

第07章 数组

目录

- ▶ 一维数组
- ▶ 数组作为函数参数
- ▶ 向量
- ▶ C-字符串与string

数组

数组

▶ 数组

- ▶ 数组是**有序数据**的集合
- ▶ 特点
 - ▶ 所有数据元素**类型**相同
 - ▶ 数据元素类型可以是基本数据类型和构造数据类型
 - ▶ 由数组名和下标确定具体的数据元素

▶ 一维数组

▶ 二维数组

▶ 高维数组(C++很少使用)

▶ 字符数组和字符串

一维数组的定义

▶ 一维数组的定义

▶ 格式: 类型 数组名[数组长度];

- ▶ 长度——只能为常量、常量表达式, **不能为变量**
- ▶ 类型——数组中元素的类型
- ▶ 数组名——遵循标识符命名规则, 表示数组中元素的起始位置

▶ 例如: `int a[6];`

- ▶ 分配了六个单元, 每个单元存放一个 `int` (即4个Byte), 而且各单元**连续**!

▶ 注意:

- ▶ **C++不允许对数组的大小作动态定义。**
- ▶ **如: `int a=5; int array[a];` // 错误**

a →	0x7824	35	a[0]
	0x7828	46	a[1]
	0x782C	17	a[2]
	0x7830	5	a[3]
	0x7834	78	a[4]
	0x7838	100	a[5]

一维数组的使用

▶ 一维数组的使用

- ▶ 格式：名称[下标];

- ▶ 说明：

- 1) 下标范围：0~长度-1

- 2) **越界问题**：C/C++中不检查越界，但越界写数据可能造成系统崩溃

- 3) 数组不能整体引用，只能引用其中的元素

- ▶ 例如：int a[3];

- a[2]=19;

- a[3] = 20; **// 错误，下标越界**

- 4) 下标可以是常量、变量、表达式或函数

▶ **注意：**

- ▶ 使用数组时，程序员**必须自己检查数组越界问题**。

- ▶ 使用数组时，数组**下标可以是常量，变量，表达式，甚至是函数**。

一维数组的初始化

- ▶ 部分初始化：
 - ▶ 例如： `int a[20]={3,2,5,6,9};` 其余元素为0
- ▶ 全部初始化：——数组长度可以省略
 - ▶ 例如： `int a[3]={3,4,6};`
`int b[]={23,45,67,89};` // 长度为4
 - ▶ `int a[10]={0,1,2,3,4};` 相当于
`int a[10]={0,1,2,3,4,0,0,0,0,0};`
- ▶ 对数组全部元素初始化为0方式
 - ▶ 例如： `int a[5]={0,0,0,0,0};`
或者： `int a[5] = {0};`
- ▶ **说明：** 如果不给数组设置初值，则为随机数

C++中三种基本排序算法

- ▶ 冒泡排序
- ▶ 选择排序
- ▶ 插入排序
- ▶ 特点：
 - ▶ 运行效率比较低，时间复杂度为 $O(n^2)$ ，其中n指要排序的数据项数。
 - ▶ 小数据量：与快速排序的时间效率差别不大

冒泡排序

```
const int LEN=10;
```

```
int main(){
```

```
    int i, temp, k, n, a[LEN];
```

```
    for(i=0; i<LEN; i++)    cin>>a[i];
```

```
    for(k=0; k<LEN-1; k++){
```

```
        for(i=0; i<LEN-k-1; i++) {
```

```
            if(a[i]> a[i+1]) {
```

```
                temp = a[i];
```

```
                a[i] = a[i+1];
```

```
                a[i] = temp;
```

```
            }
```

```
        }
```

```
    }
```

```
    for(i=0; i<LEN; i++)    cout<<a[i];
```

```
    return 0
```

数组的操作

选择排序

```
const int LEN=10;
int main(){
    int i,index,temp, k, a[LEN];
    srand(time(NULL));
    for(i=0; i<LEN; i++)    a[i] = rand()%1000;
    for(k=0; k<LEN-1; k++){
        index = k;
        for(i=k+1; i<LEN; i++) {
            if(a[i]< a[index])
                index=i;
        }
        if (index != k)
            {temp=a[index]; a[index]=a[k]; a[k]=temp; }
    }
    for(i=0; i<LEN; i++)    cout<<setw(5)<<a[i];
    return 0;
}
```

