**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软件工程2105

学 号： 8202210810

姓 名： 阎庆阳

**实验报告成绩：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **实验** | **实验一** | **实验二** | **实验三** | **实验四** | **实验五** | **总评** |
| **成绩** |  |  |  |  |  |  |

**批阅教师：**

**实验三 函数**

**一、实验目的**

本实验主要培养、训练学生对函数的理解，要求：

1. 掌握函数的定义、声明的方法；

2. 掌握函数的编写要求；

3. 掌握函数的调用方法；

4. 掌握函数参数的传递方法；

5. 掌握变量的作用域；

6. 掌握多文件编程方法。

**二、实验内容与要求**

1、输入自然数m和n，

（1）求他们的最大公约数（或称最大公因数）。

要求输入、输出在主函数中进行，求公约数由函数实现。

1. 在函数中求最大公约数与最小公倍数。（提示：使用引用参数）

2. 编写程序满足：声明一个函数，判断一个整数是否为素数，使用如下函数头：

bool is\_prime(int num) ,如果num是素数函数返回true，否则返回false；

利用函数is\_prime找出前200个素数，并按每行10个输出：

     2     3      5      7    11    13    17    19    23    29

3、编程实现摄氏温度到华氏温度的转换：

编写一个头文件，包含下面两个函数：

double celsius\_to\_fah(double cel)    //摄氏温度到华氏温度

double fahrenheit\_to\_cels(double fah) //华氏温度到摄氏温度

实现头文件，并编写测试程序，调用函数显示如下结果：

Celsius    Fahrenheit   |   Fahrenheit       Celsius

40.0       105.0        |   120.0            48.89

39.0       102.0        |   110.0            43.33

……       ……        |   ……             ……

31.0        87.8        |   30.0             -1.11

（测试程序为主模块，即main( )函数所在的CPP文件，头文件mytemperature.h只有函数声明；函数定义写在另一CPP文件mytemperature.cpp）

4、创建名为mytriangle.h的头文件，包括：

bool is\_valid(double side1,double side2,double side3)

double\_area(double side1,double side2, double side3)

面积=sqrt(s(s-side1)(s-side2)(s-side3))

其中s=(side1+side2+side3)/2

写测试程序：读取三角形三边长，如输入合法，计算面积，否则输出错误信息。

（测试程序为主模块，即main( )函数所在的CPP文件，头文件mytriangle.h只有函数声明；函数定义写在另一CPP文件mytriangle.cpp）

**3与4选一个完成**

5、猴子吃桃：猴子第一天摘若干桃子，当即吃了一半，还不过瘾，又吃了一个。第二天又将剩下的桃子吃掉一半，又多吃一个，以后每天如此，到第10天，发现只剩最后一个桃子，问，第一天猴子共摘多少桃子（用递归实现）。

**三、实验思考题**

1. 本实验中函数中返回的值为什么与函数类型一致？

2. 本实验中主函数调用函数时采用的是何种传递方式？

解答：

1. 函数的返回值类型用于指示函数应该返回的数据类型。因为函数的类型表示函数运行后返回的结果，若结果为int则返回int，若无返回值则为void。
2. 第一题涉及到了引用传递，其他均为值传递。在值传递方式中，调用函数时传递的参数是变量的值的副本，而不是原始变量本身。这意味着在函数内部对参数的修改不会影响到原始变量的值。在引用传递中，函数参数接收的是原始变量的引用，即直接使用原始变量作为函数参数，在函数内部可以直接修改原始变量的值。

**四、算法分析，程序结果**

#### 1.

**代码：**

#include <iostream>

using namespace std;

// 最大公约数

int gcd(int a, int b) {

if (b == 0)

return a;

return gcd(b, a % b);

}

// 最小公倍数

void gcdAndLcm(int m, int n, int& gcdResult, int& lcmResult) {

gcdResult = gcd(m, n);

lcmResult = m \* n / gcdResult;

}

int main() {

int m, n;

cout << "请输入两个自然数: ";

cin >> m >> n;

int gcdResult, lcmResult;

gcdAndLcm(m, n, gcdResult, lcmResult);

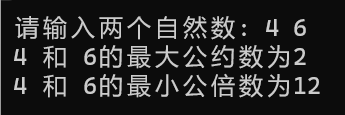
cout << m << " 和 " << n << "的最大公约数为" << gcdResult << endl;

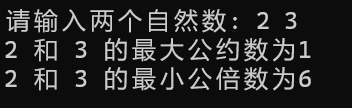
cout << m << " 和 " << n << "的最小公倍数为" << lcmResult << endl;

return 0;

}

**运行结果截图：**





#### 2.

**代码：**

#include <iostream>

#include <cmath>

using namespace std;

bool is\_prime(int num) {

if (num <= 1)

return false;

for (int i = 2; i \* i <= num; i++) {

if (num % i == 0) return false;

}

return true;

}

int main() {

int sum = 0;

int num = 2;

while (sum < 200) {

if (is\_prime(num)) {

sum++;

cout << num << " ";

if (sum % 10 == 0) cout << endl;

