高级语言C++程序设计
Lecture 5 数组和字符串

李雨森
南开大学 计算机学院
2021

数组

南开大学

倒序问题

接受键盘输入的100个整数,然后将它们按和原顺序相反的顺序输出

- 如何存放这100个整数?
- 定义100个int型的变量, n1, n2, n3, ……n100，用来存放这100个整数？

使用数组 ！

数组

- 可以用来表达类型相同的元素的集合，集合的名字就是数组名
- 数组里的元素都有编号，元素的编号叫下标；通过数组名和下标，就能访问元素
- 一维数组的定义方法如下：

```
类型名  数组名[元素个数];
```

- “元素个数”必须是常量，不能是变量，而且其值必须是正整数，元素的个数成为“数组的长度”

数组

```
int  a[100];
```

名字为a的数组，有100个元素，每个元素都是一个int类型的变量

a[0]	a[1]	a[2]	.....	a[99]

- 100个元素在内存里一个挨一个连续存放，a数组占用的内存大小为100*sizeof(int)
- 数组下标从0开始，N个元素的数组，下标从0至N-1

倒序问题

接受键盘输入的100个整数，然后将它们按和原来顺序相反的顺序输出

```
#define NUM 100
int a[NUM]; //使用符号常量，便于修改
int main() {
    for(int i = 0; i < NUM; i ++)
        cin >> a[i];
    for(int i = NUM-1; i >= 0; i--)
        cout << a[i] <<“ ”;
    return 0;
}
```

数组的初始化

- 数组定义时，可以给数组中的元素赋初值

```
类型名  数组名[元素个数]={值, 值, .....值};
```

{ }中的各个数据值即为各元素的初值，值之间用逗号间隔

```
int a[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
```

效果：a[0]=0; a[1]=1; … a[9]=9;

数组的初始化

- 数组初始化时，{ }中的值的个数可以少于元素个数。相当于只给前面部分元素赋值，而后面的元素，其存储空间里的每个字节被写入二进制0

```
int a[10] = {0, 1, 2, 3, 4};
```

效果：只给a[0]～a[4]5个元素赋值，而后5个元素自动赋值0

易错点总结

- 数组初始化时，元素个数不能超过数组大小

```
int scores [5] = {89, 90, 100, 91, 92, 93};
//错误，元素个数超过数组大小
```

- 数组定义时，数组大小不能是变量

```
int size = 5;
int scores[size]; //错误，size是变量，不是常量
```

- 初始化列表只能在数组定义时使用

```
int scores[5];
scores = {89, 90, 78, 82, 95}; //错误，不允许使用初始化列表对数组元素进行赋值
```

易错点总结

- 一个数组不能被另外一个数组初始化

```
int a_scores[5] = {89, 90, 78, 82, 95};
int b_scores[] = a_scores; //错误，一个数组不能被另外一个数组初始化
```

- 数组之间不能赋值

```
int a_scores[5], b_scores[5];
a_scores = b_scores;//错误，数组之间不能进行赋值
```

- 数组下标不能越界

```
int scores[5] = {1,2,3,4,5};
cout<<scores[5]; //错误，数组下标越界
```

更多例子

整数型数组

```
int a[] = {0, 1, 2, 3, 4};
```

如果有初始化列表，数组大小可省略

浮点型数组

```
float b[10] = {0.1, 1.2};
```

数组元素可以是各种数据类型

字符型数组

```
char c[10] = { ‘a’ , ‘b’ , ‘c’ };
```

例子-冒泡排序

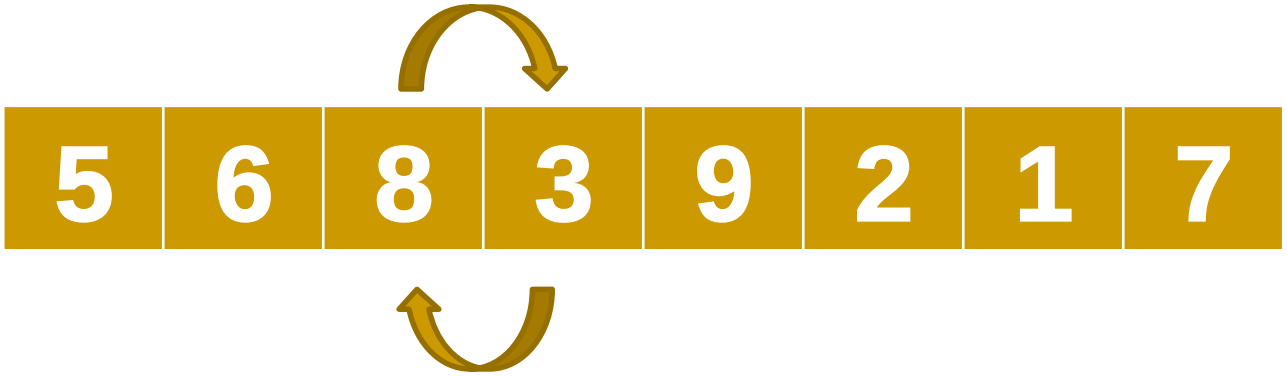
给定一个数组A，使用冒泡排序法对里面的数从小到大排序

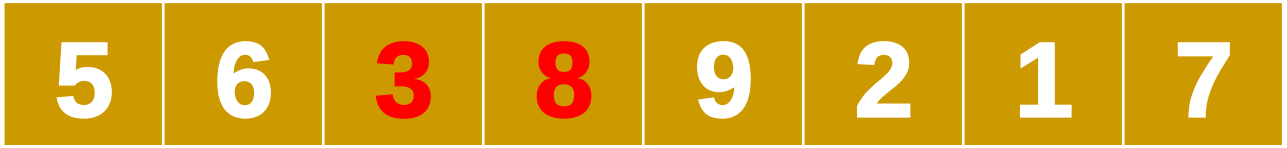
例子-冒泡排序

第一轮：从A[0]开始到A[n-2]结束，比较A[i]和A[i+1]，如果A[i]>A[i+1]，交换A[i]和A[i+1]