头文件
:ctime

头文件
:cstdlib

交换
元素

插入排序

```
// 插入排序
for(k=1; k<LEN; k++){
    temp = a[k];
    for(i=k-1; i>=0; i--){
        if (temp<a[i])
            a[i+1] = a[i];
        else{
            a[i+1] = temp;
            break;
        }
    }
    if (i<0)
        a[0] = temp;
}
```

将元素往后移一个位置

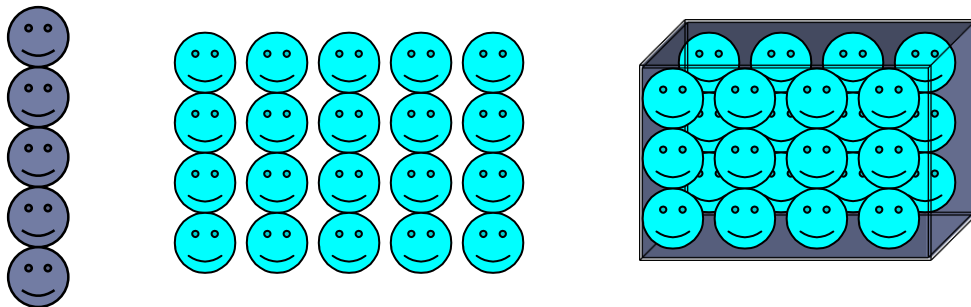
寻找temp(a[k])
插入的合适位置

当temp(a[k])的
插入位置为a[0]
时的特殊处理

二维数组

▶ 多维数组的空间想象

- ❖ 一维数组： 一列长表或一个向量
- ❖ 二维数组： 一个表格或一个平面矩阵
- ❖ 三维数组： 三维空间的一个方阵
- ❖ 多维数组： 多维空间的一个数据列阵



二维数组的定义

▶ 格式

▶ <类型> 名称[长度1][长度2];

▶ 说明:

1) 长度1和长度2——常量表达式，
其中长度1表示行数，长度2表示列数

2) 内存中二维数组的元素按**行存放**

3) 也可以用一维数组的方式引用二维数组元素。(指针)

▶ 例如：整型数组 **b[3][3]={
{1,2,3}, {4,5,6}, {7,8,9} };**

地址	值	数组元素
3000H	1	b[0][0]
3004H	2	b[0][1]
3008H	3	b[0][2]
300CH	4	b[1][0]
3010H	5	b[1][1]
3014H	6	b[1][2]
3018H	7	b[2][0]
301CH	8	b[2][1]
3020H	9	b[2][2]

数组小结

- ▶ 数组是一种用于存放集合数据的数据类型，能够实现
对数据的随机访问，使用时注意几点：
 - ① 在程序中，数组的**长度在定义时就确定了**，不能发生改变，也就是不可以定义可变长度的数组。
 - ② 数组使用时需要特别注意**下标越界**错误，数组的最大下标是其长度-1
 - ③ C++中提供了另外一种类型(**向量vector**)来实现数组功能，vector功能更完善，建议常用。

数组作为函数参数

- ▶ 函数参数传递方向

 - ▶ 实参→形参

- ▶ 函数参数传递方式

 - ▶ 传值

 - ▶ 传地址

- ▶ 数组作为函数参数

 - ▶ 传地址方式

 - ▶ 例如: `void sort(int []);`

 - ▶ 说明:

 - ▶ 参数为数组名: 实际传递的是数组的首地址

 - ▶ 函数可以通过地址操作主调函数的内存空间

 - ▶ `sort`函数操作主调函数的实参数组

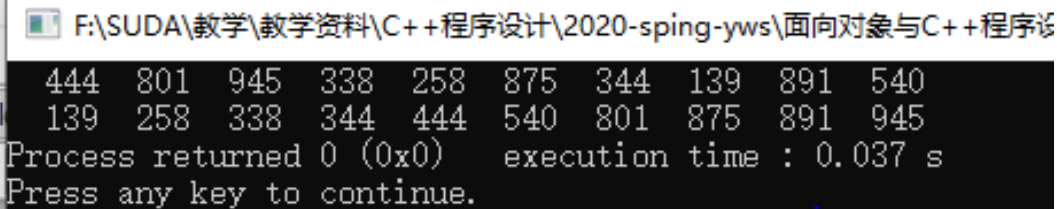
问题: 此函数设计有什么不足?

