; j<=n; j++)
12             if(a[i][j]>Max) {
13                 Max=a[i][j];
14                 (4) ;
15             }
16     cout<<"max="<<Max<<endl<<x<<" "<<y<<endl;
```

例2：稀疏矩阵

大部分元素是0的矩阵称为稀疏矩阵，如果用二维数组存放稀疏矩阵会极大的浪费存储空间，所以通常用数字所在的位置记录数据。

【输入】

第一行2个整数， n 和 m 。第二行开始是 $n*m$ 的稀疏矩阵。

【输出】

简记形式的稀疏矩阵。

【样例输入】

```
3 5
0 0 0 0 5
0 0 4 0 0
1 0 0 0 1
```

【样例输入】

```
1 5 5
2 3 4
3 1 1
3 5 1
```

稀疏矩阵

```
#include<bits/stdc++.h>
using namespace std;

int main()
{
    int n,m,x;
    cin>>n>>m;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
        {
            cin >>x;
            if (x!=0) cout <<i<<" "<<j<<" "<<x<<endl;
        }
    return 0;
}
```

例：图像相似度

给出两幅相同大小的黑白图像（用0-1矩阵）表示，求它们的相似度。

说明：若两幅图像在相同位置上的像素点颜色相同，则称它们在该位置具有相同的像素点。两幅图像的相似度定义为相同像素点数占总像素点数的百分比。

输入

第一行包含两个整数 m 和 n ，表示图像的行数和列数，中间用单个空格隔开。 $1 \leq m \leq 100, 1 \leq n \leq 100$ 。

之后 m 行，每行 n 个整数0或1，表示第一幅黑白图像上各像素点的颜色。相邻两个数之间用单个空格隔开。

之后 m 行，每行 n 个整数0或1，表示第二幅黑白图像上各像素点的颜色。相邻两个数之间用单个空格隔开。

输出

一个实数，表示相似度（以百分比的形式给出），精确到小数点后两位。

样例输入

```
3 3
1 0 1
0 0 1
1 1 0

1 1 0
0 0 1
0 0 1
```

图像相似度

```
3  int m,n,x,y,a[101][101],b[101][101];
4  int main() {
5      cin>>m>>n;
6      for(int i=1; i<=m; i++)//读入a数组
7          for(int j=1; j<=n; j++)
8              cin>>a[i][j];
9      int s(1);
10     for(int i=1; i<=m; i++) //读入b数组
11         for(int j=1; j<=n; j++) {
12             cin>>b[i][j];
13             if((2))s++;//统计相似度
14         }
15     printf("%.2lf\n",100.0*(3));//注意精度及百分比
16     return 0;
```

2、二维数组下标 与数据位置的关系

$A[i][j]$: i 相同表示同行
 j 相同表示同列

例3：《最大差值》

输入n行m列数据，查找每行最大值并求和，查找每列最小值并求和，输出他们的差值。（ $n, m < 100$ ）以上所有数字均为非负整数。

【输入样例】

3 3

2 4 9

0 7 4

2 8 5

【输出样例】

16

样例说明：每行最大值的和： $9+7+8$

每列最小值的和： $0+4+4$

$24-8=16$

伪代码：

- 1、按行读入数据；
 - 2、求第X行的最大值函数 `int fmax(int x){ }`
 - 3、求第X列的最小值函数 `int fmin (int x){ }`
 - 4、调用函数求N行的最大值和
 - 5、调用函数求N列的最小值和
 - 6、输出 `max-min`
-

参考代码:

```
3 int a[105][105],n,m;
4 int fmax(int x) { //找 x 行的最大值
5     int Max=a(1);
6     for(int j=1; j<=(2); j++)
7         Max=max(Max,a[x][j]);
8     return Max;
9 }
10 int fmin(int y) { //找 y 列的最小值
11     int Min=a[1][y];
12     for(int i=1; (3); i++)
13         Min=min(Min,a[i][y]);
14     return Min;
15 }
```

```
16 int main() {
17     cin>>n>>m;
18     for(int i=1; i<=n; i++) //读入数据
19         for(int j=1; j<=m; j++) cin>>a[i][j];
20     int s1=0,s2=0;
21     for(int i=1; i<=n; i++) //每一行的最大值和
22         s1=(4);
23     for(int i=1; i<=m; i++) //每一列的最小值和
24         s2=s2+(5);
25     cout<<s1-s2<<endl;
26     return 0;
}
```

例4：《计算矩阵边缘元素之和》

输入一个整数矩阵，计算位于矩阵边缘的元素之和。所谓矩阵边缘的元素，就是第一行和最后一行的元素以及第一列和最后一列的元素。

输入：第一行分别为矩阵的行数 m 和列数 n （ $m < 100$ ， $n < 100$ ），两者之间以一个空格分开。

接下来输入的 m 行数据中，每行包含 n 个整数，整数之间以一个空格分开。

输出：对应矩阵的边缘元素和

样例输入

3 3

3 4 1

3 7 1

2 0 1

样例输出

15

3	4	1
3	7	1
2	0	1

分析：

- 1、第一行+最后一行+第一列+最后一列？
 - 2、四边上的点位置有没有规律可找？
-

算法1:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int a[110][110];
    int m,n,s=0;
    cin>>m>>n;
    for (int i=1;i<=m;i++)
        for (int j=1;j<=n;j++)
            cin>>a[i][j];
    for (int i=1;i<=n;i++)
    {
        s+=a[1][i];
        s+=a[m][i];
    }
    for (int i=2;i<m;i++)
    {
        s+=a[i][1];
        s+=a[i][n];
    }
    cout<<s<<endl;
    return 0;
}
```