例子-冒泡排序

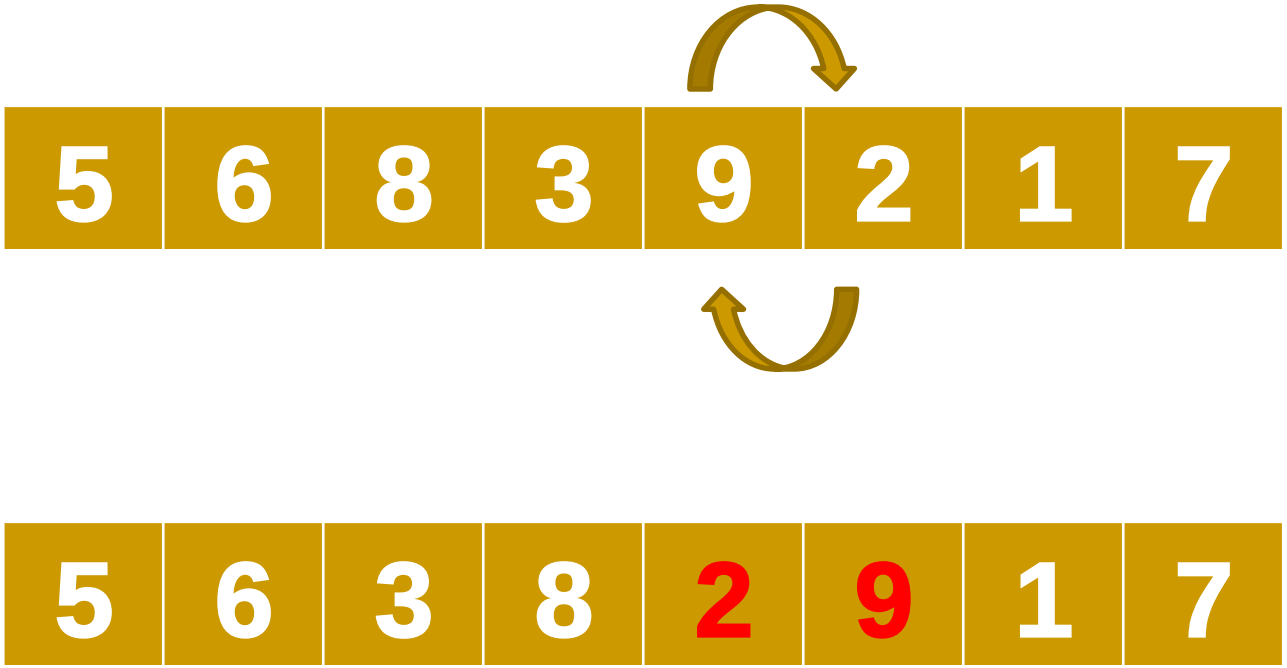
第一轮：从A[0]开始到A[n-2]结束，比较A[i]和A[i+1]，如果A[i]>A[i+1]，交换A[i]和A[i+1]





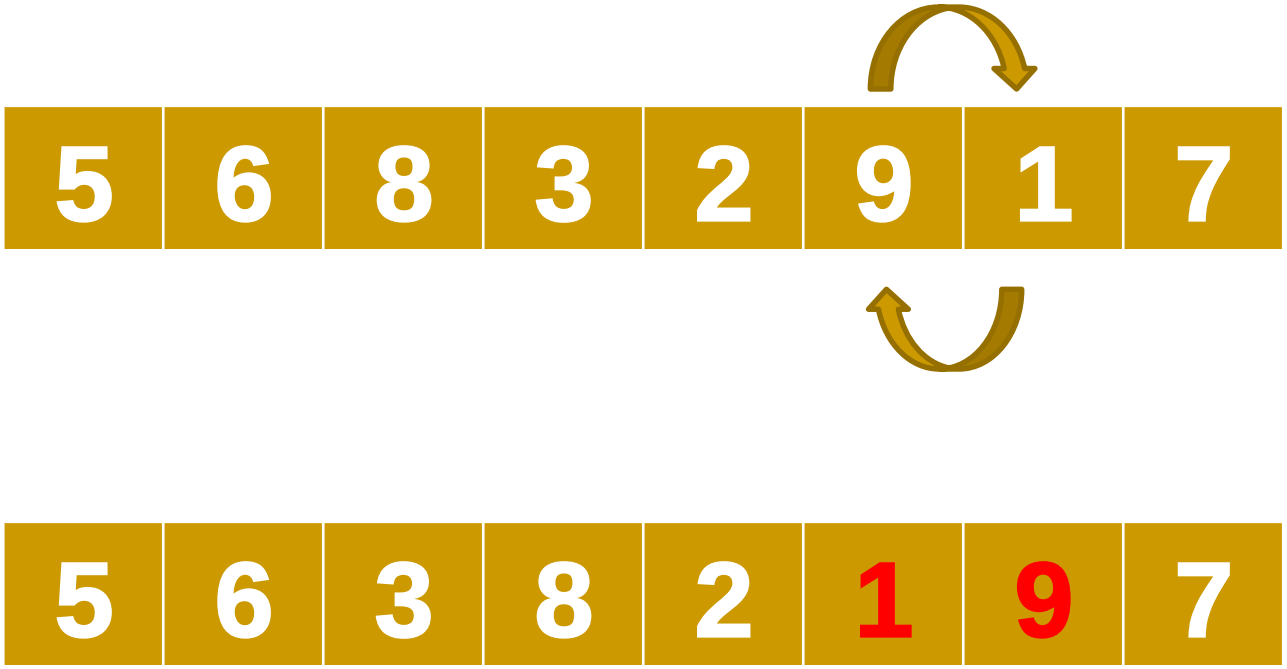
例子-冒泡排序

第一轮：从A[0]开始到A[n-2]结束，比较A[i]和A[i+1]，如果A[i]>A[i+1]，交换A[i]和A[i+1]