}

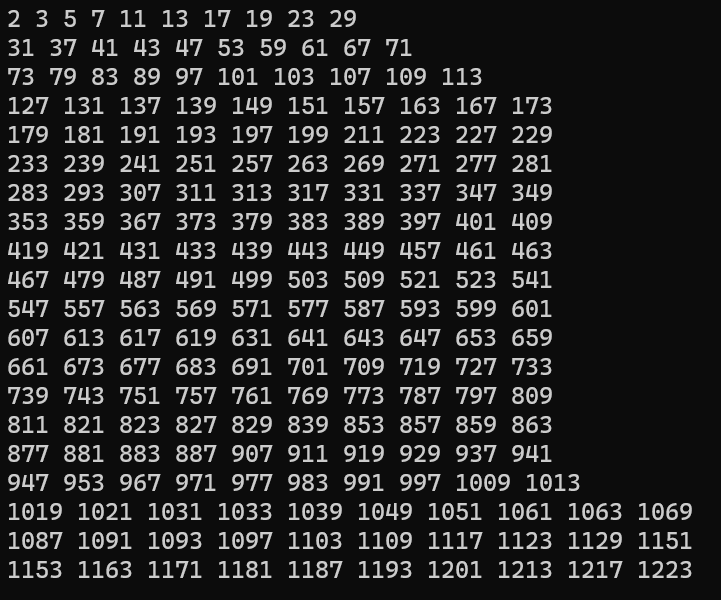
num++;

}

return 0;

}

**运行结果：**



#### 3.

**代码：**

mytemperature.h

#pragma once

#ifndef MYTEMPERATURE\_H

#define MYTEMPERATURE\_H

double celsius\_to\_fah(double cel); //摄氏温度到华氏温度

double fahrenheit\_to\_cels(double fah); //华氏温度到摄氏温度

#endif

test3\_3.cpp

#include <iostream>

#include "mytemperature.h"

using namespace std;

//摄氏温度到华氏温度

double celsius\_to\_fah(double cel) {

return cel \* 9 / 5 + 32;

}

//华氏温度到摄氏温度

double fahrenheit\_to\_cels(double fah) {

return (fah - 32) \* 5 / 9;

}

int main() {

cout << "Celsius Fahrenheit | Fahrenheit Celsius" << endl;

cout << "-----------------------------------------------------" << endl;

for (double cel = 40.0, fah = 105.0; cel >= 31.0; cel--, fah -= 3) {

cout << fixed;

cout.precision(1);

if (celsius\_to\_fah(cel) >= 100)

cout << cel << " " << celsius\_to\_fah(cel) << " | " << fah << " " << fahrenheit\_to\_cels(fah) << endl;

else

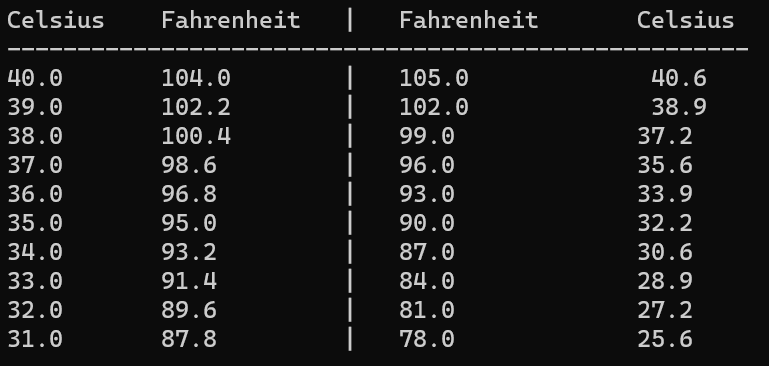
cout << cel << " " << celsius\_to\_fah(cel) << " | " << fah << " " << fahrenheit\_to\_cels(fah) << endl;

}

return 0;

}

**运行结果：**



#### 5.

**代码：**

#include <iostream>

using namespace std;

int calculatePeaches(int day) {

// 第十天时，只有1个桃子

if (day == 10) {

return 1;

}

else {

// 每天吃剩下的桃子数量是前一天吃剩下的桃子数量的一半减1

return 2 \* (calculatePeaches(day + 1) + 1);

}

}

int main() {

int totalPeaches = calculatePeaches(1);

cout << "第一天猴子共摘了 " << totalPeaches << " 个桃子" << endl;

return 0;

}

**运行结果：**

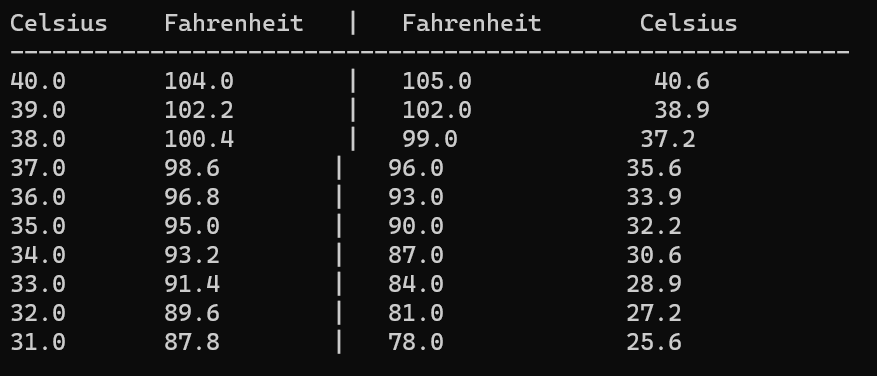


**对于第三题“实验思考题”的解答写在了题目后面。**

**五、遇到的问题与解决方法**

在第二题中，由于考虑到运算效率，在i<=时即可返回，如对于数字93，并不需要考虑93%2一直到93%92是否等于0，仅需到93/即可。但我在使用for循环时，条件语句为for(int i=2; i<=sqrt(num);i++)，导致数字2一直被认为不是素数，后来改变了策略，用i\*i<=num作为条件，才让数字2正常被识别为素数。

