**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软件工程2404班

学 号： 8209240413

姓 名： 廖甜甜

**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？

函数返回值与函数类型一致主要基于以下几个重要原因：

1.类型安全：保证了程序的类型安全性。如果返回值的类型可以随意与函数声明的类型不一致，可能会导致类型不匹配的错误难以察觉，增加程序出错的风险。

2.编译器优化：编译器可以根据函数声明的类型进行相应的优化。例如，对于数值类型，如果知道返回值的确切类型，编译器可以生成更高效的机器码。

3.代码可读性和可维护性：使代码更具可读性和可理解性。其他开发者在阅读和使用该函数时，能够根据函数类型准确预期返回值的类型，减少误解和错误使用的可能性。

4.函数接口的明确性：明确了函数与外部交互的接口规范。有助于构建清晰、一致和可靠的代码结构，方便团队协作和代码复用。

总之，函数返回值与函数类型一致有助于维持程序的正确性、效率、可读性和可维护性，是良好编程实践的重要原则之一。

1. 本实验中主函数调用函数时采用的是何种传递方式？

值传递方式。

**四、算法分析，程序结果**

**2.1.1：**#include<iostream>

using namespace std;

int gys(int a, int b)

{

if (a > b)

{

while (b != 0)

{

int temp = b;

b = a % b;

a = temp;

}

return a;

}

if (b > a)

{

while (a != 0)

{

int temp = a;

a = b % a;

b = temp;

}

return b;

}

}

int main()

{

int m;

int n;

cout << "请输入两个数m,n:" << endl;

cin >> m;

cin >> n;

if (m == 0 || n == 0)

{

cout << "m和n无最大公约数，请输入正确的数值！" << endl;

}

if (m != 0 && n != 0)

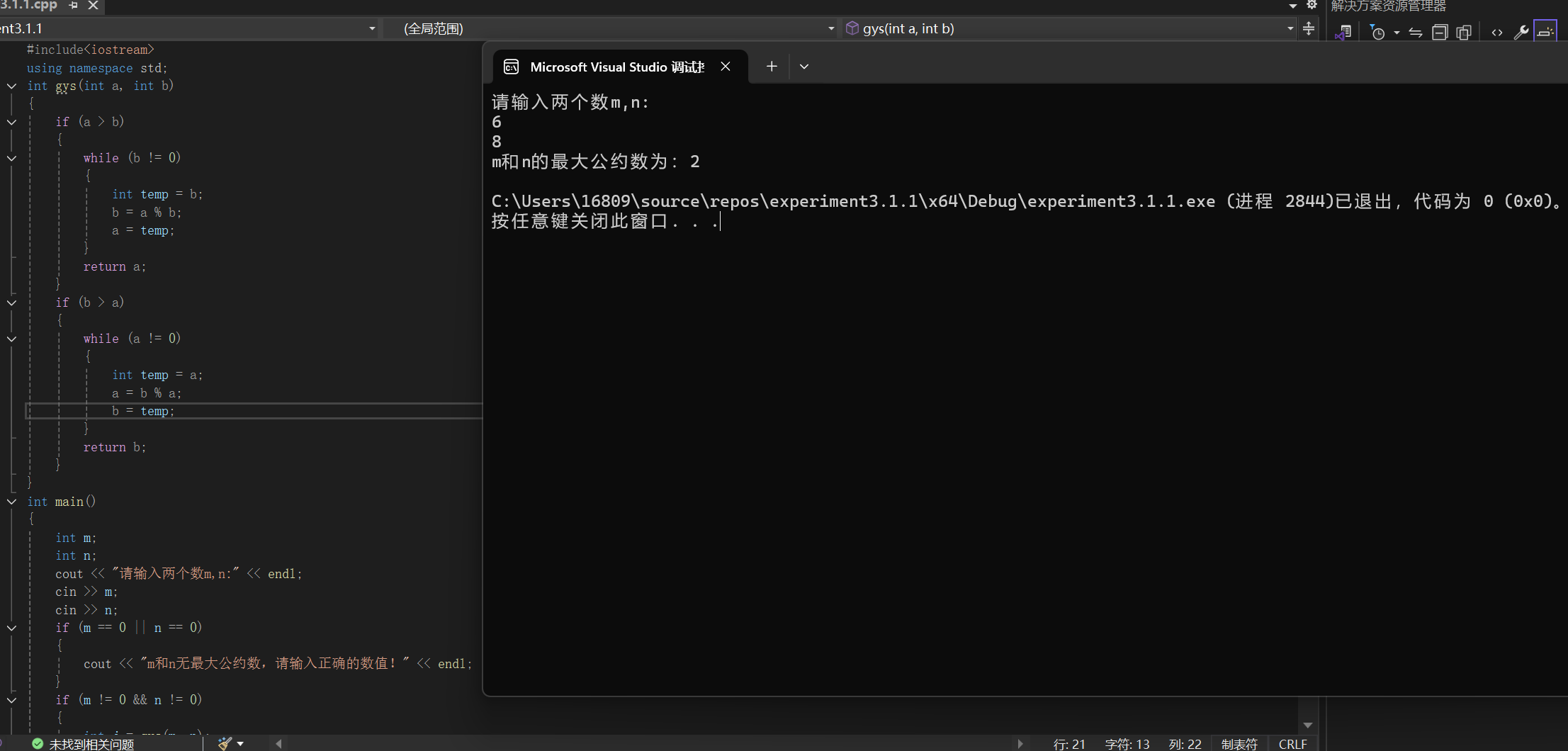
{

int j = gys(m, n);

cout << "m和n的最大公约数为：" << j << endl;

}

}



2.1.2:#include<iostream>

using namespace std;

int gys(int a, int b)

{

if (a > b)

{

while (b != 0)

{

int temp = b;

b = a % b;

a = temp;

}

return a;

}

if (b > a)

{

while (a != 0)

{

int temp = a;

a = b % a;

b = temp;

}

return b;

}

}

int gbs(int a, int b)

{

int k = (a / gys(a, b)) \* (b / gys(a, b)) \* gys(a, b);

return k;

}

int main()

{

int m;

int n;

cout << "请输入两个数m,n:" << endl;

cin >> m;

cin >> n;

if (m == 0 || n == 0)

{

cout << "m和n无最大公约数和最小公倍数，请输入正确的数值！" << endl;

}

if (m != 0 && n != 0)

{

int j = gys(m, n);

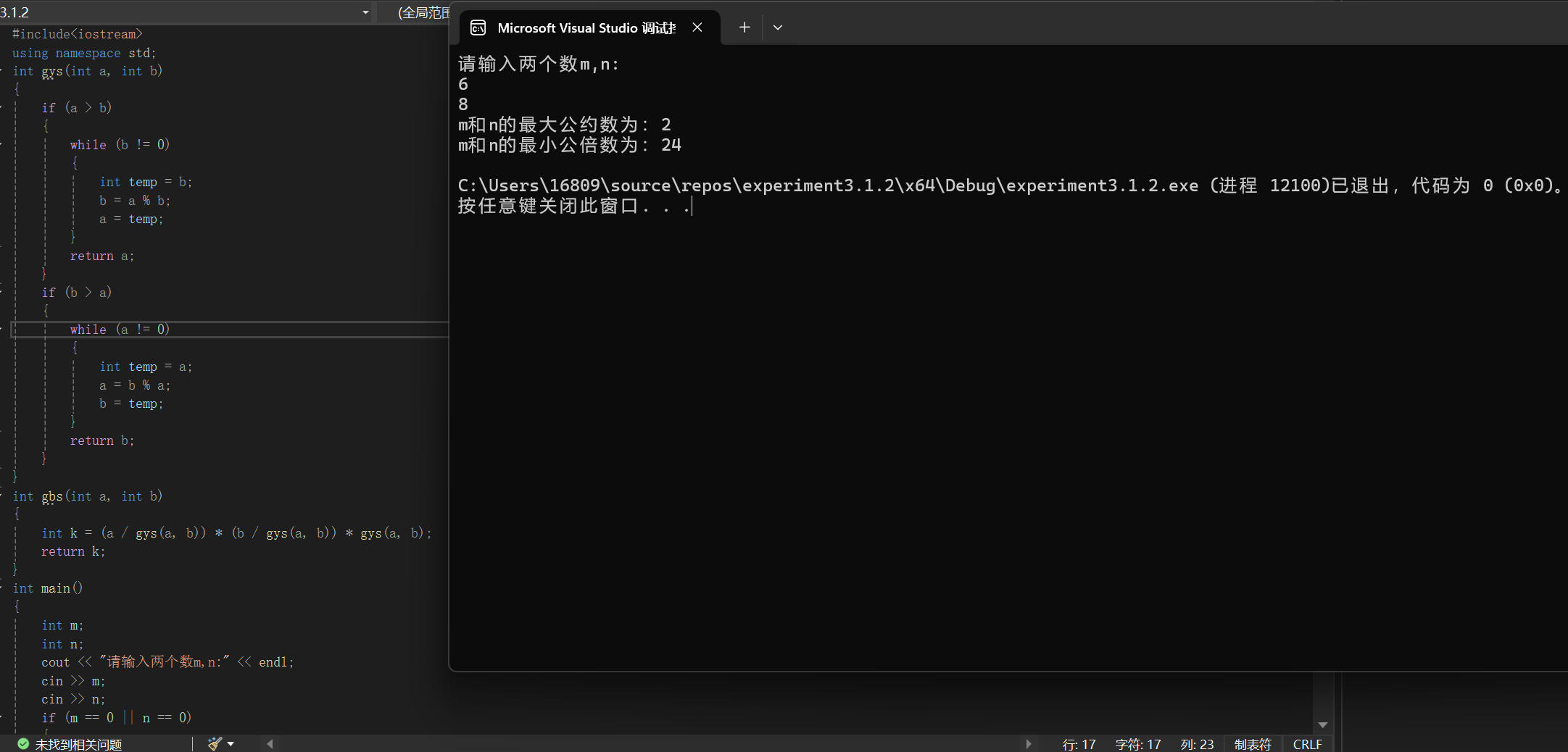
int k = gbs(m, n);

cout << "m和n的最大公约数为：" << j << endl;

cout << "m和n的最小公倍数为：" << k << endl;