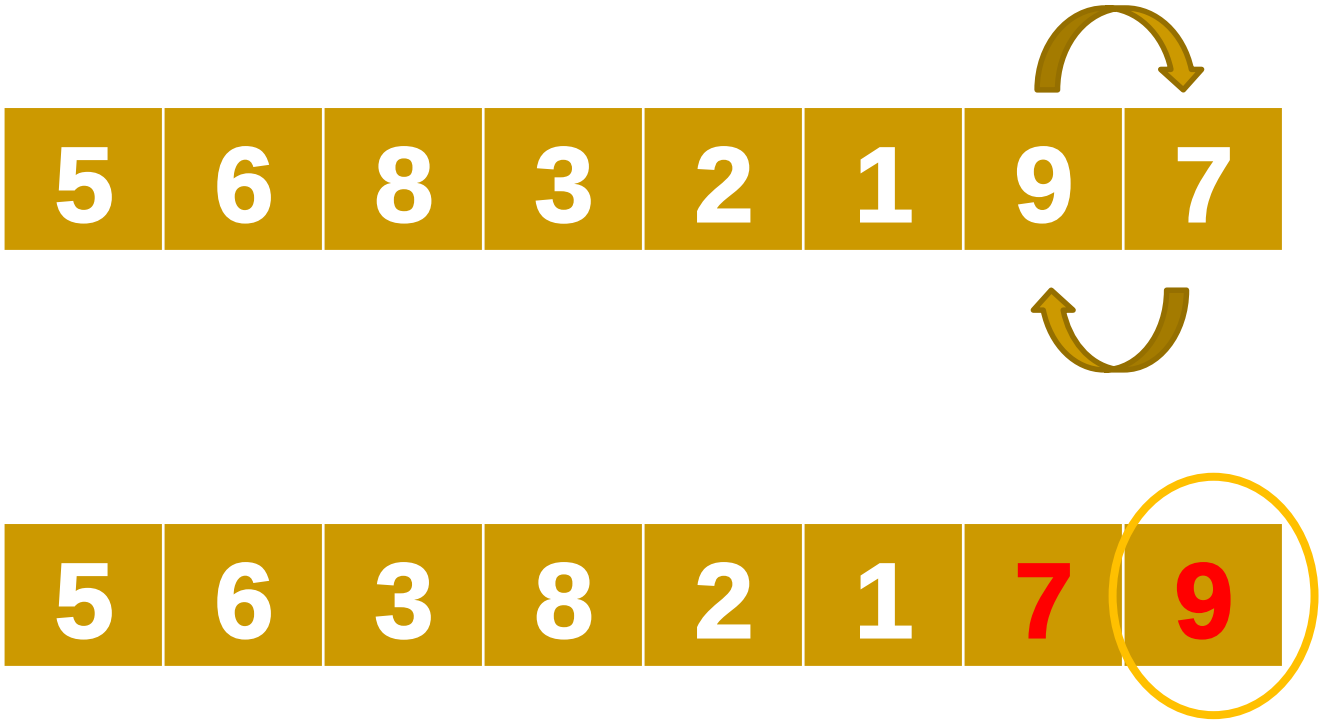
例子-冒泡排序

第一轮：从A[0]开始到A[n-2]结束，比较A[i]和A[i+1]，如果A[i]>A[i+1]，交换A[i]和A[i+1]



例子-冒泡排序

第一轮：从A[0]开始到A[n-2]结束，比较A[i]和A[i+1]，如果A[i]>A[i+1]，交换A[i]和A[i+1]



经过第一轮交换，最大的数被换到数组最右边！

例子-冒泡排序

第二轮：从A[0]开始到A[n-3]结束，比较A[i]和A[i+1]，如果A[i]>A[i+1]，交换A[i]和A[i+1];
第二轮结束后，倒数第二大的数换到a[n-2]

第三轮：从A[0]开始到A[n-4]结束，比较A[i]和A[i+1]，如果A[i]>A[i+1]，交换A[i]和A[i+1];
第三轮结束后，倒数第三大的数换到a[n-3]

…

第n-1轮：从A[0]开始到A[0]结束，比较A[i]和A[i+1]，如果A[i]>A[i+1]，交换A[i]和A[i+1];
第n-1轮结束后，数组从小到大排好序

例子-冒泡排序

```
#define n 8
int main() {
    int temp;
    int A[n] = {5,8,6,3,9,2,1,7};
    for(int j = 1; j <= n-1; j ++) //第j轮
                //从A[0]到A[n-j-1]
        for(int i = 0; i < n-j; i ++){
            if(A[i] > A[i+1]) {//比较A[i]和A[i+1]
                temp = A[i];
                A[i] = A[i+1];
                A[i+1] = temp;
            }
        }
    return 0;
}
```

字符串

南开大学

C++中的字符串

字符串有三种形式：

- （1）用双引号括起来的字符串常量

```
#include <iostream>
using namespace std;
int main() {
    cout<< “hello world!” <<endl;
    return 0;
}
```

C++中的字符串

字符串有三种形式：

- （2）C++标准模板库里的string

```
#include <iostream>
using namespace std;
int main() {
    string s;
    cin>>s; //键盘输入字符串，存放在s中
    cout<< s <<endl;
    return 0;
}
```

C++中的字符串

字符串有三种形式：

- （3）以‘\0’结尾的字符数组

```
#include <iostream>
using namespace std;
int main() {
    //s是一个字符数组，同时也是一个字符串
    char s[10] = {'a','b','c','\0'};
    cout<<s<<endl;
    return 0;
}
```

‘\0’是ASCII为0的特殊字符，表示字符串结尾

C++中的字符串

字符数组和字符串

普通字符数组，不包含'\0'

```
char s1[] = {'a','b','c'};
```

如果s是字符数组，以下两种初始化方式是字符串：

```
char s[] = {'a','b','c','\0'}; //直接加'\0'

char s[] = "abc"; //用字符串常量初始化，默认在末尾
                  自动加上'\0'，数组大小为4
```

C++中的字符串

字符串的输出

- 直接输出

```
char s[10] = {'a','b','c','\0'};
cout<<s; //cout可以直接输出整个字符串"abc"
```

- 使用循环语句

```
char s[10] = {'a','b','c','\0'};
int i = 0;
while(s[i] != '\0') { //判读是否到结尾
    cout<<s[i]; //逐个字符输出
    i++;
}
```

C++中的字符串

字符串举例

```
char carr[5]={‘1’,‘2’,‘3’,‘4’,‘5’}; //carr为普通字符数组
char str1[5]={‘I’,‘ ’,‘c’,‘a’,‘n’}; //str1为普通字符数组
char str2[6]={'I',' ','c','a','n','\0'};//str2为字符串
char str3[6]= "I can"; //str3为字符串
cout<<"carr=";
for(int i=0; i<5; i++) //输出数组前5个元素值
    cout<<carr[i];
cout<<endl;
cout<<"str1=";
for(i=0; i<5; i++) //输出数组前5个元素值
    cout<<str1[i];
cout<<endl;
cout<<“str2=”<<str2<<endl; //对字符串直接输出
cout<<"str3="<<str3<<endl; //对字符串直接输出
```

