**计算机程序设计基础（C++)**

**实验报告**

专业班级：软件工程2402

学 号：8209240201

姓 名：徐昱隆

**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？

**四、算法分析，程序结果**

1. #include <iostream>

using namespace std;

// 求最大公约数的函数

int gcd(int m, int n) {

while (n!= 0) {

int temp = n;

n = m % n;

m = temp;

}

return m;

}

// 求最大公约数和最小公倍数的函数，使用引用参数

void gcdAndLcm(int m, int n, int& gcdValue, int& lcmValue) {

gcdValue = gcd(m, n);

lcmValue = m \* n / gcdValue;

}

int main() {

int m, n, gcdValue, lcmValue;

cout << "请输入第一个自然数 m: ";

cin >> m;

cout << "请输入第二个自然数 n: ";

cin >> n;

gcdValue = gcd(m, n);

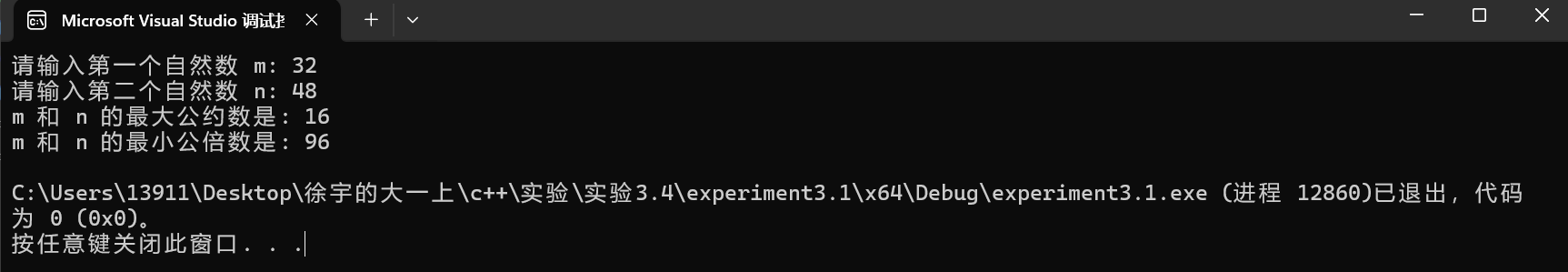
cout << "m 和 n 的最大公约数是: " << gcdValue << endl;

gcdAndLcm(m, n, gcdValue, lcmValue);

cout << "m 和 n 的最小公倍数是: " << lcmValue << endl;

return 0;

}



2.#include <iostream>

using namespace std;

// 函数用于判断一个数是否为素数

bool is\_prime(int num) {

if (num <= 1) {

return false; // 1 以下的数不是素数

}

for (int i = 2; i \* i <= num; i++) { // 从 2 到 num 的平方根遍历

if (num % i == 0) { // 如果能被整除，不是素数

return false;

