# 5、 数组

## 5.1 概述

所谓数组，就是一个集合，里面存放了相同类型的数据元素

**特点1：**数组中的每个数据元素都是相同的数据类型

**特点2：**数组是由连续的内存位置组成的


## 5.2 一维数组

### 5.2.1 一维数组定义方式

**一维数组定义的三种方式：**

1. `数据类型 数组名[ 数组长度 ];`
2. `数据类型 数组名[ 数组长度 ] = { 值1, 值2 ...};`
3. `数据类型 数组名[ ] = { 值1, 值2 ...};`

**示例**

```
#include<iostream>
#include<ctime> //time系统时间头文件
using namespace std;

int main()
{
```

```cpp

    //数组
    /*

    1. 数据类型  数组名[ 数组长度 ];
    2. 数据类型  数组名[ 数组长度 ] = { 值1, 值2 ...};
    3. 数据类型  数组名[ ] = { 值1, 值2 ...};
    */
    //1. 数据类型  数组名[数组长度];
    int arr[5];
    //给数组中的元素进行赋值
    //数组元素的下标是从0开始索引的
    arr[0] = 10;
    arr[1] = 20;
    arr[2] = 30;
    arr[3] = 40;
    arr[4] = 50;
    //访问数组元素
    cout << arr[0] << endl;
    cout << arr[1] << endl;
    cout << arr[2] << endl;
    cout << arr[3] << endl;
    cout << arr[4] << endl;

    // 2. 数据类型  数组名[ 数组长度 ] = { 值1, 值2 ...};
    //如果在初始化数据的时候，没有全部填写完，会用0进行填充剩余的数据
    int arr2[5] = { 10,20,30,40,50 };
    //cout << arr2[0] << endl;
    //cout << arr2[1] << endl;
    //cout << arr2[2] << endl;
    //cout << arr2[3] << endl;
    //cout << arr2[4] << endl;
    for (int i = 0; i < 5; i++)
    {
        cout << arr2[i] << endl;
    }
    //3. 数据类型  数组名[] = { 值1, 值2 ... };
    //定义数组的时候，必须有初始长度
    int arr3[] = { 10,20,30,40,50 };
    for (int j = 0; j < 5; j++)
    {
        cout << arr2[j] << endl;
    }

    system("pause");
    return 0;
}
```

> 总结1：数组名的命名规范与变量名命名规范一致，不要和变量重名

> 总结2：数组中下标是从0开始索引

## 5.2.2 一维数组数组名

**一维数组名称的用途：**

1. **可以统计整个数组在内存中的长度**
2. **可以获取数组在内存中的首地址**

**示例：**

```
#include<iostream>
#include<ctime> //time系统时间头文件
using namespace std;
int main()
{
    //数组名用途
    //1、可以统计整个数组在内存的长度
    int arr[10] = { 1,2,3,4,5,6,7,8,9,10 };
    cout << "整个数组占用内存空间为：" << sizeof(arr) << endl;
    cout << "每个元素占用内存空间：" << sizeof(arr[0]) << endl;
    cout << "数组中元素的个数：" << sizeof(arr) / sizeof(arr[0]) << endl;

    //2、可以获取数组在内存中的首地址
    cout << "数组首地址：" << arr << endl;
    cout << "数组中第一个元素地址为：" <<&arr[0]<< endl;

    //数组名是常量，不可以进行赋值操作
    system("pause");
    return 0;
}
```

> 注意：数组名是常量，不可以赋值

> 总结1：直接打印数组名，可以查看数组所占内存的首地址

> 总结2：对数组名进行sizeof，可以获取整个数组占内存空间的大小

**练习案例1**：五只小猪称体重

**案例描述：**

在一个数组中记录了五只小猪的体重，如：int arr[5] = {300,350,200,400,250};