第三题中，考虑到输出的华摄度数值可能为两位数或三位数，导致输出的效果不美观，如下图所示，竖线无法对齐，我考虑加入一层判断，若华摄度为两位数，则紧接着在后面多输出一个空格，使得输出更加整洁。



**六、体会**

本次实验的内容主要是函数式编程，虽然在c++编程时经常会用到函数，但是像是第一题中要求使用引用参数的情况比较少，第一题的编写又加深了我对引用的理解与使用。第三题要求编写头文件并实现，虽然头文件只需要将函数声明添加进去即可，但是因为接触的比较少，我还是去网上搜索了一下，发现标准的写法需要添加#ifndef、#define和#endif，目的是防止该头文件被重复引用，但什么是重复引用呢？是不能在不同的两个文件中使用include来包含这个头文件吗？其实真正的意思是指一个头文件在同一个cpp文件中被include了多次，这种错误常常是由于include嵌套造成的。

虽然实验的整体实现难度不大，但仍然在这次的实验中受益匪浅，尤其是注意到了头文件中一般不被注意到的规范。

**实验四 数组与指针**

【**实验目的**】

1、进一步加深对数组的理解，掌握数组的定义方法；

2、掌握数组的处理方法、数组作为函数参数的使用方法，以及搜索与排序的应用。

3、掌握指针的概念、指针变量定义格式以及指针的运算；

4、掌握指针与数组、函数的关系；

5、理解内存动态分配的含义、熟练掌握内存动态分配方法；

6、掌握递归函数的定义方法。

【实验内容与步骤】

**（一）数组**

1、打印不同的数：

编写一个程序，读入10个数，输出其中不同的数（即如果一个数出现多次，只打印一次）。

提示：读入的数如果是一个新的值，则将其存入一个数组。否则，将其丢弃。输入完毕后，数组中保存的就是不同的数。

下面是一个运行样例：

Enter

Enter ten numbers: 1 2 3 2 1 6 3 4 5 2

The distinct numbers are: 1 2 3 6 4 5

2、起泡排序：

利用起泡排序算法编写一个排序函数。起泡排序算法分若干趟对数组进行处理。每趟处理中，对相邻元素进行比较。若为降序，则交换；否则，保持原顺序。此技术被称为起泡排序（bubble sort）或下沉排序（sinking sort），因为较小的值逐渐地“冒泡”到上部，而较大值逐渐下沉到底部。

算法可描述如下：

bool changed = true;

do

{

changed = false;

for (int j = 0; j < listSize – 1; j++)

if (list[j] > list[j+1])

{

swap list[j] with list[j+1];

changed = true;

}

} while (changed);

很明显，循环结束后，列表变为升序。容易证明do循环最多执行listSize – 1次。

编写测试程序，读入一个含有10个双精度数字的数组，调用函数并显示排列后的数字。

3、游戏：存物柜问题：

一个学校有100个存物柜，100个学生。开学第一天所有存物柜都是关闭的。第一个学生（记为S1）来到学校后，打开所有的存物柜。第二个学生S2，从第二个存物柜（记为L2）开始，每隔两个存物柜，将它们关闭。第三个学生S3从第三个存物柜L3开始，每隔三个，将它们的状态改变（开着的关上，关着的打开）。学生S4，从L4开始，每隔四个改变它们的状态。学生S5，从L5开始，每隔五个改变状态。依此类推，直至学生S100改变L100的状态。

当所有学生完成这个过程，那些存物柜是开着的？编写一个程序求解此问题，显示所有开着的柜子号码，号码之间用一个空格隔开。

提示：使用一个100个布尔型元素的数组，每个元素代表存物柜是开（true）或关（false）。最初所有的储物柜都是关闭的。

4、合并两个排列好的数组：

编写如下函数，合并两个排列好的数组，形成一个新的排列好的数组。

void merge(const int list1[], int size1, const int list2[], int size2, int list3[])

使用size1+size2次比较实现函数。编写测试程序，提示用户输入两个排列好的数组，并显示合并以后的数组。下面是一个运行样例。注意，输入数据的第一个数字是数组的元素数，而不是数组的一部分。假定数组大小不超过80。

Enter

Enter

Enter list1: 5 1 5 16 61 111

Enter list1: 4 2 4 5 6

The merged list is 1 2 4 5 5 6 16 61 111

5、检验子串：

编写如下函数，检验C字符串s1是否是C字符串s2的子串。如果匹配，返回s1在s2中的下标，否则返回–1。

int indexOf(const char s1[], const char s2[])