}

}

return true; // 否则是素数

}

int main() {

int count = 0; // 用于计数找到的素数个数

int num = 2; // 从 2 开始判断

while (count < 200) { // 当找到的素数个数小于 200 时

if (is\_prime(num)) { // 如果是素数

cout << num << " "; // 输出该素数

count++; // 计数加 1

if (count % 10 == 0) { // 如果输出满 10 个素数

cout << endl; // 换行

}

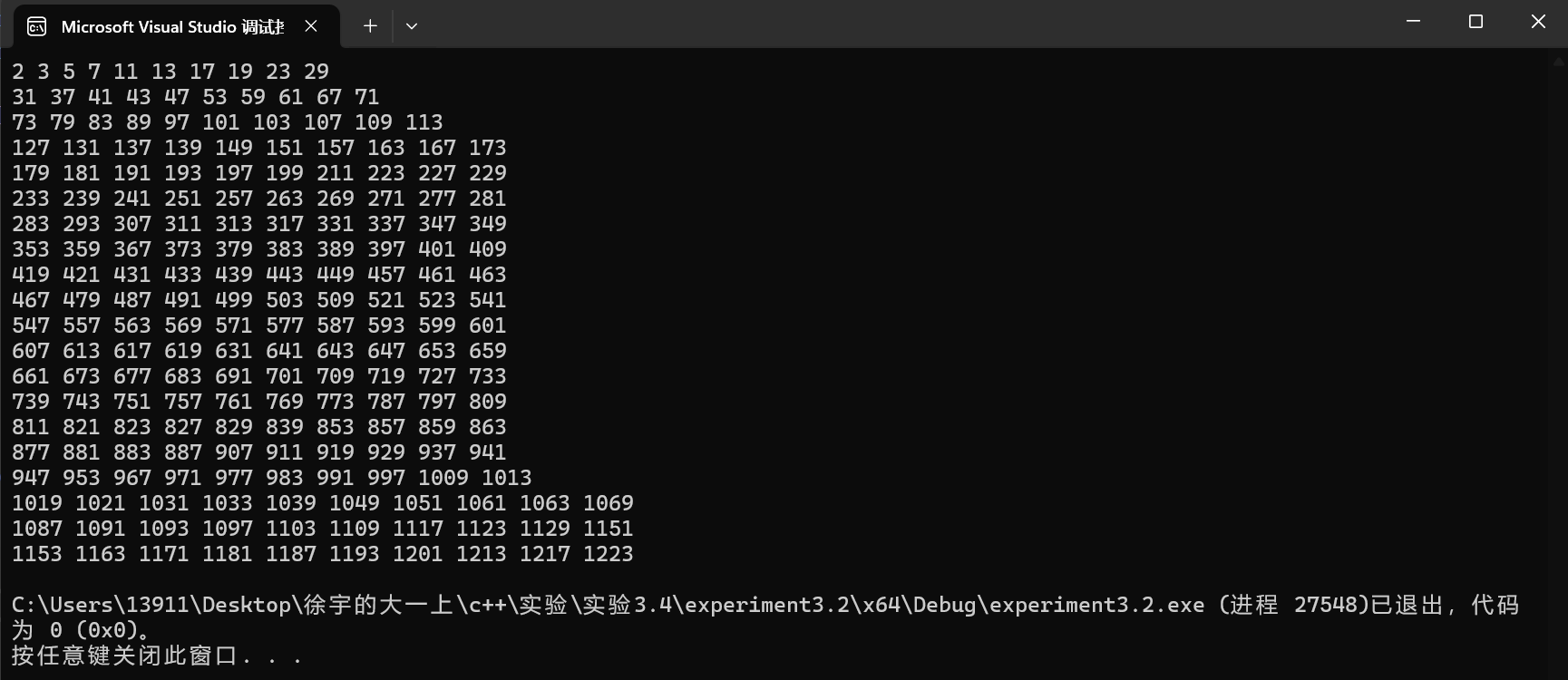
}

num++; // 检查下一个数

}

return 0;

}

****

3. #pragma once

double celsius\_to\_fah(double cel);

double fahrenheit\_to\_cels(double fah);

#include "mytemperature.h"

double celsius\_to\_fah(double cel) {

return cel \* 9.0 / 5.0 + 32;

}

double fahrenheit\_to\_cels(double fah) {

return (fah - 32) \* 5.0 / 9.0;

}

#include <iostream>

#include "mytemperature.h"

Using namespace std;

int main() {

cout << "Celsius\tFahrenheit\t|\tFahrenheit\tCelsius" << endl;

double celsius\_values[] = {40.0, 39.0, 31.0};

double fahrenheit\_values[] = {105.0, 102.0, 87.8};

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

double celsius = celsius\_values[i];

double fahrenheit = fahrenheit\_values[i];

double fah\_converted = celsius\_to\_fah(celsius);

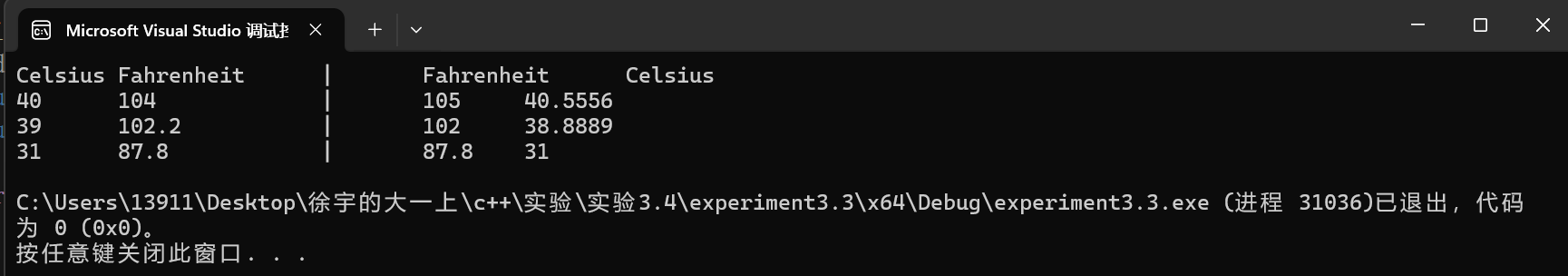
double cel\_converted = fahrenheit\_to\_cels(fahrenheit);

cout << celsius << "\t" << fah\_converted << "\t\t|\t" << fahrenheit << "\t" << cel\_converted << endl;

}

return 0;

}



4.

5. #include <iostream>

using namespace std;

int main() {

int day = 10; // 表示第 10 天

int remainingPeaches = 1; // 第 10 天剩下 1 个桃子

while (day > 1) { // 从第 10 天往前推到第 2 天

remainingPeaches = (remainingPeaches + 1) \* 2; // 计算前一天剩下的桃子数

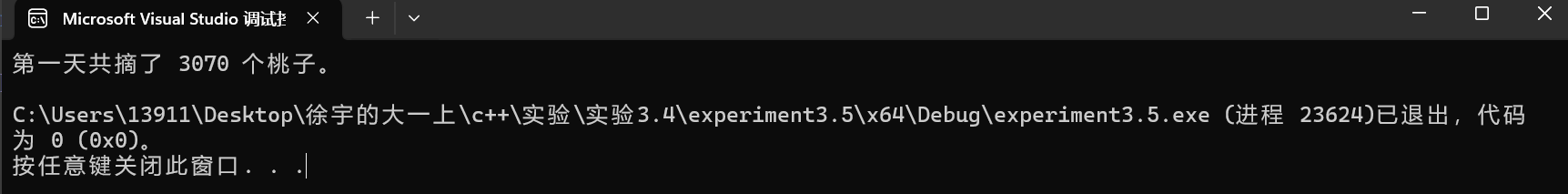
day--;

}

remainingPeaches = (remainingPeaches + 1) \* 2; // 计算第一天摘下的桃子数

cout << "第一天共摘了 " << remainingPeaches << " 个桃子。" << endl;

return 0;

}

**五、遇到的问题与解决方法**

1. 写第二题的时候遇到了以下问题

（1）素数判断效率问题：如果使用简单的方法，即检查从2到 num-1 的所有数是否能整除 num ，那么对于大数来说效率非常低。后来经过在网上查找相关资料了解到可以优化算法，只查到sqrt(num) ，因为如果 num 有一个因子大于它的平方根，那么它必然有一个因子小于或等于它的平方根。

