**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软工2405

学 号： 8209240518

姓 名： 曹雨佳

**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**

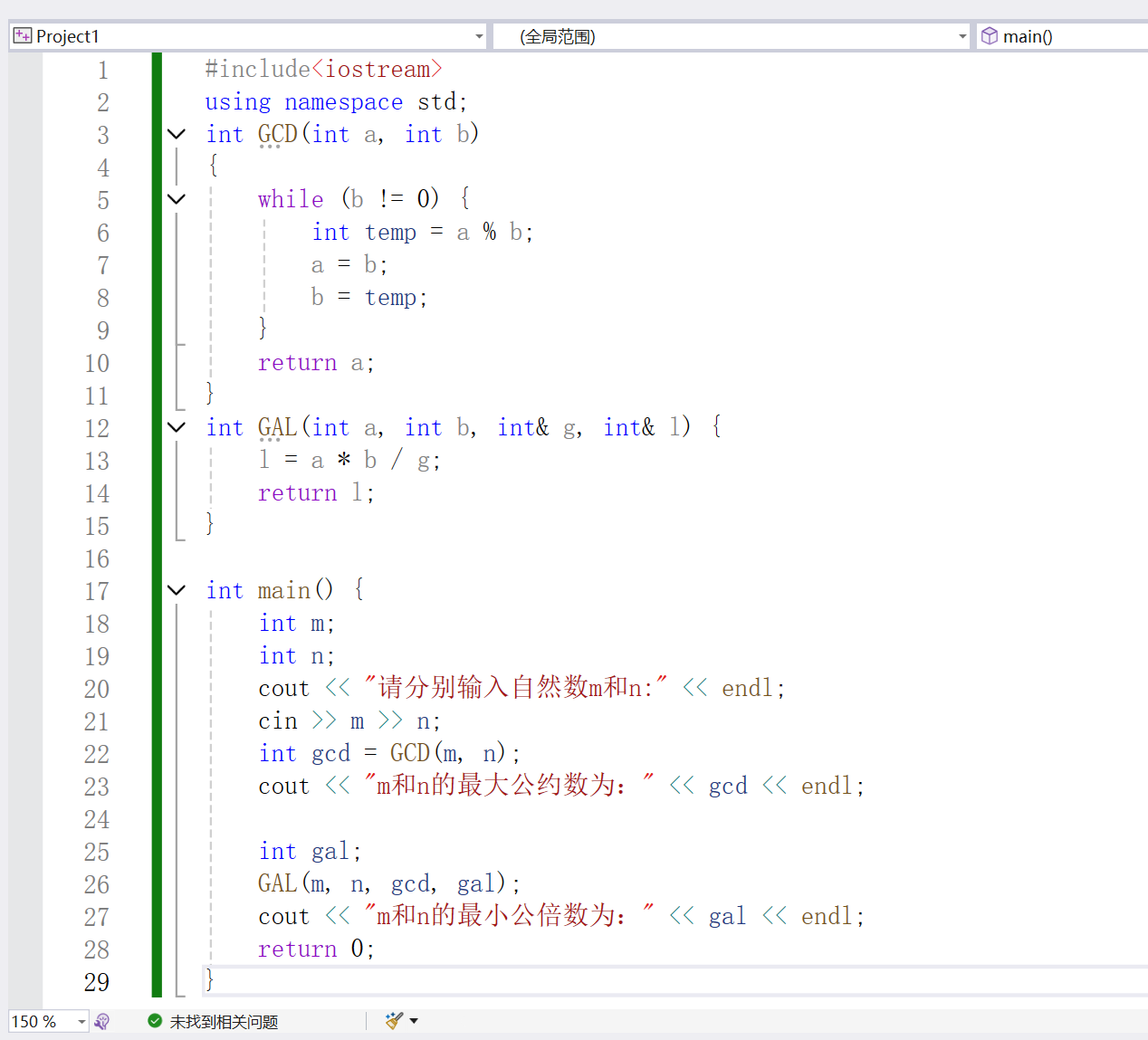
1. 本实验中函数中返回的值为什么与函数类型一致？

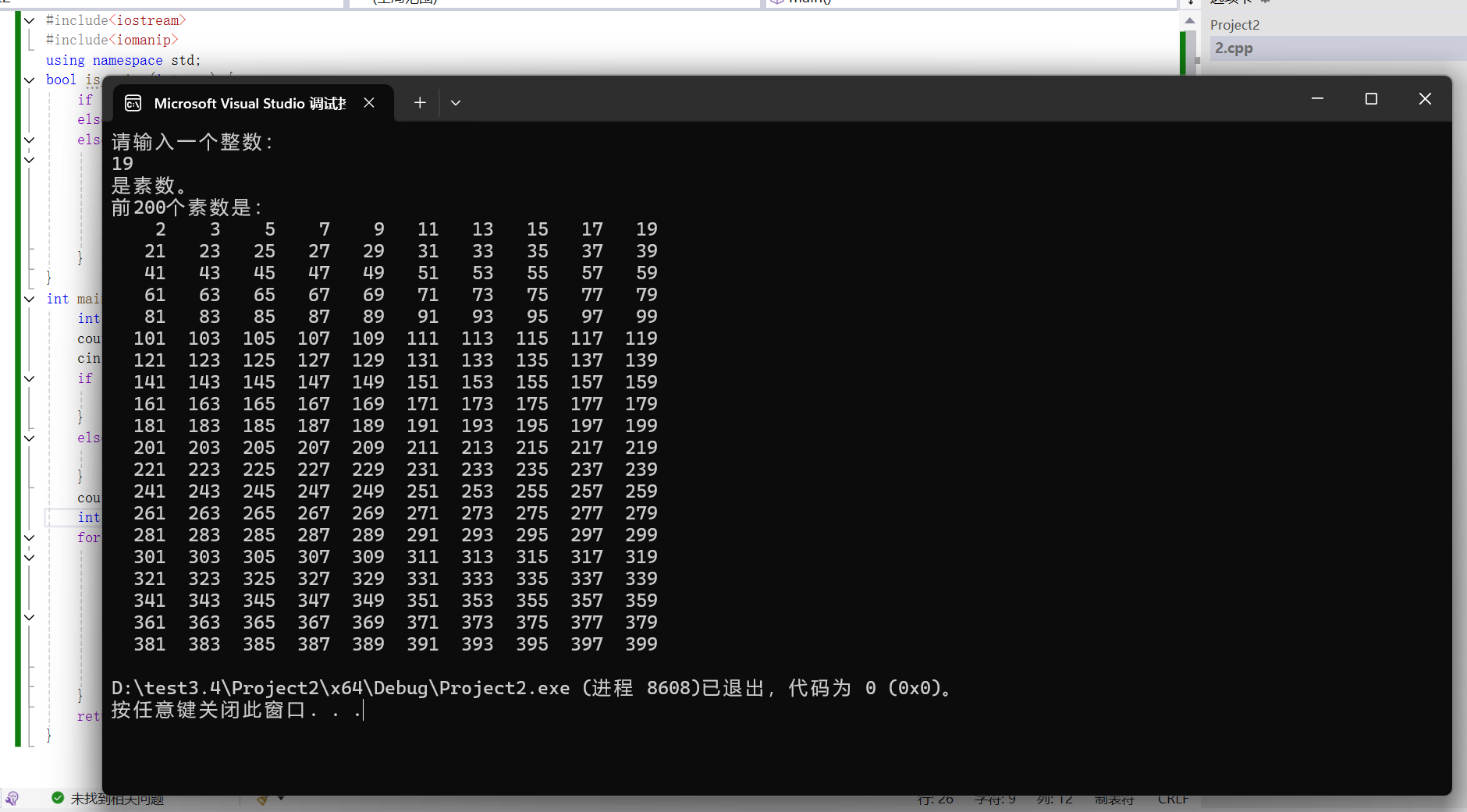
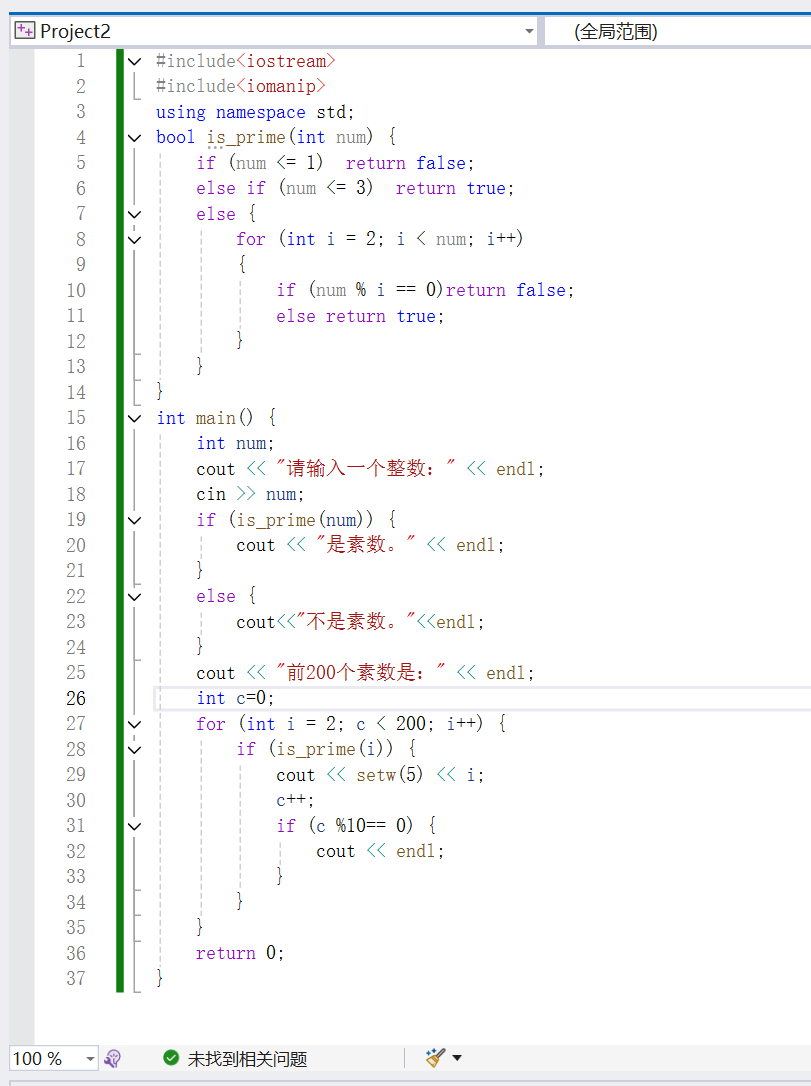
函数定义时声明了特定的返回类型，这是为了让调用者明确知道该函数执行完毕后会返回何种类型的数据，以便正确地接收和使用这个返回值。如果返回值类型和函数声明的不一致，编译器会报错，因为这违背了函数设计的规范和语法要求，无法保证程序按照预期去处理返回的数据。

