

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

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4
5  int main()
6  {
```

```

7
8 //数组
9 /*
10
11 1. 数据类型 数组名[ 数组长度 ];
12 2. 数据类型 数组名[ 数组长度 ] = { 值1, 值2 ... };
13 3. 数据类型 数组名[ ] = { 值1, 值2 ... };
14 */
15 //1. 数据类型 数组名[数组长度];
16 int arr[5];
17 //给数组中的元素进行赋值
18 //数组元素的下标是从0开始索引的
19 arr[0] = 10;
20 arr[1] = 20;
21 arr[2] = 30;
22 arr[3] = 40;
23 arr[4] = 50;
24 //访问数组元素
25 cout << arr[0] << endl;
26 cout << arr[1] << endl;
27 cout << arr[2] << endl;
28 cout << arr[3] << endl;
29 cout << arr[4] << endl;
30
31 // 2. 数据类型 数组名[ 数组长度 ] = { 值1, 值2 ... };
32 //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
33 int arr2[5] = { 10,20,30,40,50 };
34 //cout << arr2[0] << endl;
35 //cout << arr2[1] << endl;
36 //cout << arr2[2] << endl;
37 //cout << arr2[3] << endl;
38 //cout << arr2[4] << endl;
39 for (int i = 0; i < 5; i++)
40 {
41     cout << arr2[i] << endl;
42 }
43 //3. 数据类型 数组名[] = { 值1, 值2 ... };
44 //定义数组的时候, 必须有初始长度
45 int arr3[] = { 10,20,30,40,50 };
46 for (int j = 0; j < 5; j++)
47 {
48     cout << arr2[j] << endl;
49 }
50
51 system("pause");
52 return 0;
53 }

```

总结1: 数组名的命名规范与变量命名规范一致, 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：

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4  int main()
5  {
6      //数组名用途
7      //1、可以统计整个数组在内存的长度
8      int arr[10] = { 1,2,3,4,5,6,7,8,9,10 };
9      cout << "整个数组占用内存空间为：" << sizeof(arr) << endl;
10     cout << "每个元素占用内存空间：" << sizeof(arr[0]) << endl;
11     cout << "数组中元素的个数：" << sizeof(arr) / sizeof(arr[0]) << endl;
12
13     //2、可以获取数组在内存中的首地址
14     cout << "数组首地址：" << arr << endl;
15     cout << "数组中第一个元素地址为：" << &arr[0] << endl;
16
17     //数组名是常量，不可以进行赋值操作
18     system("pause");
19     return 0;
20 }
```

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。

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4  int main()
5  {
6      // 1、创建5只小猪体重的数组
7      int arr[5] = {300,350,200,400,250};
8
9      // 2、从数组中找出最大值
```

```

10 //先假定一个最大值, arr[0]
11 int max = 0;
12 for (int i = 0; i < 5; i++)
13 {
14     //cout << arr[i] << endl;
15     //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
16     if (arr[i]>max)
17     {
18         max = arr[i];
19     }
20 }
21 // 3、打印最大值
22 cout << "最重的小猪体重为: " << max << endl;
23 system("pause");
24 return 0;
25 }

```

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);

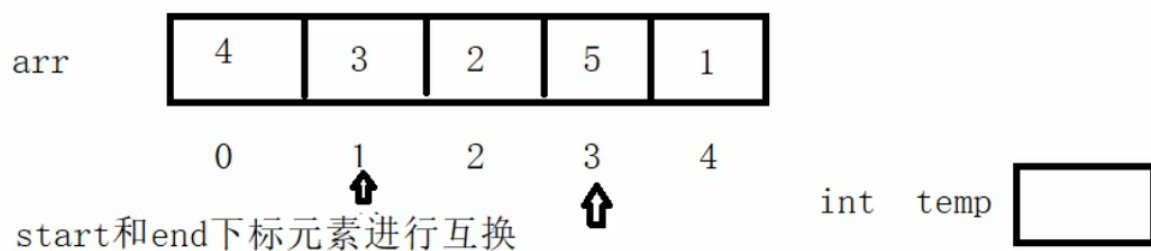
int start = 0;

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

int start = 0; //起始元素下标

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



