**计算机程序设计基础（C++)**

**实验报告**

专业班级：软件工程2401

学 号：8209240125

姓 名：朱 艺 丹

**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）。

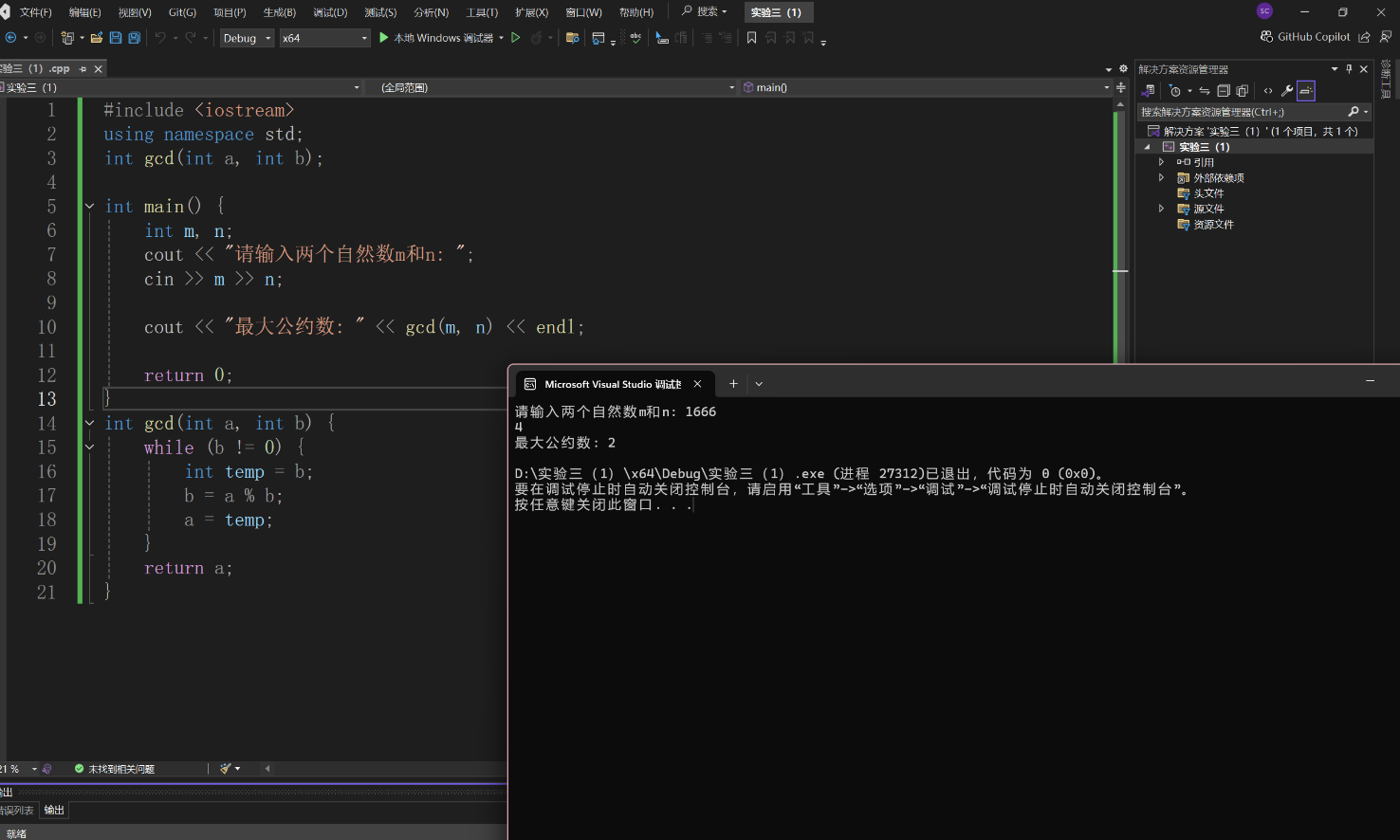
**三、实验思考题**

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.（1）**

#include <iostream>

using namespace std;

int gcd(int a, int b);

int main() {

int m, n;

cout << "请输入两个自然数m和n: ";

cin >> m >> n;

cout << "最大公约数: " << gcd(m, n) << endl;

return 0;

}

int gcd(int a, int b) {

while (b != 0) {

int temp = b;

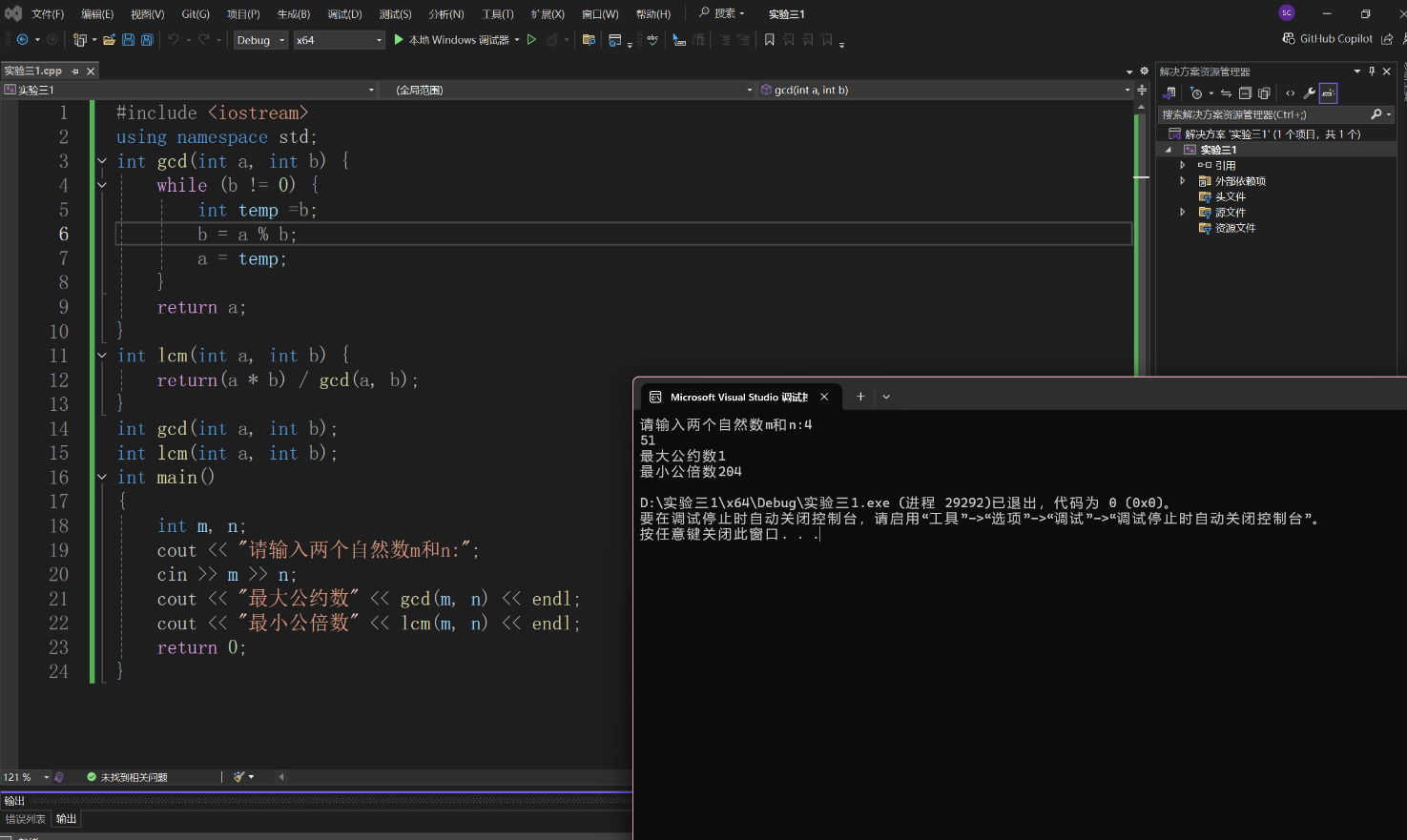
b = a % b;

a = temp;