字符串的输入

- 用cin可以将字符串读入字符数组，cin读入到空格为止，并会自动在字符串的末尾加上 '\0'

```
#include <iostream>
using namespace std;
int main() {
    char s[10];
    cin>>s;  //从键盘度入一个字符串
    cout<<s;
    return 0;
}
```

输入：Hello ↙
输出：Hello

输入：Hello World ↙
输出：Hello

字符串的输入

- 在数组长度不足的情况下，cin可能导致数组越界

```
#include <iostream>
using namespace std;
int main() {
    char s[5];
    cin>>s;  //若输入”12345”，则数组越界
    cout<<s;
    return 0;
}
```

字符串的输入

- 用gets读入带空格的字符串

```
gets(char buf[]);
```

读入一行字符到buf，自动添加 ‘\0’ ，遇到换行符或EOF停止

```
#include <iostream>
using namespace std;
int main() {
    char s[10];
    gets(s); // 读入带空格的字符串到s
    cout<<s;
    return 0;
}
```

字符串的输入

- 用getline读入带空格的字符串

```
cin.getline(char buf[], int bufSize);
```

读入一行（行长度不超过bufSize-1)字符到buf，自动添加‘\0’，回车换行符不会被读入

```
#include <iostream>
using namespace std;
int main() {
    char s[10];
    cin.getline(s, 10);// 读入最多9个字符到s
    cout<<s;
    return 0;
}
```

字符串的输入

- 用getline读入带空格的字符串

cin.getline(char buf[], int bufSize);

cin和getline混用，容易出现下面错误：

```
#include <iostream>
using namespace std;
int main() {
    char s[10];
    char s2[10];
    cin>>s;
    cout<<s<<endl;
    cin.getline(s2, 10);
    cout<<s2<<endl;
}
```

输入：Hello ↙
输出：Hello
程序结束

当输入Hello并回车时，程序输出完Hello就结束了，并没有等待用户第二个输入

字符串的输入

- 用getline读入带空格的字符串

```
cin.getline(char buf[], int bufSize);
```

cin和getline混用，容易出现下面错误：

```
#include <iostream>
using namespace std;
int main() {
    char s[10];
    char s2[10];
    cin>>s;
    cout<<s<<endl;
    cin.getline(s2, 10);
    cout<<s2<<endl;
}
```

输入：Hello ↙
输出：Hello
程序结束

这是因为cin不能读入回车，将回车留在缓冲区，导致getline读入缓冲区中的回车直接结束

字符串的输入

- 用getline读入带空格的字符串

```
cin.getline(char buf[], int bufSize);
```

cin和getline混用，容易出现下面错误：

```
#include <iostream>
using namespace std;
int main() {
    char s[10];
    char s2[10];
    cin>>s;
    cout<<s<<endl;
    cin.sync(); //清空缓冲区
    cin.getline(s2, 10);
    cout<<s2<<endl;
}
```

输入：Hello ↙
输出：Hello
输入：world ↙
输出：world
程序结束

解决方案是cin结束后清空缓冲区

字符串库函数

- 使用字符串函数需要 #include <cstring>

字符串拷贝

```
strcpy(char dest[], char src[]);//拷贝src到dest
```

字符串比较大小

```
int strcmp(char s1[], char s2[]);//返回0则相等
```

求字符串长度

```
int strlen(char s[]);//不包含'\0'
```

字符串拼接

```
strcat(char s1[], char s2[]);//s2拼接到s1后面
```

字符串库函数

字符串拷贝

原型：strcpy(str1, str2);

功能：将字符串str2复制到字符串str1中，并覆盖str1原始字符串

注意：1）字符串str2会覆盖str1中的全部字符，2）字符串str2的长度不能超过str1

```
char str1[] = "We are csdn!";
char str2[] = "Hello!";
strcpy(str1, str2); //str1会变为Hello!
```

字符串库函数

字符串比较

原型：strcmp(str1, str2);

功能：比较两个字符串，如果相等，则返回0；若str1大于str2，返回一个正数；若str1小于str2，返回一个负数；

```
char str1[20] = "Wearecsdn!";
char str2[] = "Wearea!";
int cmp = strcmp(str1, str2); //cmp为正数
```

字符串库函数

字符串长度

原型：strclen(str1);

功能：计算str1的长度

```
char str1[20] = "We are csdn!";
int size = strlen(str1); //size为12
```

字符串库函数

字符串连接

原型：strcat(str1, str2);

功能：将字符串str2添加到字符串str1的尾部，也就是拼接两个字符串

注意：拼接之后的长度不能超过字符串str1所在数组的长度

```
char str1[20] = "We are csdn!";
char str2[] = "Hello!";
strcat(str1, str2); //str1会变为We are csdn!Hello!
```

字符串库函数

```
#include <iostream>
#include <cstring>
using namespace std;
int main() {
    char s1[30];
    char s2[40], char s3[100];
    strcpy(s1, “Hello”);//拷贝”Hello”到s1,
                        s1=“Hello”
    strcpy(s2, s1); //拷贝s1到s2, s2=“Hello”
    strcat(s1, “ world”);//将” world”接到s1
                         尾部, s1 = “Hello
                         World”
    cout<<s1<<s2<<endl;
```

字符串库函数

```
if(strcmp(s1, s2))//如果s1小于等于s2(按字典序)
        cout<<s1;
    else
        cout<<s2;
    cout<<strlen(s2); //输出s2的长度
    return 0;
}
```

例子-简单字符串操作

键盘输入字符串，统计其中数字字符的个数

```
#include<iostream>
#include<cstdio>
#include<cstring>
using namespace std;
int main(){
  char a1[100]; //存字符串的数组
  gets(a1); //读入字符串
  int len = strlen(a1); //字符串长度
  int num = 0; //数字字符个数
  for(int i = 0; i < len; i ++) {
      if(a1[i] >= '0' && a1[i] <= '9')
      num ++;
  }
  cout<<num<<endl;
  return 0;
}
```