（2）输出格式问题：：需要按照每行10个素数的格式输出，这可能需要在输出时进行额外的计数和换行处理。在经过学长的指点下，了解到可以使用一个计数器来跟踪当前输出的素数数量，并在达到10个时输出换行符。

**六、体会**

在 C++函数章节的实验课中，我收获颇丰。函数就像是程序里的一个个小工具，各有特定功能，能让复杂的编程任务变得条理清晰。

编写函数时，要先明确其功能与输入输出，就像做数学题先确定解题思路。比如计算一组数的平均值，函数的输入就是这组数，输出则是平均值。定义函数参数和返回值类型很关键，这能保证函数被正确调用和使用。

函数的调用让代码可重复利用，减少冗余。在大型项目里，能提高开发效率与代码可读性。例如，多次用到打印三角形图案的功能，写成函数后，每次只需调用，不必重复代码。

实验中，我也碰到不少问题。像参数传递错误，值传递时对形参的修改不会影响实参，引用传递才会，没分清就容易出错。还有局部变量与全局变量的作用域问题，若同名，局部变量会覆盖全局变量，易造成逻辑错误。

调试过程让我学会用断点调试工具，逐行执行代码，观察变量变化，从而快速定位错误。这使我明白，编程不仅是写代码，更是解决问题的过程。

通过这章实验课，我对 C++函数有了深入理解，编程思维和动手能力都得到提升。以后我会多练习，更好地掌握函数的运用，为编写复杂程序奠定基础。

**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 “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\_\_\_\_\_\_\_\_

(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. #include<iostream>

using namespace std;

int main()

{

int arr[10];//定义数组

int temparr[10];//定义临时数组

int num = 0;//定义显示数

int temp = 0;//定义临时变量记录是否有数据重复

cout << "请输入数组中的10个数" << endl;//提示输入数据

for (int i = 0; i < 10; i++)//搭建循环输入数据

{

cin >> temparr[i];//输入数据

}

for (int i = 0; i < 10; i++)//搭建循环判断赋值

{

temp = 0;//将临时记录变量赋为0

for (int j = 0; j < i; j++)//创建内层循环

{

if (temparr[i] == temparr[j])//如果目前的数值和前面的数值有重复

{

temp++;//临时变量+1

}

}

if (temp == 0)//如果临时变量为0 即不存在重复

{

arr[num] = temparr[i];//将不重复的数值在数组中的最新一位赋值

num++;//显示数+1

}

}

cout << "您输入的数字是(已删去重复值)" << endl;//赋值后对数组进行打印输出

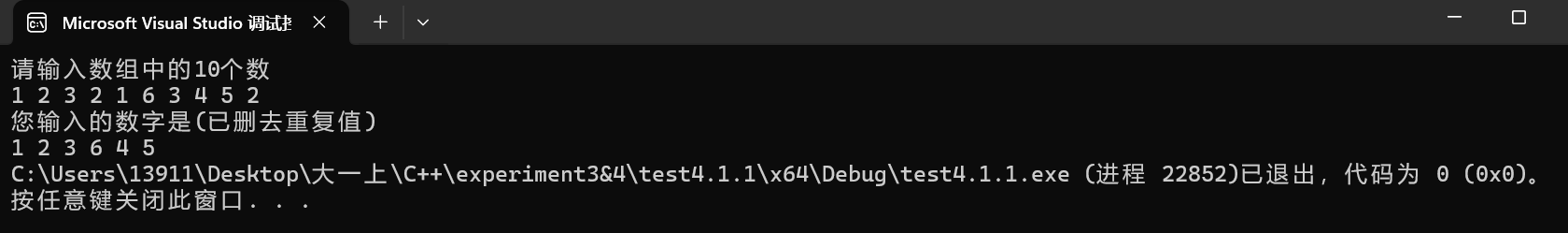
for (int i = 0; i < num; i++)//只打印在显示数内的数据

{

cout << arr[i] << ' ';//打印数据

}

return 0;

}

2. #include <iostream>

using namespace std;

// 交换两个元素的函数

void swap(double& a, double& b) {

double temp = a;

a = b;

b = temp;

}

// 冒泡排序函数

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]) {

swap(list[j], list[j + 1]);

changed = true;

}

}

} while (changed);

}

// 测试程序

int main() {

double arr[10];

cout << "请输入 10 个双精度数字：" << endl;

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

cin >> arr[i];

}

bubbleSort(arr, 10);

cout << "排序后的数字为：" << endl;

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

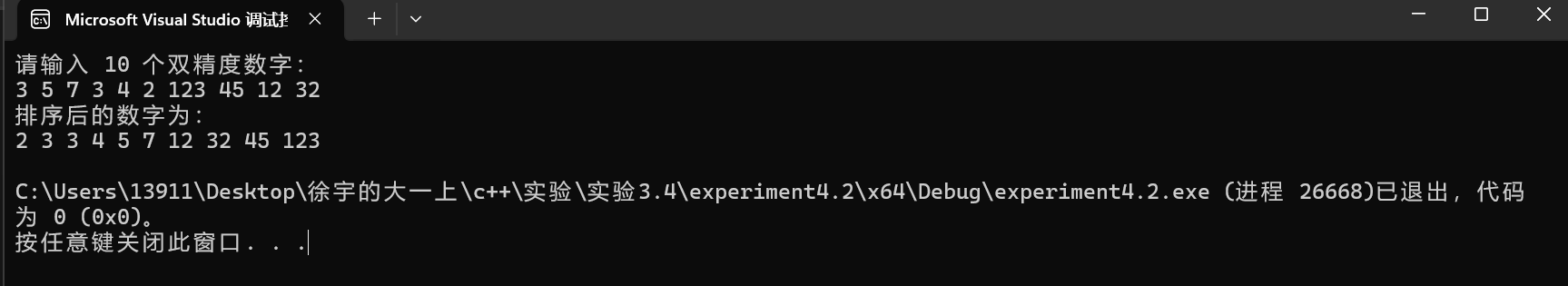
cout << arr[i] << " ";