找出并打印最重的小猪体重。

**思想：**访问数组中每个元素，如果这个元素比我认定的最大值要打，更新最大值。

```
#include<iostream>
#include<ctime> //time系统时间头文件
using namespace std;
int main()
{
    // 1、创建5只小猪体重的数组
    int arr[5] = {300,350,200,400,250};

    // 2、从数组中找出最大值
```

```cpp
    //先假定一个最大值，arr[0]
    int max = 0;
    for (int i = 0; i < 5; i++)
    {
        //cout << arr[i] << endl;
        //如果访问的数组中元素比我认定的最大值还要大，更新最大值
        if (arr[i]>max)
        {
            max = arr[i];
        }

    }
    // 3、打印最大值
    cout << "最重的小猪体重为：" << max << endl;
    system("pause");
    return 0;
}
```

**练习案例2：**数组元素逆置

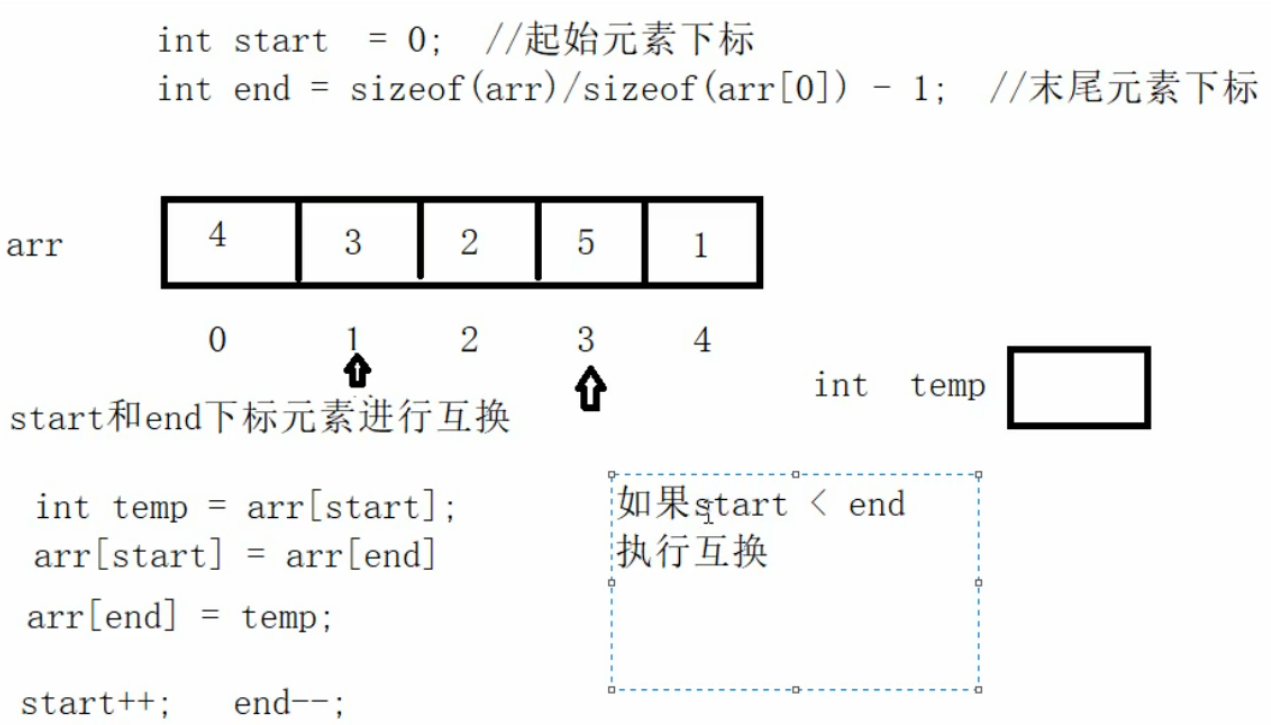
**案例描述：**请声明一个5个元素的数组，并且将元素逆置.

