# 二维数组的声明

当一维数组元素的类型也是一维数组时，便构成了“数组的数组”，即二维数组。

二维数组声明的格式如下：

```
1. 数据类型 数组名[维度 1 大小][维度 2 大小];
```

其中维度 1 和维度 2 的大小都必须由常量定义。例如：

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

则表示 a 是二维数组，共有 3×5=15 个元素。这 15 个元素可以看成一个有 3 行、5 列的表格。

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

我们也可以把这个表格称为矩阵（matrix）。

# 二维数组的引用

二维数组的数组元素引用与一维数组元素引用类似，区别在于二维数组元素的引用必须给出两个下标。

引用的格式为：

```
1. 数组名[下标1][下标2]
```

显然，每个下标表达式取值不应超出下标所指定的范围，否则会导致致命的越界错误。

设有定义：

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

则表示 a 是二维数组，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] |

第 1 行第 1 列对应数组元素 a[0][0]。

a[2][3]即表示第 3 行第 4 列的元素。

第 n 行第 m 列对应数组元素 a[n-1][m-1]。

# 二维数组的初始化

二维数组的初始化和一维数组类似。

**【方法 1】**

可以将每一行分开来写在各自的括号里

```
int d1[4][2] =
{
    {1, 0},
    {0, 1},
    {-1, 0},
    {0, -1}
};
```

或

```
int d2[4][2] = { {1, 0}, {0, 1}, {-1, 0}, {0, -1} };
```

**【方法 2】**

可以把所有数据写在一个括号里

```
int d3[4][2] = {1, 0, 0, 1, -1, 0, 0, -1};
```

一般情况下，采用第一种方式进行初始化，更方便人来阅读。

# 多维数组

下标的个数并不局限在一个或二个，可以任意多个。

当定义的数组下标有多个时，我们称为多维数组。

如定义一个三维数组 a:

```
1.  int a[100][3][5];
```

如定义一个四维数组 b:

```
1.  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。

# 矩阵的输入输出

**【问题描述】**

输入一个$M \times N$的矩阵。

输出这个矩阵。

**【输入格式】**

第一行包含两个整数 n 和 m，表示矩阵的行数和列数。

接下来 n 行，每行 m 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 1~1000 之间。

**【输出格式】**

n 行，每行 m 个整数，为输入的矩阵。相邻两个整数之间用单个空格隔开。

**【数据范围】**

$$1 \leq M, N \leq 10$$

**【输入样例】**

```
3 4
1 2 3 4
5 6 7 8
9 10 11 12
```

**【输出样例】**

```
1 2 3 4
5 6 7 8
9 10 11 12
```

**【参考程序】**

```
#include<iostream>
using namespace std;
const int M = 10;
const int N = 10;
int a[M][N];
int main()
```

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

# 矩阵的行列互换

**【问题描述】**

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。

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。

**【输入格式】**

第一行包含两个整数 n 和 m，表示矩阵的行数和列数。

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。

**【输出格式】**

第一行，两整数，分别是输出矩阵的行数和列数。

接下来 m 行，每行 n 个整数，为矩阵的转置。

相邻两个整数之间用单个空格隔开。

**【数据范围】**

$$1 \leq n,m \leq 10$$

**【输入样例】**

```
3 4
1 2 3 4
5 6 7 8
9 10 11 12
```

**【输出样例】**

```
4 3
1 5 9
2 6 10
3 7 11
4 8 12
```

**【参考程序】**

```
#include<iostream>
```

```
using namespace std;
const int M = 10;
const int N = 10;
int a[M][N];
int main()
{
    int m, n;
    cin >> m >> n;
    //输入矩阵元素
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            cin >> a[i][j];
        }
    }
    //输出新矩阵的行数和列数
    cout << n << " " << m << endl;
    //输出矩阵元素
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            cout << a[j][i] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

# 矩阵对角线

**【问题描述】**

已知一个 N×N 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。

**【输入样例】**

```
3
1 1 1
1 1 1
1 1 1
```

**【输出样例】**

```
11 1 11
1 11 1
11 1 11
```

**【分析】**

每个方阵都有两条对角线，难点在于对角线的元素怎么确定。

**【参考程序】**

```
#include<iostream>
using namespace std;
const int N = 10;
int main()
{
    int a[N][N];
    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++)
```

```
19.     {
20.         for (int j=0; j<n; j++)
21.         {
22.             //寻找对角线的特征
23.             if ( (i==j) || (i+j==n-1) )
24.             {
25.                 a[i][j] += 10;
26.             }
27.         }
28.     }
29.     //输出矩阵元素
30.     for (int i=0; i<n; i++)
31.     {
32.         for (int j=0; j<n; j++)
33.         {
34.             cout << a[i][j] << " ";
35.         }
36.         cout << endl;
37.     }
38.     return 0;
39. }
```