二维数组

关键词：

- 顺序初始化
- 按行初始化
- 初始化可以省略第一维参数
- 内存中顺序存储
- 二维数组传递需要大小和列的数量

二维数组的声明和初始化

二维数组的声明和一维数组是类似的，不同之处只是多了一个下标：

数据类型数组名[行数][列数]；

要注意，二维数组的下标也都是从0开始的。

二维数组的初始化分为两种，一种是顺序初始化，一种是按行初始化，我们来看一段程序，就能够对它们有所了解了：

```
#include "iostream.h"
#include "iomanip.h"
int main()
{
   int array13={4,2,5,6};//顺序初始化
   int array23={{4,2},{5},{6}};//按行初始化
   cout <<"array1" <<endl;
   for (int i=0;i<3;i++)//输出数组array1
   {
      for (int j=0;j<2;j++)
     {
         cout <<setw(2) <<array1i;
      }
      cout <<endl;
   }
   cout <<"array2" <<endl;
   for (int k=0;k<3;k++)//输出数组array2
   {
      for (int l=0;l<2;l++)
      {
         cout <<setw(2) <<array2k;
      }
      cout <<endl;
   }
   return 0;
}
```

运行结果：

```
array1
4 2
5 6
13 4
array2
4 2
5 8
6 8
```

我们可以看出，

所谓按顺序初始化就是先从左向右再由上而下地初始化，即第一行所有元素都初始化好以后再对第二行初始化。而按行初始化则是用一对大括号来表示每一行，跳过前一行没有初始化的元素，在行内从左向右地进行初始化。

对于没有初始化的元素，则都是一个不确定的值。

省略第一维的大小

我们在第一节学到，一维数组的大小可以省略。可是二维数组的元素个数是行数和列数的乘积，如果我们只告诉电脑元素个数，电脑无法知道究竟这个数组是几行几列。所以，C++规定，在声明和初始化一个二维数组时，只有第一维（行数）可以省略。比如：

int array[][3]={1,2,3,4,5,6};

相当于：

int array[2][3]={1,2,3,4,5,6};

二维数组在内存中的存储情况

先前已经说明，内存是依靠地址来确定内存中的唯一一个存储单元的，即只有一个参数。所以在内存中，所有的数据都是像一维数组那样顺序存储的。那么具有两个下标的二维数组是怎样存放到内存中的呢？

在内存中，先将二维数组的第一行按顺序存储，接着就是第二行的数据，然后是第三行的数据.....

向函数传递二维数组

我们知道，数组作为参数传递给函数的是数组首元素的地址。对于二维数组来说亦是如此。不过有两个问题，

一个是我们必须让函数知道行数和列数

，这就像我们要让函数知道一维数组的大小一样，防止发生越界访问。

另一个就是我们必须让电脑知道这个二维数组是怎样的一个表格，即必须告知数组的列数。

这和只能省略二维数组的行数道理是一样的。下面我们就来看一个向函数传递二维数组的程序：

```
#include "iostream.h"
#include "iomanip.h"
void disp(int a,int r,int c);//告知数组的列数
int main()
{
   int array3={4,2,5,6,3,1};
   cout <<"array" <<endl;
   disp(array,3,2);
   return 0;
}
void disp(int a,int r,int c)
{
   for (int i=0;i<r;i++)
   {
      for (int j=0;j<c;j++)
      {
        cout <<setw(2) <<ai;
      }
      cout <<endl;
   }
}
运行结果：
array
4 2
5 6
3 1
```