}

}



2.2:#include<iostream>

using namespace std;

bool is\_prime(int num)

{

if (num == 1 || num == 2)

{

return true;

}

else

{

int i = 2;

for (i = 2; i < num; i++)

{

if (num % i == 0)

{

return false;

}

else

{

return true;

}

}

}

}

int main()

{

int j = 0;

int a = 1;

for (a = 1;j<200; a++)

{

if (is\_prime(a))

{

cout << a <<" ";

j++;

if (j % 10 == 0)

{

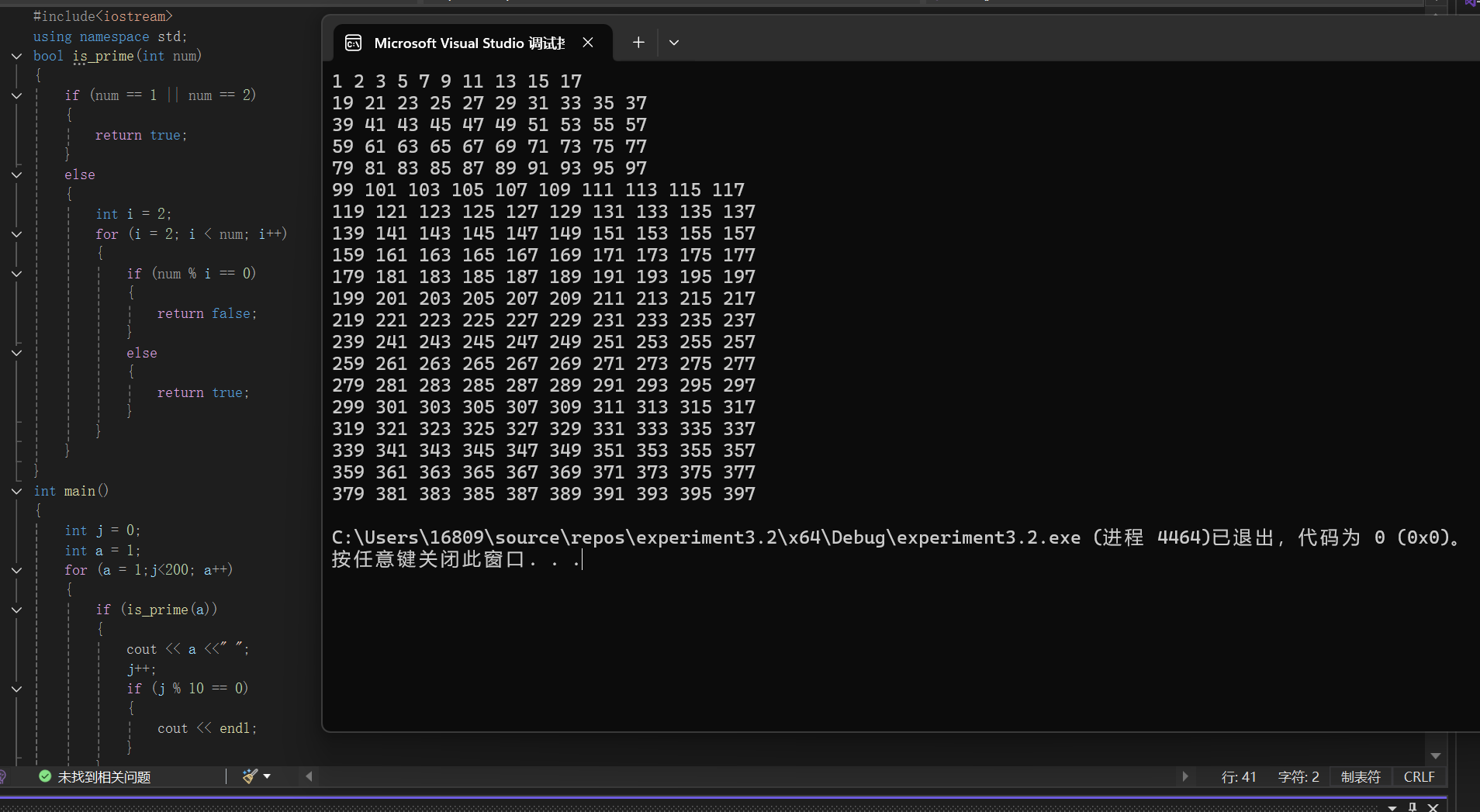
cout << endl;

}

}

}

}



2.3：头文件mytemperature.h：#pragma once

#include<iostream>

using namespace std;

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

CPP文件mytemperature.cpp：#include<iostream>

#include"mytemperature.h"

using namespace std;

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

{

double fah = cel \* 9 / 5 + 32;

return fah;

}

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

{

double cel = (fah - 32) \* 5 / 9;

return cel;

}

主模块：#include<iostream>

#include"mytemperature.h"

using namespace std;

int main()

{

double cel[10] = { 40.0,39.0,38.0,37.0,36.0,35.0,34.0,33.0,32.0,31.0 };

double fah[10] = { 120.0,110.0,100.0,90.0,80.0,70.0,60.0,50.0,40.0,30.0 };

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

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

{

cout << cel[i] << " ";

double f=celsius\_to\_fah(cel[i]);

cout << f << " "<<"|"<<" ";

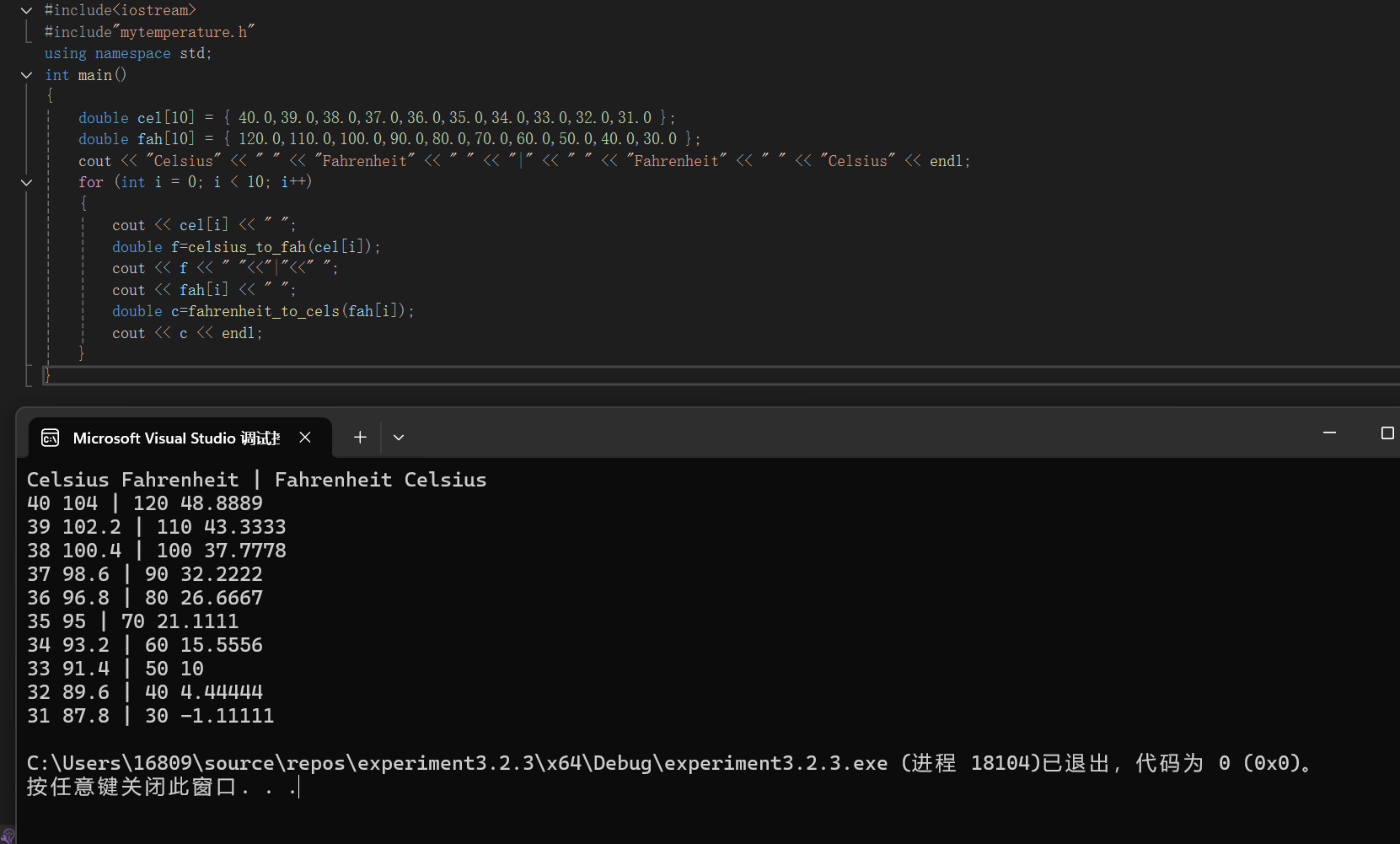
cout << fah[i] << " ";

double c=fahrenheit\_to\_cels(fah[i]);

cout << c << endl;

}

}



2.4;mytriangle.h的头文件：#pragma once

#include<iostream>

#include<cmath>

using namespace std;

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

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

CPP文件mytriangle.cpp：#include<iostream>

#include<cmath>

using namespace std;

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

{

if (side1 + side2 > side3 && side1 + side3 > side2 && side2 + side3 > side1 && side1 - side2 < side3 && side2 - side1 < side3 && side1 - side3 < side2 && side3 - side1 < side2 && side2 - side3 < side1 && side3 - side2 < side1)