编写测试程序，读入两个C字符串，检验C字符串s1是否是C字符串s2的子串。下面是程序的运行样例：

Enter

Enter

Enter

Enter the first string: welcome

Enter the second string: We welcome you!

indexOf(“welcome”, “We welcome you!”) is 3

Enter

Enter the first string: welcome

Enter the second string: We invite you!

indexOf(“welcome”, “We invite you!”) is –1

6、字符串中每个字母出现的次数：

请使用如下函数头编写函数，数出字符串中每个字母出现的次数。

void count(const char s[], int counts[])

counts是一个有26个元素的整数数组。const[0]，const[1]，…，const[25]分别记录a，b，…，z出现的次数。字母不分大小写，例如字母A和字母a都被看作a。

编写测试程序，读入字符串并调用count函数，显示非零的次数。下面是程序的一个运行样例：

Enter

Enter a string: Welcome to New York!

c: 1 times

e: 3 times

k: 1 times

l: 1 times

m: 1 times

n: 1 times

o: 3 times

r: 1 times

t: 1 times

w: 2 times

y: 1 times

**（二）指针**

1、上机验证下列程序的运行结果（有错误的话自己补充完善）

(1) void main()

｛

int i,j,\*pi,\*pj; //此处的\*表示定义指针变量，而非间接运算符

pi=&i;

pj=&j;

i=5;j=7;

cout<<i<<’\t’<<j<<’\t’<<pi<<’\t’<<pj;

cout<<&i<<’\t’<<\*&i<<’\t’<<&j<<’\t’<<\*&j;

}

运行结果：

上述结果中，pi与&i,pj与&j是地址值，随编译程序而变化，不确定。

(2) int main() //C语言程序，要了解

{

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

int \*p,i;

p=a; //将数组a首地址送给p

for (i=0;i<3;i++)

printf("%d,%d,%d,%d\n",a[i],p[i],\*(p+i),\*(a+i)); //与cout功能差不多

}

运行结果：

1,1,1,1

2,2,2,2

3,3,3,3

通过这两道题目，希望学生掌握数组元素与指向数组的指针的不同。

a[i]表示数组中下标为i的元素。

a[i]←p[i]←\*(p+i)←\*(a+i)

a是数组名，表示数组首地址，(p+i)表示数组中第i个元素的地址，\*(p+i) 相当于a[i]。

(3)通过如下的问题理解递归函数的定义与调用（递归未讲，可以后做）

#include <iostream>

using namespace std; //#include “stdio.h”

void f(char \*st,int i)

{

st[i]=’\0’;

cout<<st; // printf(“%s\n”,st);

if (i>1) f(st,i-1);

}

void main()

{

char st[]=”abcd”;

f(st,4);

}

补充完整，运行时输出为\_\_\_abcdabcaba\_\_\_\_\_

(4)下面程序的主函数中能保证p[0]输出1，p[1]输出2吗？如何修改以保证之（提示：在函数f中使用new生成动态数组；在main中用delete释放。）

#include<iostream>

using namespace std;

int \*f()

{

int list[]={1,2,3,4};

return list;

}

void main()

{

int \*p=f();

cout<<p[0]<<endl;

cout<<p[1]<<endl;

}

2、程序设计

(1)编写函数检查字符串s1是否为字符串s2的子串，若是，返回第一次匹配的下标，否则返回-1。在主程序中输入字符串s1与s2，调用函数实现。

函数原型：int indexof(const char \*s1,const char \*s2);

(2)编写一个函数将以字符串形式表示的一个16进制数转换为10进制数，并在主函数中测试。函数原型 int parseHex(const char \*const hexString);

如：调用函数 parseHex(“A5”);返回165

1. 主程序中建立一动态数组（使用new），数组元素及元素个数由键盘输入，动态调试观察指针及指针指向的内容；设计一个函数对数组由小到大排序；主程序中用指针方式输出数组元素；最后释放数组内存（delete）。

【完成实验报告】

**实验报告只要求写程序设计部分**

**三、算法分析，程序结果**

**（一）数组**

#### 1.

**代码：**

#include <iostream>

using namespace std;

int main() {

cout << "Enter ten numbers: ";

int array[10];

int num;

int len = 0;

bool flag;

for (int i = 0; i < 10; i++) {

flag = true;

cin >> num;

for (int j = 0; j < i; j++) {

// 判断数组中num是否已存在

if (array[j] == num) {

flag = false;

break;

}

}

// 若数组中不存在num，则将num加入数组

if (flag) {

array[len] = num;

len++;

}

}

cout << "The distinct numbers are: ";

for (int i = 0; i < len; i++) {

cout << array[i] << " ";

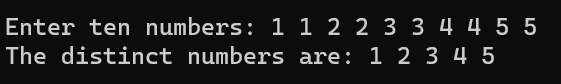
}

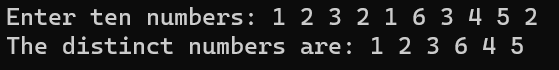
cout << endl;

return 0;