}

cout << endl;

return 0;

}



3. #include <iostream>

int main() {

bool lockers[100] = {false}; // 初始化数组，所有存物柜都关闭

for (int i = 1; i <= 100; i++) { // 模拟每个学生的操作

for (int j = i - 1; j < 100; j = j + i) { // 从第 i 个存物柜开始，每隔 i 个改变状态

lockers[j] =!lockers[j];

}

}

for (int i = 0; i < 100; i++) { // 找出开着的存物柜

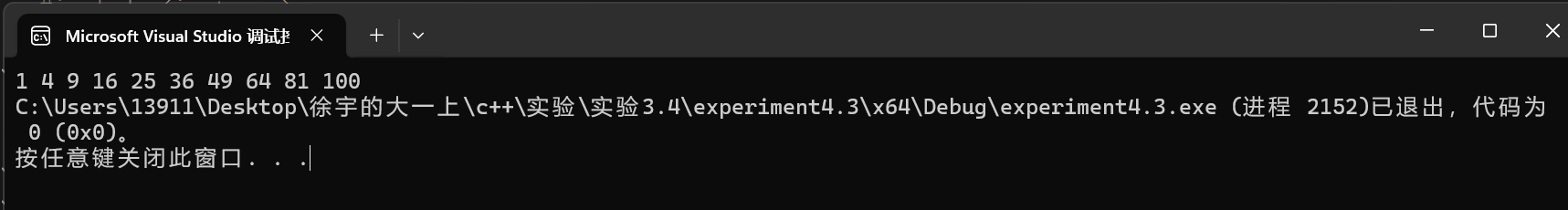
if (lockers[i]) {

std::cout << (i + 1) << " "; // 输出开着的存物柜编号

}

}

return 0;

}

4. #include <iostream>

#include <vector>

using namespace std;

// 合并两个排序好的数组

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

int i = 0, j = 0, k = 0;

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

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

list3[k++] = list1[i++];

} else {

list3[k++] = list2[j++];

}

}

while (i < size1) {

list3[k++] = list1[i++];

}

while (j < size2) {

list3[k++] = list2[j++];

}

}

int main() {

int size1, size2;

cout << "Enter the number of elements in list1: ";

cin >> size1;

vector<int> list1(size1);

cout << "Enter list1: ";

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

cin >> list1[i];

}

cout << "Enter the number of elements in list2: ";

cin >> size2;

vector<int> list2(size2);

cout << "Enter list2: ";

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

cin >> list2[i];

}

// 合并后的数组大小为两个数组大小之和

vector<int> list3(size1 + size2);

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

cout << "The merged list is: ";

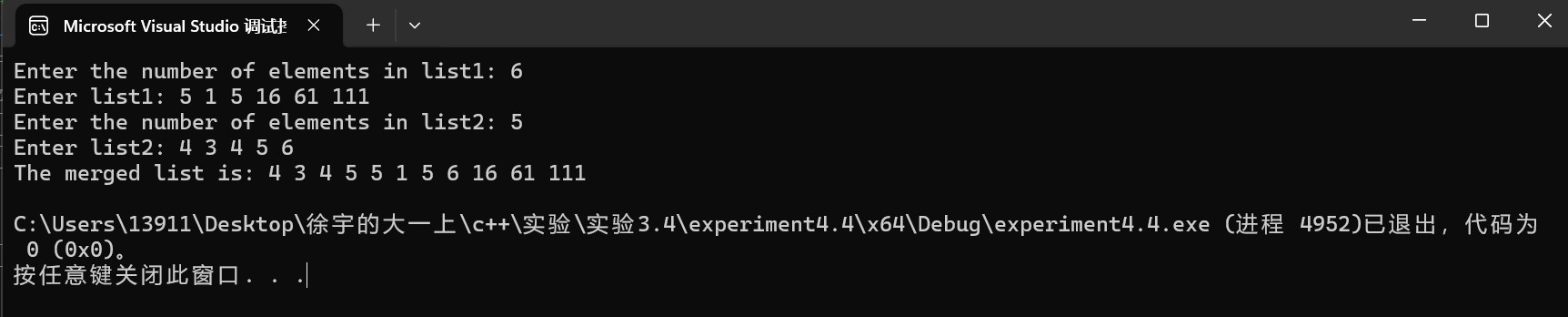
for (int i = 0; i < size1 + size2; ++i) {

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

}

cout << endl;

return 0;

}

5. #include <iostream>

#include <cstring> // 用于使用 strstr 函数

using namespace std;