{

return true;

}

else

{

return false;

}

}

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

{

double s = (side1 + side2 + side3) / 2;

double mj = sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));

return mj;

}

主模块：#include<iostream>

#include"mytriangle.h"

using namespace std;

int main()

{

double side1;

double side2;

double side3;

cout << "请输入三角形的三条边：" << endl;

cin >> side1;

cin >> side2;

cin >> side3;

if (is\_valid(side1, side2, side3))

{

double mj = \_area(side1, side2, side3);

cout << "此三边能构成三角形，三角形的面积为：" << mj << endl;

}

else

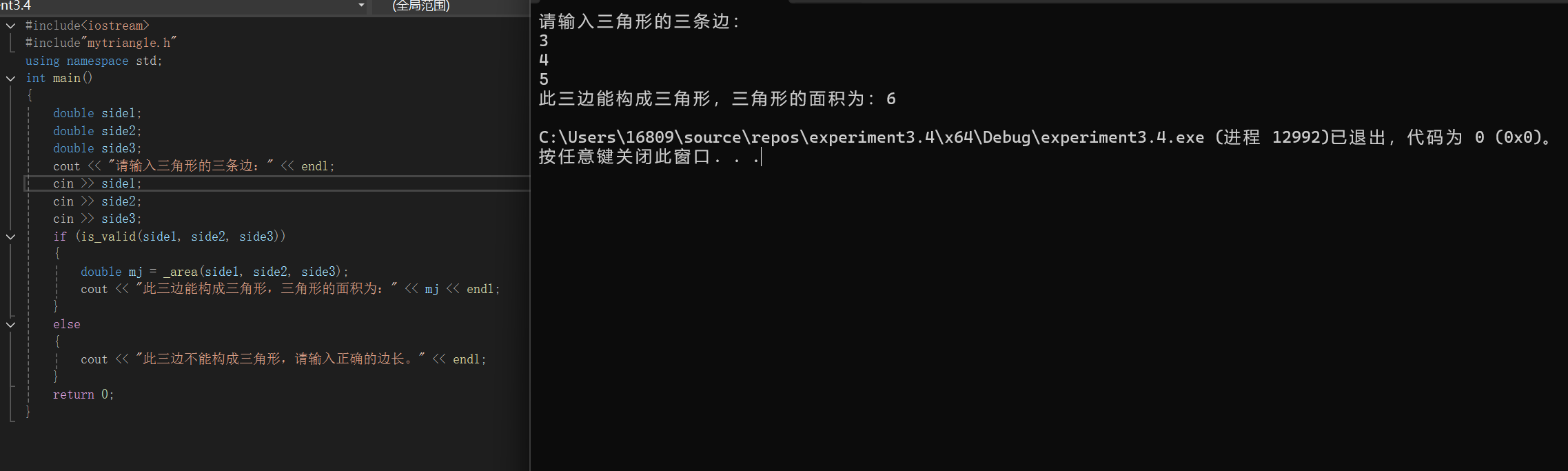
{

cout << "此三边不能构成三角形，请输入正确的边长。" << endl;

}

return 0;

}



3.5：#include<iostream>

using namespace std;

int gs(int day)

{

if (day == 10)

{

return 1;

}

else

{

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

}

}

int main()

{

cout << "第一天猴子共摘桃子" << gs(1) << "个" << endl;

}



1. **遇到的问题与解决方法**

**看不懂代码，在vs中输入时一直报错，打字速度较慢。看书查找知识点，加以变通运用，并运用了科技的手段，上网查找询问，AI查错，思考改正，最终修改成功得以运行。**

**六、体会**

**还是要多多动手进行实操，写代码训练，独立思考做题，只看书念书一味听讲是不行的，要有自己的理解，自己上手会发现很多不足与错误然后加以改正，有不会的可以适当借助科技的力量，上网查找询问，AI查错，进行学习借鉴，也要有自己的思考，提高打字速度，思考能力，熟练掌握技能。**

**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**

**4.1.1:**#include<iostream>

using namespace std;

int main()

{

int sz[10];

int gs = 0;

cout << "Enter ten numbers : " << endl;

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

{

int number;

cin >> number;

bool different = true;

for (int j = 0; j < gs; j++)

{

if (sz[j] == number)

{

different = false;

break;

}

}

if (different)

{

sz[gs++] = number;

}

}

cout << "The distinct numbers are: ";

for (int x = 0; x < gs; x++)

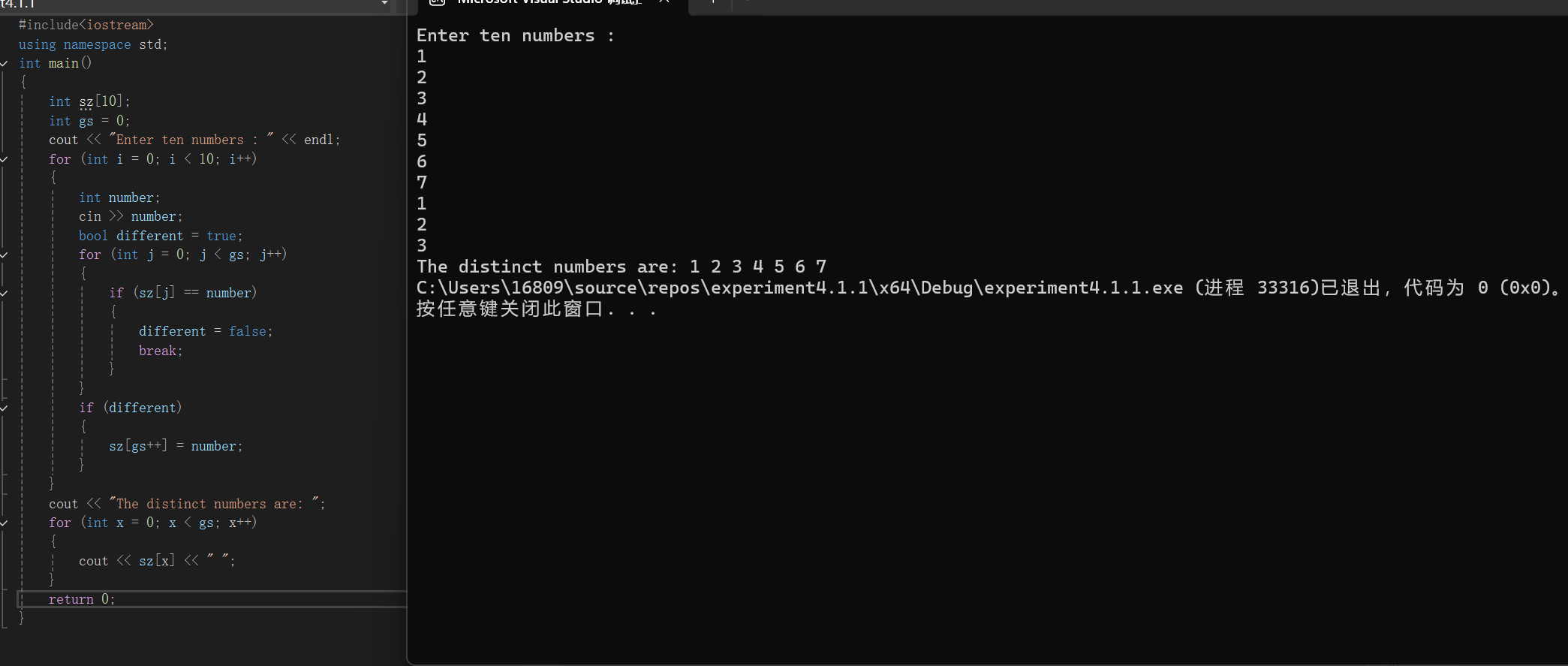
{

cout << sz[x] << " ";

}

return 0;

}



4.1.2:#include <iostream>

using namespace std;

void swap(double & a,double & b)

{

double temp = a;

a = b;

b = temp;

}

void mp(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);

}

int main()

{

double list[10];

cout << "请输入十个双精度数字：" << endl;

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

{

double number;

cin >> number;

list[i] = number;

}

mp(list, 10);

cout << "冒牌排序后的数组为：";