}

**运行结果：**





#### 2.

**代码：**

#include <iostream>

using namespace std;

void bubbleSort(double list[], int listSize) {

bool changed = true;

do {

changed = false;

for (int j = 0; j < listSize - 1; j++) {

if (list[j] > list[j + 1]) {

// 交换list[j]和list[j+1]

double temp = list[j];

list[j] = list[j + 1];

list[j + 1] = temp;

changed = true;

}

}

} while (changed);

}

int main() {

const int size = 10;

double numbers[size];

cout << "请输入10个双精度数字: " << endl;

for (int i = 0; i < size; i++) {

cin >> numbers[i];

}

bubbleSort(numbers, size);

cout << "排序后的数字：" << endl;

for (int i = 0; i < size; i++) {

cout << numbers[i] << " ";

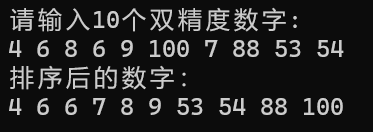
}

cout << endl;

return 0;

}

**运行结果：**



#### 3.

**代码：**

#include <iostream>

using namespace std;

int main()

{

bool states[101];

for (int i = 1; i <= 100; i++)

states[i] = true;

for (int i = 2; i <= 100; i++) {

for (int j = i; j <= 100; j += i) {

states[j] = !states[j];

}

}

cout << "开着的柜子号码为：" << endl;

int sum = 0;

for (int i = 1; i <= 100; i++) {

if (states[i]) {

cout << i << " ";

sum++;

if (sum % 10 == 0)

cout << endl;

}

}

return 0;

}

**运行结果：**



#### 4.

**代码：**

#include <iostream>

using namespace std;

void merge(const int list1[], int size1, const int list2[], int size2, int list3[]) {

int i = 0; // 指向list1的索引

int j = 0; // 指向list2的索引

int k = 0; // 指向list3的索引

// 比较两个数组，将较小的元素放入list3中，直到一个数组被完全遍历

while (i < size1 && j < size2) {

if (list1[i] <= list2[j]) {

list3[k] = list1[i];

i++;

}

else {

list3[k] = list2[j];

j++;

}

k++;

}

// 将剩余的元素放入list3中

while (i < size1) {

list3[k] = list1[i];

i++;

k++;

}

while (j < size2) {

list3[k] = list2[j];

j++;

k++;

}

}

int main() {

int size1, size2;

cout << "Enter list1: ";

cin >> size1;

int list1[80];

for (int i = 0; i < size1; i++) {

cin >> list1[i];

}

cout << "Enter list2: ";

cin >> size2;

int list2[80];

for (int i = 0; i < size2; i++) {

cin >> list2[i];

}

int list3[160]; // 假设最终的合并数组不会超过160个元素

// 调用合并函数并输出结果

merge(list1, size1, list2, size2, list3);

cout << "The merged list is ";

for (int i = 0; i < size1 + size2; i++) {

cout << list3[i] << " ";

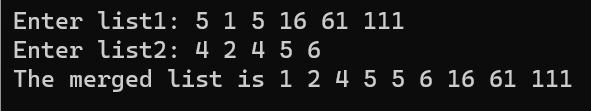
}

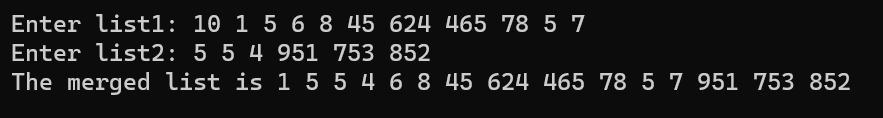
cout << endl;

return 0;

}

**运行结果：**





#### 5.

**代码：**

#include <iostream>

using namespace std;

const int MAX\_SIZE = 100; // 字符串的最大尺寸

int indexOf(const char s1[], const char s2[]) {

int i = 0, j = 0;

int matchIndex = -1; // 匹配的下标

int currentIndex = 0;// 当匹配到后，重新匹配的位置

bool flag = false;//若匹配到了相同的字符串

while (s2[j] != '\0') {

if (s1[i] == s2[j]) {

// 若匹配到相同字符，此时是s1第一个字符匹配，则重置匹配下表为此时s2的下标

if (i == 0) {

matchIndex = j;

currentIndex = j;

}

flag = true;

i++;

}

else {

// 若发现为匹配，则重置匹配状态

i = 0;

matchIndex = -1;

if (flag) {

j = currentIndex;

flag = false;

}

}

if (s1[i] == '\0') {

// s1已经完全匹配

break;

}

j++;

}

return matchIndex;

}

int main()

{

char s1[MAX\_SIZE];

char s2[MAX\_SIZE];

cout << "Enter the first string: ";

cin.getline(s1, MAX\_SIZE);

cout << "Enter the second string: ";

cin.getline(s2, MAX\_SIZE);

cout << "indexOf(“" << s1 << "”, “" << s2 << "”) is ";

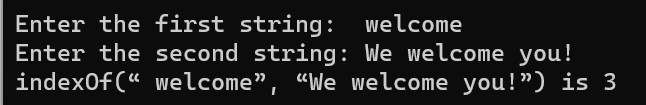
int index = indexOf(s1, s2);

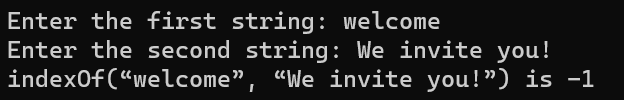
cout << index << endl;