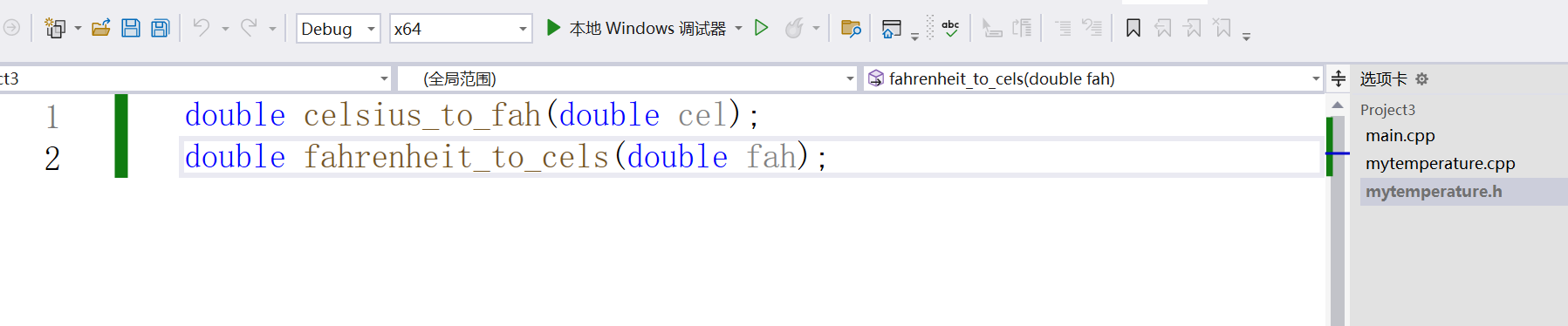
1. 本实验中主函数调用函数时采用的是何种传递方式？

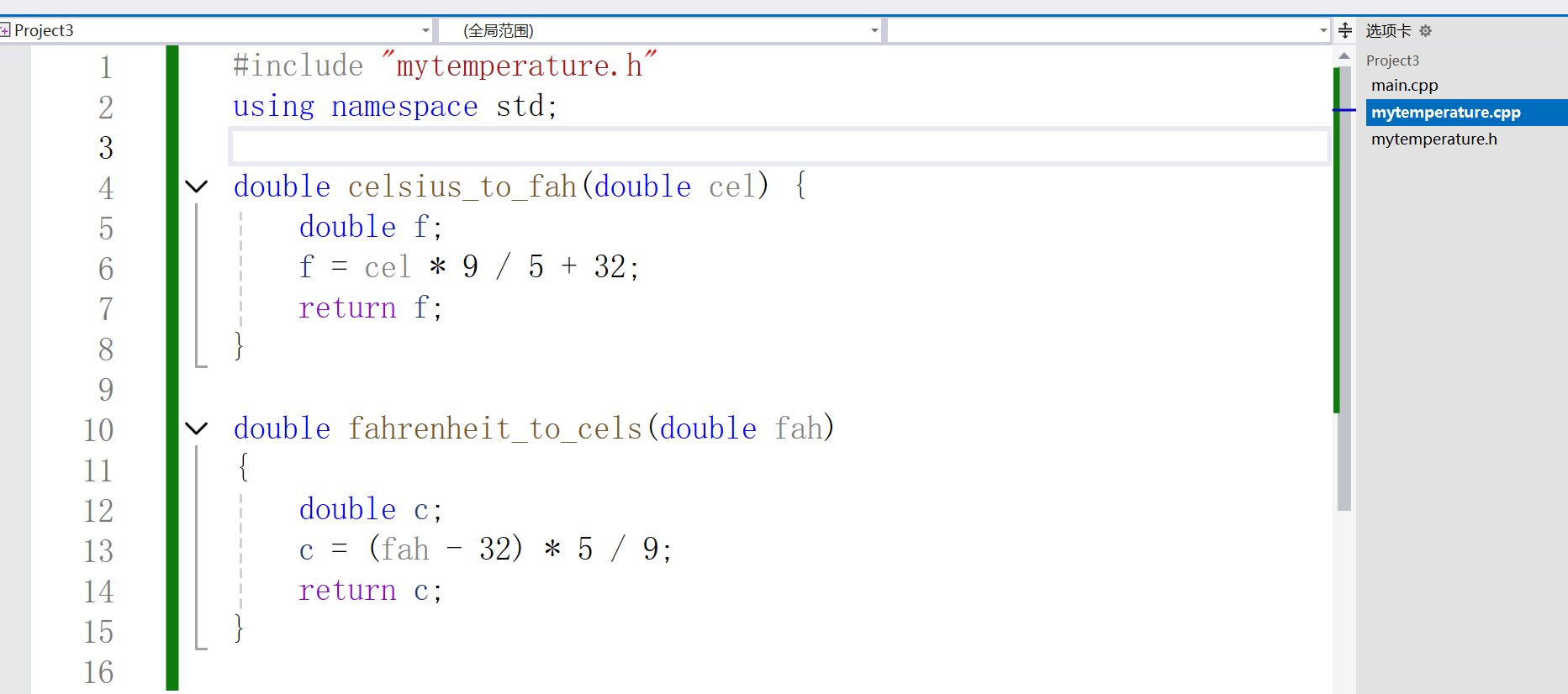
在本实验不同的函数调用场景中，有的采用值传递方式，比如传递普通的参数（像求最大公约数等部分，如果只是简单传递自然数  m  和  n  等基本类型的参数，就是值传递），值传递时是将实参的值复制一份传递给形参，形参在函数内的改变不会影响到原来的实参。还有的采用引用传递方式，例如在求最大公约数与最小公倍数部分提示使用引用参数，引用传递相当于给实参起了一个别名，函数内对引用参数的操作实际就是对实参本身的操作，可以通过这种方式实现从函数内部改变外部传入的变量的值等功能。