// 检验 s1 是否是 s2 的子串

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

const char\* found = strstr(s2, s1);

if (found) {

return found - s2; // 计算 s1 在 s2 中的起始下标

}

return -1; // 如果 s1 不是 s2 的子串，返回 -1

}

int main() {

const int maxLen = 256;

char str1[maxLen];

char str2[maxLen];

cout << "Enter the first string: ";

cin.getline(str1, maxLen);

cout << "Enter the second string: ";

cin.getline(str2, maxLen);

int result = indexOf(str1, str2);

if (result != -1) {

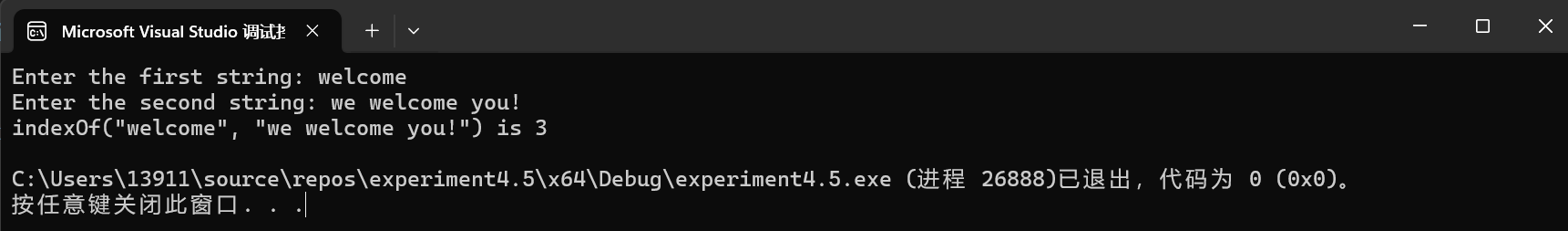
cout << "indexOf(\"" << str1 << "\", \"" << str2 << "\") is " << result << endl;

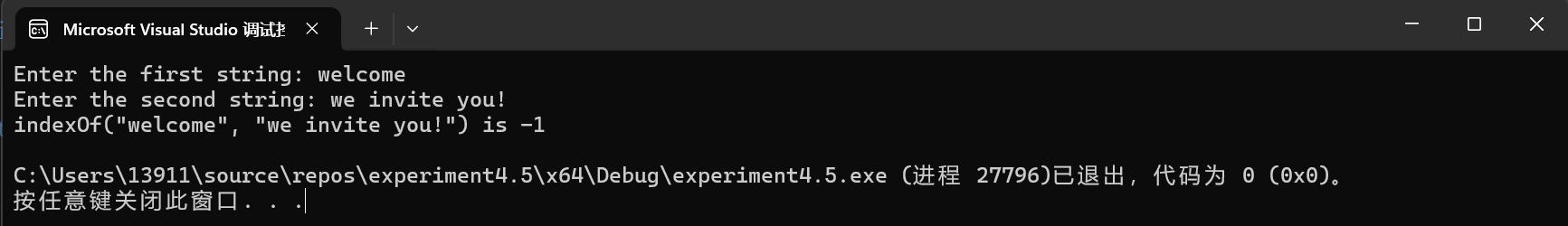
} else {

cout << "indexOf(\"" << str1 << "\", \"" << str2 << "\") is -1" << endl;

}

return 0;

}



6. #include <iostream>

#include <cstring>

using namespace std;

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

for (int i = 0; s[i] != '\0'; ++i) {

char c = tolower(s[i]); // 将字符转换为小写

if (c >= 'a' && c <= 'z') {

counts[c - 'a']++; // 更新对应字母的计数

}

}

}

int main() {

const int SIZE = 26;

int counts[SIZE] = {0}; // 初始化计数数组

char str[100]; // 假设字符串长度不超过99

cout << "Enter a string: ";

cin.getline(str, 100); // 读取字符串

count(str, counts); // 调用count函数

// 打印非零的计数

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

if (counts[i] > 0) {

cout << char('a' + i) << ": " << counts[i] << endl;

}

}

return 0;

}

****

（二）指针

1. #include <iostream>

using namespace std;

void main() {

int i, j, \*pi, \*pj; // 定义两个整数和两个指针

pi = &i; // pi指向i的地址

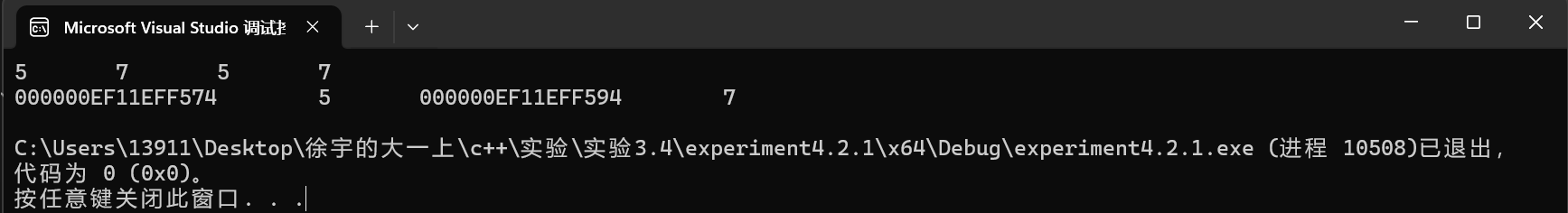
pj = &j; // pj指向j的地址

i = 5; // 给i赋值5

j = 7; // 给j赋值7

cout << i << '\t' << j << '\t' << \*pi << '\t' << \*pj << endl; // 输出i和j的值，以及通过pi和pj访问的值

cout << &i << '\t' << \*&i << '\t' << &j << '\t' << \*&j << endl; // 输出i和j的地址，以及通过&i和&j访问的值