}

return a;

}

**（2）**

#include <iostream>

using namespace std;

int gcd(int a, int b) {

while (b != 0) {

int temp =b;

b = a % b;

a = temp;

}

return a;

}

int lcm(int a, int b) {

return(a \* b) / gcd(a, b);

}

int gcd(int a, int b);

int lcm(int a, int b);

int main()

{

int m, n;

cout << "请输入两个自然数m和n:";

cin >> m >> n;

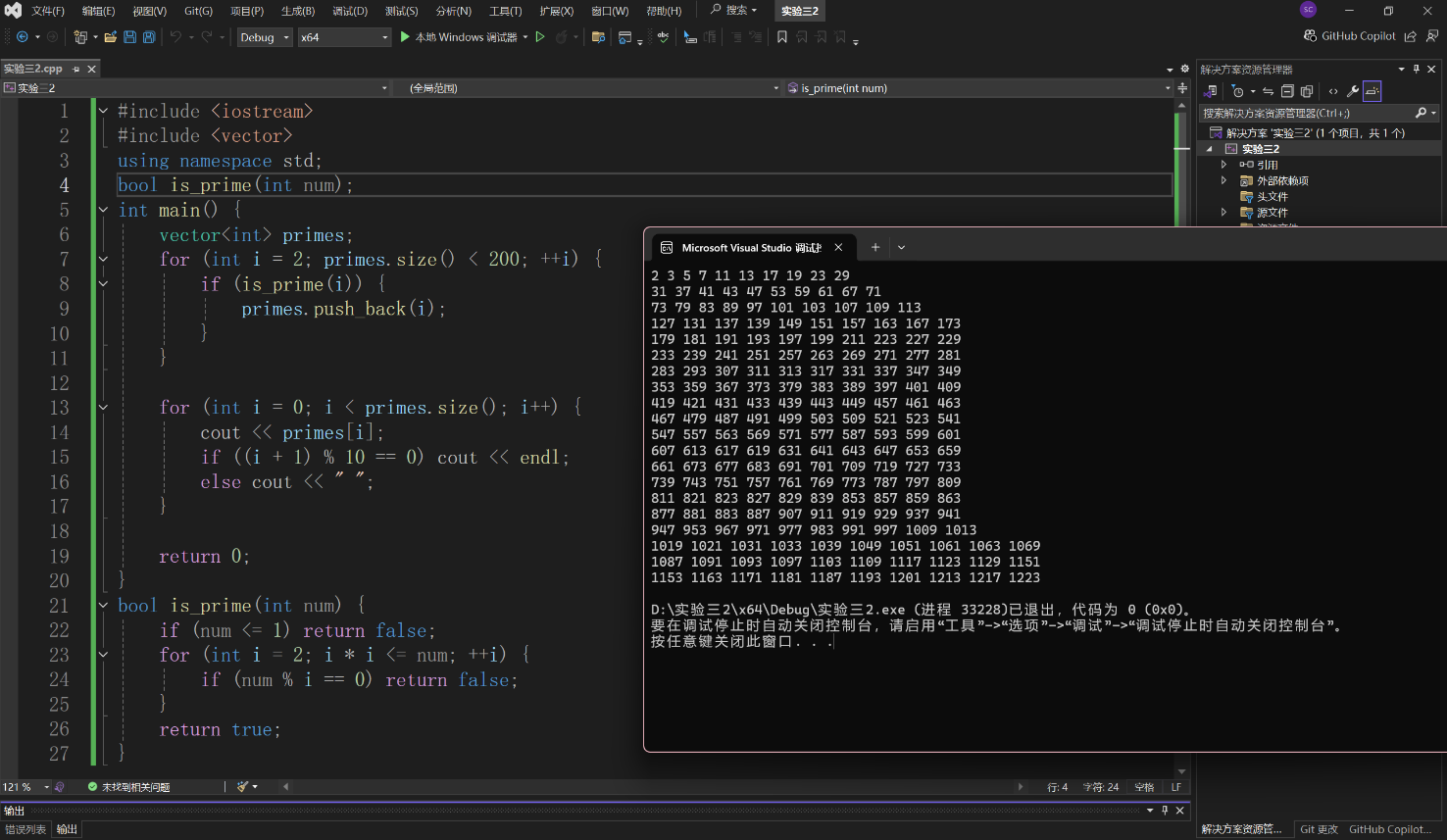
cout << "最大公约数" << gcd(m, n) << endl;

cout << "最小公倍数" << lcm(m, n) << endl;

return 0;

}

**2.**

****

#include <iostream>

#include <vector>

using namespace std;

bool is\_prime(int num);

int main() {

vector<int> primes;

for (int i = 2; primes.size() < 200; ++i) {

if (is\_prime(i)) {

primes.push\_back(i);

}

}

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

cout << primes[i];

if ((i + 1) % 10 == 0) cout << endl;

else cout << " ";

}

return 0;

}

bool is\_prime(int num) {

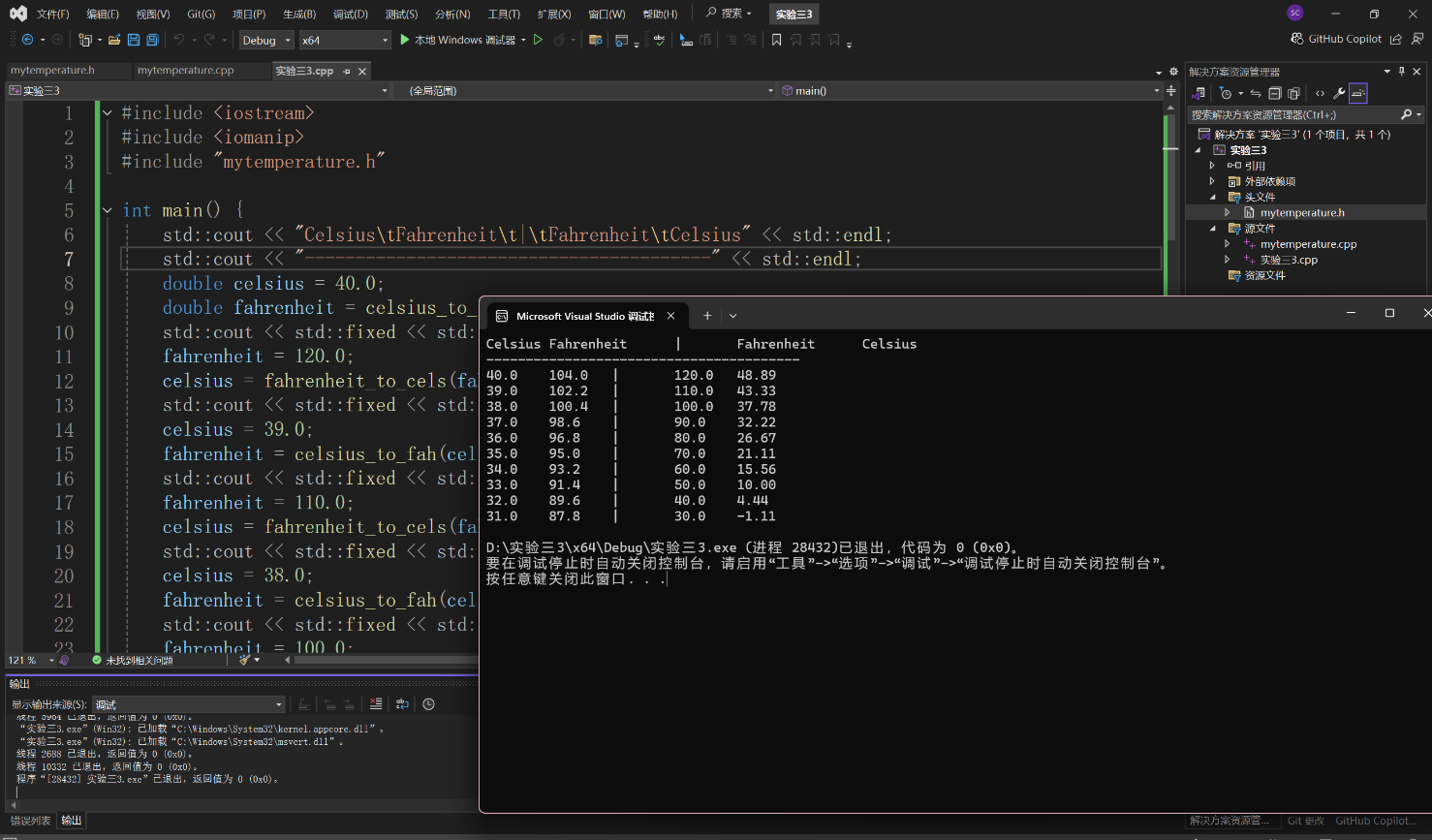
if (num <= 1) return false;