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

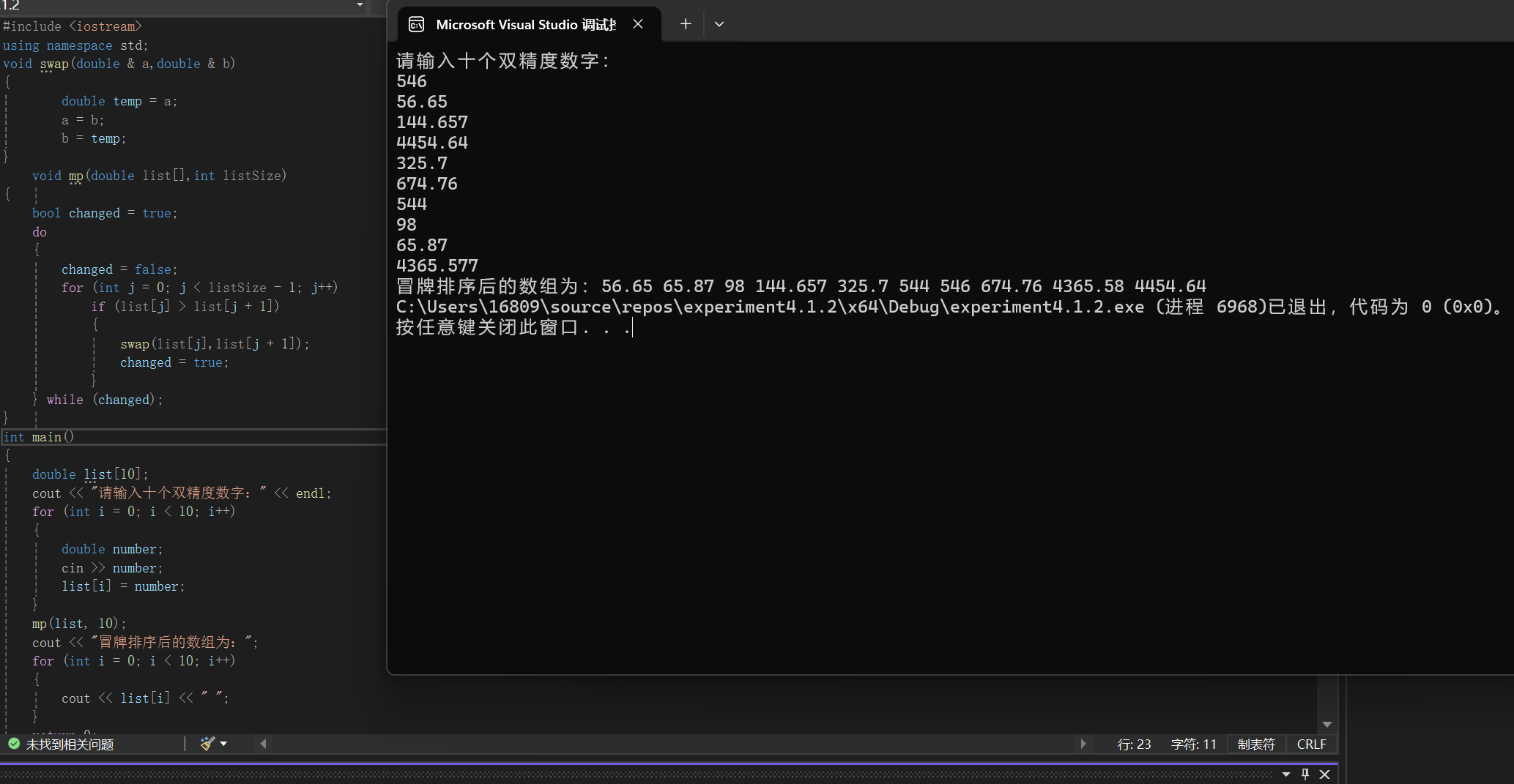
{

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

}

return 0;

}



4.1.3：#include <iostream>

using namespace std;

int main()

{

bool arr[100] = { false };

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

{

for (int L= S - 1;L < 100;L += S)

{

arr[L] = !arr[L];

}

}

cout << "开着的储物柜号码为：";

for (int i = 0; i < 100; i++)

{

if (arr[i])

{

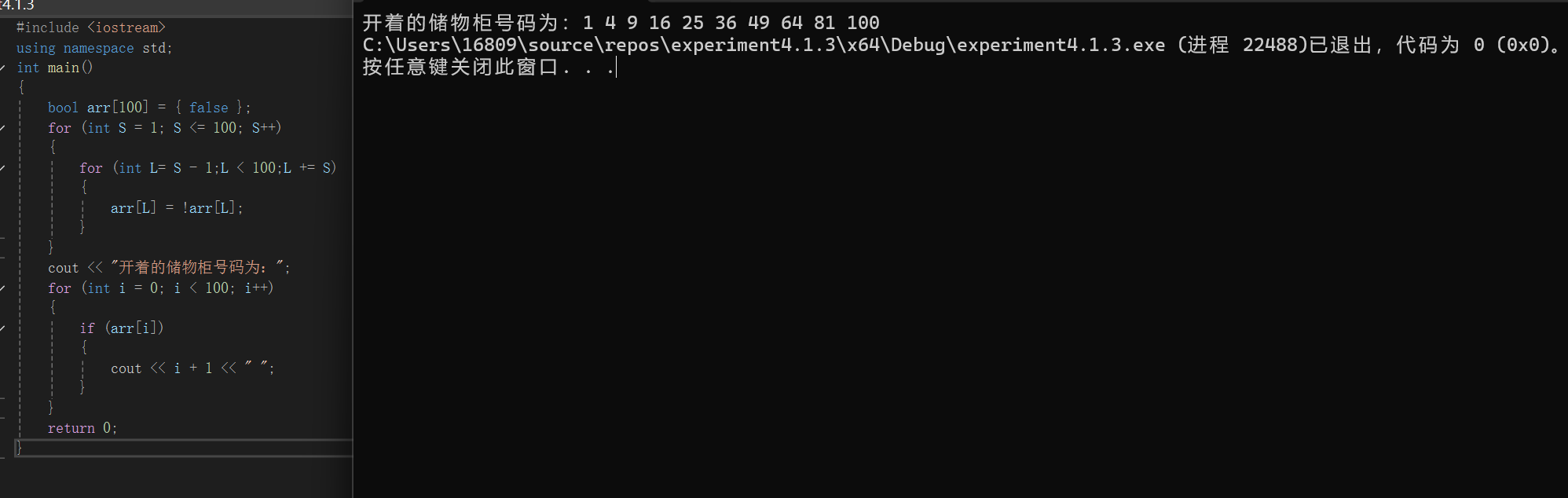
cout << i + 1 << " ";

}

}

return 0;

}



4.1.4：#include <iostream>

using namespace std;

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[k])

{

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 list1[80], list2[80], size1, size2, list3[160];

cout << "Enter list1: ";

size1 = sizeof(list1) / sizeof(list1[0]);

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

{

cin >> list1[i];

size1 = sizeof(list1) / sizeof(list1[0]);

}

cout << "Enter list2: ";

size2 = sizeof(list2) / sizeof(list2[0]);

for (int j = 0; j < size2; j++)

{

cin >> list2[j];

size2 = sizeof(list2) / sizeof(list2[0]);

}

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

cout << "The merged list is ";

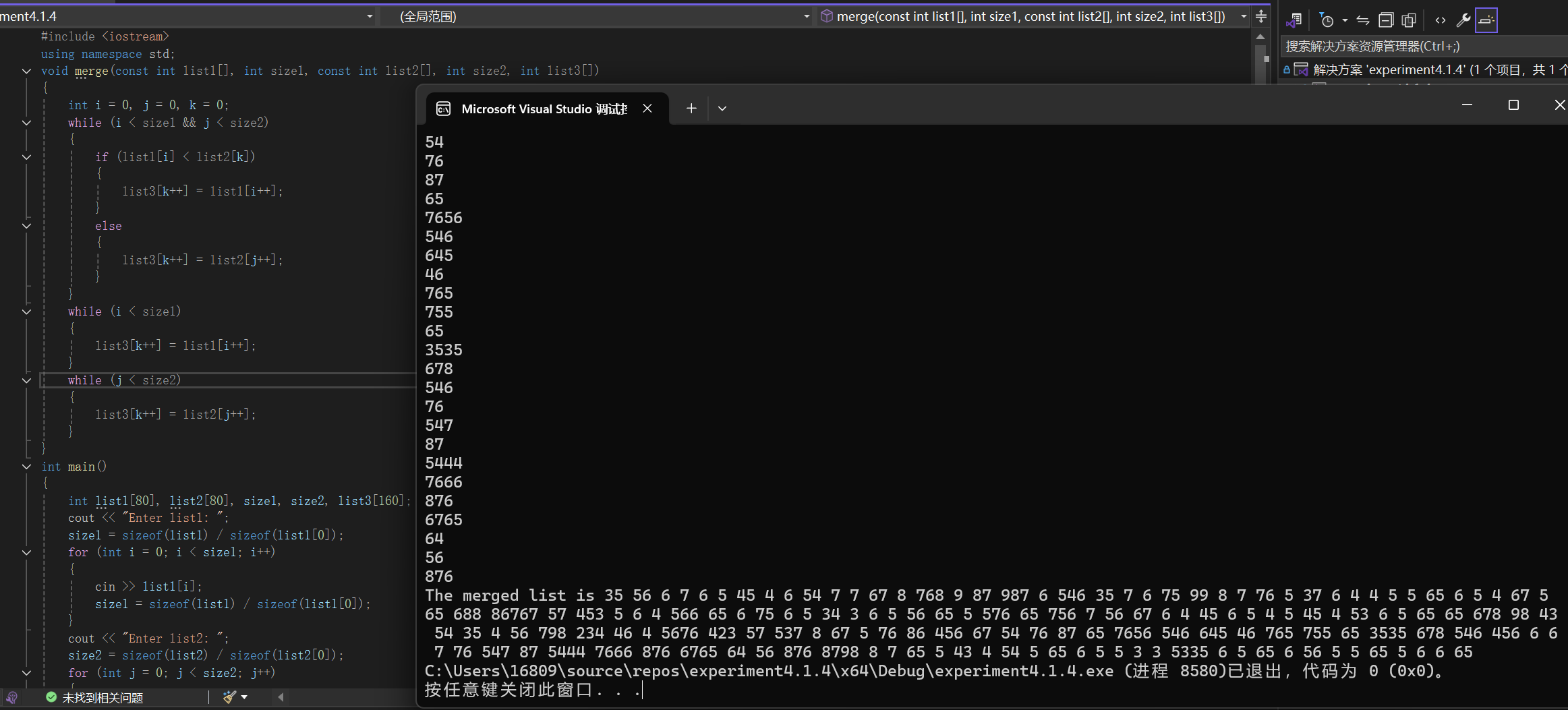
for (int k = 0; k < size1+size2; k++)

{

cout << list3[k] << " ";

}

}



4.1.5：#include<iostream>

#include<string>

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()

{

char s1[100], s2[100];

cout << "Enter the first string: ";

cin.getline(s1,100);

cout << "Enter the second string: ";

cin.getline(s2,100);

int result = indexOf(s1, s2);

if (result != -1)