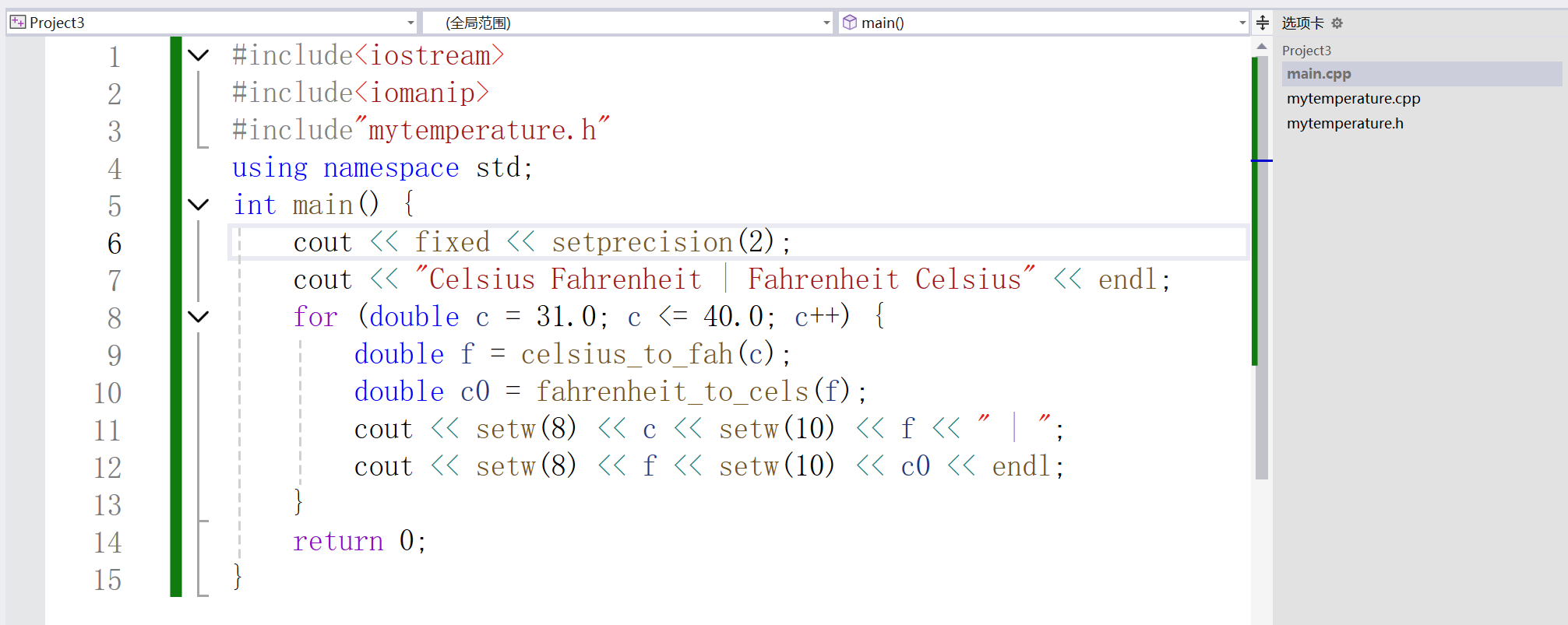
**四、算法分析，程序结果**

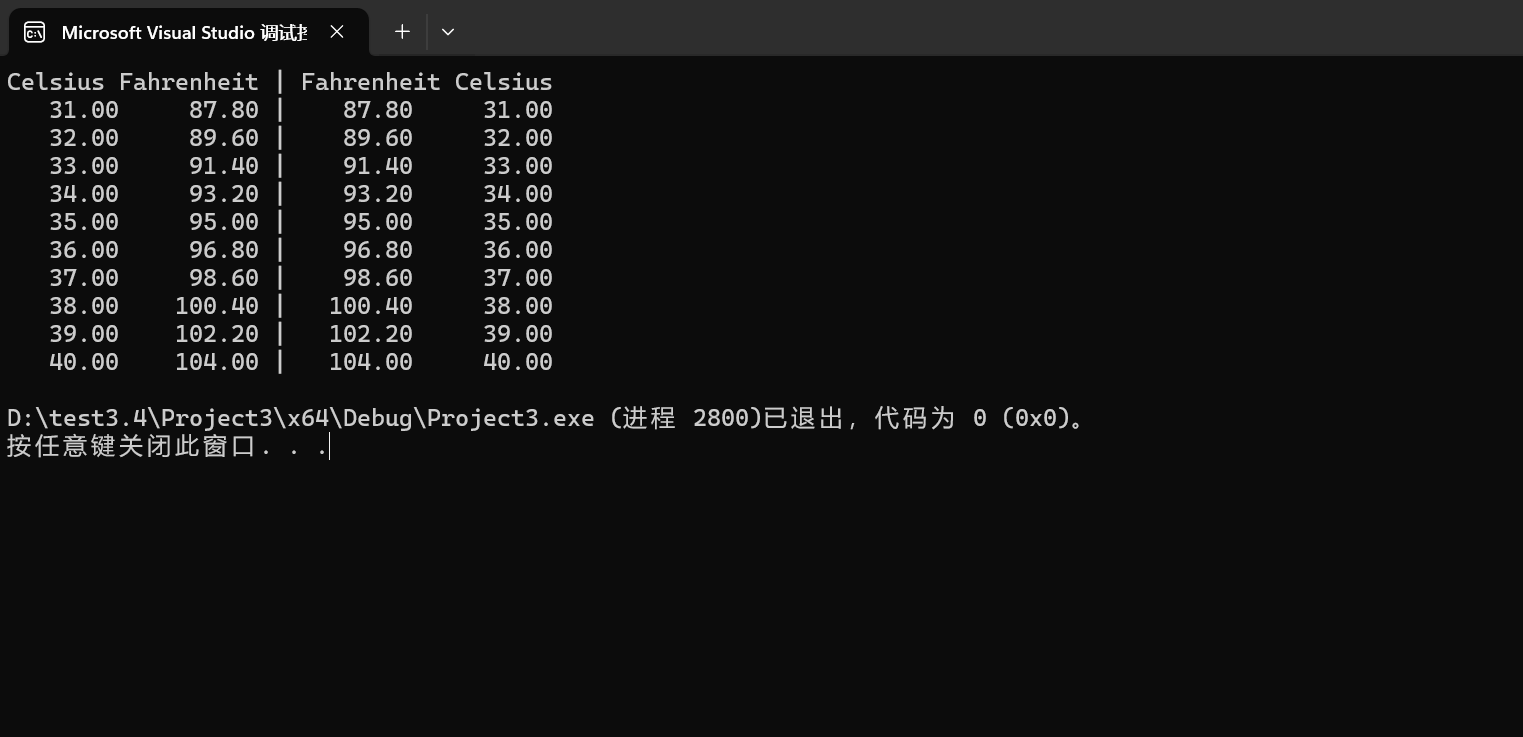
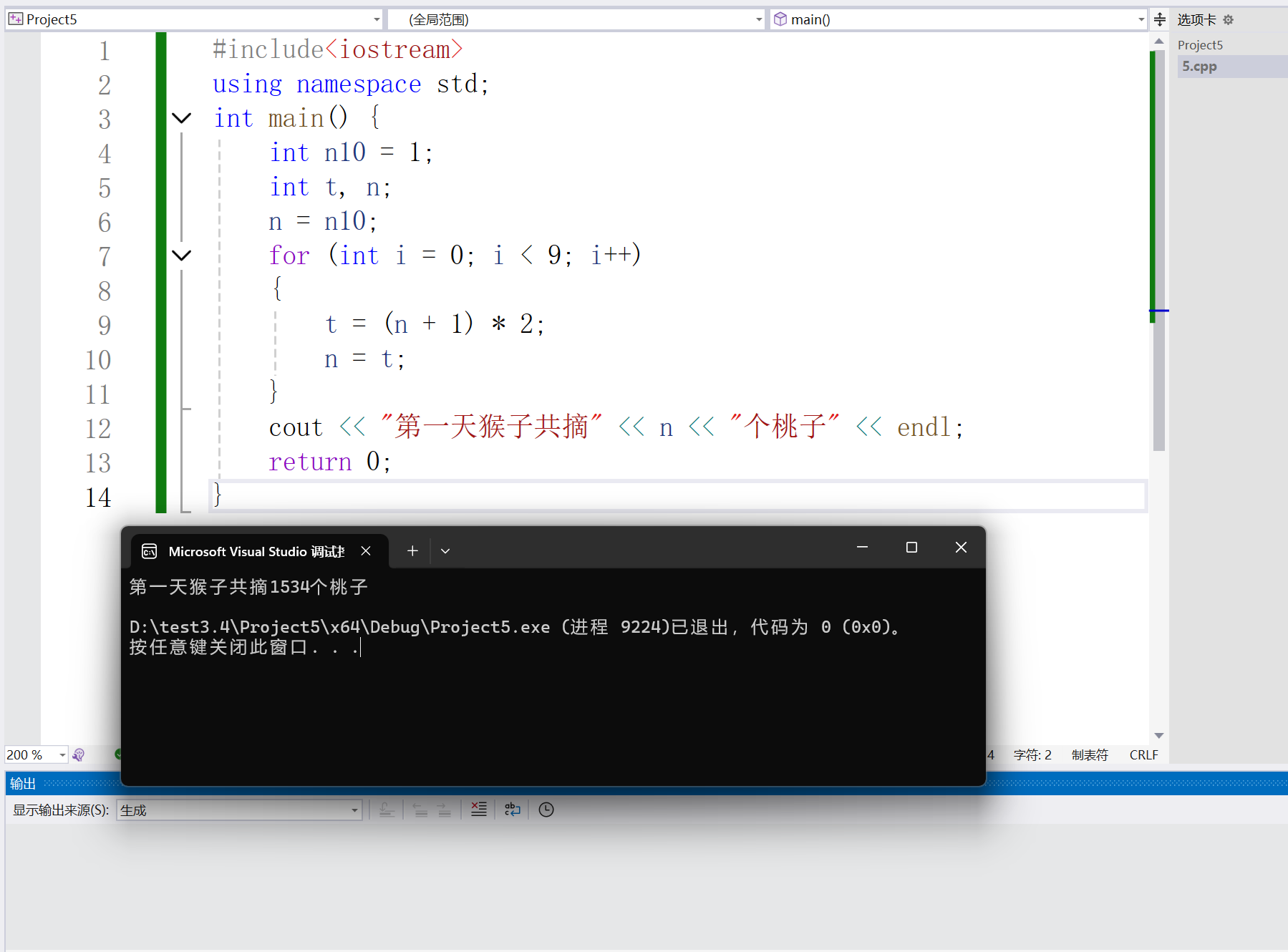
**1.**