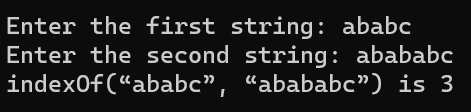
return 0;

}

**运行结果：**







#### 6.

**代码：**

#include <iostream>

#include <cstring>

#include <cctype>

using namespace std;

void count(const char s[], int counts[]) {

int len = strlen(s); // 获取字符串长度

for (int i = 0; i < len; i++) {

if (isalpha(s[i])) { // 检查字符是否为字母

char c = tolower(s[i]); // 将字符转换为小写

counts[c - 'a']++; // 增加对应字母的计数

}

}

}

int main() {

const int SIZE = 26;

int counts[SIZE] = { 0 }; // 初始化计数数组

char s[100]; // 假设字符串最大长度为100

cout << "Enter a string: ";

cin.getline(s, 100); // 读取输入的字符串

count(s, counts); // 调用计数函数

for (int i = 0; i < SIZE; i++) {

if (counts[i] > 0) { // 仅显示非零次数的字母

char c = 'a' + i;

cout << c << ": " << counts[i] << " times" << endl;

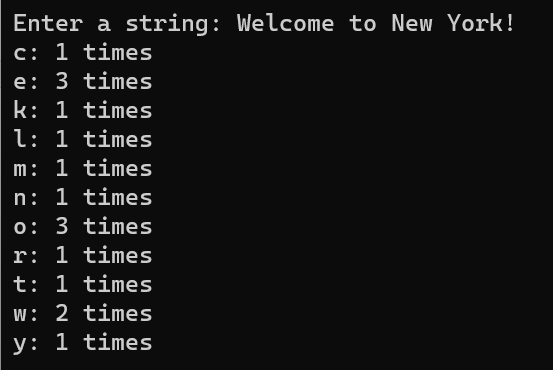
}

}

return 0;

}

**运行结果：**



**（二）指针**

#### 1.

（1）

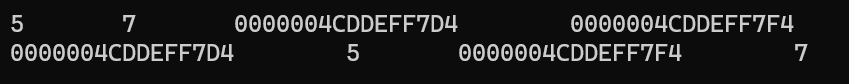
**修改：**

第一处：”{”为中文符号，修改为”{”英文符号；

第二处：程序未添加#include <iostream> using namespace std;无法使用cout方法；

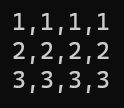
第三处：输出语句中引号均为中文符号，修改其为英文符号。

**运行结果：**



（2）

**运行结果：**



**（3）**

**填空处已在前面题目中完成。**

**（4）**

以上函数不能保证p[0]输出1，p[1]输出2，输出结果如下：



原因： f() 函数中创建的数组 list 是一个局部变量，其生命周期仅在 f() 函数内部。当 f() 函数返回时，list 数组将不再有效，因此返回的指针 p 将引用无效的内存位置，访问其中的元素将导致未定义行为。

**修改代码：**

#include <iostream>

using namespace std;

int\* f()

{

int\* list = new int[4]; // 在堆上动态分配内存

list[0] = 1;

list[1] = 2;

list[2] = 3;

list[3] = 4;

return list;

}

int main()

{

int\* p = f();

cout << p[0] << endl; // 输出1

cout << p[1] << endl; // 输出2

delete[] p; // 释放动态分配的内存

return 0;

}

**修改后运行结果：**



#### 2.

**（1）**

**代码：**

#include <iostream>

#include <cstring>

using namespace std;

int indexof(const char\* s1, const char\* s2) {

int len1 = strlen(s1);

int len2 = strlen(s2);

for (int i = 0; i <= len2 - len1; i++) {

int j;

for (j = 0; j < len1; j++) {

if (s2[i + j] != s1[j]) {

break;

}

}

if (j == len1) {

return i;

}

}

return -1;

}

int main() {

const int SIZE = 100;

char s1[SIZE];

char s2[SIZE];

cout << "Enter s1: ";

cin.getline(s1, SIZE);

cout << "Enter s2: ";

cin.getline(s2, SIZE);

int result = indexof(s1, s2);

if (result == -1) {

cout << "s1 不是 s2 的子串" << endl;

}

else {

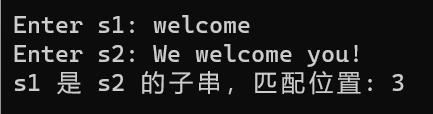
cout << "s1 是 s2 的子串，匹配位置: " << result << endl;

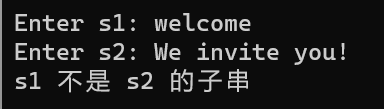
}

return 0;

}

**运行结果：**





**（2）**

**代码：**

#include <iostream>

using namespace std;

int parseHex(const char\* const hexString) {

int result = 0;

int i = (hexString[0] == '0' && hexString[1] == 'x') ? 2 : 0;

while (hexString[i] != '\0') {

int digit;

if (hexString[i] >= '0' && hexString[i] <= '9') {

digit = hexString[i] - '0';

}

else if (hexString[i] >= 'A' && hexString[i] <= 'F') {

digit = hexString[i] - 'A' + 10;