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

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

}

return true;

}

**3.**

**mytemperature.h:**

#ifndef MYTEMPERATURE\_H

#define MYTEMPERATURE\_H

double celsius\_to\_fah(double cel);

double fahrenheit\_to\_cels(double fah);

#endif

**mytemperature.cpp:**

#include "mytemperature.h"

double celsius\_to\_fah(double cel) {

return (cel \* 9.0 / 5.0) + 32.0;

}

double fahrenheit\_to\_cels(double fah) {

return (fah - 32.0) \* 5.0 / 9.0;

}

**主模块**：

#include <iostream>

#include <iomanip>

#include "mytemperature.h"

int main() {

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

std::cout << "----------------------------------------" << std::endl;

double celsius = 40.0;

double fahrenheit = celsius\_to\_fah(celsius);

std::cout << std::fixed << std::setprecision(1) << celsius << "\t" << fahrenheit << "\t|\t";

fahrenheit = 120.0;

celsius = fahrenheit\_to\_cels(fahrenheit);

std::cout << std::fixed << std::setprecision(1) << fahrenheit << "\t" << std::setprecision(2) << celsius <<std::endl;

celsius = 39.0;

fahrenheit = celsius\_to\_fah(celsius);

std::cout << std::fixed << std::setprecision(1) << celsius << "\t" << fahrenheit << "\t|\t";

fahrenheit = 110.0;

celsius = fahrenheit\_to\_cels(fahrenheit);

std::cout << std::fixed << std::setprecision(1) << fahrenheit << "\t" << std::setprecision(2) << celsius << std::endl;

celsius = 38.0;

fahrenheit = celsius\_to\_fah(celsius);

std::cout << std::fixed << std::setprecision(1) << celsius << "\t" << fahrenheit << "\t|\t";

fahrenheit = 100.0;

celsius = fahrenheit\_to\_cels(fahrenheit);

std::cout << std::fixed << std::setprecision(1) << fahrenheit << "\t" << std::setprecision(2) << celsius << std::endl;

celsius = 37.0;

fahrenheit = celsius\_to\_fah(celsius);

std::cout << std::fixed << std::setprecision(1) << celsius << "\t" << fahrenheit << "\t|\t";

fahrenheit = 90.0;

celsius = fahrenheit\_to\_cels(fahrenheit);

std::cout << std::fixed << std::setprecision(1) << fahrenheit << "\t" << std::setprecision(2) << celsius << std::endl;

celsius = 36.0;

fahrenheit = celsius\_to\_fah(celsius);

std::cout << std::fixed << std::setprecision(1) << celsius << "\t" << fahrenheit << "\t|\t";

fahrenheit = 80.0;

celsius = fahrenheit\_to\_cels(fahrenheit);

std::cout << std::fixed << std::setprecision(1) << fahrenheit << "\t" << std::setprecision(2) << celsius << std::endl;

celsius = 35.0;

fahrenheit = celsius\_to\_fah(celsius);

std::cout << std::fixed << std::setprecision(1) << celsius << "\t" << fahrenheit << "\t|\t";

fahrenheit = 70.0;

celsius = fahrenheit\_to\_cels(fahrenheit);

std::cout << std::fixed << std::setprecision(1) << fahrenheit << "\t" << std::setprecision(2) << celsius << std::endl;

celsius = 34.0;

fahrenheit = celsius\_to\_fah(celsius);

std::cout << std::fixed << std::setprecision(1) << celsius << "\t" << fahrenheit << "\t|\t";

fahrenheit = 60.0;

celsius = fahrenheit\_to\_cels(fahrenheit);

std::cout << std::fixed << std::setprecision(1) << fahrenheit << "\t" << std::setprecision(2) << celsius << std::endl;

celsius = 33.0;

fahrenheit = celsius\_to\_fah(celsius);

std::cout << std::fixed << std::setprecision(1) << celsius << "\t" << fahrenheit << "\t|\t";

fahrenheit = 50.0;

celsius = fahrenheit\_to\_cels(fahrenheit);

std::cout << std::fixed << std::setprecision(1) << fahrenheit << "\t" << std::setprecision(2) << celsius << std::endl;

celsius = 32.0;

fahrenheit = celsius\_to\_fah(celsius);

std::cout << std::fixed << std::setprecision(1) << celsius << "\t" << fahrenheit << "\t|\t";

fahrenheit = 40.0;

celsius = fahrenheit\_to\_cels(fahrenheit);

std::cout << std::fixed << std::setprecision(1) << fahrenheit << "\t" << std::setprecision(2) << celsius << std::endl;

celsius = 31.0;

fahrenheit = celsius\_to\_fah(celsius);

std::cout << std::fixed << std::setprecision(1) << celsius << "\t" << fahrenheit << "\t|\t";

fahrenheit = 30.0;