**2.**

**3.**

****

****

**5.**

**五、遇到的问题与解决方法**

1. 问题：在使用引用参数求最大公约数与最小公倍数时，一开始不太理解引用的概念，导致程序出现逻辑错误，结果不正确。

解决方法：重新复习了引用的相关知识，仔细对照教材和网上的示例代码，搞清楚引用实际上就是变量的别名，在函数定义和调用时正确地使用了引用语法，经过多次调试，最终得到了正确的结果。

2. 问题：在编写判断素数的函数时，对于边界条件的判断出现失误，比如  1  这个数的判断出现错误情况。

解决方法：查阅资料明确了素数的定义是大于  1  且除了  1  和它自身外不能被其他自然数整除的数，所以在函数中专门针对  1  以及小于等于  0  的数做了特殊处理，修改代码逻辑后，函数能正确判断素数了。

3. 问题：在进行摄氏温度和华氏温度转换的多文件编程时，出现了链接错误，找不到函数定义的情况。

解决方法：仔细检查了头文件和对应的函数定义文件（ .cpp  文件）的命名、包含关系以及函数声明和定义是否匹配等情况，发现是在包含头文件时路径出现了一点小问题，修改正确后，成功完成了温度转换的测试并且能正确显示结果。

**六、体会**

通过这次实验，对函数相关的知识有了更深入的理解和掌握。函数的定义、声明、调用以及参数传递等每一个环节都非常重要，任何一处出现问题都可能导致整个程序无法正确运行。多文件编程的部分一开始觉得很复杂，但是经过不断调试和解决遇到的各种问题后，体会到了它在代码组织、模块划分方面的优势，可以让代码结构更加清晰，便于维护和扩展。同时，在遇到问题时，回顾基础知识、查阅资料以及耐心调试都是非常有效的解决办法，这也让我明白了编写程序不仅要掌握理论知识，更要注重实践操作，通过实际动手去发现问题、解决问题，才能真正提升自己的编程能力。

**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);

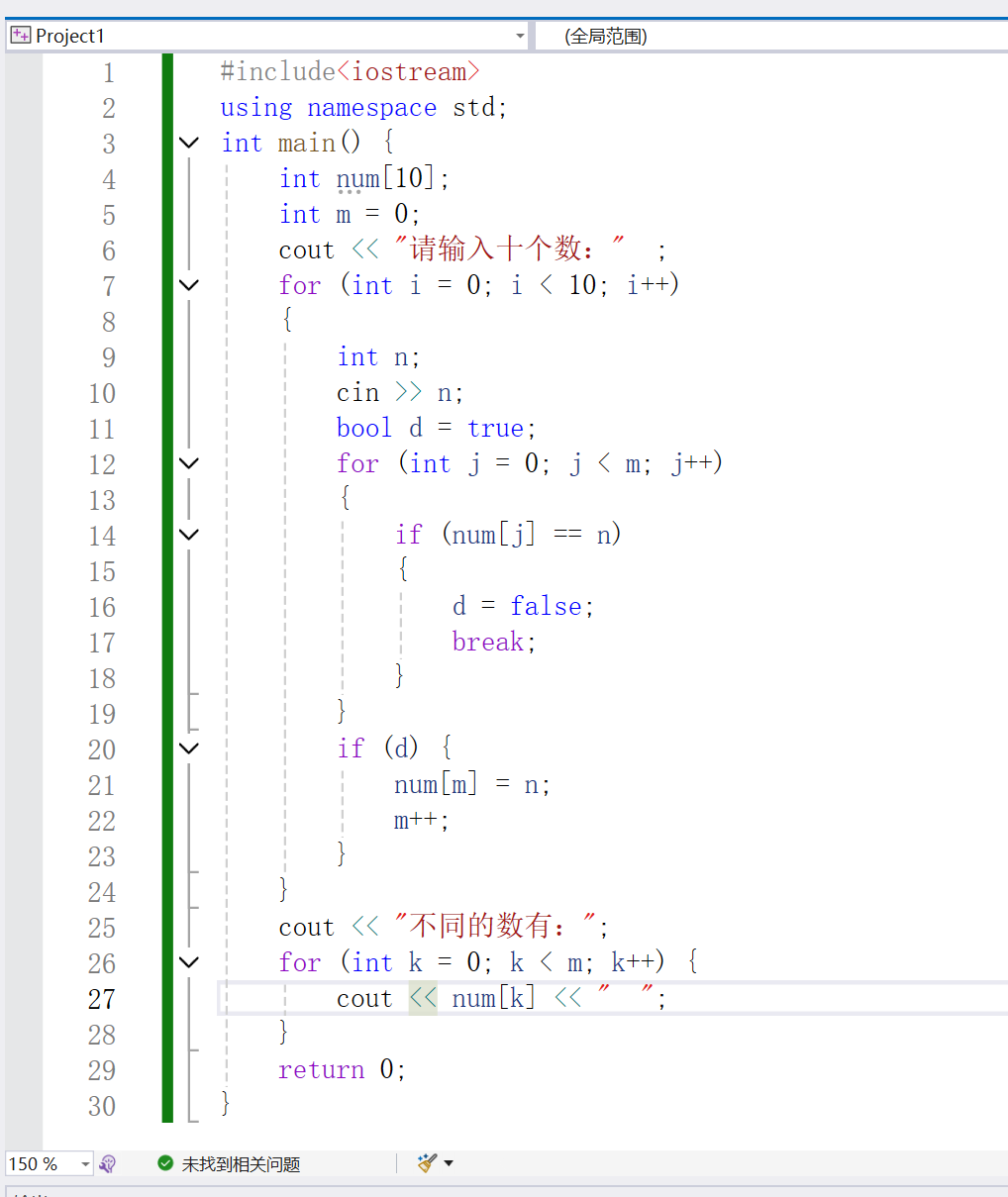
如：调用函数 parseHex(“A5”);返回165

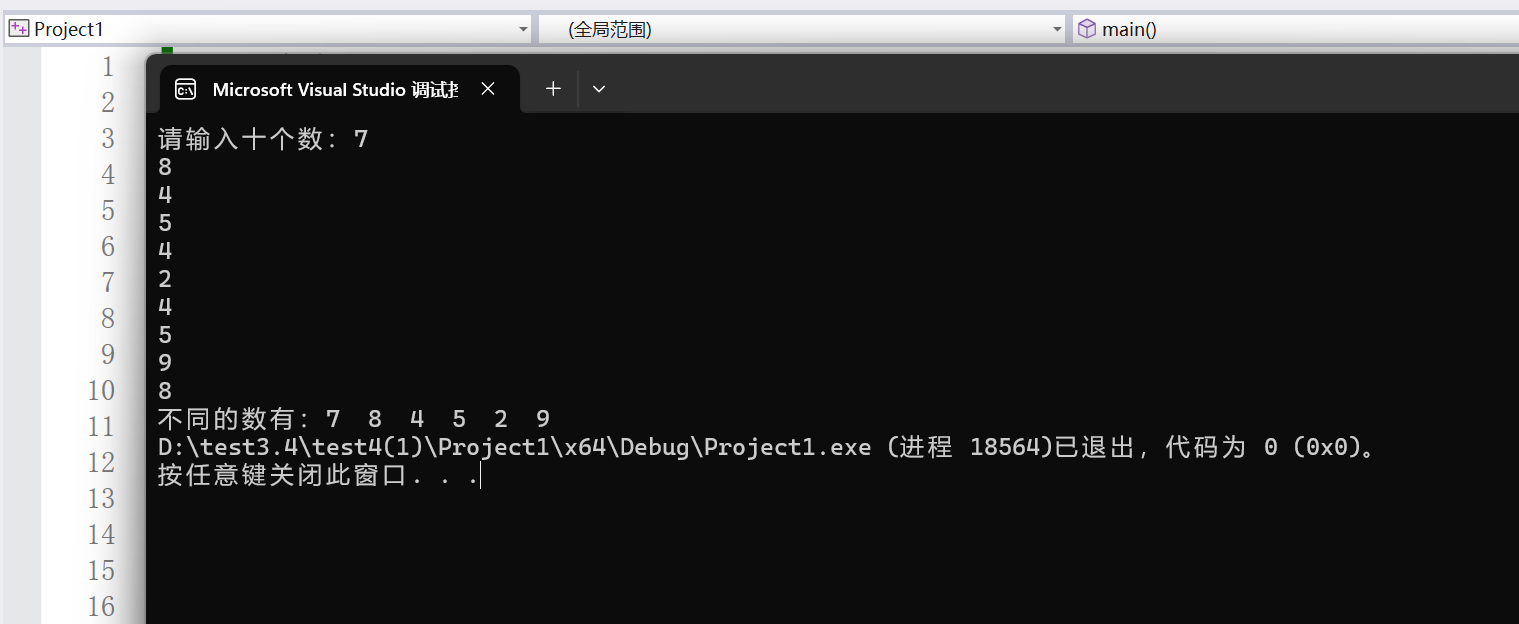
1. 主程序中建立一动态数组（使用new），数组元素及元素个数由键盘输入，动态调试观察指针及指针指向的内容；设计一个函数对数组由小到大排序；主程序中用指针方式输出数组元素；最后释放数组内存（delete）。