}

2. #include <iostream>

using namespace std;

int main() {

int a[] = {1, 2, 3}; // 定义一个整型数组

int \*p, i; // 定义一个整型指针p和整型变量i

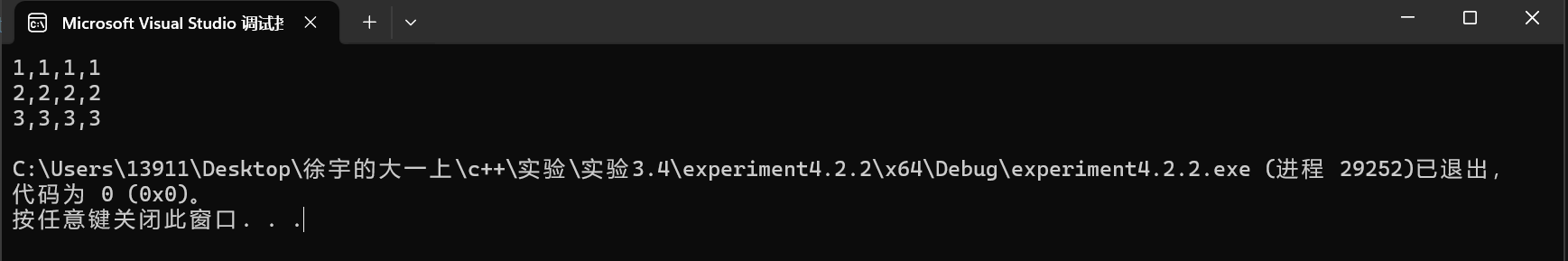
p = a; // 将数组a的首地址赋给指针p

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

printf("%d,%d,%d,%d\n", a[i], p[i], \*(p + i), \*(a + i)); // 打印数组元素的不同访问方式

}

return 0; // 标准main函数返回值

}

3. #include <iostream>

using namespace std;

void f(char \*st, int i) {

st[i] = '\0'; // 在第i个字符位置放置字符串结束符

cout << st; // 打印字符串

if (i > 1) f(st, i - 1); // 递归调用，减少i的值

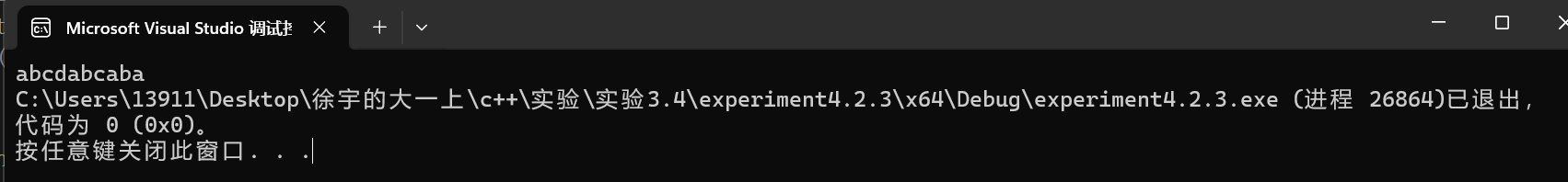
}

int main() {

char st[] = "abcd"; // 定义字符数组并初始化

f(st, 4); // 调用函数f，传入字符串和长度

return 0;

}

4. #include <iostream>

using namespace std;

int\* f() {

int\* list = new int[4] {1, 2, 3, 4}; // 使用new在堆上分配内存

return list;

}

int main() {

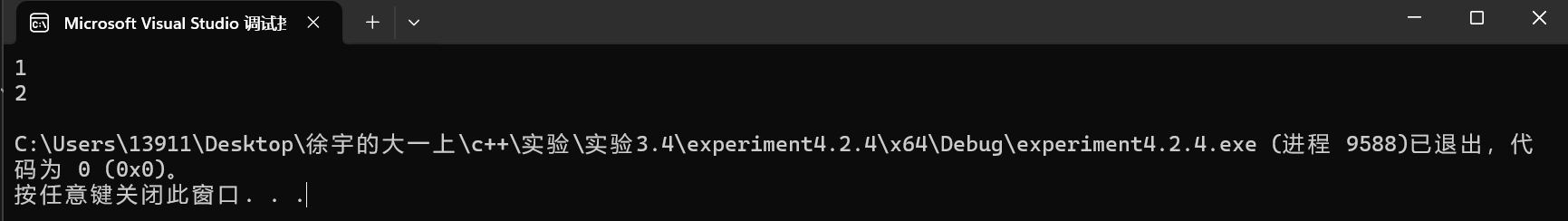
int\* p = f();

cout << p[0] << endl; // 输出1

cout << p[1] << endl; // 输出2

delete[] p; // 释放内存

return 0;

}

（三）程序设计

1. #include <iostream>

#include <cstring> // 为了使用 strstr 函数

using namespace std;

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

// 使用 strstr 函数查找 s1 在 s2 中第一次出现的位置

const char \*pos = strstr(s2, s1);

if (pos != nullptr) {

// 如果找到了，返回 s1 在 s2 中的起始位置（下标）

return pos - s2;

}

// 如果没找到，返回 -1

return -1;