{

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

}

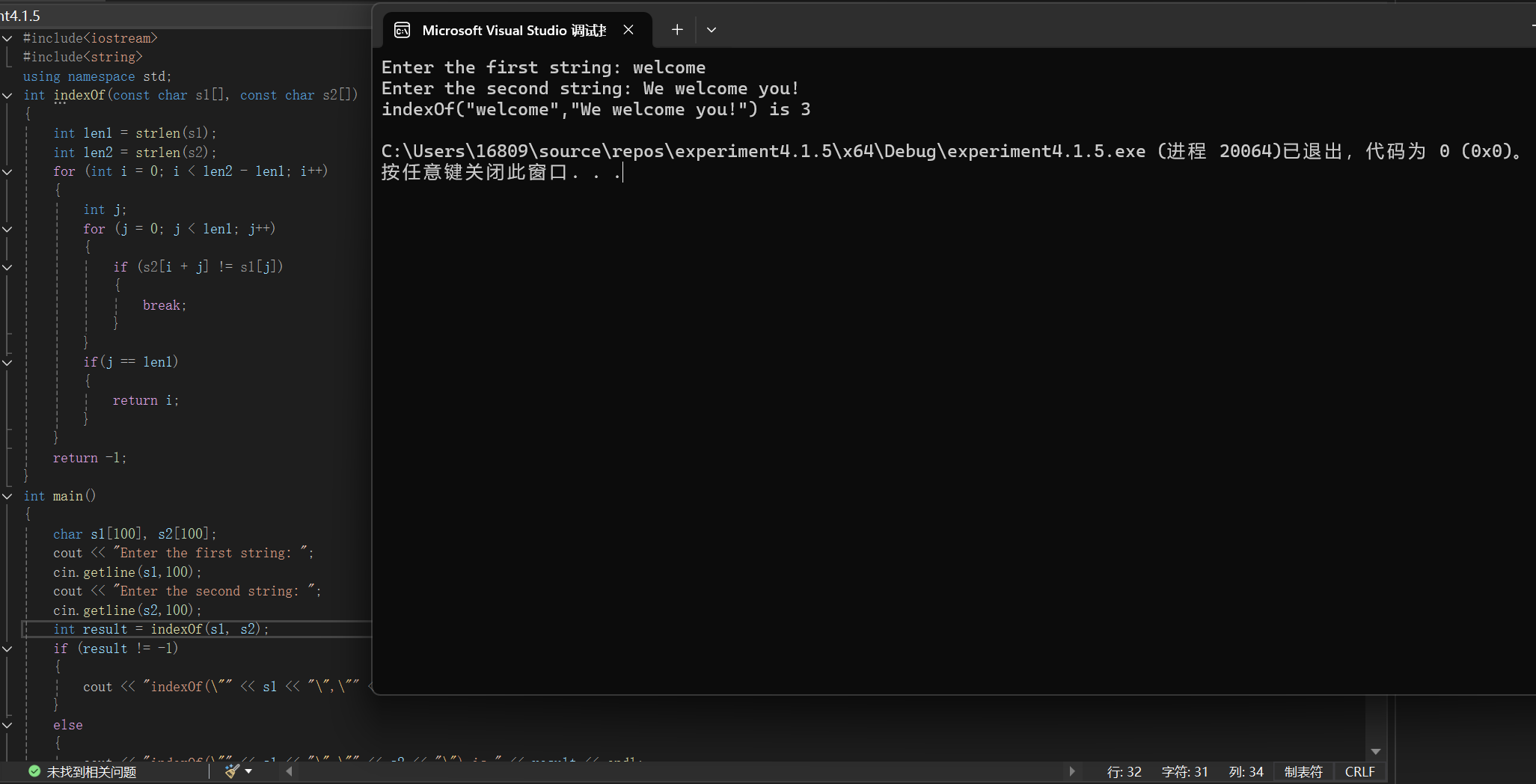
else

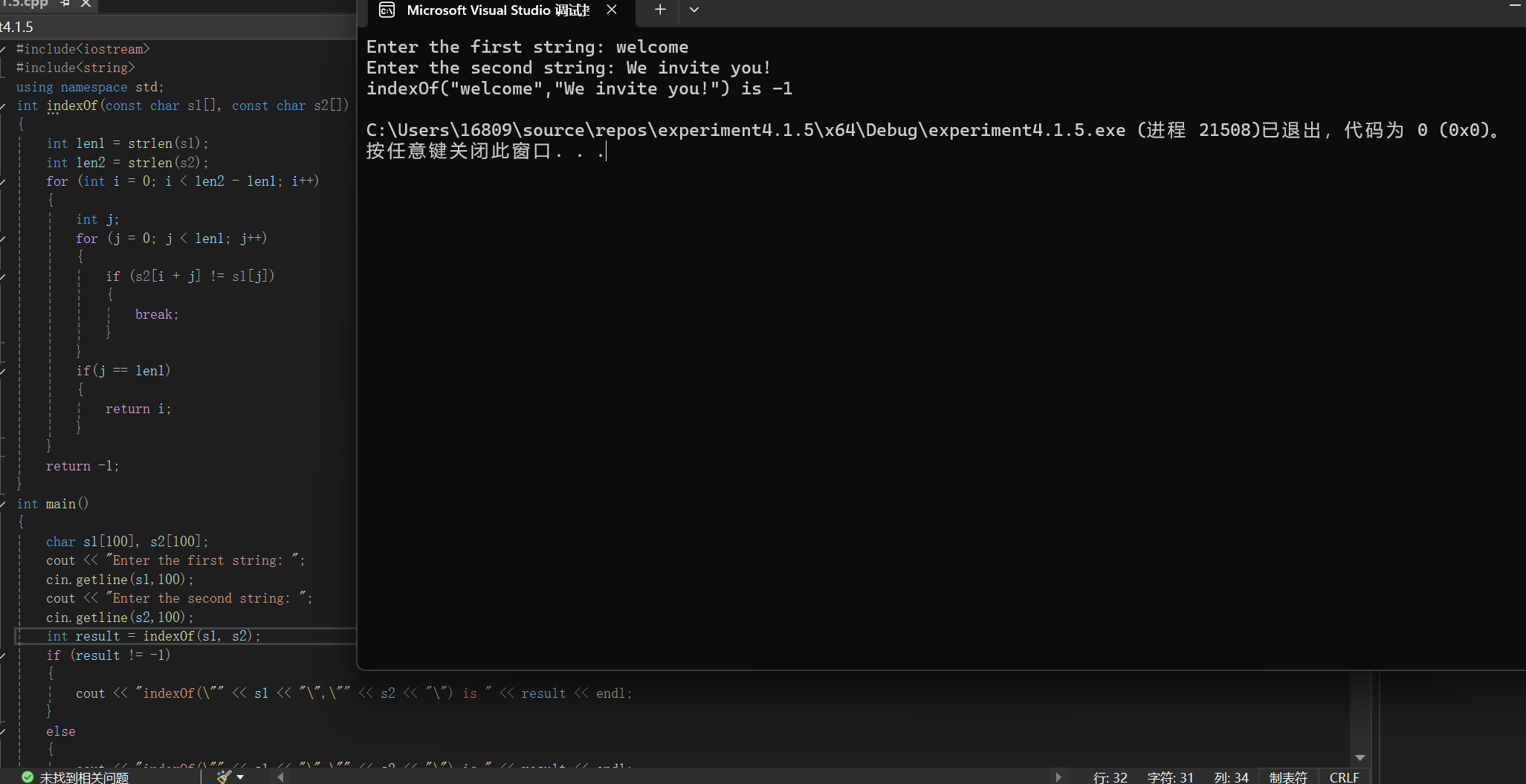
{

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

}

}





4.1.6:#include<iostream>

#include<cctype>

#include<string>

using namespace std;

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

{

int sy;//索引，用来指二十六个字母顺序位置的

for (int i = 0; s[i] != '\0'; i++)//遇到空格就停

{

if (isalpha(s[i]))//函数，看是否为字母

{

sy = tolower(s[i]) - 'a';//函数，将字母转为小写，然后找索引

counts[sy]++;

}

}

}

int main()

{

const int size = 26;//二十六个字母

int counts[size] = { 0 };//初始化数组

cout << "Enter a string: ";

string input;

getline(cin, input);

count(input.c\_str(), counts);//函数，将string类型转为const char\*类型，返回指针

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

{

if (counts[i] > 0)

{

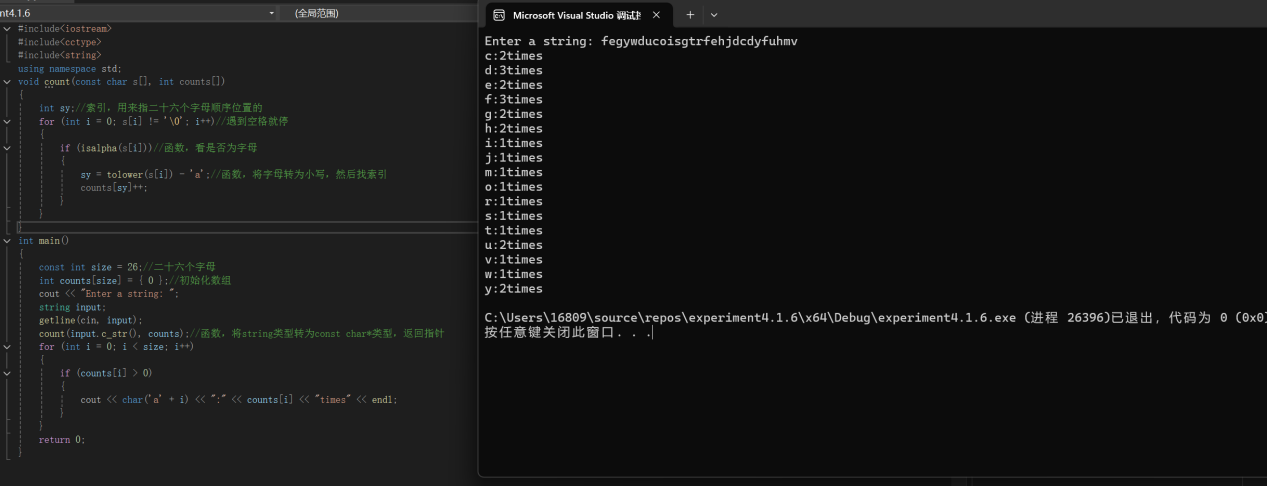
cout << char('a' + i) << ":" << counts[i] << "times" << endl;