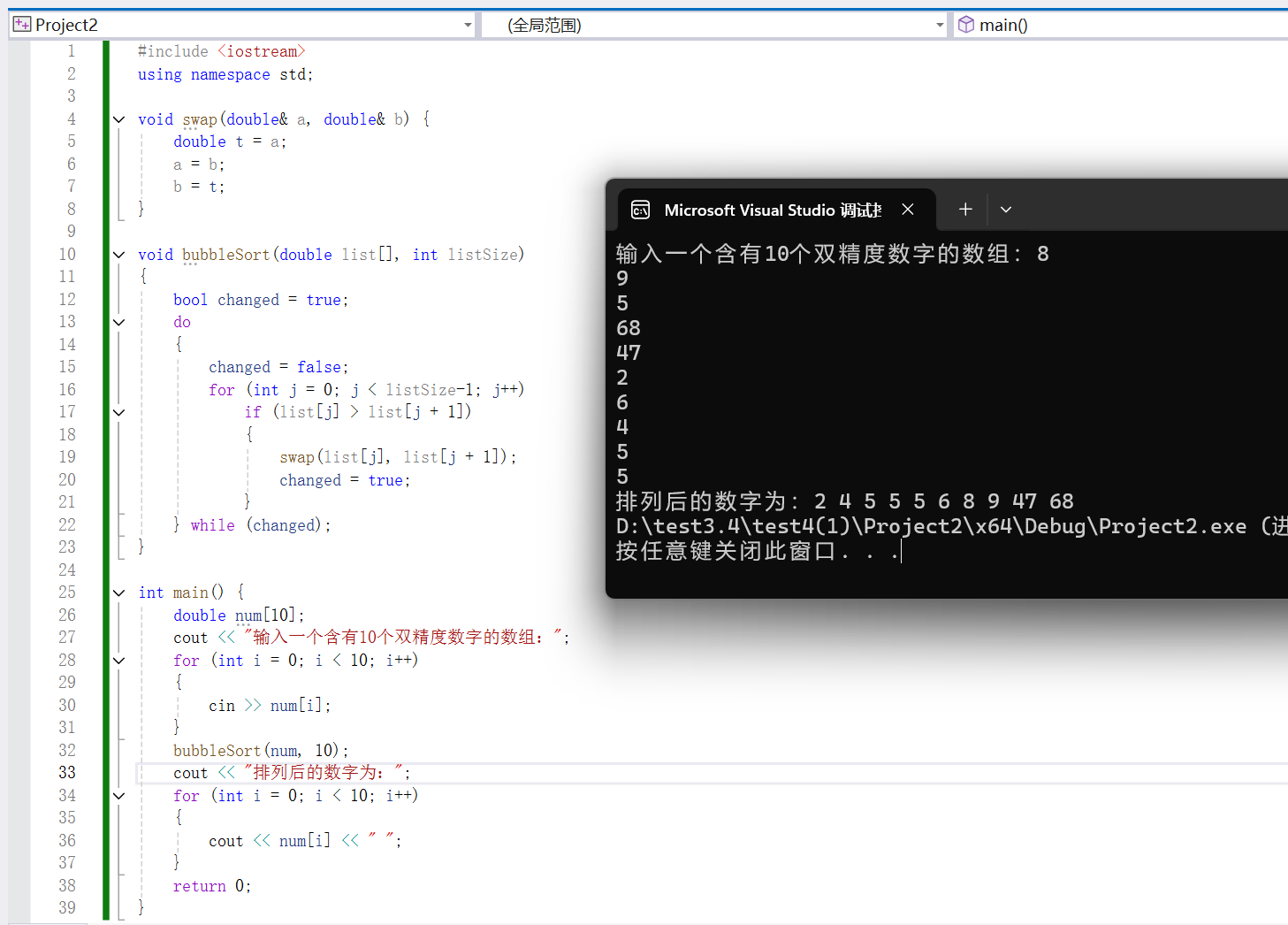
【完成实验报告】

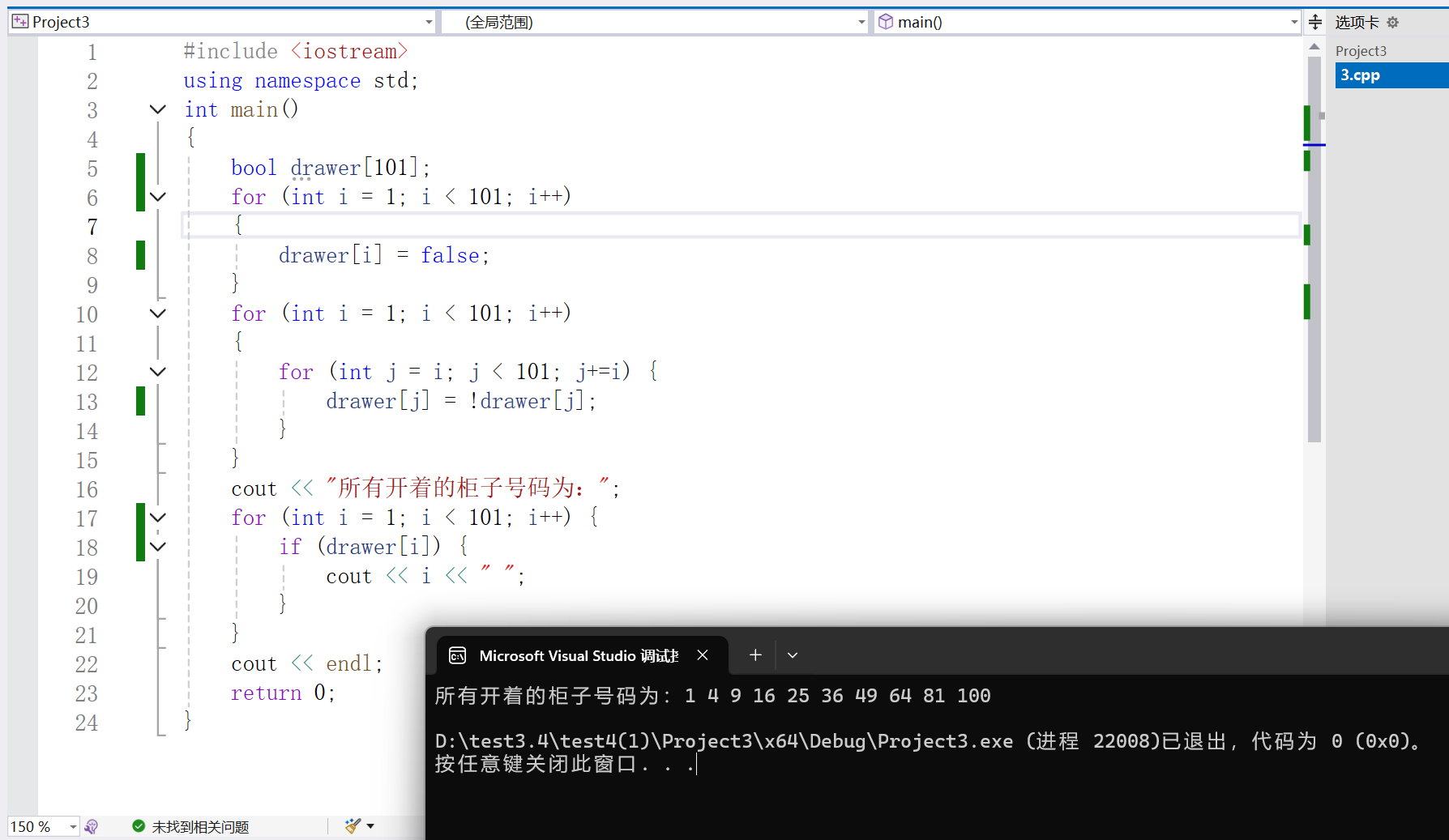
**实验报告只要求写程序设计部分**

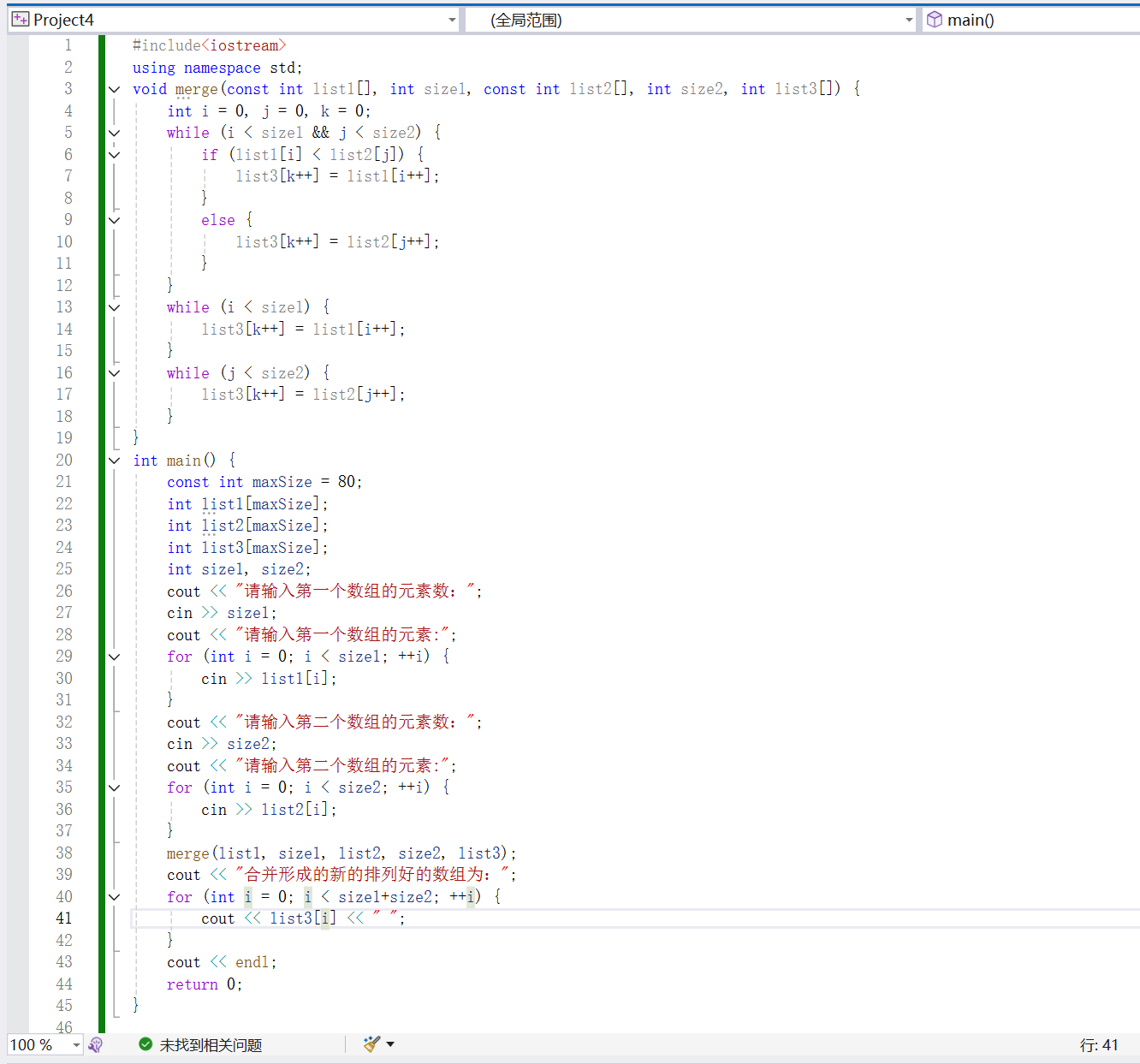