`void sort(int[], int);`

数组作为函数参数——举例

```
void sort(int array[], int len)
{
    int i, k, temp;
    // 插入排序
    for(k=1; k<len; k++){
        temp = array[k];
        for(i=k-1; i>=0; i--){
            if (temp<array[i])
                array[i+1] = array[i];
            else{
                array[i+1] = temp;
                break;
            }
        }
        if (i<0)
            array[0] = temp;
    }
}
```

```
const int LEN=10;
int main() {
    int i, temp, k, a[LEN];
    srand(time(NULL));
    for(i=0; i<LEN; i++)
        a[i] = rand()%1000;
    for(i=0; i<LEN; i++)
        cout<<setw(5)<<a[i];
    cout<<endl;
    sort(a, LEN);
    for(i=0; i<LEN; i++)
        cout<<setw(5)<<a[i];
    return 0;
}
```



F:\SUDA\教学\教学资料\C++程序设计\2020-spring-yws\面向对象与C++程序设计

444	801	945	338	258	875	344	139	891	540
139	258	338	344	444	540	801	875	891	945

Process returned 0 (0x0) execution time : 0.037 s
Press any key to continue.

向量

向量

- ▶ 容器技术
- ▶ 向量的概念
 - ▶ 向量是一种可以容纳任何合法数据类型（含用户自定义）的,包含各种常见的数组操作、且长度可动态变化的数组。
- ▶ 向量的定义和初始化
 - ▶ 向量支持多种定义和初始化的方法

```
vector<int> a(10), b;    // 元素个数: a为10, b为0
vector<int> c(10, 2);    // c有10个元素, 每个元素值为2
vector<int> d(c);
int array[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
vector<int> e(array, array+5);
// begin()返回一个迭代器, 容器类才有的成员函数
vector<int> f(e.begin(), e.begin() + 3);
```

向量

- ▶ 遍历器的概念及其作用
 - ▶ begin和end的含义
 - ▶ 声明头文件: **iterator**
 - ▶ 定义: `vector<int>::iterator it; // it类似于指针`
 - ▶ 使用: `it = e.begin(); cout<<*it; *it=100;`
- ▶ 向量的输入和输出:
 - ▶ 不允许整体输入输出, 只能单个元素的输入输出
- ▶ 向量的相关操作
 - ▶ 和数组不同, 向量的操作都是通过成员函数来进行的

向量的相关操作

```
vector<int> e(5,2), b;  
e[2] = 100;           //下标范围: [0, e.size()-1]  
e.at(2) = 200;        // 参数范围: [0, e.size()-1]  
e.push_back(2);       // 将整数2存入到e的末端  
e.pop_back();         // 删除最后一个元素  
b.assign(e.begin(), e.end() - 2); //把e的元素赋值给b  
int x = b.back();      // 返回b的最后一个元素值  
int y = b.front();     // 返回b的第一个元素值  
b.clear();            // 清除b, b.size()返回0  
b.empty();  
b.resize(10);         // 扩充b的元素个数为10, 并设置元素值为0  
b.resize(20, 2);      //扩充b的元素个数为20, 扩充的10个元素设置为2  
e == b;               // 比较: 向量每一个对应元素都必须相等  
e.erase(e.begin()+1); // 删除某一个元素, 或几个元素  
e.erase(e.begin(), e.begin()+2); // 删除[0,2)之间的元素
```

向量长度的问题

- ▶ 向量的长度是可以动态变化的
 - ▶ `resize()`
 - ▶ `capacity()`: // 向量可以存储的元素个数，不一定与`size()`相同
- ▶ 自动扩展容量和手动扩展容量的选择
- ▶ 二维向量
 - ▶ 概念：类似于二维数组
 - ▶ 定义方法：注意`vector<vector<int> > dim_array;`
 - ▶ 两个向量的交换操作 `a.swap(b)`