```

int temp = arr[start];
arr[start] = arr[end]
arr[end] = temp;

start++; end--;

```

如果start < end
执行互换

```

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

```

```

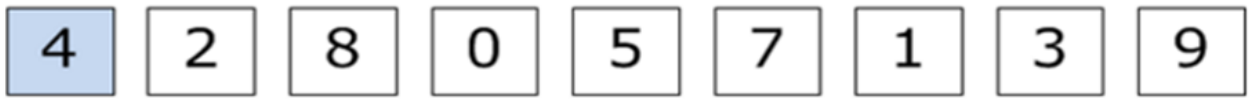
4  int main()
5  {
6      //实现数组元素逆置
7      // 1、创建数组
8      int arr[5] = { 1,2,3,4,5 };
9      cout << "元素数组逆置前结果: " << endl;
10     for (int i = 0; i < 5; i++)
11     {
12         cout << arr[i] << endl;
13     }
14
15     // 2、实现逆置
16     // 2.1 记录起始下标位置
17     // 2.2 记录结束下标位置
18     // 2.3 记录起始下标与结束下标的元素互换
19     // 2.4 起始位置++, 结束位置--
20     // 2.5 循环执行2.1操作, 知道起始位置>=结束位置
21     int start = 0; //起始下标
22     int end = sizeof(arr) / sizeof(arr[0]) - 1; //结束下标
23
24     while (start<end)
25     {
26         //实现元素互换
27         int temp = arr[start];
28         arr[start] = arr[end];
29         arr[end] = temp;
30
31         //下标更新
32         start++;
33         end--;
34
35     }
36
37     // 3、打印逆置后的数组
38     cout << "数组元素逆置后结果: " << endl;
39     for (int i = 0; i < 5; i++)
40     {
41         cout << arr[i] << endl;
42     }
43
44     system("pause");
45     return 0;
46 }

```

5.2.3 冒泡排序

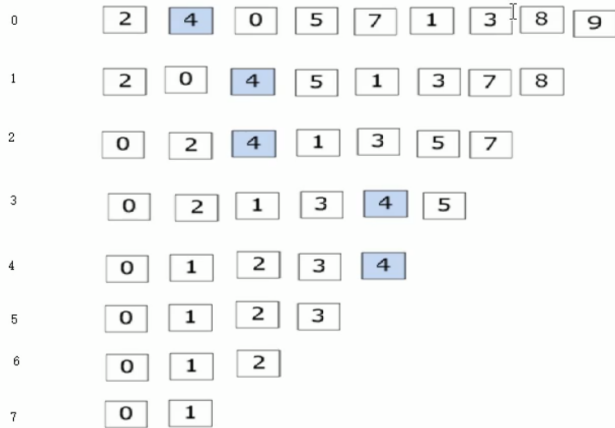
作用：最常用的排序算法，对数组内元素进行排序

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

排序轮数



对比次数

8
7
6
5
4
3
2
1

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

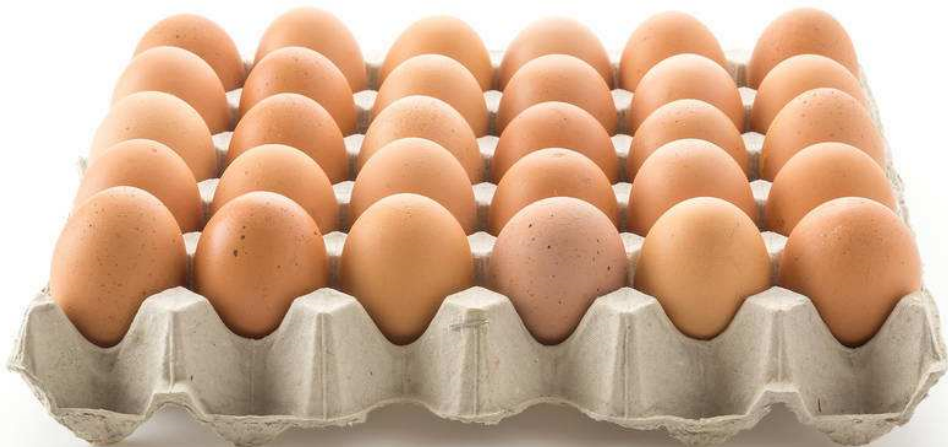
排序总轮数 = 元素个数 - 1;
每轮对比次数 = 元素个数 - 排序轮数 - 1;

```
1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4  int main()
5  {
6      //利用冒泡排序实现升序序列
7      int arr[9] = { 4,2,8,0,5,7,1,3,9 };
8      cout << "排序前: " << endl;
9      for (int i = 0; i < 9; i++)
10     {
11         cout << arr[i] << "\t";
12     }
13     //开始排序
14     // 总共排序轮数为: 元素个数-1
15     for (int i = 0; i < 9-1; i++)
16     {
17         //内层循环对比次数 = 元素个数-当前轮数-1
18         for (int j = 0; j < 9-i-1; j++) //sizeof(arr)/size(arr[0])
19         {
20             //如果第一个数字, 比第二个数字大, 交换两个数字
21             if (arr[j] > arr[j + 1])
22             {
23                 int temp = arr[j];
24                 arr[j] = arr[j + 1];
25                 arr[j + 1] = temp;
26             }
27         }
28     }
29     //排序后结果
30     cout << "排序后结果: " << endl;
31     for (int i = 0; i < 9; i++)
```