**三、算法分析，程序结果**

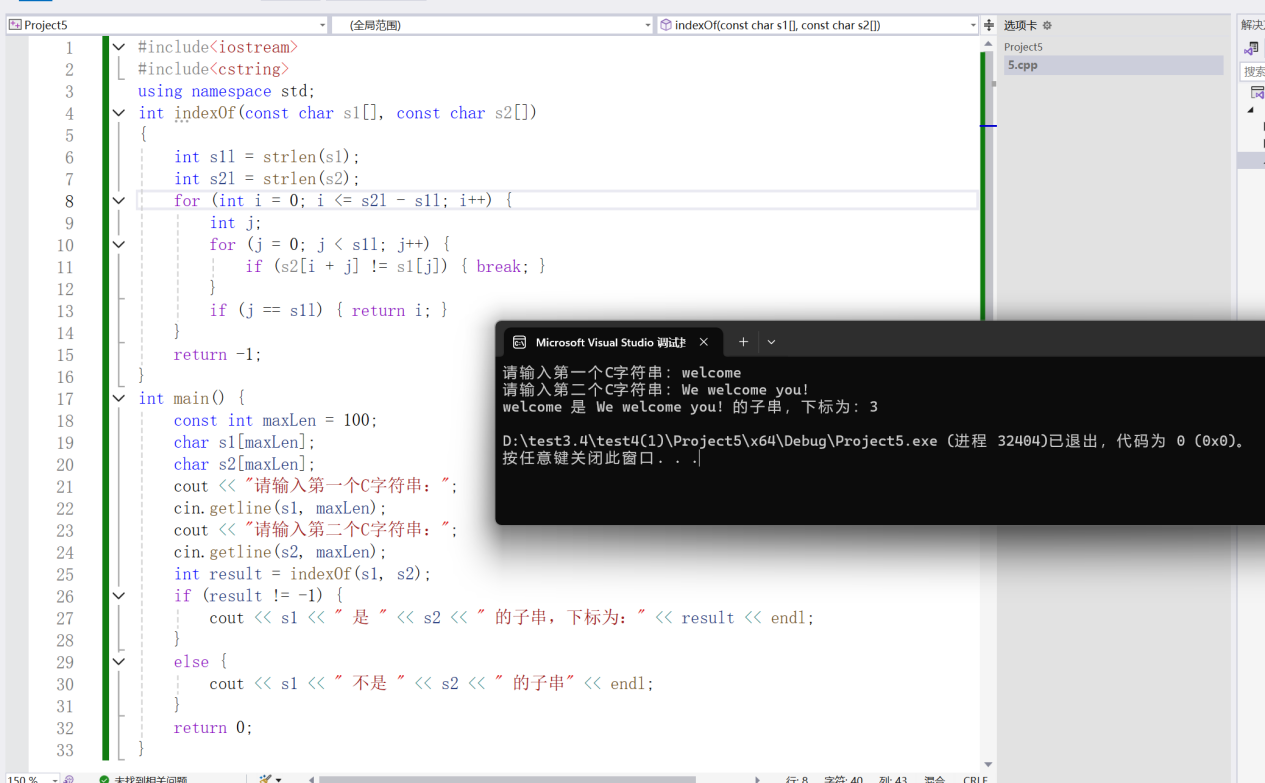
**1.1**

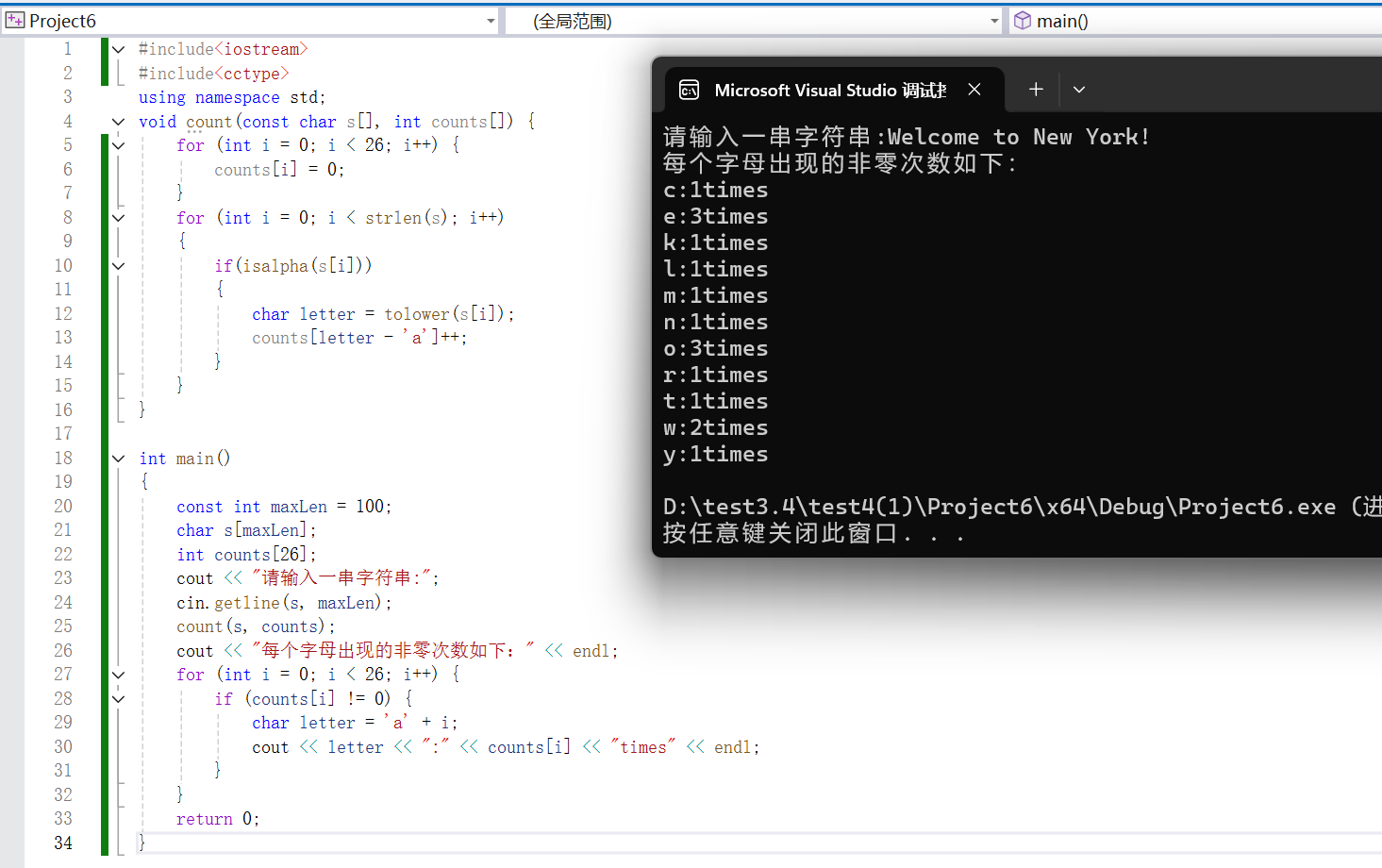