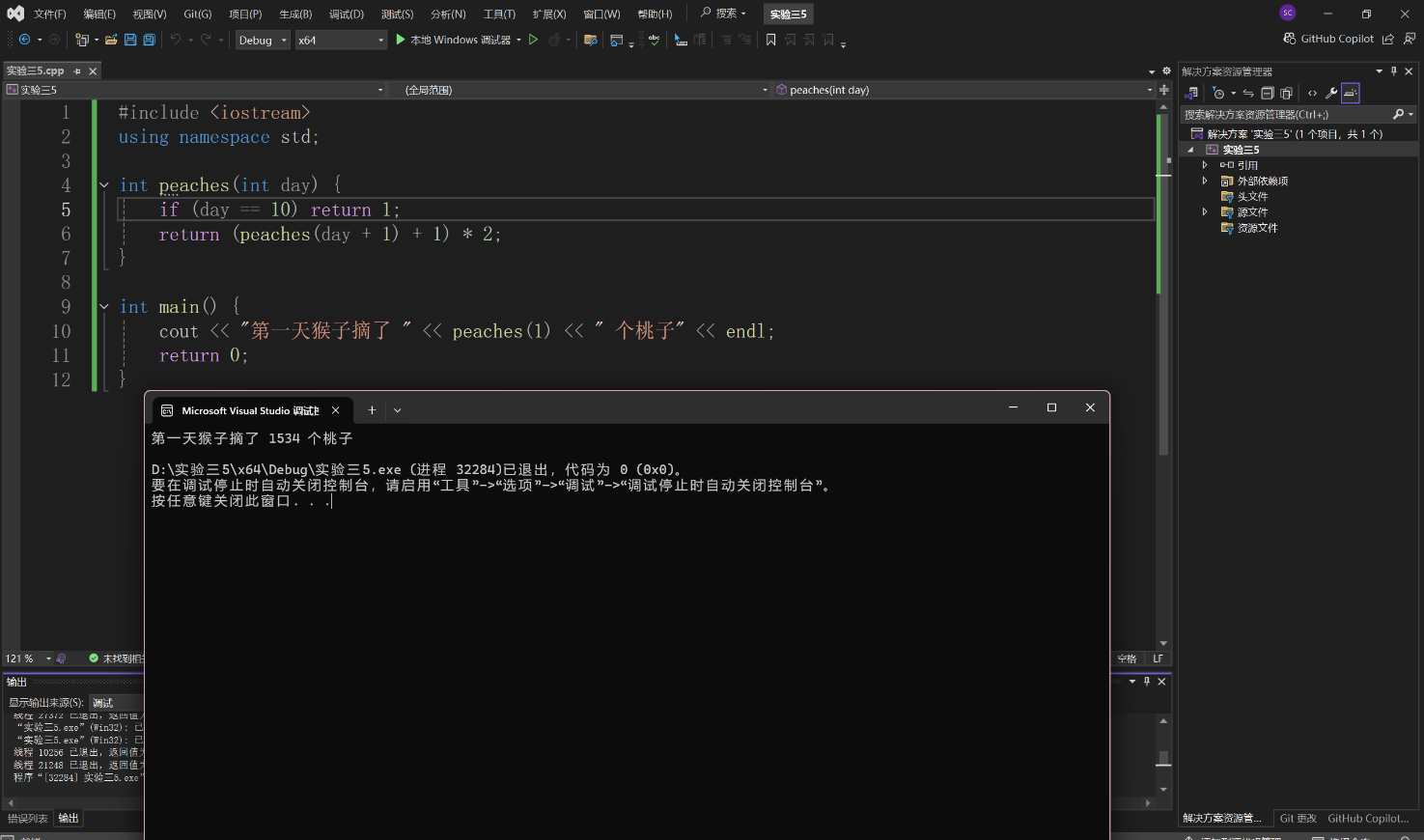
celsius = fahrenheit\_to\_cels(fahrenheit);

std::cout << std::fixed << std::setprecision(1) << fahrenheit << "\t" << std::setprecision(2) << celsius << std::endl;

return 0;

}

**5.**



#include <iostream>

using namespace std;

int peaches(int day) {

if (day == 10) return 1;

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

}

int main() {

cout << "第一天猴子摘了 " << peaches(1) << " 个桃子" << endl;

return 0;

}

**五、遇到的问题与解决方法**

**1．动态数组大小不确定导致编译错误。解决方法：使用`std::vector`或动态分配内存（`new`）来处理不确定大小的数组。**

**2．递归函数中使用局部数组导致内存问题。解决方法：在递归函数中使用`new`分配动态数组，并在适当时机使用`delete[]`释放内存。**

**3．字符串处理中，`cin.getline`读取字符串时遇到换行符问题。解决方法：确保在调用`cin.getline`之前清除输入缓冲区，或者使用`std::string`来处理输入。**

**4．递归函数中字符串反转逻辑错误。解决方法：修正递归逻辑，确保正确地从字符串末尾开始反转。**

**六、体会**

**1．在C++中，动态内存管理是一把双刃剑，它提供了灵活性，但也需要开发者谨慎管理，避免内存泄漏。**

**2．递归是解决问题的强大工具，但需要仔细设计递归逻辑，避免无限递归和栈溢出。**

**3．字符串处理是C++编程中的常见任务，正确理解字符数组和字符串类的区别对于编写高效、安全的代码至关重要。**

**4．程序设计中，函数的模块化和重用性非常重要，这有助于提高代码的可读性和可维护性。通过将功能分解为小的、可管理的函数，可以更容易地理解和调试代码。**

**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;

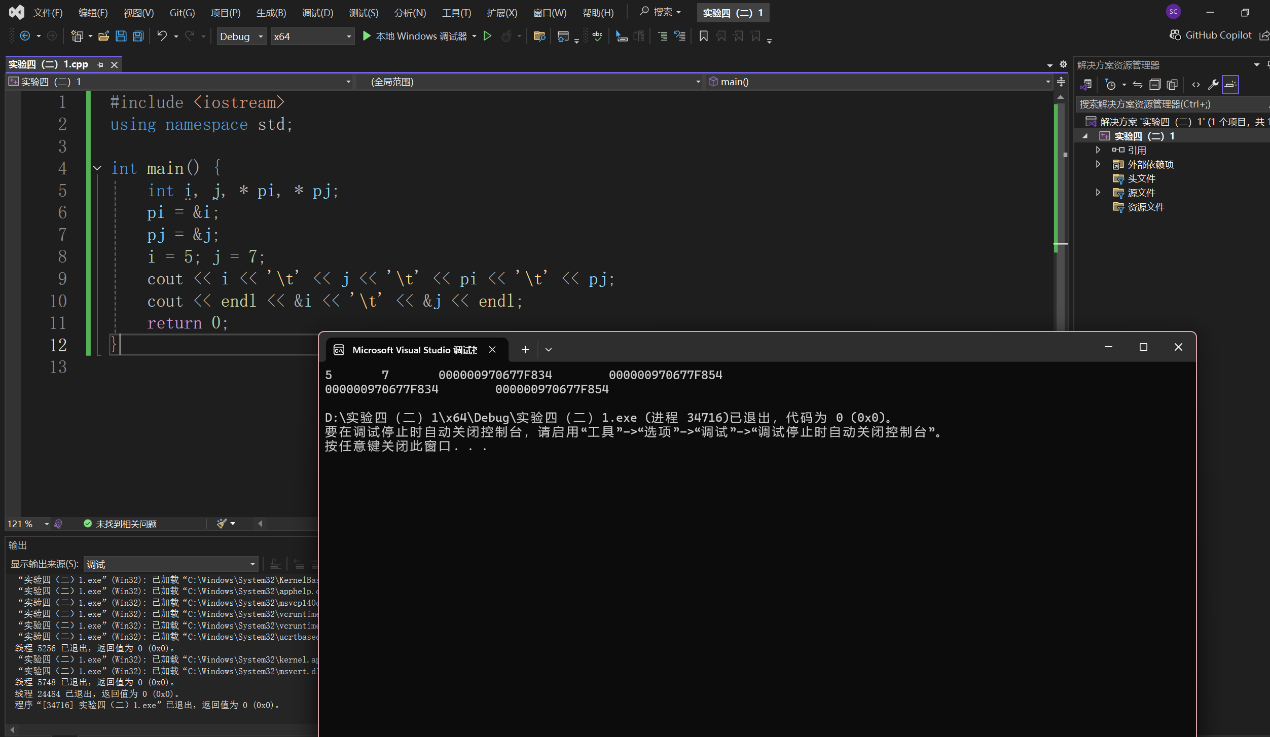
pj=&j;