}

}

return 0;

}



4.2.1.1：#include<iostream>

using namespace std;

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;

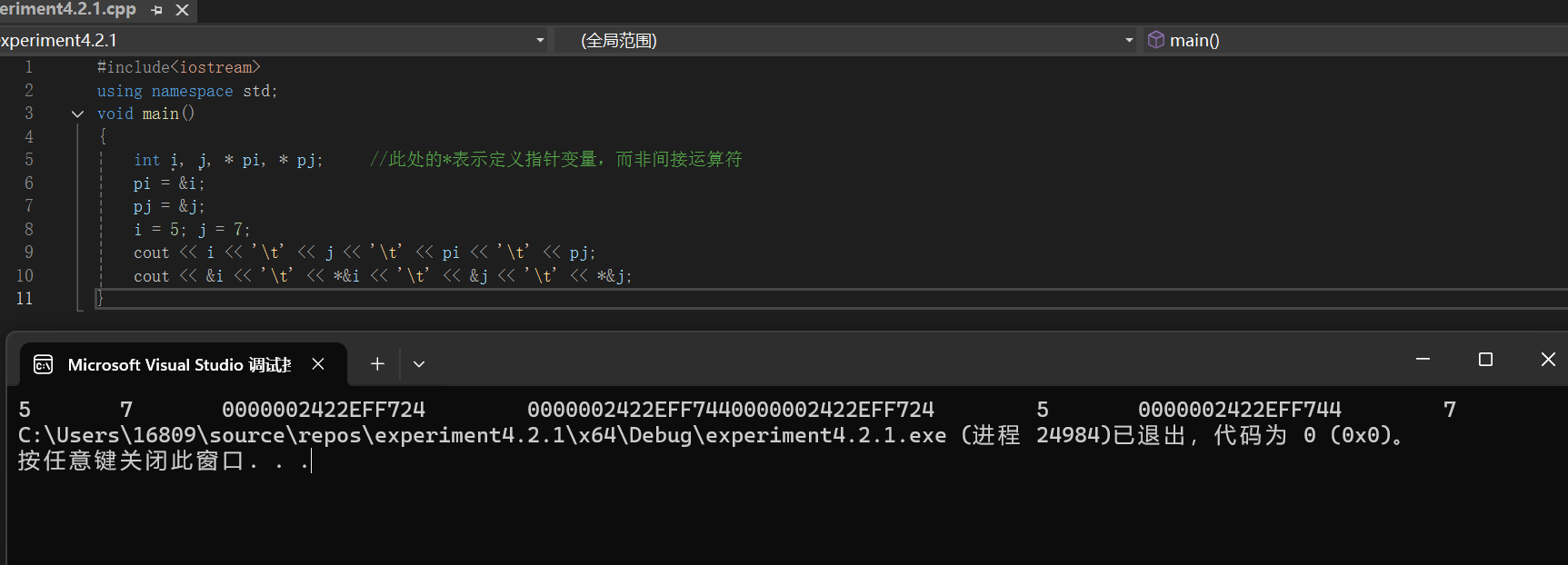
pj = &j;

i = 5; j = 7;

cout << i << '\t' << j << '\t' << pi << '\t' << pj;

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

}



4.2.1.2:#include<iostream>

using namespace std;

int main() //C语言程序，要了解

// 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]。

{

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

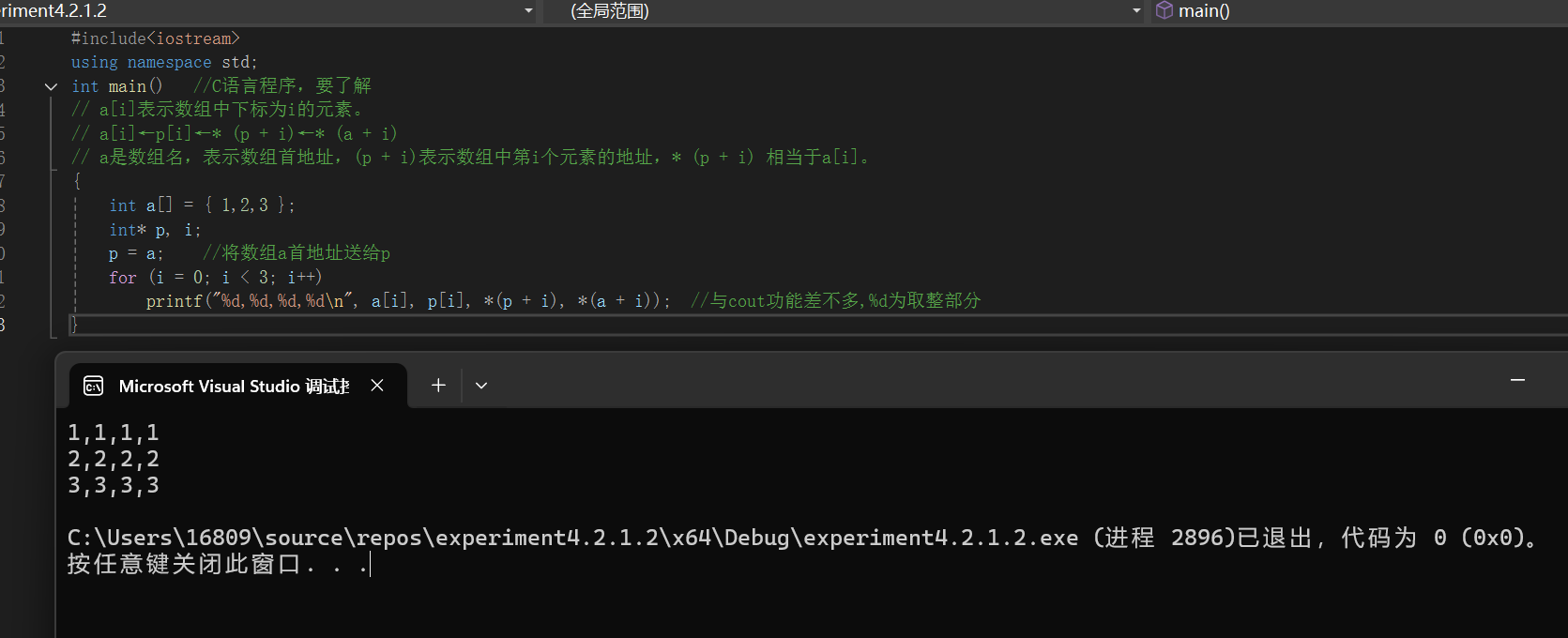
int\* p, i;

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,%d为取整部分

}



4.2.1.3:#include<stdio.h>

#include<iostream>

using namespace std;

void f(char\* st, int i)

{

st[i] = '\0';

cout << st; // printf(“%s\n”,st);

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

}

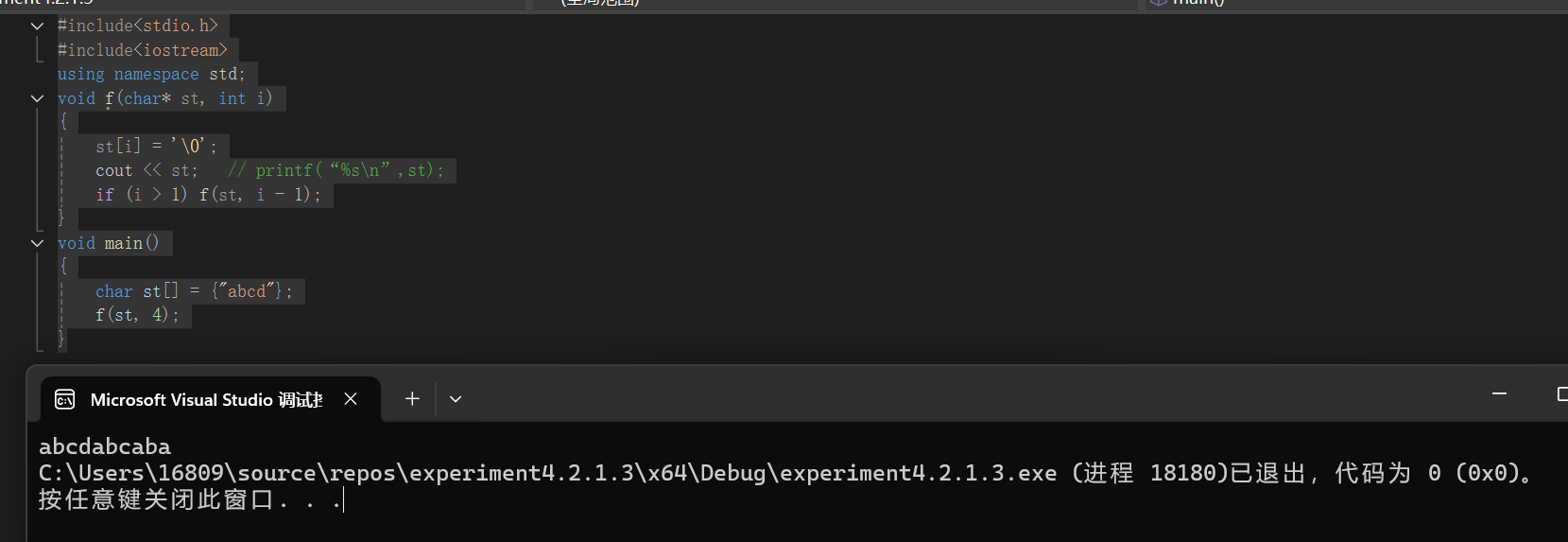
void main()