```
32     {
33         cout << arr[i] << "\t";
34     }
35
36     cout << endl;
37     system("pause");
38     return 0;
39 }
```

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：

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

```

3 using namespace std;
4 int main()
5 {
6     //二维数组定义方式
7
8     /*
9     1. 数据类型 数组名[ 行数 ][ 列数 ];
10    2. 数据类型 数组名[ 行数 ][ 列数 ] = { {数据1, 数据2 } , {数据3, 数据4 } };
11    3. 数据类型 数组名[ 行数 ][ 列数 ] = { 数据1, 数据2, 数据3, 数据4};
12    4. 数据类型 数组名[ ][ 列数 ] = { 数据1, 数据2, 数据3, 数据4};
13    */
14    //1. 数据类型 数组名[ 行数 ][ 列数 ];
15    int arr[2][3]; // 2行3列数组
16    arr[0][0] = 1;
17    arr[0][1] = 2;
18    arr[0][2] = 3;
19    arr[1][0] = 4;
20    arr[1][1] = 5;
21    arr[1][2] = 6;
22    cout << "输出每一个元素: " << endl;
23    //外层循环打印行数, 内层循环打印列数
24    for (int i = 0; i < 2; i++)
25    {
26        for (int j = 0; j < 3; j++)
27        {
28            cout << arr[i][j]<<endl;
29
30        }
31    }
32
33    //2. 数据类型 数组名[行数][列数] = { {数据1, 数据2 } , {数据3, 数据4 } };
34    int arr2[2][3] =
35    {
36        {1,2,3},
37        {4,5,6},
38    };
39    for (int i = 0; i < 2; i++)
40    {
41        for (int j = 0; j < 3; j++)
42        {
43            cout << arr2[i][j]<<" ";
44
45        }
46        cout << endl;
47    }
48
49    //3. 数据类型 数组名[行数][列数] = { 数据1, 数据2, 数据3, 数据4 };
50    int arr3[2][3] = { 1,2,3,4,5,6 };
51    for (int i = 0; i < 2; i++)
52    {
53        for (int j = 0; j < 3; j++)
54        {
55            cout << arr3[i][j] << " ";

```



```

56     }
57     cout << endl;
58 }
59
60
61 //4. 数据类型 数组名[][列数] = { 数据1, 数据2, 数据3, 数据4 };
62 int arr4[][3] = {1,2,3,4,5,6};
63 for (int i = 0; i < 2; i++)
64 {
65     for (int j = 0; j < 3; j++)
66     {
67         cout << arr4[i][j] << " ";
68     }
69     cout << endl;
70 }
71 system("pause");
72 return 0;
73 }

```

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：

```

1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  using namespace std;
4  int main()
5  {
6      //二维数组名称用途
7      //1、可以查看占用的内存空间大小
8      int arr[2][3] =
9      {
10         {1,2,3},
11         {4,5,6}
12     };
13     cout << "二维数组占用的内存空间大小: " << sizeof(arr) << endl;
14     cout << "二维数组第一行占用内存为: " << sizeof(arr[0]) << endl; //0表示行号
15     cout << "二维数组第一个元素占用内存为: " << sizeof(arr[0][0]) << endl;
16
17     cout << "二维数组的行数为: " << sizeof(arr) / sizeof(arr[0]) << endl; // 行数
18     cout << "二维数组的列数为: " << sizeof(arr[0]) / sizeof(arr[0][0]) << endl; //列数
19
20     //2、可以查看二维数组的首地址
21     cout << "二维数组的首地址为: "<< arr << endl;
22     cout << "二维数组第一行首地址为: " << arr[0] << endl;
23     cout << "二维数组第二行首地址为: " << arr[1] << endl;

```

```

24     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
25     system("pause");
26     return 0;
27 }

```

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

```

1  #include<iostream>
2  #include<ctime> //time系统时间头文件
3  #include<string>
4  using namespace std;
5  int main()
6  {
7      //二维数组的案例--考试成绩统计
8      int score[3][3] =
9      {
10         {100,100,100},
11         {90,50,100},
12         {60,70,80}
13     };
14
15     string names[3] = { "张三", "李四", "王五" };
16     //2、统计每个人的总和分数
17     for (int i = 0; i < 3; i++)
18     {
19         int sum = 0; // 统计分数总和的变量
20         for (int j = 0; j < 3; j++)
21         {
22             sum += score[i][j];
23             //cout << score[i][j] << "\t";
24         }
25     }

```

```
26         cout << names[i] << "个人的总分为: " << sum << endl;  
27         cout << endl;  
28  
29     }  
30     system("pause");  
31     return 0;  
32 }
```