算法2填空：根据行列关系边读边处理

```
3  int m,n,a[105][105];
4  int main() {
5      cin>>m>>n;
6      int s=0;
7      for(int i=1; i<=m; i++)
8          for(int j=1; j<=n; j++) {
9              cin>>a[i][j]; //边读边处理
10             if(i==1 || i==m || (1))
11                 s=(2);
12         }
13     cout<<s<<endl;
```

3、下标灵活应用获取数据

根据下标：输出、查找、比较、重组、填

充相关数据

例6 《填数》

将1到 $n \times n$ 排成一个正方形方阵，用一个小正方形框出 $m \times m$ 个数字，然后求和。
例如，将连续自然数1到 7×7 排成方阵，求出2 2 3即：起点下标是：2 2， 3×3 的一个矩形数据区：9、10、11、16、17、18、23、24、25的和。

【输入说明】

第一行是n，表示以下是n行n列数字矩阵，第二行是数据块起点的下标及m。以上所有数字均为整数。

【输出说明】

一个整数。

【输入样例】

4

3 1 2

【输出样例】

46

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	32	33	34	35
36	37	38	39	40	41	42
43	44	45	46	47	48	49

分析:

- 1、构造数组
- 2、小矩阵求和
- 3、输出

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

```
for(int i=x; i<x+m; i++) //i  
    for(int j=y; j<y+m; j++)  
        s=s+a[i][j];
```

例7：数据块

输入n行m列数据，求出其中某一块数据的和并输出。

【输入说明】

第一行是n、m，以下是n行m列数字矩阵，最后一行是数据块起点、终点的下标。以上所有数字均为整数。

【输出说明】

一个整数。

【输入样例】

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

【输出样例】

```
17
```

输出说明： $0+7+2+8=17$

同上例找到数据块的行列起始即可

```
3  int n,m,a[100][100];
4  int main() {
5      cin>>n>>m;
6      for(int i=1; i<=n; i++)
7          for(int j=1; j<=m; j++)
8              cin>>a[i][j];
9      int x,y,x2,y2,s=0;
10     cin>>x>>y>>x2>>y2;
11     for(int i=x; (1); i++) { //块区间的行
12         for(int j=y; (2); j++) //列
13             s=s+a[i][j];
14     }
15     cout<<s<<endl;
```

例9：计算鞍点

给定一个5*5的矩阵，每行只有一个最大值，每列只有一个最小值，寻找这个矩阵的鞍点。

鞍点指的是矩阵中的一个元素，它是所在行的最大值，并且是所在列的最小值。

例如：在下面的例子中（第4行第1列的元素就是鞍点，值为8）。

输入

输入包含一个5行5列的矩阵

输出

如果存在鞍点，输出鞍点所在的行、列及其值，

如果不存在，输出 “not found”

样例输入

```
11 3 5 6 9
12 4 7 8 10
10 5 6 9 11
8 6 4 7 2
15 10 11 20 25
```

样例输出

```
4 1 8
```

填空：

```
3 int a[10][10];
4 int fmax(int x) { //找 x行最大值
5     int maxh=a[x][1];
6     for(int i=2; i<=5; i++)
7         maxh=(1);
8     return maxh;
9 }
10 int fmin(int y) { //找 y列最大值
11     int minh=a[1][y];
12     for(int i=2; i<=5; i++)
13         minh=(2);
14     return minh;
15 }
```

```
16 int main() {
17     for(int i=1; i<=5; i++)
18         for(int j=1; j<=5; j++) cin>>a[i][j];
19     for(int i=1; i<=5; i++)
20         for(int j=1; j<=5; j++)
21             if(fmax(i)==a[i][j]&&(3)) {
22                 cout<<i<<" "<<j<<" "<<a[i][j]<<endl;
23                 return 0;
24             }
25     cout<<"not found"<<endl;
26     return 0;
}
```

例10：《数组清零》

二维数组小结：

- 1、二维数组数据结构的特点？
 - 2、如何输入、输出、遍历数组？
 - 3、如何灵活运用行列下标：查找、比较、计算、重构数据？
-