(如原数组元素为：1,3,2,5,4;逆置后输出结果为:4,5,2,3,1);

int start = 0;

int end = sizeof(arr)/sizeof(arr[0]) -1; //末尾元素下标

start和end下标元素进行互换，还需要一个零时的变量



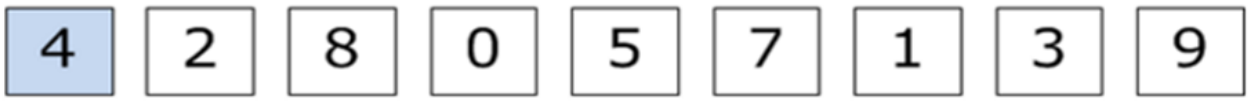

```cpp
#include<iostream>
#include<ctime> //time系统时间头文件
using namespace std;
```

```cpp
int main()
{
    //实现数组元素逆置
    // 1、创建数组
    int arr[5] = { 1,2,3,4,5 };
    cout << "元素数组逆置前结果: " << endl;
    for (int i = 0; i < 5; i++)
    {
        cout << arr[i] << endl;
    }

    // 2、实现逆置
    // 2.1 记录起始下标位置
    // 2.2 记录结束下标位置
    // 2.3 记录起始下标与结束下标的元素互换
    // 2.4 起始位置++，结束位置--
    // 2.5 循环执行2.1操作，知道起始位置>=结束位置
    int start = 0; //起始下标
    int end = sizeof(arr) / sizeof(arr[0]) - 1; //结束下标

    while (start<end)
    {
        //实现元素互换
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;

        //下标更新
        start++;
        end--;

    }

    // 3、打印逆置后的数组
    cout << "数组元素逆置后结果: " << endl;
    for (int i = 0; i < 5; i++)
    {
        cout << arr[i] << endl;
    }

    system("pause");
    return 0;
}
```

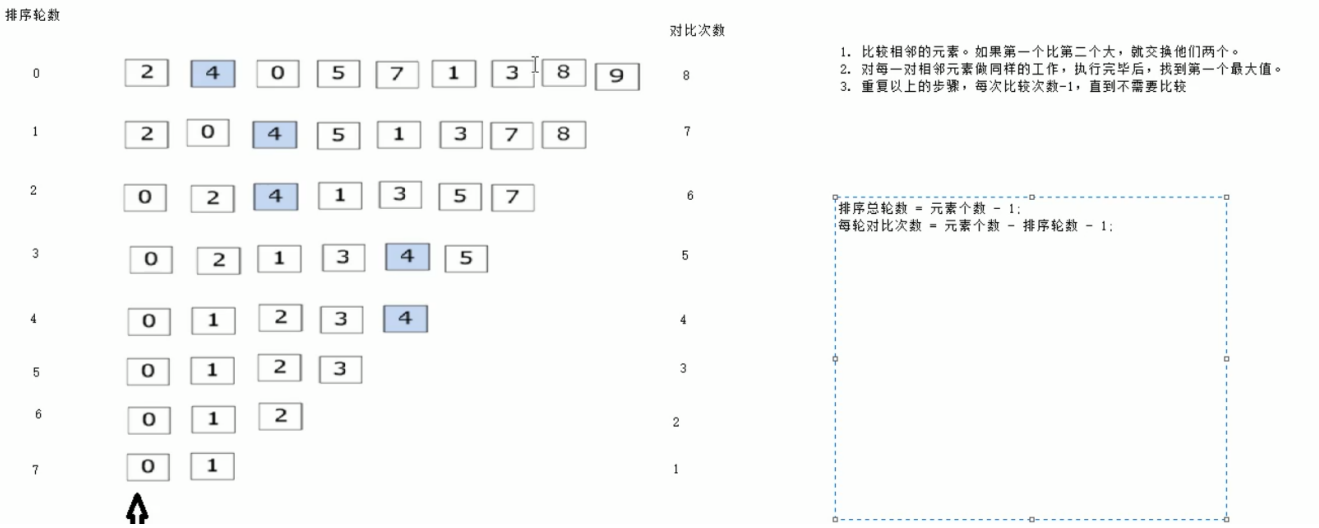
## 5.2.3 冒泡排序

**作用：** 最常用的排序算法，对数组内元素进行排序

1. 比较相邻的元素。如果第一个比第二个大，就交换他们两个。
2. 对每一对相邻元素做同样的工作，执行完毕后，找到第一个最大值。
3. 重复以上的步骤，每次比较次数-1，直到不需要比较