例子-简单字符串操作

键盘输入两个字符串，比较大小

```
#include<iostream>
#include<cstdio>
#include<cstring>
using namespace std;
int main(){
  char a[100], b[100]; //两个数组，分别存储两个字符串
  gets(a); //读入字符串a
  gets(b); //读入字符串b
  int ret = 0; //0表示相等，负数表示a<b，整数表示a>b
  int i = 0;
  while(a[i] == b[i]) { //如果相等，一直往后比
      if(a[i] == '\0') { //如果同时结束，表示两个字符串相同
          ret = 0;
          break;
      }
      i ++;
  }
```

例子-简单字符串操作

键盘输入两个字符串，比较大小

```
    ret = a[i] - b[i];  //如果某个字符不相等，跳出循环，不相等
    cout<<ret<<endl;    的两个字符的ASCII相减就能代表大小
    return 0;
}
```

一维数组综合示例 - 高精度加法

求两个不超过100位的非负整数的和

难点：数据过大，基本数据类型会溢出

思路：

1. 以字符串的形式读入两个大整数
2. 从字符串中大整数的每一位转为整数，放到整数型数组中
3. 按照加法逻辑，对两个整数数组的每一位对应相加，并按照规则进位

一维数组综合示例 - 高精度加法

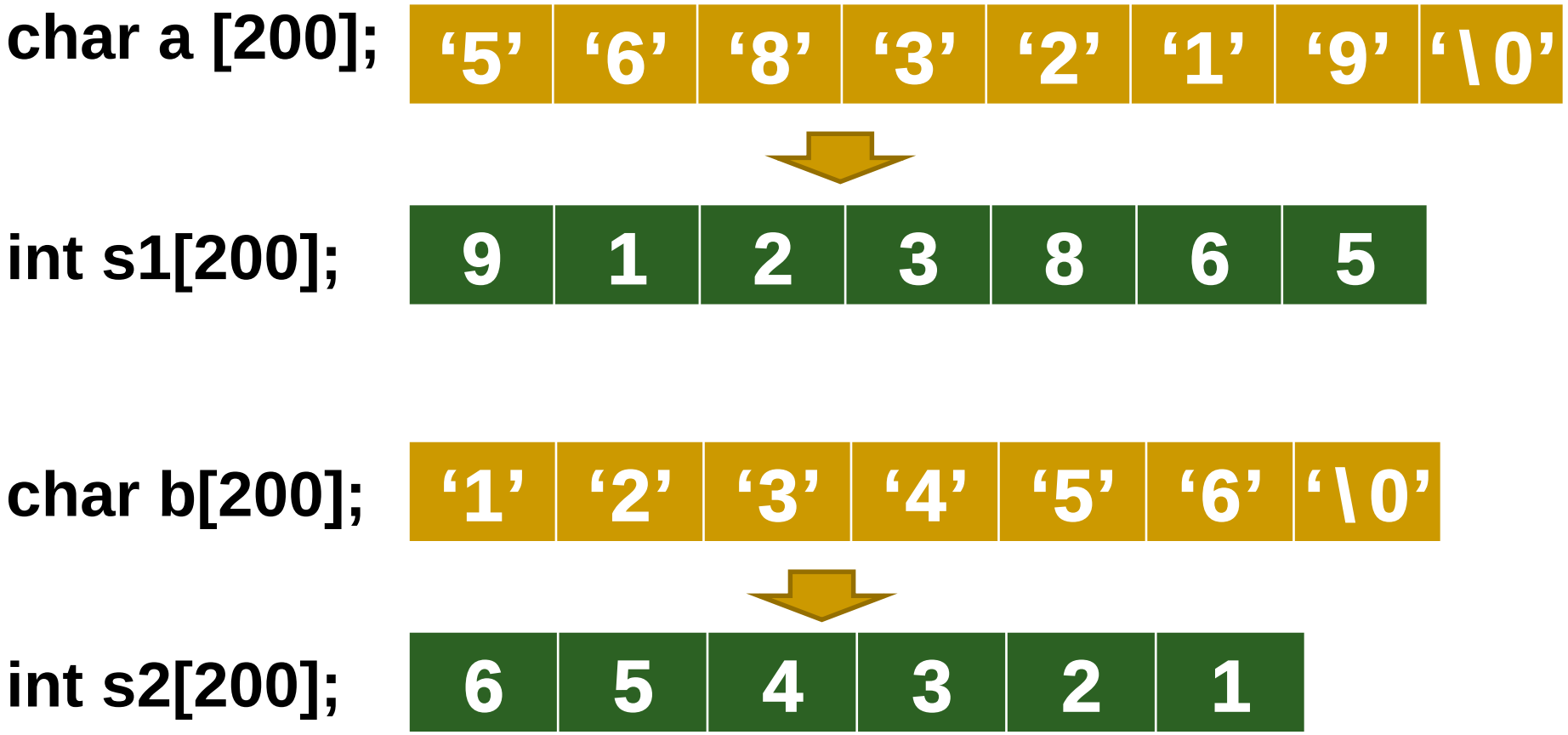
1. 以字符串的形式读入两个大整数

char a[200];

char b[200];