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）。

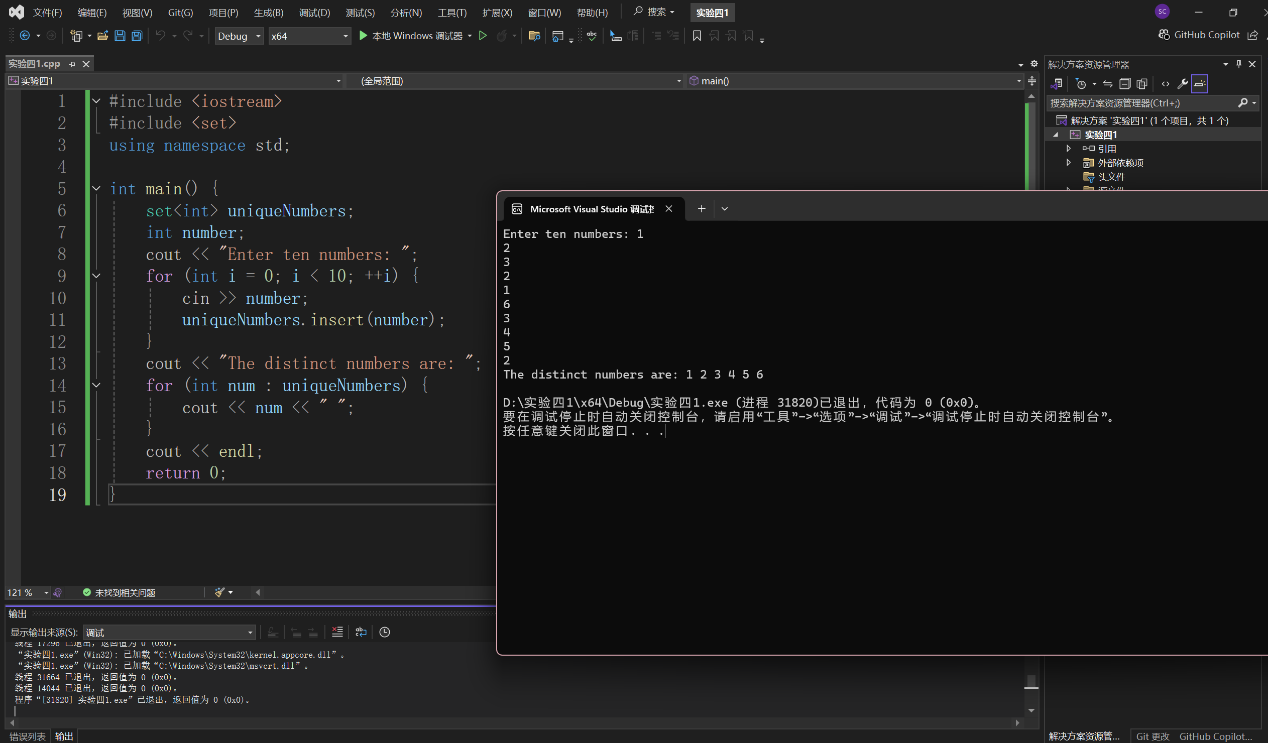
【完成实验报告】

**实验报告只要求写程序设计部分**

**三、算法分析，程序结果**

**（一）**

**1.**

****#include <iostream>

#include <set>

using namespace std;

int main() {

set<int> uniqueNumbers;

int number;

cout << "Enter ten numbers: ";

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

cin >> number;

uniqueNumbers.insert(number);

}

cout << "The distinct numbers are: ";

for (int num : uniqueNumbers) {

cout << num << " ";

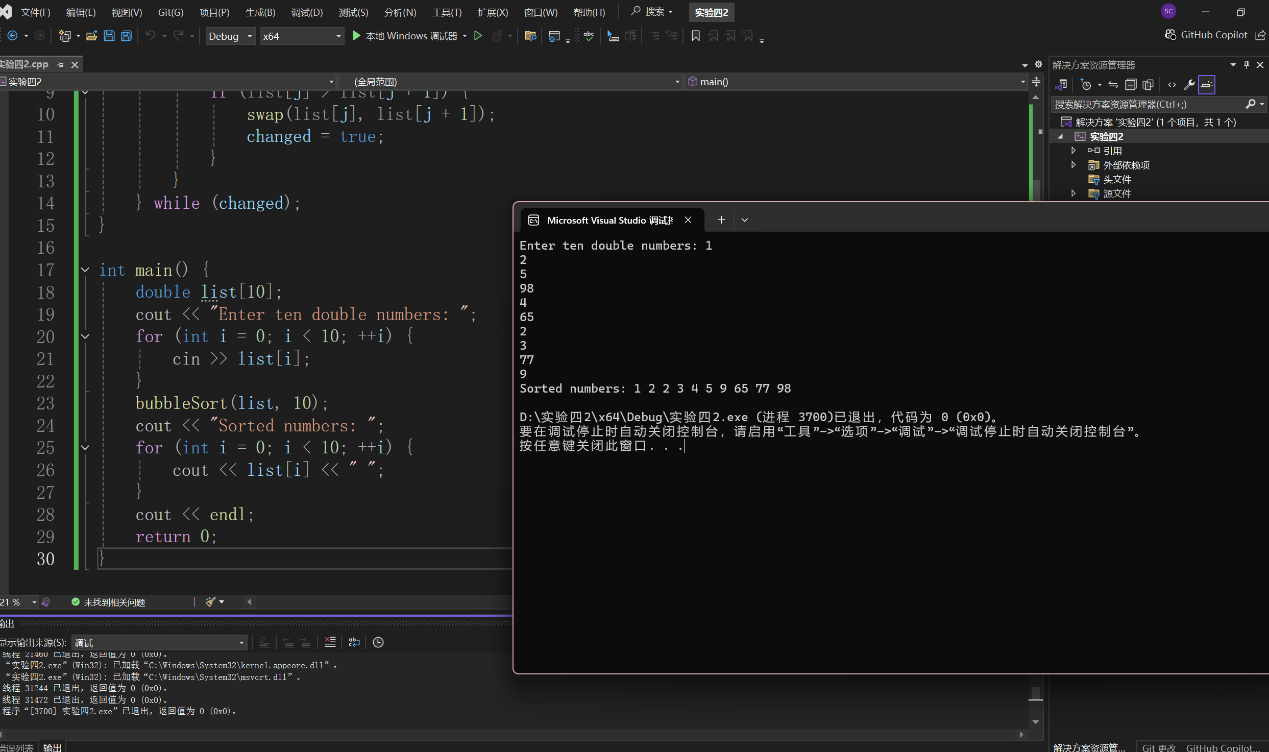
}

cout << endl;

return 0;

}

**2.**

#include <iostream>

using namespace std;

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

bool changed;

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);

}

int main() {

double list[10];

cout << "Enter ten double numbers: ";

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

cin >> list[i];

}

bubbleSort(list, 10);

cout << "Sorted numbers: ";

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

cout << list[i] << " ";