}

int main() {

const char \*s1, \*s2;

cout << "Enter string s1: ";

cin >> s1;

cout << "Enter string s2: ";

cin >> s2;

int result = indexof(s1, s2);

if (result != -1) {

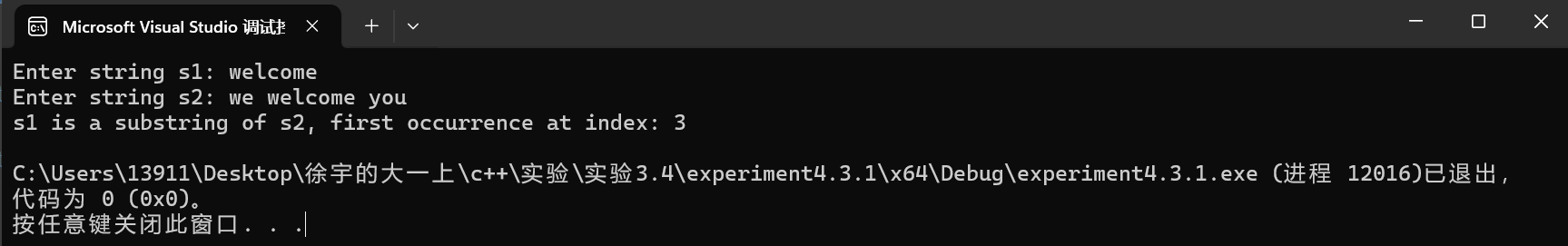
cout << "s1 is a substring of s2, first occurrence at index: " << result << endl;

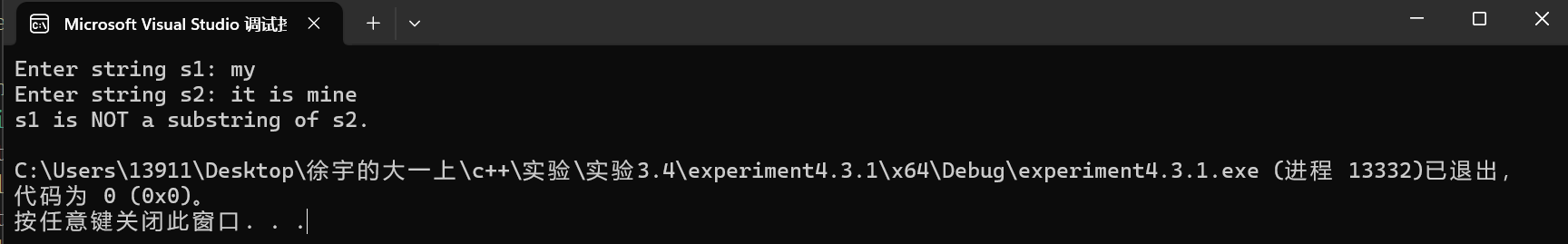
} else {

cout << "s1 is NOT a substring of s2." << endl;

}

return 0;

}



2. #include<iostream>

#include<cmath>

using namespace std;

int parseHex(const char\* const hexString)//定义函数

{

int s\_len = strlen(hexString);//定义数组长度

int num = 0;//定义计数=0

for (int i = 0; i < s\_len; i++)//建立循环

{

if (hexString[i] >= '0' && hexString[i] <= '9')//对于0-9的数值

{

num = num + (((int(hexString[i]))-48) \* pow(16, s\_len - i - 1));//转换成int类型的0-9 根据相应位数乘上16的x-1次方

}

else if (hexString[i] >= 'A' && hexString[i] <= 'F')//对于A-F的数值

{

num = num +(((int(hexString[i])) - 55) \* pow(16, s\_len - i - 1));//转换为int类型的10-15 根据相应位数乘上16的x-1次方

}

else if (hexString[i] >= 'a' && hexString[i] <= 'f')//对于a-f的小写数值

{

num = num +(((int(hexString[i])) - 87) \* pow(16, s\_len - i - 1));//转换为int类型的10-15 根据相应位数乘上16的x-1次方

}

else {//对于其他字符

cout << "数值不是16进制" << endl;//提示字符串不合法

return 0;//返回错误值

}

}

return num;

}

int main()

{

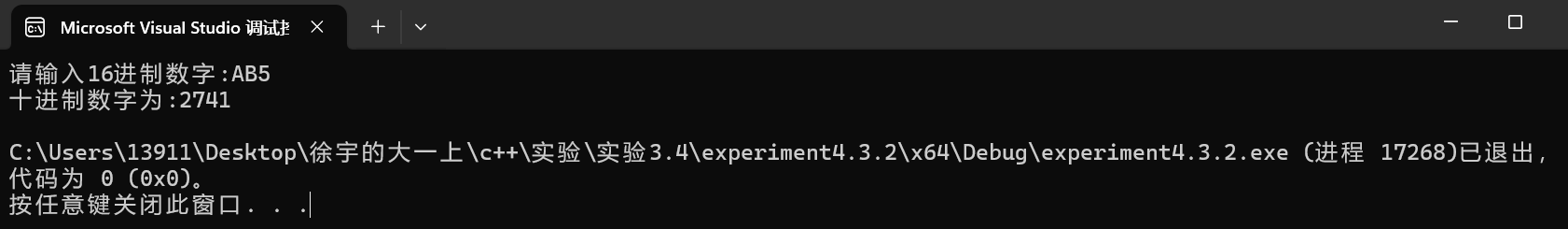
char s[50];//定义数组

cout << "请输入16进制数字:";//输入数组

cin.getline(s, 50);

int dec=parseHex(s);//调用函数

cout << "十进制数字为:" << dec << endl;//输出十进制数字

}

3. #include <iostream>

#include <algorithm> // 为了使用 sort 函数

using namespace std;

// 函数声明

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

int main() {

int size;

cout << "Enter the number of elements: ";

cin >> size; // 用户输入数组大小

// 使用 new 分配动态数组

int\* arr = new int[size];

cout << "Enter " << size << " elements:" << endl;

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

cin >> arr[i]; // 用户输入数组元素

}

// 排序数组

sortArray(arr, size);