**示例： 将数组 { 4,2,8,0,5,7,1,3,9 } 进行升序排序**

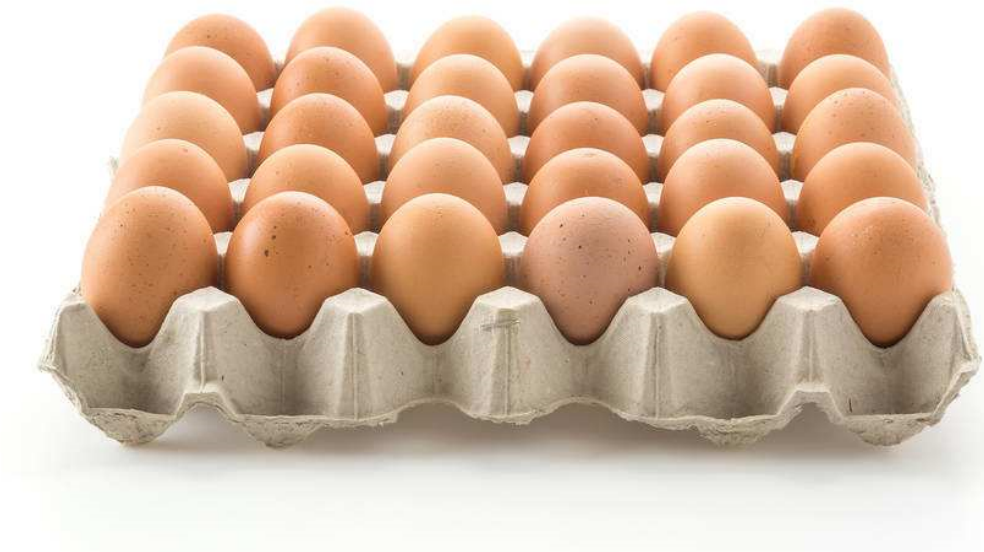



```cpp
#include<iostream>
#include<ctime> //time系统时间头文件
using namespace std;
int main()
{
    //利用冒泡排序实现升序序列
    int arr[9] = { 4,2,8,0,5,7,1,3,9 };
    cout << "排序前: " << endl;
    for (int i = 0; i < 9; i++)
    {
        cout << arr[i] << "\t";
    }
    //开始排序
    // 总共排序轮数为: 元素个数-1
    for (int i = 0; i < 9-1; i++)
    {
        //内层循环对比次数 = 元素个数-当前轮数-1
        for (int j = 0; j < 9-i-1; j++) //sizeof(arr)/size(arr[0])
        {
            //如果第一个数字，比第二个数字大，交换两个数字
            if (arr[j] > arr[j + 1])
            {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
    //排序后结果
    cout << "排序后结果: " << endl;
    for (int i = 0; i < 9; i++)
```

```
    {
        cout << arr[i] << "\t";
    }

    cout << endl;
    system("pause");
    return 0;
}
```

## 5.3 二维数组

二维数组就是在一维数组上，多加一个维度。



### 5.3.1 二维数组定义方式

二维数组定义的四种方式：

1. 数据类型 数组名[ 行数 ][ 列数 ];
2. 数据类型 数组名[ 行数 ][ 列数 ] = { {数据1, 数据2 } , {数据3, 数据4 } };
3. 数据类型 数组名[ 行数 ][ 列数 ] = { 数据1, 数据2, 数据3, 数据4};
4. 数据类型 数组名[ ][ 列数 ] = { 数据1, 数据2, 数据3, 数据4};