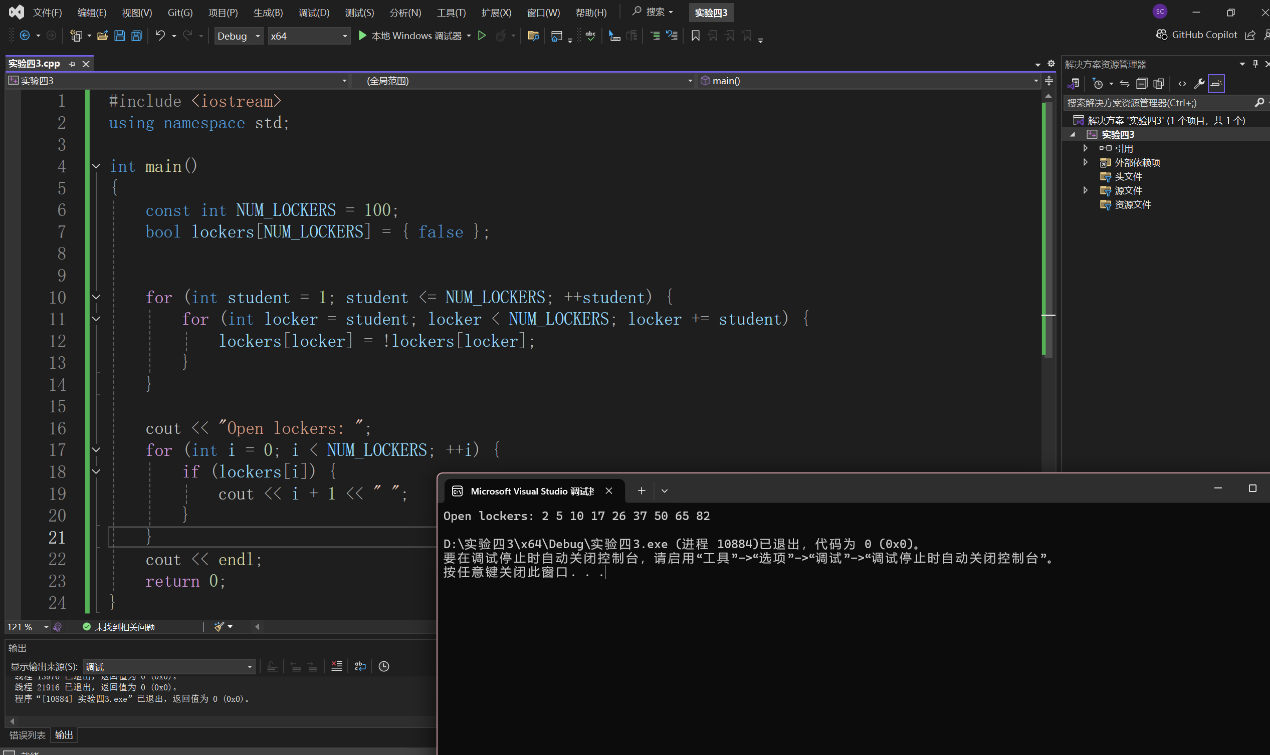
}

cout << endl;

return 0;

}

**3.**



#include <iostream>

using namespace std;

int main()

{

const int NUM\_LOCKERS = 100;

bool lockers[NUM\_LOCKERS] = { false };

for (int student = 1; student <= NUM\_LOCKERS; ++student) {

for (int locker = student; locker < NUM\_LOCKERS; locker += student) {

lockers[locker] = !lockers[locker];

}

}

cout << "Open lockers: ";

for (int i = 0; i < NUM\_LOCKERS; ++i) {

if (lockers[i]) {

cout << i + 1 << " ";

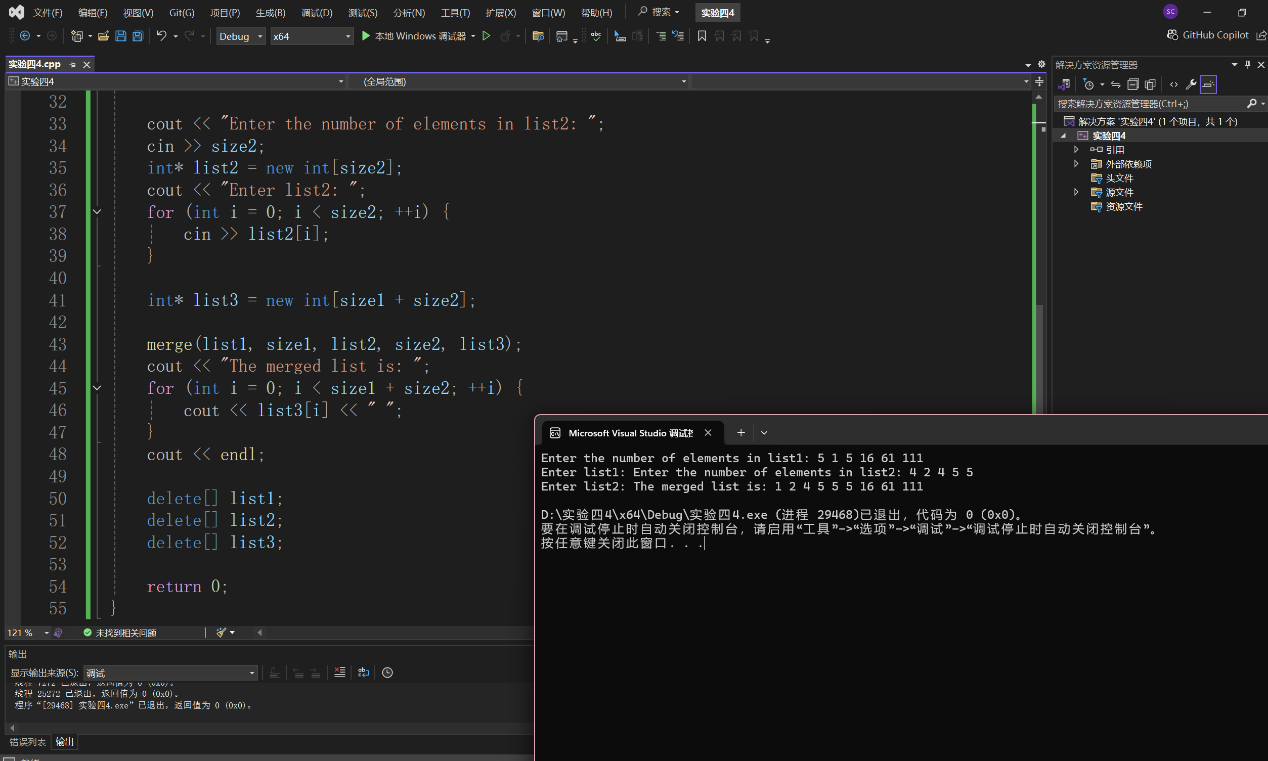
}

}

cout << endl;

return 0;

}

**4.**

#include <iostream>

using namespace std;

void merge(int\* list1, int size1, 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;

int\* list1 = new int[size1];

cout << "Enter list1: ";

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

cin >> list1[i];

}

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

cin >> size2;

int\* list2 = new int[size2];

cout << "Enter list2: ";

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

cin >> list2[i];

}

int\* list3 = new int[size1 + size2];

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

cout << "The merged list is: ";

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

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

}

cout << endl;

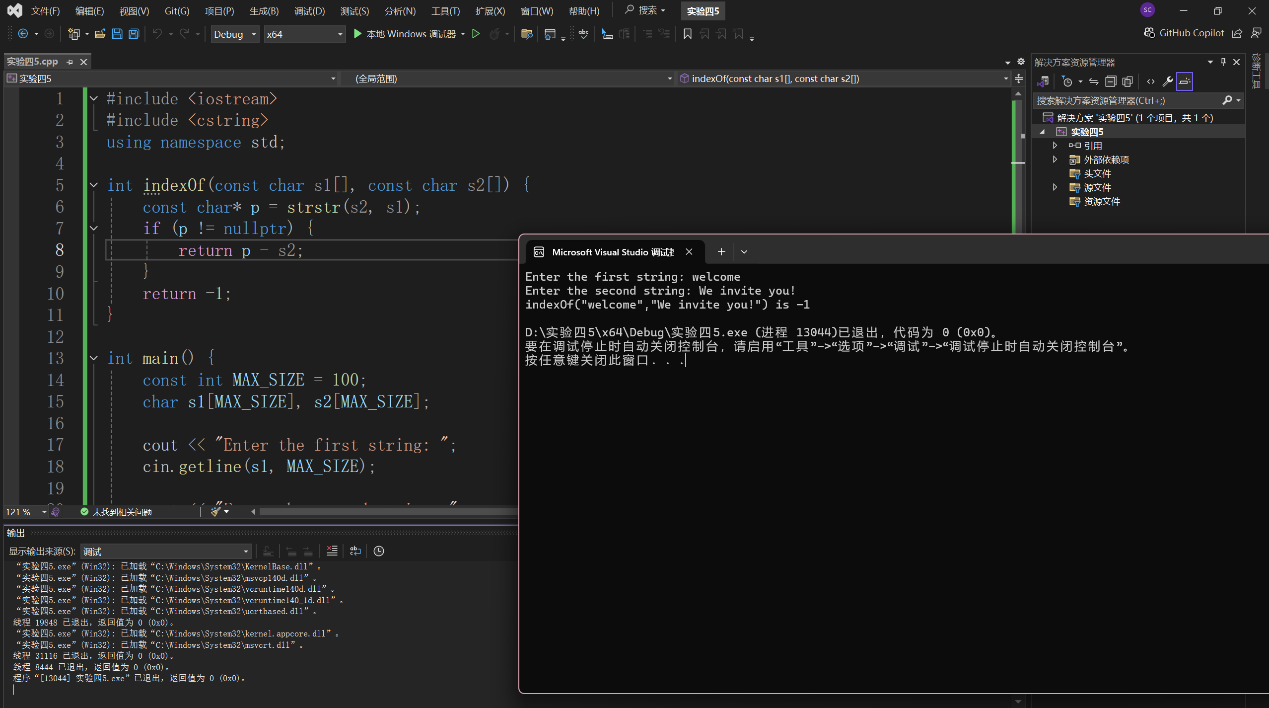
delete[] list1;

delete[] list2;

delete[] list3;

return 0;