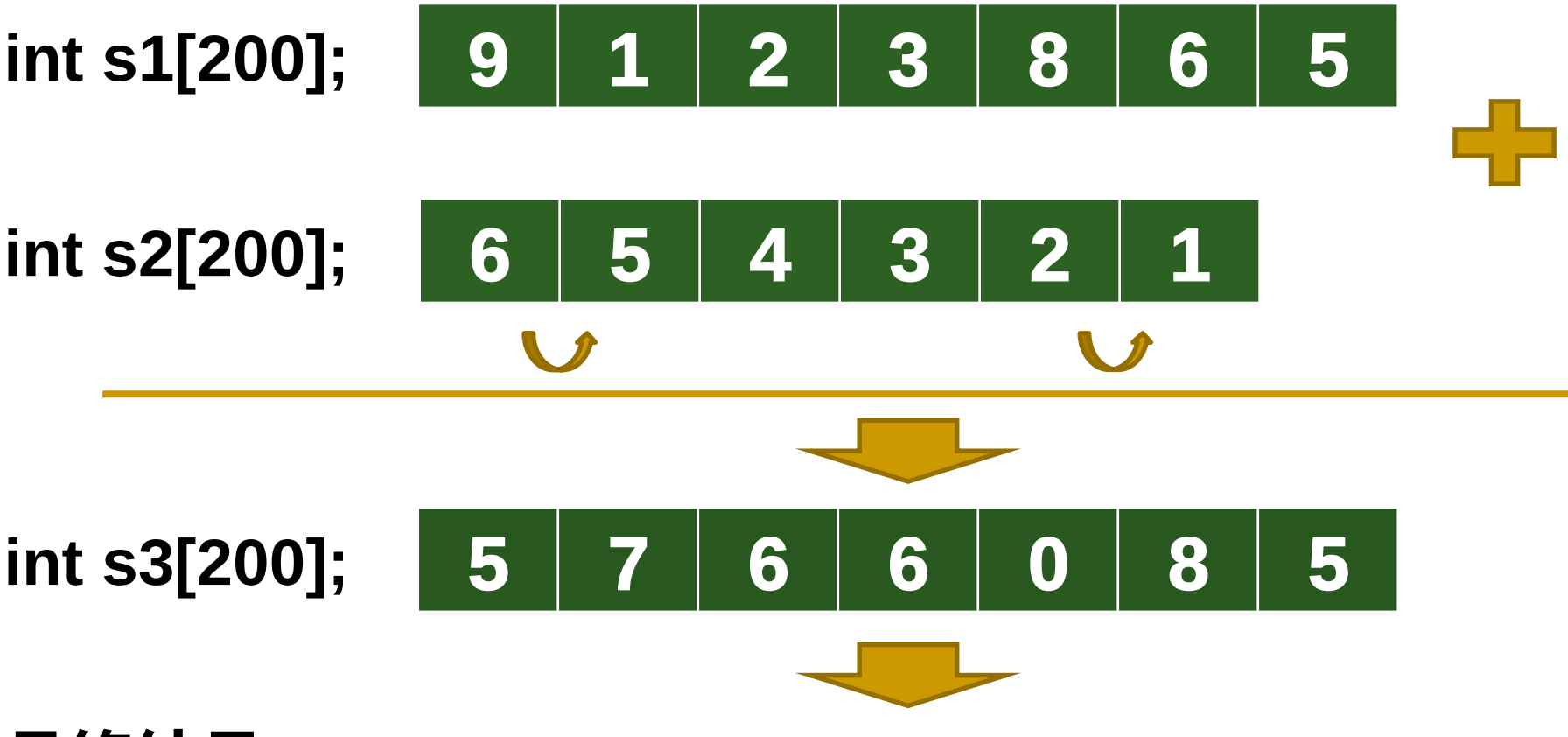
一维数组综合示例 - 高精度加法

2. 从字符串将大整数的每一位转为整数，并存到相应的整数型数组



一维数组综合示例 - 高精度加法

3. 按照加法逻辑，对大整数的每一位对应相加，并按照规则进位



一维数组综合示例 - 高精度加法

```
#include<iostream>
#include<cstdio>
#include<cstring>
using namespace std;
int main(){
  char a1[200],b1[200]; //存放两个数的字符串数组
  int a[200],b[200],c[200]; //整数数组
  int a1_len,b1_len,lenc,i,x;
  memset(a,0,sizeof(a)); //数组初始化为0
  memset(b,0,sizeof(b));
  memset(c,0,sizeof(c));

  gets(a1); //输入第一个数
  gets(b1); //输入第二个数
```

一维数组综合示例 - 高精度加法

```
a1_len = strlen(a1); //第一个数的位数
b1_len = strlen(b1); //第二个数的位数

for (i = 0;i <= a1_len-1; i++) {
    a[a1_len-i]=a1[i]-48;//转为整数，放入a数组
}

for (i = 0;i <= b1_len-1; i++) {
    b[b1_len-i]=b1[i]-48; //转为整数，放入b数组
}
```

一维数组综合示例 - 高精度加法

```
lenc = 1;
x = 0;

while(lenc <=a1_len || lenc <=b1_len) {
    c[lenc] = a[lenc]+b[lenc]+x; //两数相加
    x = c[lenc]/10;  //要进的位
    c[lenc] = c[lenc]%10;   //进位后的数
    lenc++;   //数组下标加1
}

c[lenc] = x;
if (c[lenc]==0) {
    lenc--; //处理最高进位
}
```

一维数组综合示例 - 高精度加法

```
    for (i = lenc; i >= 1; i--) {
        cout<<c[i]; //输出结果
    }
    cout<<endl;
    return 0;
}
```

二维数组

南开大学

二维数组

定义N行M列的二维数组：

类型名 数组名[N][M];

- 行数N、列数M必须是常量，不能是变量，而且其值必须是正整数
- 数组有N行，每一行都有M个元素，NxM个元素在内存里是一个挨一个连续存放的
- 数组占用了一片连续的存储空间，大小为N*M*sizeof(元素类型)

二维数组

定义2行3列的二维数组：

```
int  a[2][3];
```

a[0][0]	a[0][1]	a[0][2]	a[1][0]	a[1][1]	a[1][2]

- 数组有2行，每一行都有3个元素，6个元素在内存里是一个挨一个连续存放的
- a[i][j]表示数组的第i行、第j列元素，i称为行下标，j称为列下标，行下标和列下标都从0开始

二维数组

定义2行3列的二维数组：

```
int  a[2][3];
```

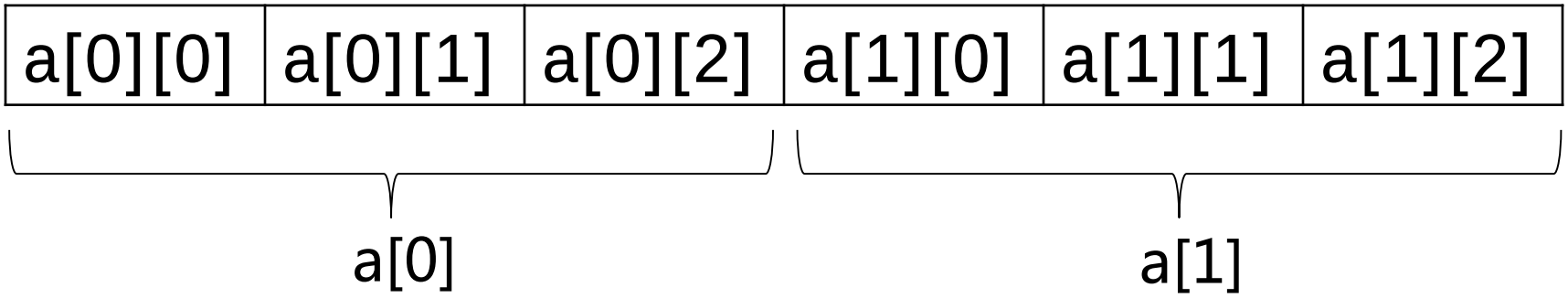
a[0][0]	a[0][1]	a[0][2]	a[1][0]	a[1][1]	a[1][2]