}

else if (hexString[i] >= 'a' && hexString[i] <= 'f') {

digit = hexString[i] - 'a' + 10;

}

else {

break;

}

result = result \* 16 + digit;

i++;

}

return result;

}

int main() {

const int SIZE = 100;

char hexString[SIZE];

cout << "16进制数: ";

cin.getline(hexString, SIZE);

int result = parseHex(hexString);

cout << "16进制数 " << hexString << " 表示为10进制为: " << result << endl;

return 0;

}

**运行结果：**





**（3）**

**代码：**

#include <iostream>

using namespace std;

void sortArray(int\* arr, int size) {

for (int i = 0; i < size - 1; i++) {

for (int j = 0; j < size - i - 1; j++) {

if (\*(arr + j) > \*(arr + j + 1)) {

int temp = \*(arr + j);

\*(arr + j) = \*(arr + j + 1);

\*(arr + j + 1) = temp;

}

}

}

}

void printArray(int\* arr, int size) {

for (int i = 0; i < size; i++) {

cout << \*(arr + i) << " ";

}

cout << endl;

}

int main() {

int size;

cout << "请输入数组长度: ";

cin >> size;

int\* arr = new int[size];

cout << "请输入数组元素: ";

for (int i = 0; i < size; i++) {

cin >> \*(arr + i);

}

sortArray(arr, size);

cout << "排序后 : ";

printArray(arr, size);

delete[] arr;

return 0;

