一维数组

2023年9月10日 22:40

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 ...};

```
//定义方式3
//数据类型 数组名[] = {值1, 值2 , 值3 ...};
int score3[] = { 100,90,80,70,60,50,40,30,20,10 };

for (int i = 0; i < 10; i++)
{
    cout << score3[i] << endl;
}

system("pause");

return 0;
```

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

2：数组中下标是从0开始索引

一维数组数组名的用途

2023年9月12日 21:06

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

```
int main() {

    //数组名用途
    //1、可以获取整个数组占用内存空间大小
    int arr[10] = { 1,2,3,4,5,6,7,8,9,10 };

    cout << "整个数组所占内存空间为：  " << sizeof(arr) << endl;
    cout << "每个元素所占内存空间为：  " << sizeof(arr[0]) << endl;
    cout << "数组的元素个数为：  " << sizeof(arr) / sizeof(arr[0]) << endl;

    //2、可以通过数组名获取到数组首地址
    cout << "数组首地址为：  " << (int)arr << endl;
    cout << "数组中第一个元素地址为：  " << (int)&arr[0] << endl;
    cout << "数组中第二个元素地址为：  " << (int)&arr[1] << endl;

    //arr = 100; 错误，数组名是常量，因此不可以赋值


    system("pause");

    return 0;
```

数组练习案例

2023年9月12日 21:17

五只小猪称体重案例

2023年9月12日 21:17

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。

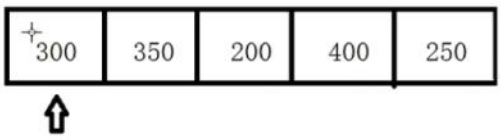
练习案例2：数组元素逆置

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 arr[5] = {300, 350, 200, 400, 250}
```

```
int max = 0;
```



访问数组中每个元素，如果这个元素比我认定的最大值要大，更新最大值

```cpp
#include<iostream>
using namespace std;
int main() {
    int arr[5] = { 300,350,400,200,340 };
    int max = 0;
    for (int i = 0; i< 5; i++) {
        //cout << arr[i] << endl;
        if (arr[i] > max) {
            max = arr[i];
        }

    }
    cout << max << endl;



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

位置互换案例

2023年9月12日 21:17

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。

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);

```
//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，直到起始位置 >= 结束位置
```

```cpp
#include<iostream>
using namespace std;
int main() {
    int arr[5] = { 300,350,400,200,340 };
    cout << "数组逆置前的位置：" << endl;
    for (int i = 0; i < 5; i++) {
        cout << arr[i]<<"  ";}
    cout << endl;
    int start = 0;
    int end = sizeof(arr) / sizeof(arr[0]) - 1;
    while (start < end) {
        int temp = arr[start];  //12与14行代码的作用：交换首尾的值
        arr[start] = arr[end];
        arr[end] = temp;
        start++;end--;}  //更新下标
    cout << "数组逆置后的位置；" << endl;
    for (int j = 0; j < 5; j++) {
        cout << arr[j] << "  ";}
    system("pause");
    return 0;
}
```

★ 牢记12到15行代码及其含义

冒泡排序

2023年9月12日　　22:25

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

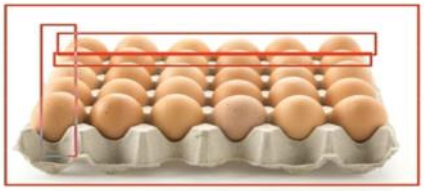

```
#include<iostream>
using namespace std;
int main() {
    int arr[10] = {4,3,6,7,8,9,2,1,0,5 };
    cout << "排序前： " << endl;
    for (int i = 0; i < 10; i++) {
        cout << arr[i] << "  ";}
    cout << endl;
    for (int i = 0; i < 9; i++) {
        for (int j = 0; j < 9 - i; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;}}}
    cout << "排序后： " << endl;
    for (int i = 0; i < 10; i++) {
        cout << arr[i] << "  ";}
    system("pause");
    return 0;}
```

二维数组

2023年9月13日 17:29

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

```
//方式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}
};
```

二维数组组名的应用

2023年9月13日 17:50

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

```
cout << "二维数组大小:  " << sizeof(arr) << endl;
cout << "二维数组一行大小:  " << sizeof(arr[0]) << endl;
cout << "二维数组元素大小:  " << sizeof(arr[0][0]) << endl;

cout << "二维数组行数:  " << sizeof(arr) / sizeof(arr[0]) << endl;
cout << "二维数组列数:  " << sizeof(arr[0]) / sizeof(arr[0][0]) << endl;

//地址
cout << "二维数组首地址: " << arr << endl;
cout << "二维数组第一行地址: " << arr[0] << endl;
cout << "二维数组第二行地址: " << arr[1] << endl;

cout << "二维数组第一个元素地址: " << &arr[0][0] << endl;
cout << "二维数组第二个元素地址: " << &arr[0][1] << endl;
```

```
//二维数组数组名
int arr[2][3] =
{
    {1,2,3},
    {4,5,6}
};

cout << "二维数组大小:  " << sizeof(arr) << endl;
cout << "二维数组一行大小:  " << sizeof(arr[0]) << endl;
cout << "二维数组元素大小:  " << sizeof(arr[0][0]) << endl;

cout << "二维数组行数:  " << sizeof(arr) / sizeof(arr[0]) << endl;
```

```
cout << "二维数组行数:  " << sizeof(arr) / sizeof(arr[0]) << endl;
cout << "二维数组列数:  " << sizeof(arr[0]) / sizeof(arr[0][0]) << endl;

//地址
cout << "二维数组首地址: " << arr << endl;
cout << "二维数组第一行地址: " << arr[0] << endl;
cout << "二维数组第二行地址: " << arr[1] << endl;

cout << "二维数组第一个元素地址: " << &arr[0][0] << endl;
cout << "二维数组第二个元素地址: " << &arr[0][1] << endl;

system("pause");

return 0;
```

考试成绩统计案例

2023年9月13日　17:50

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

```cpp
#include<iostream>
using namespace std;
#include<string>  //使用string 字符串时要用的头文件
int main() {
    int scores[3][3] = { {100,100,100},
                         {90,90,90},
                         {80,80,80} };
    string names[3] = {"张三","李四","王五"};
    string names1[3] = { "语文","数学","英语" };
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout <<names[i]<< "的"<<names1[j]<<"成绩为："<<scores[i][j] << "  ";}
        cout << endl;}
    for (int i = 0; i < 3; i++) {
        int sum = 0;
        for (int j = 0; j < 3; j++) {
            sum += scores[i][j];}  //“+=”的定义要理解好，即sum这个量再加上scores[i][j]的量
        cout << names[i] << "的总分为：" << sum;
        cout << endl;}
    system("pause");
    return 0;}
```

```
C:\Users\yf213\source\repos\
张三的语文成绩为：100  张三的数学成绩为：100  张三的英语成绩为：100
李四的语文成绩为：90  李四的数学成绩为：90  李四的英语成绩为：90
王五的语文成绩为：80  王五的数学成绩为：80  王五的英语成绩为：80
张三的总分为：300
李四的总分为：270
王五的总分为：240
请按任意键继续. . .
```