{

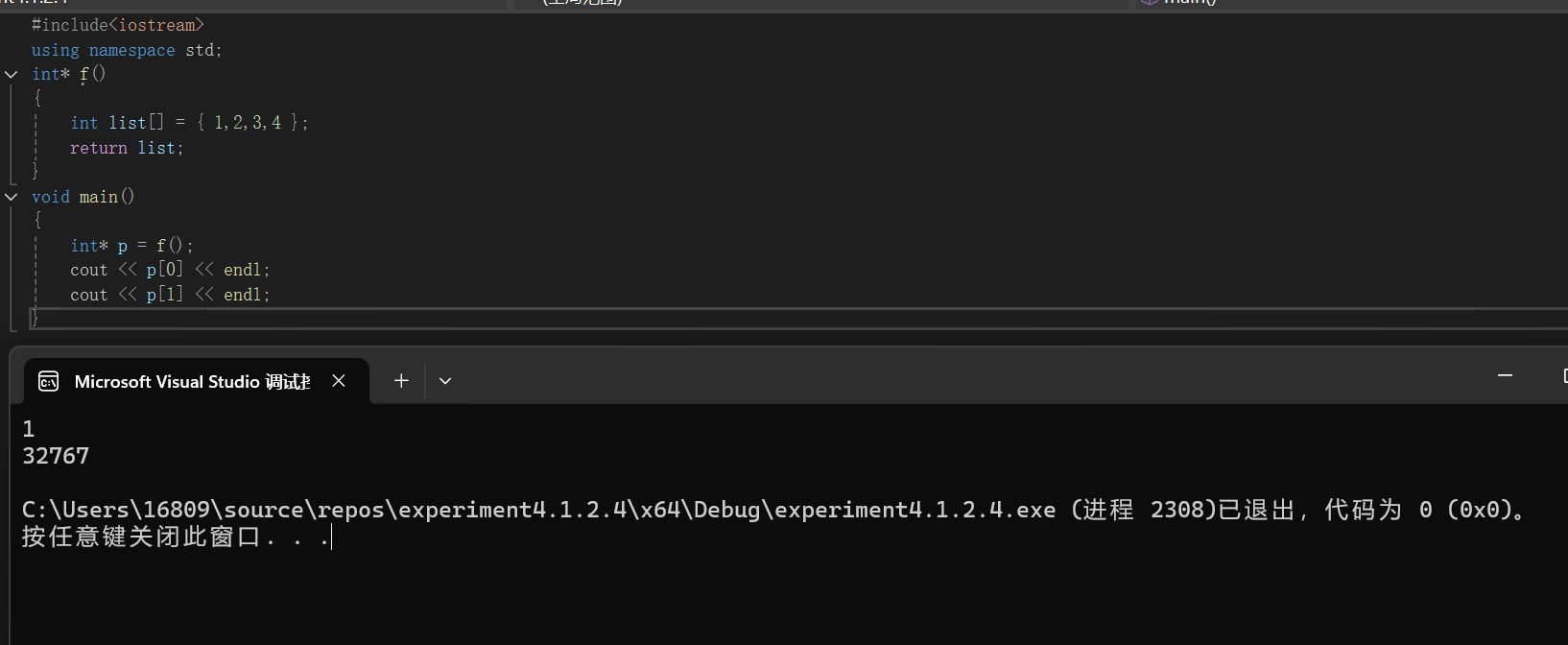
char st[] = {"abcd"};

f(st, 4);

}



4.2.1.4：不能。因为list数组为局部变量，储存在栈区中，函数执行完毕内存就被自动释放，返回的指针成为悬空指针，第一次打印正确的数字是因为编译器做了保留，第二次这个数据就不再保留，为非法操作。结果如下：



修改后的代码：#include<iostream>

using namespace std;

int\* f()

{

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

return list;

}

void main()

{

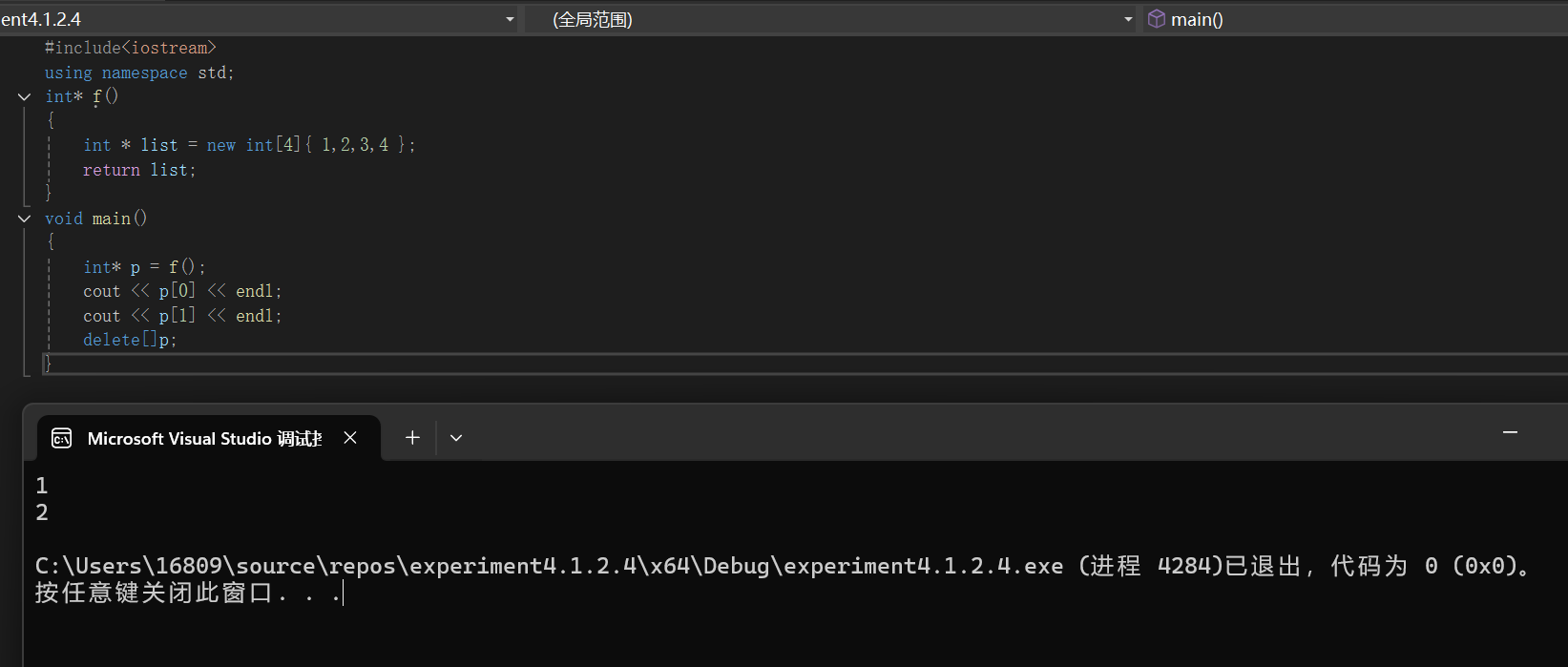
int\* p = f();

cout << p[0] << endl;

cout << p[1] << endl;

delete[]p;

}



程序设计部分：

4.2.2.1：#include <iostream>

#include<string>

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 = 0;

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

{

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

{

break;

}

}

if (j = len1)

{

return i;

}

}

return -1;

}

int main()

{

char s1[100], s2[100];

cout << "Enter the first string :";

cin.getline(s1, 100);

cout << "Enter the second string: ";

cin.getline(s2, 100);

int result = indexof(s1, s2);

if (result != -1)

{

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

}

else

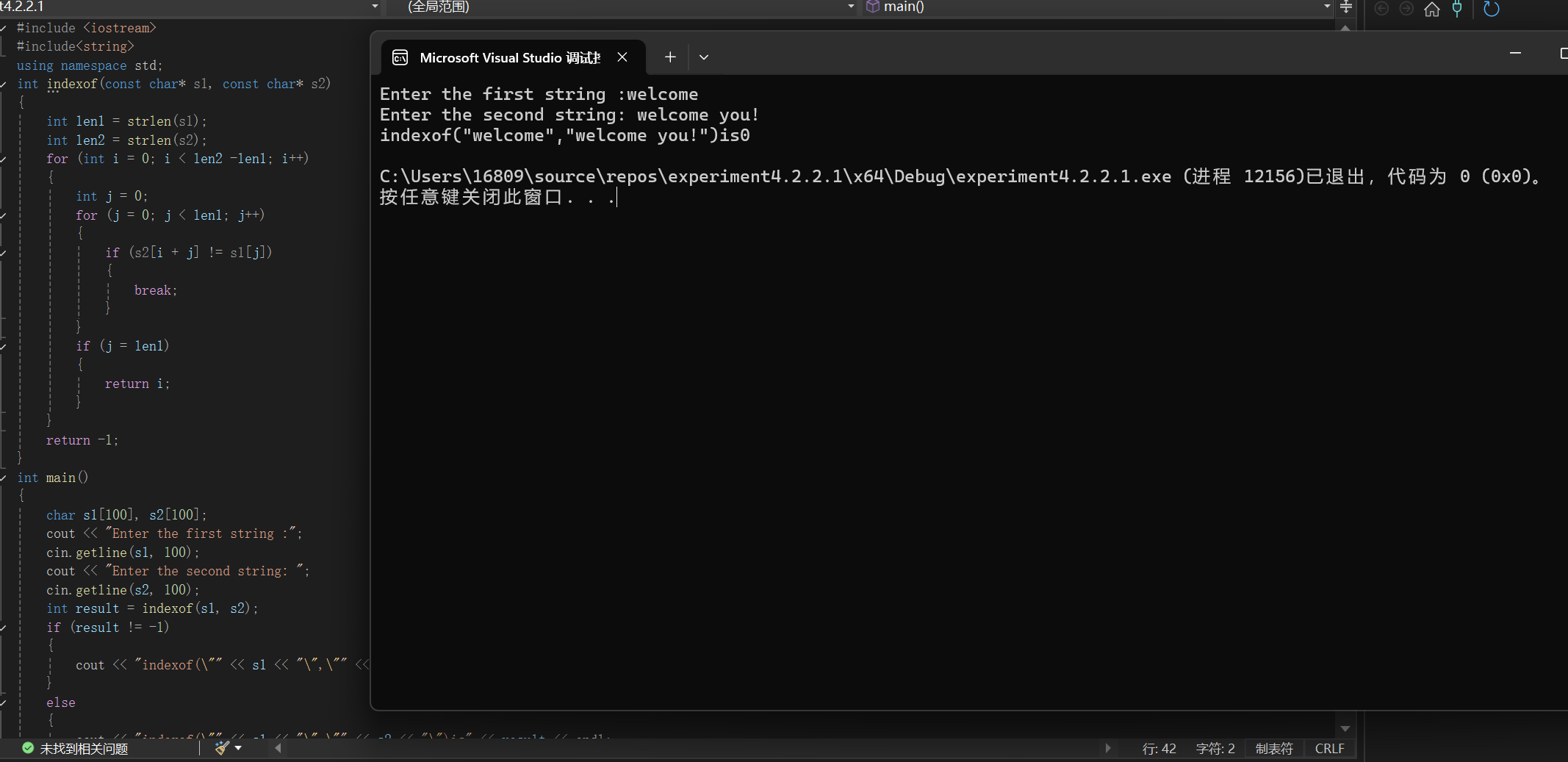
{

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

}

return 0;