// 输出数组元素

cout << "Sorted array elements:" << endl;

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

cout << arr[i] << " ";

}

cout << endl;

// 释放动态数组内存

delete[] arr;

return 0;

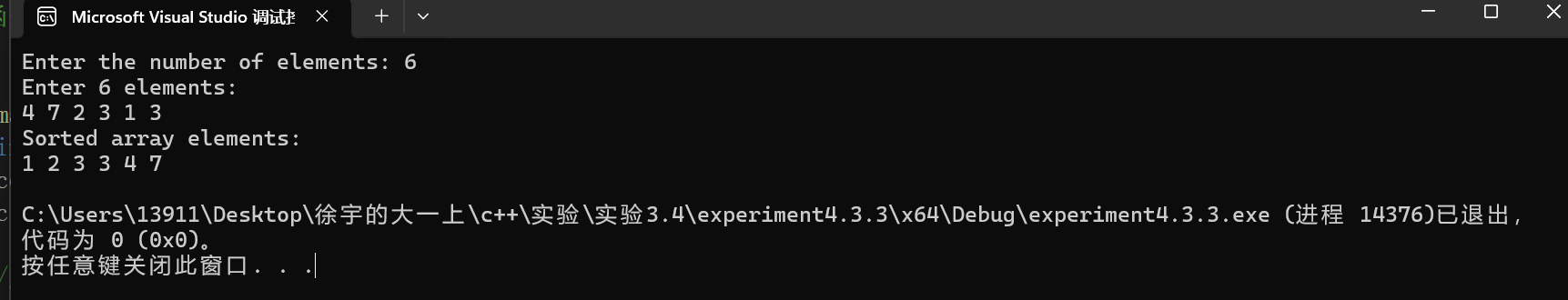
}

// 定义 sortArray 函数，使用标准库的 sort 函数对数组进行排序

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

sort(arr, arr + size);

}

**四、遇到的问题与解决方法**

1.写第三题的时候遇到了以下问题：

（1）数组初始化问题：：需要确保所有存物柜初始状态都是关闭的。在查阅资料后了解到可以使用布尔数组，并在程序开始时将所有元素初始化为`false`。

（2）状态改变逻辑问题：需要正确实现每个学生对存物柜状态的改变。在经过网上查询解决方法了解到使用循环来模拟每个学生的行为，根据学生的编号来确定他们改变状态的起始存物柜和间隔。

（3）数组索引越界问题：如果学生编号超过存物柜数量，可能会导致数组索引越界。在经过网上查询解决方法了解到在改变状态时，应确保索引不会超过数组的最大索引。

（4）输出格式问题：需要按照要求的格式输出开着的存物柜号码。在经过网上查询解决方法了解到在输出时，使用循环遍历数组，并在找到开着的存物柜时输出其索引，同时确保输出格式正确。

2. 写第六题的时候遇到了以下问题：

（1）字符大小写问题：如果字符串中的字母有大小写之分，直接比较会得到不同的结果。在查阅资料后了解到可以在统计之前，将所有字符转换为同一种大小写，通常转换为小写。

（2）非字母字符处理问题：如果字符串中包含非字母字符，它们可能会影响统计结果。在查阅资料后了解到可以在统计时忽略非字母字符，只对字母进行计数。

（3） 数组初始化问题：如果数组没有正确初始化，可能会包含垃圾值。故在函数开始时，将数组的所有元素初始化为0。

**五、体会**

C++数组与指针章节实验心得

在 C++数组与指针章节的实验课中，我仿佛踏入了一片充满挑战与惊喜的编程天地，对这两个重要概念有了深刻的认识与感悟。

数组，就像是一个个紧密排列的存储单元，能够一次性处理大量同类型的数据，为数据的批量操作提供了极大的便利。通过实验，我学会了如何巧妙地定义、初始化数组，以及灵活运用循环结构对数组元素进行访问、修改与处理。然而，在处理多维数组时，复杂的下标运算常常让我陷入困惑，稍不留意就会出现越界错误，导致程序崩溃或产生不可预期的结果。这使我深刻体会到在使用数组时，必须严谨细致地计算下标，确保每一次访问都在合法的范围内。

指针，则宛如一把精准的导航钥匙，能够直接指向内存中的特定位置，赋予了程序更高的效率和更强的操控能力。在实验过程中，我逐渐掌握了指针的声明、赋值以及通过指针访问和修改数据的技巧。指针与数组之间千丝万缕的联系更是让我大开眼界，原来数组名本身就是一个指向数组首元素的常量指针，这种特殊的关系为数据处理带来了更多的灵活性和多样性。但同时，指针的不当使用也极易引发内存泄漏、悬空指针等严重问题，这些错误往往隐藏得很深，难以察觉和调试，给编程工作带来了巨大的挑战。

在不断摸索与实践的过程中，我逐渐领悟到数组与指针犹如编程世界中的双刃剑，它们既能赋予程序强大的功能和高效的性能，但若使用不当，也会引发诸多棘手的问题。通过反复调试程序、查阅资料和请教老师同学，我逐渐积累了经验，学会了如何巧妙地避开陷阱，让数组与指针为我所用。

这一章节的实验课不仅提升了我的编程技能，更培养了我严谨的编程思维和耐心细致的调试习惯。我深知，在 C++编程的漫漫征途中，数组与指针只是冰山一角，但它们所蕴含的编程理念和技巧却具有深远的意义。我将带着这份对知识的敬畏和探索精神，继续砥砺前行，努力在编程的海洋中畅游，探寻更多未知的奥秘与宝藏。