****

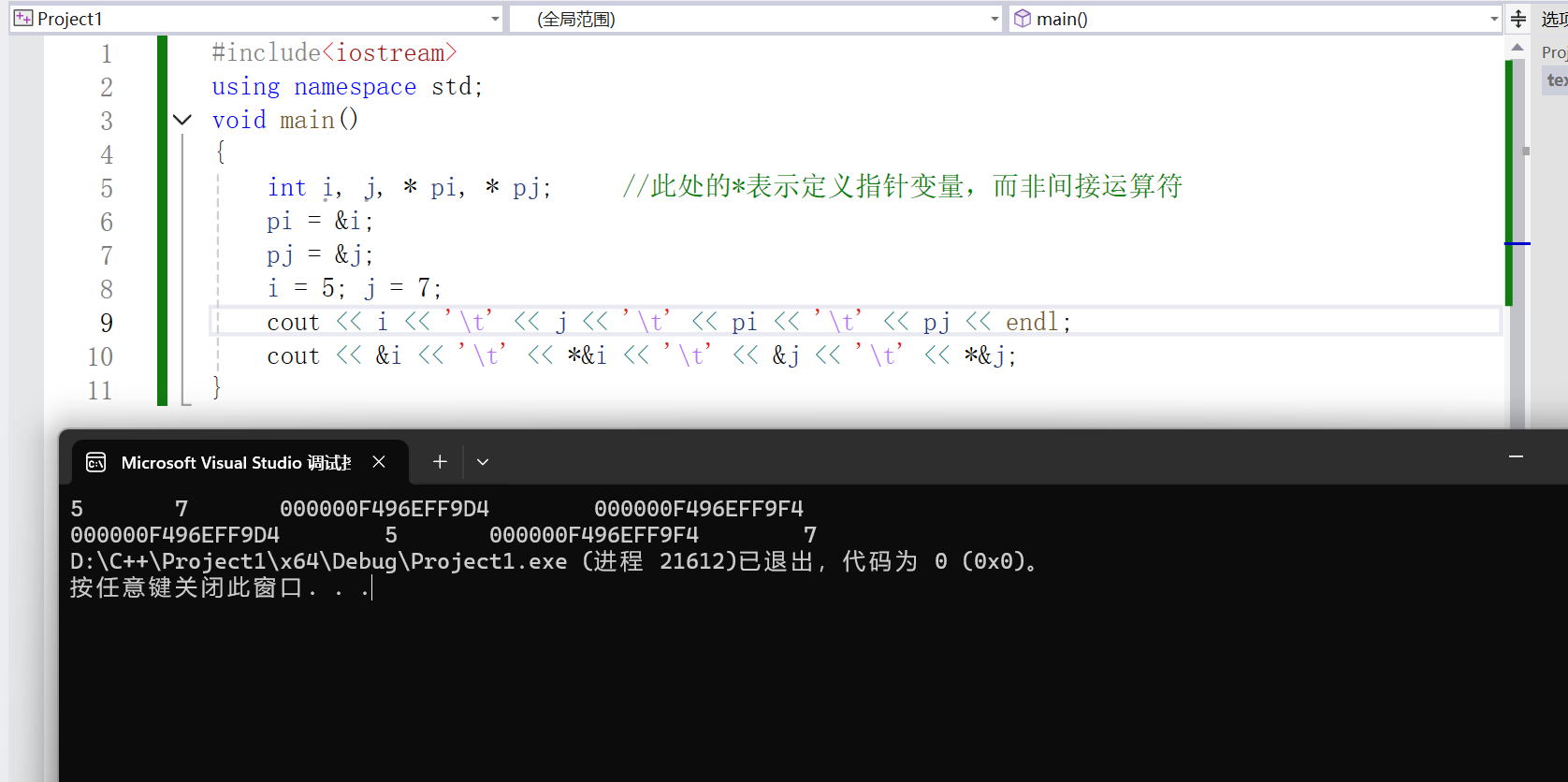
**1.2**

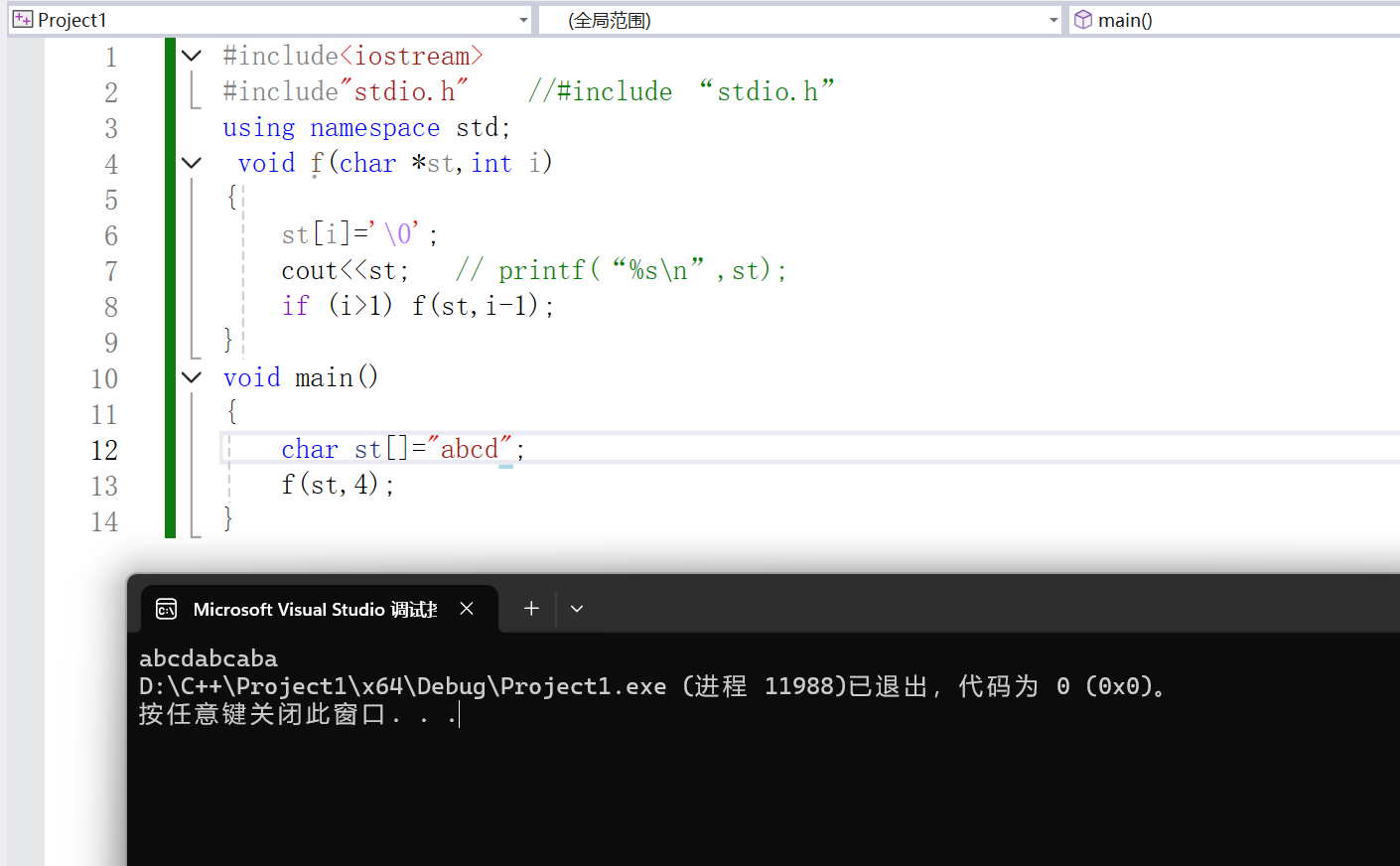
**1.3**

**1.4**

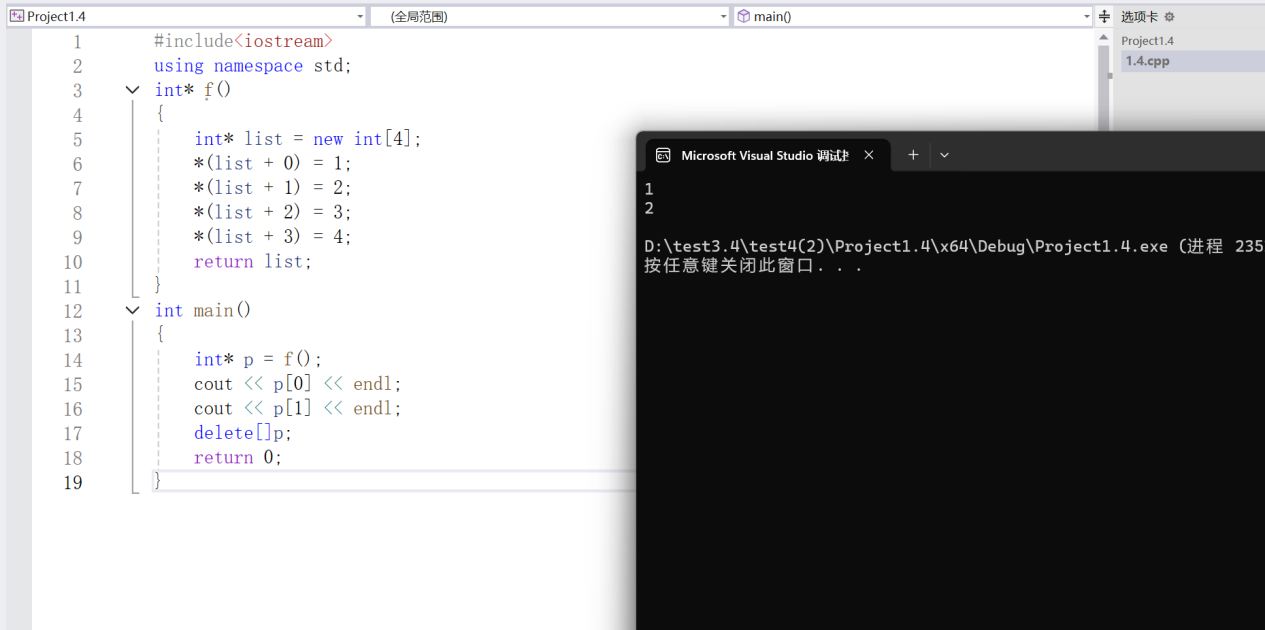