> 建议：以上4种定义方式，利用第二种更加直观，提高代码的可读性

**示例：**

```
#include<iostream>
#include<ctime> //time系统时间头文件
```

```cpp
using namespace std;
int main()
{
    //二维数组定义方式

    /*
    1. 数据类型  数组名[ 行数 ][ 列数 ];
    2. 数据类型  数组名[ 行数 ][ 列数 ] = { {数据1, 数据2 } , {数据3, 数据4 } };
    3. 数据类型  数组名[ 行数 ][ 列数 ] = { 数据1, 数据2, 数据3, 数据4};
    4. 数据类型  数组名[  ][ 列数 ] = { 数据1, 数据2, 数据3, 数据4};
    */
    //1. 数据类型  数组名[ 行数 ][ 列数 ];
    int arr[2][3]; // 2行3列数组
    arr[0][0] = 1;
    arr[0][1] = 2;
    arr[0][2] = 3;
    arr[1][0] = 4;
    arr[1][1] = 5;
    arr[1][2] = 6;
    cout << "输出每一个元素: " << endl;
    //外层循环打印行数，内层循环打印列数
    for (int i = 0; i < 2; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            cout << arr[i][j]<<endl;

        }
    }

    //2. 数据类型  数组名[行数][列数] = { {数据1, 数据2 } , {数据3, 数据4 } };
    int arr2[2][3] =
    {
        {1,2,3},
        {4,5,6},
    };
    for (int i = 0; i < 2; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            cout << arr2[i][j]<<" ";

        }
        cout << endl;
    }

    //3. 数据类型  数组名[行数][列数] = { 数据1, 数据2, 数据3, 数据4 };
    int arr3[2][3] = { 1,2,3,4,5,6 };
    for (int i = 0; i < 2; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            cout << arr3[i][j] << " ";
```

```
        }
        cout << endl;
    }

    //4. 数据类型  数组名[][列数] = { 数据1, 数据2, 数据3, 数据4 };
    int arr4[][3] = {1,2,3,4,5,6};
    for (int i = 0; i < 2; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            cout << arr4[i][j] << " ";

        }
        cout << endl;
    }
    system("pause");
    return 0;
}
```

> 总结：在定义二维数组时，如果初始化了数据，可以省略行数

## 5.3.2 二维数组数组名

- 查看二维数组所占内存空间
- 获取二维数组首地址

**示例：**

```
#include<iostream>
#include<ctime> //time系统时间头文件
using namespace std;
int main()
{
    //二维数组名称用途
    //1、可以查看占用的内存空间大小
    int arr[2][3] =
    {
        {1,2,3},
        {4,5,6}
    };
    cout << "二维数组占用的内存空间大小: " << sizeof(arr) << endl;
    cout << "二维数组第一行占用内存为: " << sizeof(arr[0]) << endl; //0表示行号
    cout << "二维数组第一个元素占用内存为: " << sizeof(arr[0][0]) << endl;

    cout << "二维数组的行数为: " << sizeof(arr) / sizeof(arr[0]) << endl; // 行数
    cout << "二维数组的列数为: " << sizeof(arr[0]) / sizeof(arr[0][0]) << endl; //列数

    //2、可以查看二维数组的首地址
    cout <<"二维数组的首地址为: "<< arr << endl;
    cout << "二维数组第一行首地址为: " << arr[0] << endl;
    cout << "二维数组第二行首地址为: " << arr[1] << endl;
```

```
    cout << "二维数组第一个元素首地址： " << &arr[0][0] << endl;// 具体元素的地址，需要加一个取地
符
    system("pause");
    return 0;
}
```

> 总结1：二维数组名就是这个数组的首地址

> 总结2：对二维数组名进行sizeof时，可以获取整个二维数组占用的内存空间大小

## 5.3.3 二维数组应用案例

**考试成绩统计：**

案例描述：有三名同学（张三，李四，王五），在一次考试中的成绩分别如下表，**请分别输出三名同学的总成绩**

| | 语文 | 数学 | 英语 |
|---|---|---|---|
| 张三 | 100 | 100 | 100 |
| 李四 | 90 | 50 | 100 |
| 王五 | 60 | 70 | 80 |

1、创建一个二维3行3列数组

2、统计考试成绩，让每行的3列数据相加，统计出来一个综合

```
#include<iostream>
#include<ctime> //time系统时间头文件
#include<string>
using namespace std;
int main()
{
    //二维数组的案例--考试成绩统计
    int score[3][3] =
    {
        {100,100,100},
        {90,50,100},
        {60,70,80}

    };

    string names[3] = { "张三","李四","王五" };
    //2、统计每个人的总和分数
    for (int i = 0; i < 3; i++)
    {
        int sum = 0; // 统计分数总和的变量
        for (int j = 0; j < 3; j++)
        {
            sum += score[i][j];
            //cout << score[i][j] << "\t";
        }
```

```
        cout << names[i] << "个人的总分为: " << sum << endl;
        cout << endl;

    }
    system("pause");
    return 0;
}
```