

# cpp22 二维数组

## 1 二维数组的定义和使用

- 二维数组的声明: 数据类型 数组名[维度1的大小][维度2的大小];

```
int a[3][5];
```

表示a是二维数组，共有3×5=15 个元素。

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

- 但是实际内部数据存储的仍然是连续数据，只是逻辑上可以理解为二维结构罢了
- 二维数组的引用: 数组名[下标1][下标2];

```
int a[3][5];
```

共有3×5=15 个元素，它们是：

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]

- 二维数组的初始化方式：（前面两种建议使用，第三种不建议使用）  
可以将每一行分开来写在各自的括号里：

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

建议采用

或：

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

建议采用

也可以把所有数据写在一个括号里：

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

不建议采用

- 多维数组：

三维数组a：`int a[100][3][5];`

四维数组b：`int b[100][100][3][5];`

多维的数组引用赋值等操作与二维数组类似。

2 二维数组的输入和输出（注意要写双层循环）

- 问题描述：

<p><b>【问题描述】</b></p> <ul style="list-style-type: none"><li>• 输入一个 <math>M \times N</math> 的矩阵。</li><li>• 输出这个矩阵。</li></ul>	<p><b>【数据范围】</b></p> <p><math>1 \leq M, N \leq 10</math></p>
<p><b>【输入格式】</b></p> <ul style="list-style-type: none"><li>• 第一行包含两个整数 <math>n</math> 和 <math>m</math>，表示矩阵的行数和列数。</li><li>• 接下来 <math>n</math> 行，每行 <math>m</math> 个整数，表示矩阵的元素。相邻两个整数之间用单个空格隔开，每个元素均在 <math>1 \sim 1000</math> 之间。</li></ul>	<p><b>【输入样例】</b></p> <pre>3 4 1 2 3 4 5 6 7 8 9 10 11 12</pre>
<p><b>【输出格式】</b></p> <ul style="list-style-type: none"><li>• <math>n</math> 行，每行 <math>m</math> 个整数，为输入的矩阵。相邻两个整数之间用单个空格隔开。</li></ul>	<p><b>【输出样例】</b></p> <pre>1 2 3 4 5 6 7 8 9 10 11 12</pre>

- 代码示例：

```
1.  #include<iostream>
2.  using namespace std;
3.  const int M = 10;
4.  const int N = 10;
5.  int a[M][N];
6.  int main()
7.  {
8.      int m, n;
9.      cin >> m >> n;
10.     //输入矩阵元素
11.     for (int i = 0; i < m; i++)
12.     {
13.         for (int j = 0; j < n; j++)
14.         {
15.             cin >> a[i][j];
16.         }
17.     }
18.     //输出矩阵元素
19.     for (int i = 0; i < m; i++)
20.     {
21.         for (int j = 0; j < n; j++)
22.         {
23.             cout << a[i][j] << " ";
24.         }
25.         cout << endl;
26.     }
27.     return 0;
28. }
```

### 3 矩阵的行列互换（观察输出的下标范围）

• 问题描述

<p><b>【问题描述】</b></p> <ul style="list-style-type: none"><li>• 输入一个n行m列的矩阵。按照行列互换的形式输出新的矩阵。</li><li>• 行列互换：输入是第i行j列的元素，输出是第j行i列的位置。</li></ul>	<p><b>【数据范围】</b></p> <p><math>1 \leq n, m \leq 10</math></p>
<p><b>【输入格式】</b></p> <ul style="list-style-type: none"><li>• 第一行包含两个整数n和m，表示矩阵的行数和列数。</li><li>• 接下来n行，每行m个整数，表示矩阵A的元素。相邻两个整数之间用单个空格隔开，每个元素均在1~1000之间。</li></ul>	<p><b>【输入样例】</b></p> <pre>3 4 1 2 3 4 5 6 7 8 9 10 11 12</pre>
<p><b>【输出格式】</b></p> <ul style="list-style-type: none"><li>• 第一行，两整数，分别是输出矩阵的行数和列数。</li><li>• 接下来m行，每行n个整数，为矩阵的转置。</li><li>• 相邻两个整数之间用单个空格隔开。</li></ul>	<p><b>【输出样例】</b><pre>4 3 1 5 9 2 6 10 3 7 11 4 8 12</pre></p>

• 代码示例：

```
1. #include<iostream>
2. using namespace std;
3. const int M = 10;
4. const int N = 10;
5. int a[M][N];
6. int main()
7. {
8.     int m, n;
9.     cin >> m >> n;
10.    //输入矩阵元素
11.    for (int i = 0; i < m; i++)
12.    {
13.        for (int j = 0; j < n; j++)
14.        {
15.            cin >> a[i][j];
16.        }
17.    }
18.    //输出新矩阵的行数和列数
19.    cout << n << " " << m << endl;
20.    //输出矩阵元素
21.    for (int i = 0; i < n; i++)
22.    {
23.        for (int j = 0; j < m; j++)
24.        {
25.            cout << a[j][i] << " ";
26.        }
27.        cout << endl;
28.    }
29.    return 0;
30. }
```

### 4 矩阵对角线

• 问题描述：

<p><b>【问题描述】</b></p> <ul style="list-style-type: none"><li>• 已知一个<math>N \times N</math>的方阵（最大不超过10），把方阵两条对角线上的元素值加上10，然后输出这个新矩阵。</li></ul>	
<p><b>【输入样例】</b></p> <pre>3 1 1 1 1 1 1 1 1 1</pre>	<p><b>【输出样例】</b></p> <pre>11 1 11 1 11 1 11 1 11</pre>

• 算法分析：

- 每个方阵都有两条对角线，想一想如何确定对角线的元素。
- 主对角线：  $i = j$
- 副对角线：  $i + j = n - 1$

	0	1	2	3	4	5	6
0							
1							
2							
3							
4							
5							
6							

• 代码示例：

```
#include<iostream>
using namespace std;
const int N = 10;
int a[N][N];
int main()
{
    int n;
    cin >> n;
    //输入矩阵元素
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            cin >> a[i][j];
        }
    }

    //更改对角线上元素的值
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            //寻找对角线的特征
            if ( (i == j) || (i + j == n - 1) )
            {
                a[i][j] += 10;
            }
        }
    }

    //输出矩阵元素
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            cout << a[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```