}



4.2.2.2：#include<iostream>

#include<string>

using namespace std;

int parseHex(const char\* const hexString)

{

int len = strlen(hexString);

int result = 0;

int power = 1;

for (int i = len - 1; i >= 0; i--)

{

char c = hexString[i];

int s;

if (c >= '0' && c <= '9')

{

s = c - '0';

}

else if (c >= 'a' && c <= 'f')

{

s = c - 'a'+10;

}

else if (c >= 'A' && c <= 'F')

{

s = c - 'A'+10;

}

else

{

cout << "输入的不是有效的十六进制字符串，请输入正确的十六进制字符串！" << endl;

return -1;

}

result += power \* s;

power \*= 16;

}

return result;

}

int main()

{

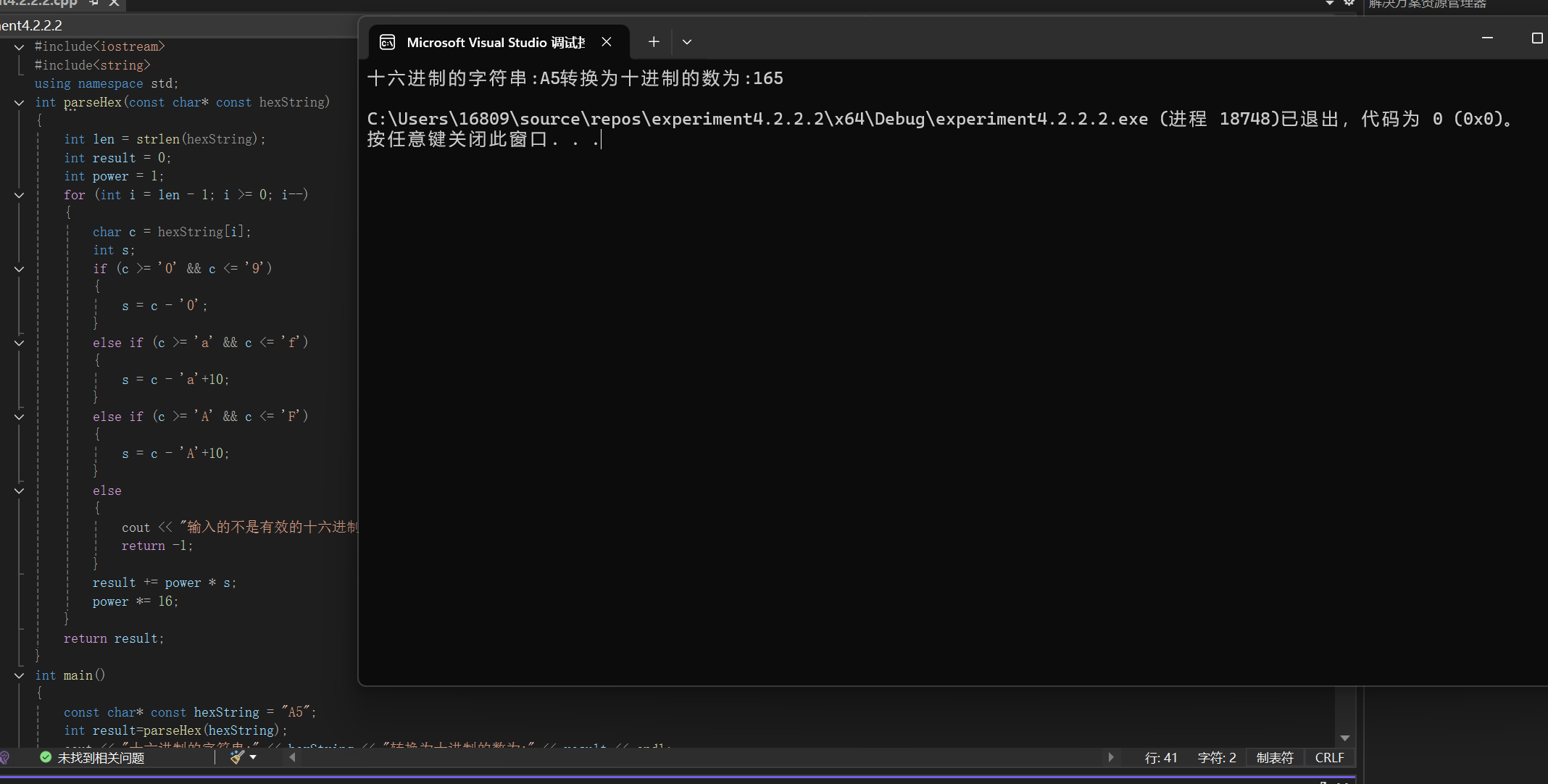
const char\* const hexString = "A5";

int result=parseHex(hexString);

cout << "十六进制的字符串:" << hexString << "转换为十进制的数为:" << result << endl;

return 0;

}



4.2.2.3：#include<iostream>

using namespace std;

void mp(int \* arr, int len)

{

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

{

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

{

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

{

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

int main()

{

int len;

cout << "请输入数组元素个数：";

cin >> len;

int\* arr = new int[len];

cout << "请输入数组元素："<<endl;

for (int j = 0; j < len; j++)

{

cin >> arr[j];

}

cout << "排序前的数组为：" << endl;

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

{

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

}

cout << endl;

mp(arr, len);

cout << "排序后的数组为：" << endl;

for (int j = 0; j < len; j++)

{

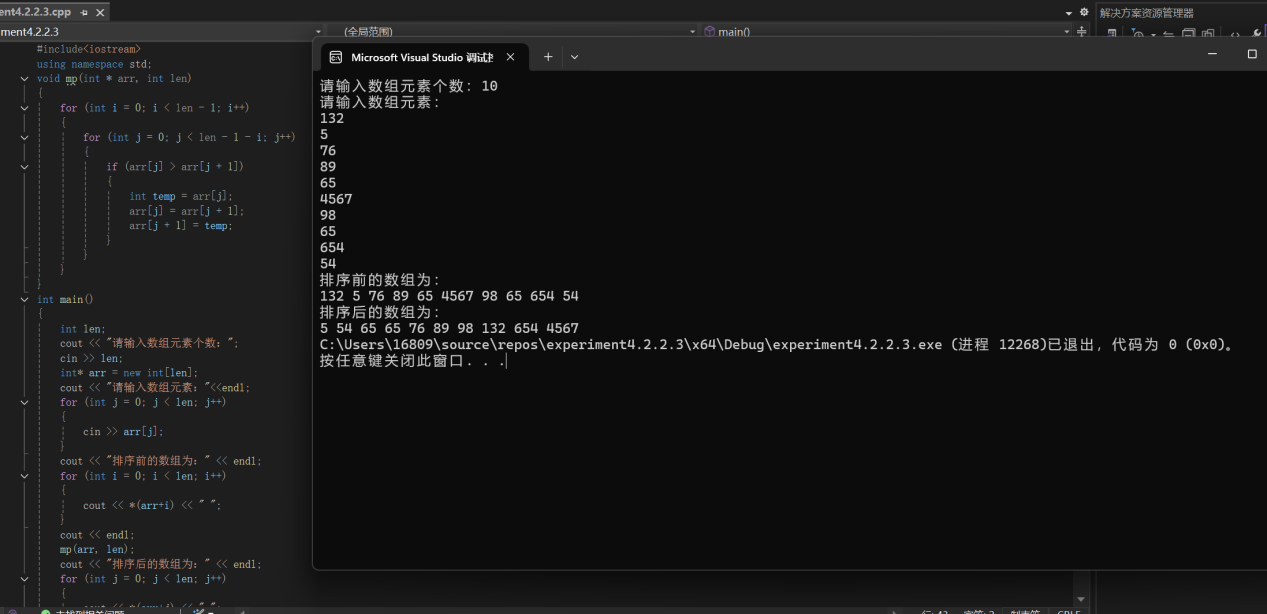
cout << \*(arr+j) << " ";

}

delete[]arr;

return 0;

}



**四、遇到的问题与解决方法**

**看不懂代码，有几题完全不会写，没有思路，不知道怎么写，在vs中输入时一直报错，打字速度较慢，许多C++常见的系统函数不会使用，字符串不会处理。看书查找知识点，加以变通运用，并运用了科技的手段，上网查找询问，AI查错，思考改正，最终修改成功得以运行。**

**五、体会**

**还是要多多动手进行实操，写代码训练，独立思考做题，只看书念书一味听讲是不行的，要有自己的理解，自己上手会发现很多不足与错误然后加以改正，有不会的可以适当借助科技的力量，上网查找询问，AI查错，进行学习借鉴，也要有自己的思考，提高打字速度，思考能力，熟练掌握技能。**