}

**5.**

#include <iostream>

#include <cstring>

using namespace std;

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

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

if (p != nullptr) {

return p - s2;

}

return -1;

}

int main() {

const int MAX\_SIZE = 100;

char s1[MAX\_SIZE], s2[MAX\_SIZE];

cout << "Enter the first string: ";

cin.getline(s1, MAX\_SIZE);

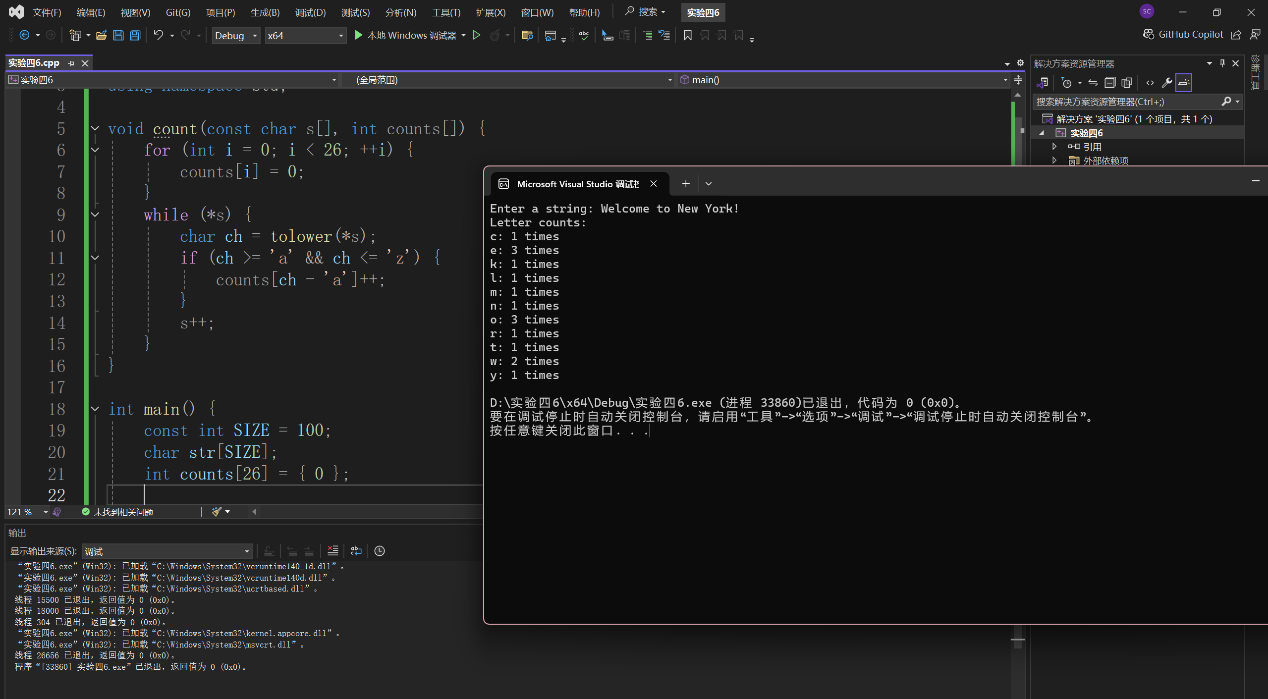
cout << "Enter the second string: ";

cin.getline(s2, MAX\_SIZE);

cout << "indexOf(\"" << s1 << "\",\"" << s2 << "\") is " << indexOf(s1, s2) << endl;

return 0;

}

**6.**

#include <iostream>

#include <cctype>

using namespace std;

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

for (int i = 0; i < 26; ++i) {

counts[i] = 0;

}

while (\*s) {

char ch = tolower(\*s);

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

counts[ch - 'a']++;

}

s++;

}

}

int main() {

const int SIZE = 100;

char str[SIZE];

int counts[26] = { 0 };

cout << "Enter a string: ";

cin.getline(str, SIZE);

count(str, counts);

cout << "Letter counts:" << endl;

for (int i = 0; i < 26; ++i) {

if (counts[i] > 0) {

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

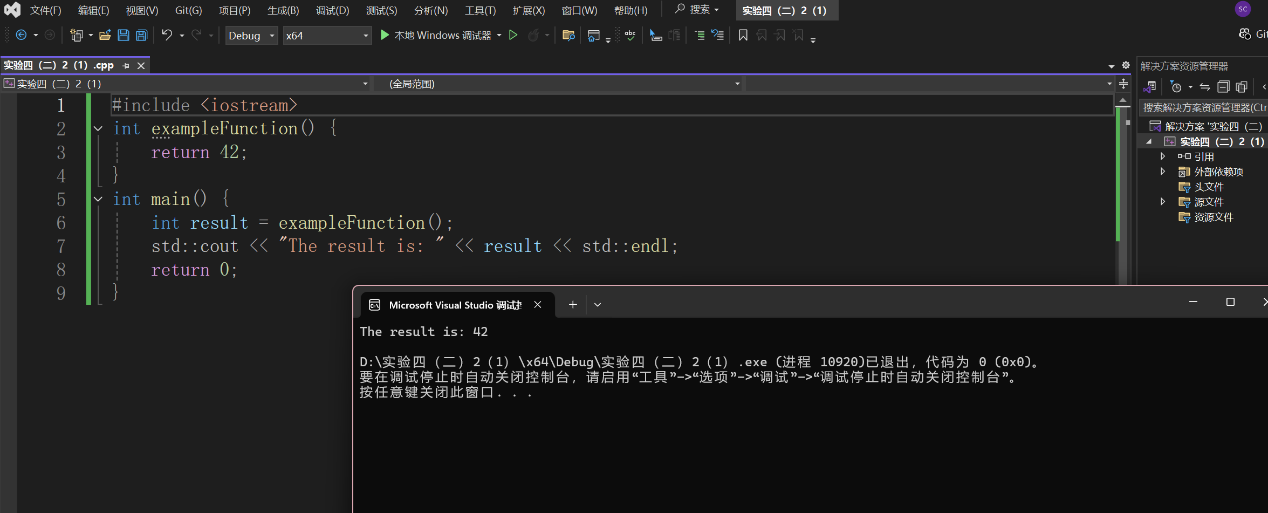
}

}

return 0;

}

**实验四（二）2. 程序设计（1）**

****#include <iostream>

int exampleFunction() {

return 42;