- 第i行的元素是a[i][0], a[i][1], a[i][2]
- 第j列的元素是a[0][j], a[1][j]

二维数组

定义2行3列的二维数组：

```
int  a[2][3];
```



- 二维数组的每一行，实际上是一个一维数组；a[0], a[1]都可以看作是一个一维数组的名字，可以直接当一维数组使用

二维数组初始化

```
int a[2][3];
```

- 以元素为单位进行初始化

```
int a[2][3] = {98, 100, 85, 86, 95, 74}; //直接给出6个元素，用一对{}
```

- 以行为单位进行初始化

```
int a[2][3] = {{98,100,85}, {86,95,74}}; //每个{}代表一行
```

二维数组初始化

```
int  a[2][3];
```

- 定义数组时行可以省略，但列不能省略

```
int scores[][3] = {{98,100,85}, {86,95,74}};
//可以直接从初始化列表中推测出有几行
```

遍历二维数组

遍历一个二维数组，将其所有元素逐行依次输出：

```
#define ROW 20
#define COL 30
int a[ROW][COL];
for(int i = 0; i < ROW; i++) {
    for(int j = 0; j < COL; j ++)
        cout<<a[i][j]<<“ “;
    cout<<endl;
}
```

矩阵乘法

编程求两个矩阵相乘的结果。输入第一行是整数m和n，表示第一个矩阵的行数和列数。接下来是一个mxn的矩阵。再下一行的是整数p和q，表示下一个矩阵是行数和列数(n=p)，再接下来就是一个p行q列的矩阵。
要求输出两个矩阵相乘的结果矩阵。

输入样例：

```
2 3
2 4 5
2 1 3
3 3
1 1 1
2 3 2
0 1 4
```

输出样例：

```
10 19 30
4 8 16
```

矩阵乘法

```
#include <iostream>
using namespace std;
#define ROW 8
#define COL 8
int a[ROW][COL]; //二维数组，存第一个矩阵
int b[ROW][COL]; //二维数组，存第二个矩阵
int c[ROW][COL]; //二维数组，存结果矩阵
int main() {
    int m, n, p, q;
    cin>>m>>n;  //输入第一个矩阵的行、列
    for(int i = 0; i < m; i++){//输入第一个矩阵
        for(int j = 0; j < n; j ++)
            cin>>a[i][j];
```

矩阵乘法

```
cin>>p>>q; //输入第二个矩阵的行、列
for(int i = 0; i < p; i ++) //输入第二个矩阵
    for(int j = 0; j < q; j ++)
        cin>>b[i][j];

for(int i = 0; i < m; i ++){
    for(int j = 0; j < q; j ++) {
        c[i][j] = 0; //计算结果矩阵的第i行j列
        for(int k = 0; k < n; k ++)
            c[i][j]+=a[i][k]*b[k][j];
    }
}
```

矩阵乘法

```
    for(int i = 0; i < m; i++){//输出结果矩阵
        for(int j = 0; j < q; j ++) {
            cout<<c[i][j]<<" ";
        }
        cout<<endl;
    }
    return 0;
}
```

二维字符数组

```
int main() {
    char arr1[3][20]={"12345", "C++ OK!",
" I can do it!"}; //arr1是二维字符数组，arr1[0]被赋值为
                  "12345"，arr1[1]被赋值为"C++ OK!"，
                  arr1[2]被赋值为"I can do it!"

    for(int i=0; i<3; i++)
        cout<<arr1[i]<<endl; //输出三个字符串
    return 0;
}
```

输出结果为：

```
12345
C++ OK!
I can do it!
```

结构体和枚举类型

南开大学

结构体(struct)