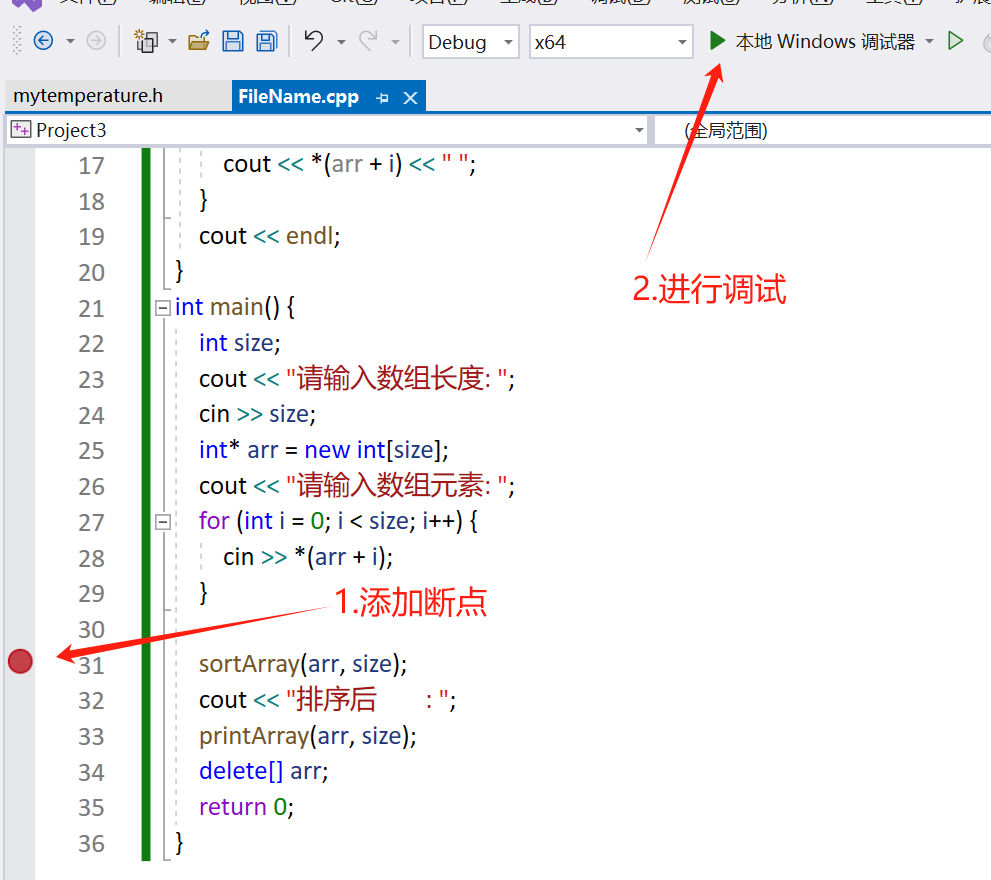
}

**运行结果：**

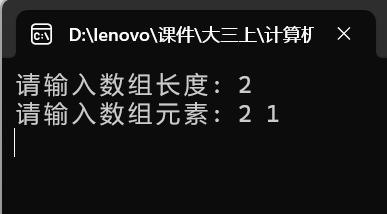


**调试结果：**

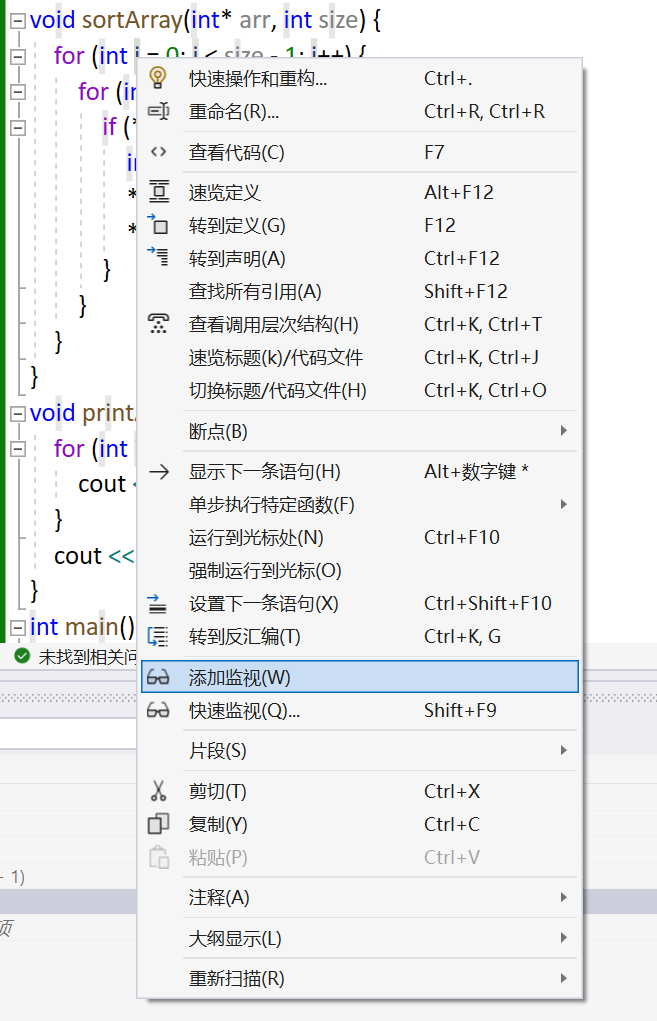
1. **添加断点，进入调试**

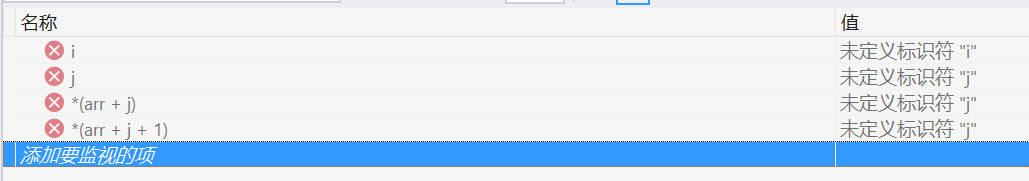


1. **初始化数组**



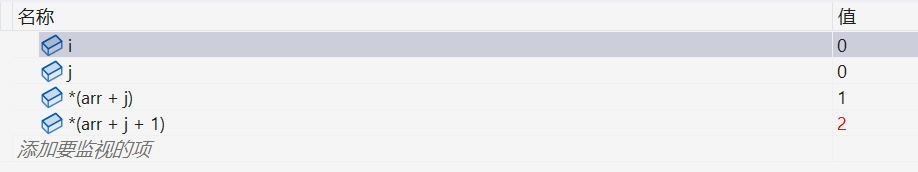
1. **添加参数进行监视**

****



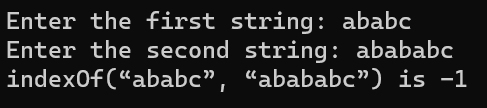
1. **监视过程**

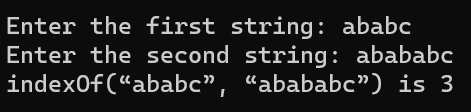




**四、遇到的问题与解决方法**

数组第五题：一开始我的匹配策略为逐个字符进行匹配，若匹配失败后则从失败的位置后重新匹配，虽然第五题给出的例子能够顺利执行，给出正确的返回值，但匹配项如下：s1=”ababc”，s2=”abababc”，则在匹配过程中，当index=4时，会匹配失败，此时会从index=5的地方重新匹配，显然会匹配失败，如下图所示，与实际不符。修改策略为添加一个表示之前是否已有字符匹配的flag，再添加一个currentIndex，表示此时若有字符串已经匹配，如上述例子中的currentIndex=0，此时若匹配失败，则从currentIndex后一项重新匹配，即index=1处，这样得到了正确的输出结果，如下第二张图。





**五、体会**

本次实验的内容较多，主要涉及的是数组与指针的使用。

数组部分，由于本身数组的使用频次较高，实验中的六个题目的实现难度并不大，因为个人习惯，我可能将数组作为参数返回，或是将数组作为函数参数的使用较少，而将数组作为全局变量较多，所以本次实验主要还是采取了前者，一来熟悉了将数组作为参数和返回值的用法，二来也把老师课堂上讲的数组传的是地址这一理论付诸实践，加深理解。

指针部分，因为数组的使用较少，对于指针的理解也较为浅显，在上课以前，我只把指针作为指向地址的参数，但实际使用却很少，如本次实验中，将函数的形参设置为const char\*类型时，既可以传入指针变量，也可以直接将char类型数组作为参数。再比如指针第二大题的第三小问中，利用到了new动态添加元素，和delete方法释放内存，都是将之前大概理解的内容的具体实现，让我自己更加熟悉对于指针、newh和delete的使用。