}

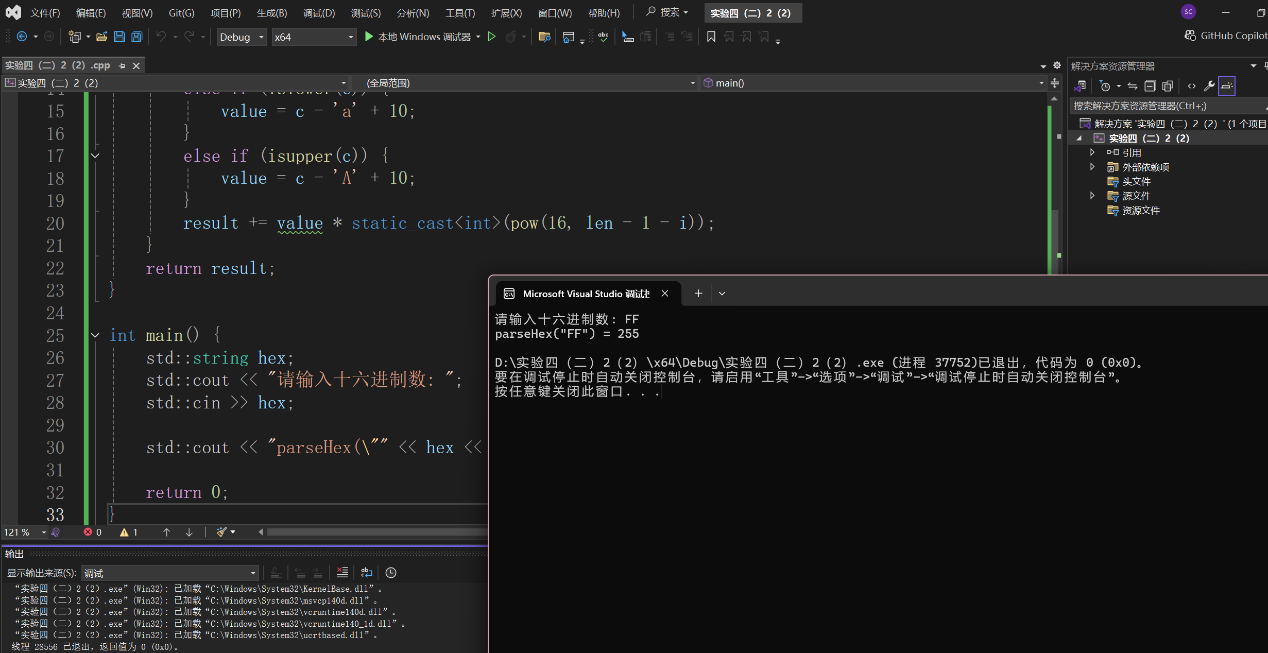
int main() {

int result = exampleFunction();

std::cout << "The result is: " << result << std::endl;

return 0;

}

**（2）**

#include <iostream>

#include <cmath>

#include <cctype>

int parseHex(const char\* const hexString) {

int result = 0;

int len = strlen(hexString);

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

char c = hexString[i];

int value;

if (isdigit(c)) {

value = c - '0';

}

else if (islower(c)) {

value = c - 'a' + 10;

}

else if (isupper(c)) {

value = c - 'A' + 10;

}

result += value \* static\_cast<int>(pow(16, len - 1 - i));

}

return result;

}

int main() {

std::string hex;

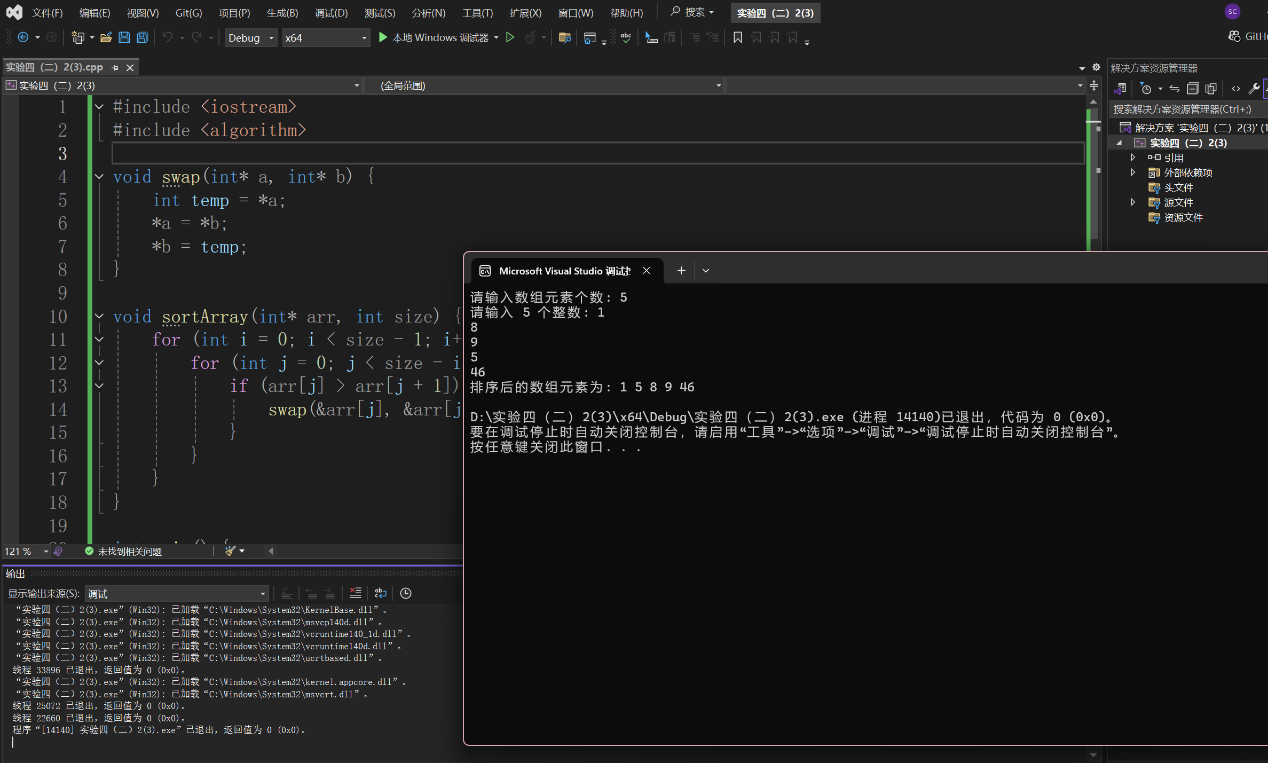
std::cout << "请输入十六进制数: ";

std::cin >> hex;

std::cout << "parseHex(\"" << hex << "\") = " << parseHex(hex.c\_str()) << std::endl;

return 0;

}

**（3）**

#include <iostream>

#include <algorithm>

void swap(int\* a, int\* b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

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]) {

swap(&arr[j], &arr[j + 1]);

}

}

}

}

int main() {

int size;

std::cout << "请输入数组元素个数: ";

std::cin >> size;

int\* arr = new int[size];

std::cout << "请输入 " << size << " 个整数: ";

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

std::cin >> arr[i];

}

sortArray(arr, size);

std::cout << "排序后的数组元素为: ";

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

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

}

std::cout << std::endl;

delete[] arr;

return 0;

}

**四、遇到的问题与解决方法**

**1.未定义标识符错误，这通常是因为缺少必要的头文件，解决方法是确保包含了正确的头文件。**

**2.尝试给常量赋值的错误，解决方法是确保不对`const`修饰的变量或字符串字面量进行修改。**

**3.对于动态内存管理，使用了`new`和`delete`来分配和释放内存，同时确保了数组的正确排序和输出。**

**五、体会**

**通过这些程序设计任务，我深刻体会到了C++中指针和数组的重要性，以及它们之间的紧密联系。指针不仅可以用来访问数组元素，还可以用于动态内存管理。此外，我也认识到了正确使用头文件和避免对常量进行修改的重要性。在编写函数时，考虑边界情况和错误处理也非常重要，这有助于编写更健壮的代码。最后，通过实践，我更加熟悉了C++的内存管理和异常处理机制。**