需求：多属性数据，如学生有姓名、年龄、身高等

如果用多个变量分别定义

```
char name[20];
int age;
int height;
```

如果属性较多，处理起来麻烦

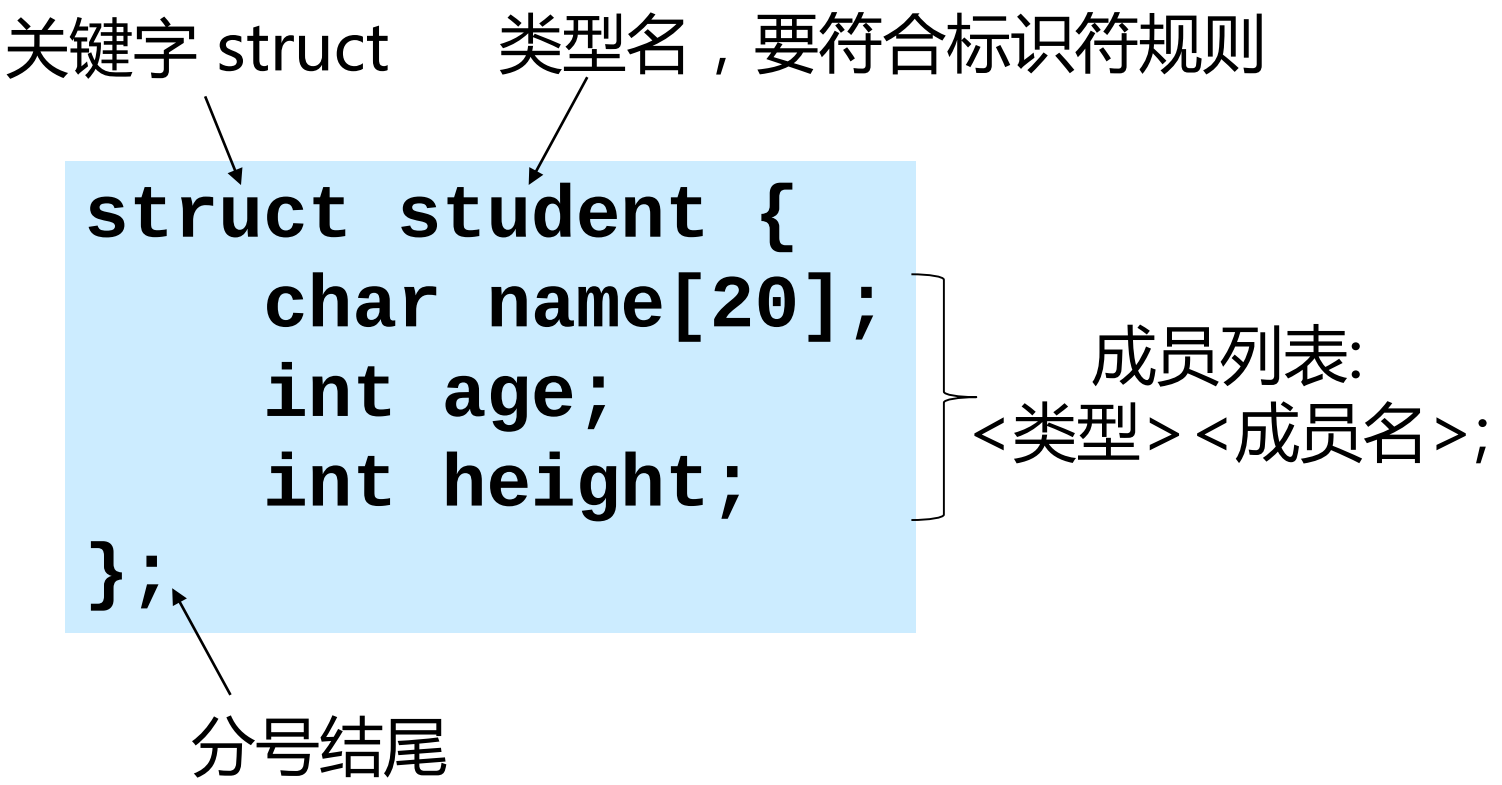
结构体

```
struct student {
    char name[20];
    int age;
    int height;
};
```

将多个属性封装在一块，
定义成一个新的数据类型

结构体(struct)

结构体定义：



结构体(struct)

结构体变量的定义和初始化：

```
struct student {
    char name[20];
    int age;
    int height;
};
```

- 类似数组初始化，利用初始化列表（花括号），给每个成员初始值

```
student s1 = {“s1”, 18, 180};
```

结构体(struct)

结构体变量的定义和初始化：

```
struct student {
    char name[20];
    int age;
    int height;
};
```

- 可以部分缺省，如果是整数，缺省值默认为 0

```
student s1={“s1”, 18};
```

结构体(struct)

结构体变量的定义和初始化：

```
struct student {
    char name[20];
    int age;
    int height;
};
```

- 初始化列表只能在变量定义时使用

```
student s1; //定义结构体变量
s1 ={“s1”, 18, 180};  //错误
```

结构体(struct)

结构体变量的访问：

```
struct student {
    char name[20];
    int age;
    int height;
};
```

```
student s1 = {“s1”, 18, 180};
cout<<s1.name<<s1.age<<s1.height;//访问成员变量

s1.age = 10; //对age重新赋值
s1.height ++; //将height加1
```

结构体(struct)

结构体变量的访问：

```
struct student {
    char name[20];
    int age;
    int height;
};
```

- 类似数组，成员只能逐个访问，不能直接访问结构体变量

```
student s1 = {“s1”, 18, 180};
cout<<s1; //error
```

结构体(struct)

结构体变量的访问：

```
struct student {
    char name[20];
    int age;
    int height;
};
```

- 不同于数组！两个相同类型的结构体变量可以相互赋值

```
student s1, s2;
s1 = s2; //结构体s2的每一个成员赋值给s1对应的成员
```

结构体(struct)

已知n个学生的注册号和成绩，计算他们的平均成绩，并列出成绩最好的前t名学生的注册号和分数

```
Input 6 student's Reg_Num & Score:
1001 88.5
1002 91
1003 85.5
1004 93.5
1005 85
1006 96

Average score:89.9
   register-number  score
1        1006        96.0
2        1004        93.5
3        1002        91.0
```

结构体(struct)

```
#include <iostream>
using namespace std;
int main(){
  const int n=6; //共有n个学生
  const int t=3; //欲找出前t名最好成绩者
  struct student{
      int index;
      float score;
  }scoretab[n]; //定义结构体，并定义结构体数组

  float sum =0;
  for (int i=0; i<n; i++){//输入注册号及成绩
    cin>>scoretab[i].index>>scoretab[i].score;
    sum += scoretab[i].score;
  }
```

结构体(struct)

```
cout.setf(ios::fixed);//设置以定点数格式输出
cout.precision(1); //点后保留1位
 //输出平均成绩
cout<<endl<<"Average score:"<<sum/n<<endl;
cout.width(25);
//输出“题头行”
cout<<"register-number score"<<endl;
```

结构体(struct)

```
for(int i=0; i<t; i++) { //找出前t名最好者
      //从i分量始 score中的最大者s
      float s = scoretab[i].score;
      int j1 = i; //j1中记录上述最大者的下标i
      //看还有否比s 更大的
      for(int j=i+1; j<n; j++){
      if(s<scoretab[j].score){//有更大的
                s=scoretab[j].score; //更大者放s
                j1=j;   //对应下标放j1
            }
      }//for j 循环体结束
     if (j1>i) {//若scoretab[i].score到scoretab[n-1].score
     //中的最大者并非score[i]时，要进行交换
           student temp  = scoretab[j1];
           scoretab [j1] = scoretab[i];
           scoretab [i] = temp;
   }
```

结构体(struct)

```
        cout.width(4);
        //输出名次号（前t名的第i+1名）
        cout<<endl<<i+1;
        cout.width(11);
        //输出第i名学生的注册号
        cout<<scoretab[i].index;
        cout.width(12);
        cout.precision(1);  //点后保留1位
        //输出第i名学生的成绩
        cout<<scoretab[i].score<<endl;
    }  //for i 循环体结束
    cout<<endl;
    return 0;
}
```

END

南開大學