空格不可省略

C-字符串和string

C-字符数组与字符串

▶ 字符数组

- ▶ 存放字符数据的数组，字符数组中的一个元素存放一个字符。

▶ 字符串

- ▶ 字符串：双引号括起的多个字符序列，例如：“abc”，“1”，“abc123”
- ▶ 字符串尾：系统自动添加'\0' //ASCII为零的字符
- ▶ 字符串的含义：字符串的起始位置
- ▶ 字符串的存储：——利用字符数组
 - ▶ `char s[10]={“Hello”};`
 - ▶ `char s[]=”Hello”;` //数组的长度为6
 - ▶ `char s[10]={‘2’,‘3’,‘4’,‘5’,‘\0’};`//字符形式的串
 - ▶ `char s[10]; s[0]=‘2’; s[1]=‘3’; s[2]=‘4’;`//字符形式，不是字符串

C-字符数组与字符串

▶ 字符串

- ▶ C-字符串的一些拷贝、比较等操作需要使用相应的库函数才能完成。
- ▶ 常用的C-字符串操作库函数
 - ▶ strcpy(目的字符数组1,源字符串2)
 - ▶ strcat(字符数组1,字符串2) // 字符串2拼接 to 字符串1后面
 - ▶ strcmp(字符串1,字符串2) // 比较两个字符串的大小
 - ▶ strlen(字符数组)
 - ▶ strlwr (字符串) // 转换为小写字母
 - ▶strupr (字符串) // 转换为大写字母
 - ▶ 头文件: string.h 或 cstring

C++中字符串的概念及其使用机制

▶ C-串结构

- ▶ 要求有结束标志
- ▶ 使用过程中可能的危险性

```
char* str1;  
char* str2 = new char[5];  
strcpy(str2, "ugly");  
strcpy(str1, str2);           // 错: str1没有空间可储  
strcpy(str2, "Hello");       // 错: str2空间不够大  
str2 = "Hello"; //错: 原来的"ugly"空间脱钩, 导致内存泄漏
```

- ▶ 操作必须配备相应的函数: 如比较/相互赋值等

▶ C++中的string

- ▶ string一串类
- ▶ 必须要包含std名空间中的string头文件。注意此string头文件和标准C中的string头文件(cstring)在性质上的差异

string类使用

using namespace std;

#include <string>

string a, s1 = "Hello ";

string s2 = "123";

a = s1;

// copy

cout<<(a==s1 ? "" : "not")<<"equal\n"; // compare

cout<<a+s2<<endl;

// concatenate

reverse(a.begin(), a.end());

// 逆转

cout<<a<<endl;

// reverse

cout<<a.replace(0,9,9,'c')<<endl; // 替换

//从0下标开始的9个字符替换为9个' c' ;

cout<<(s1.find("ell")!= -1?"":"not")<<"found\n"; // find string

cout<<(s1.find('c')!= -1?"":"not ")<<"found\n"; //find char

//找到第一个就返回该子串的首字符位置或字符的位置

STL的算法函数，头文件：**algorithm**

string类使用

```
string str1, str2;  
str1 = "this is a";  
// 计算str1的字符个数  
cout <<setw(5)<<str1.length()<<setw(5)<<str1.size();  
str1[2] = 'a';    // str1: tais is a; 下标范围: [0, str1.length()-1]  
cout<<str2.length(); // 输出: 0  
str2[1] = 'a';      // 错误, 下标越界  
str2 += "that is";  // 为string类对象添加字符的方法
```

string和C串的输入输出

- ▶ 输入的cin方法：单词方式，不读取空格、tab键、换行符
- ▶ 输入的getline方法：
 - ▶ 整句（行）输入
 - ▶ 原型一：全局函数
 - `istream& getline (istream &is , string &str , char delim);`
 - `istream& getline (istream &is , string &str);`
 - 形式参数一定要是string
 - `delim`: 终结符，遇到该字符停止读取操作，默认为回车
 - ▶ 原型二：cin的成员函数
 - `cin.getline(char *, int)`
 - 第一个参数为一个char指针，第二个参数为数组字符串长度
 - 形式参数一定要是字符数组和字符指针
- ▶ 输出的cout方法

string和C串的输入输出

```
int main()
{
    string str,str1;
    char c_str[100];
    cin>>str;                // (1)
    cout << str << endl;
    cin.get();                //(2)
    getline(cin, str);        //(3)
    cout << str << endl;
    cin.getline(c_str,8);     //(4)
    cout << c_str << endl;
    cin >> str1;              //(5)
    cout << str1 << endl;
    return 0;
}
```