**1.5**

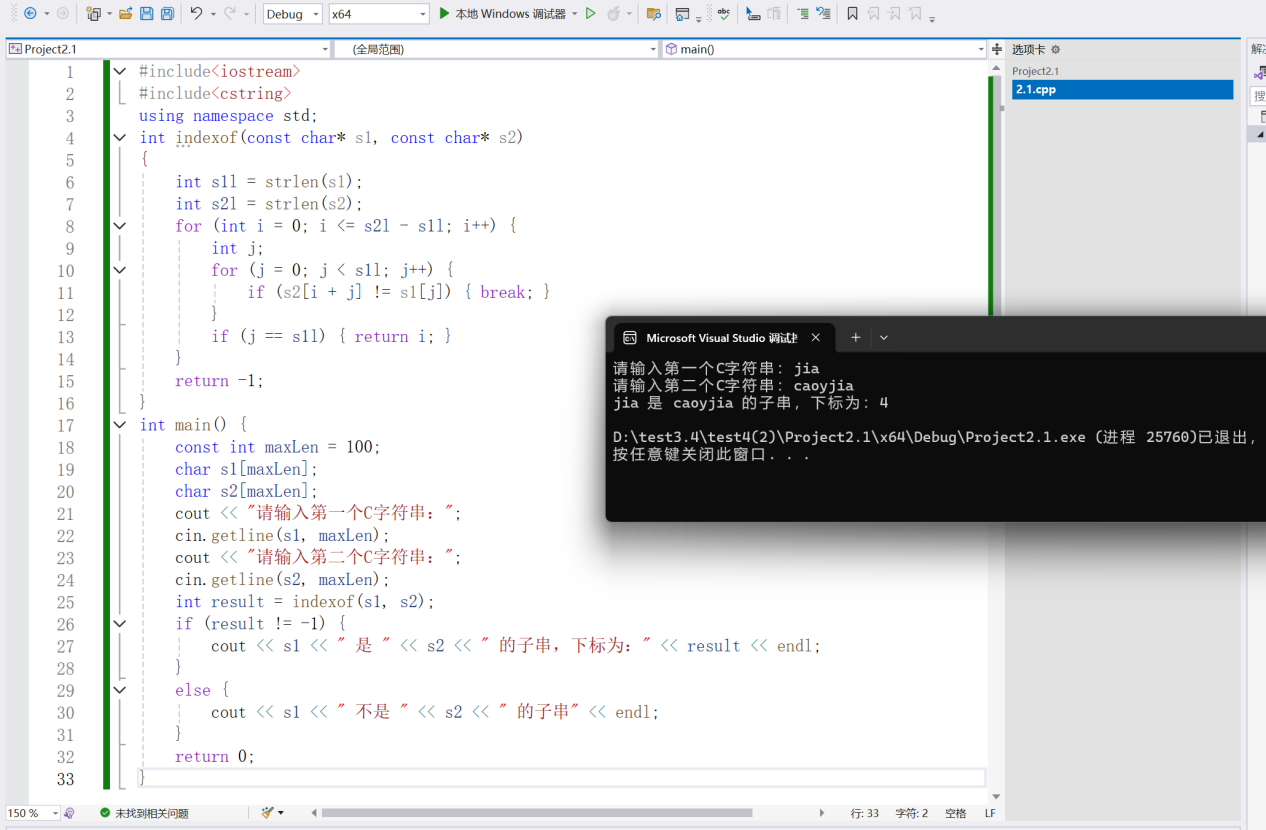
**1.6**

**2.1.1**

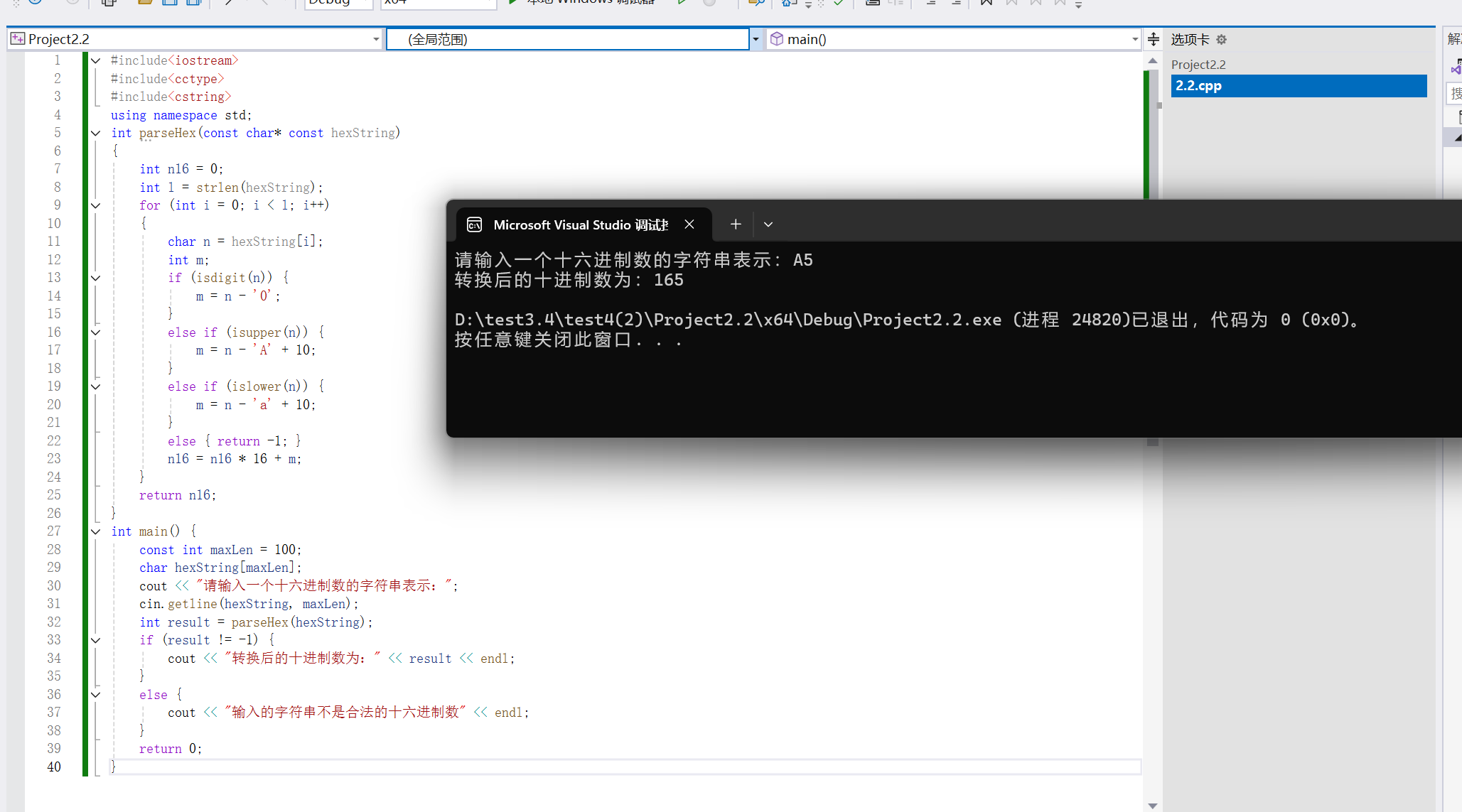
**2.1.3**

**2.1.4**

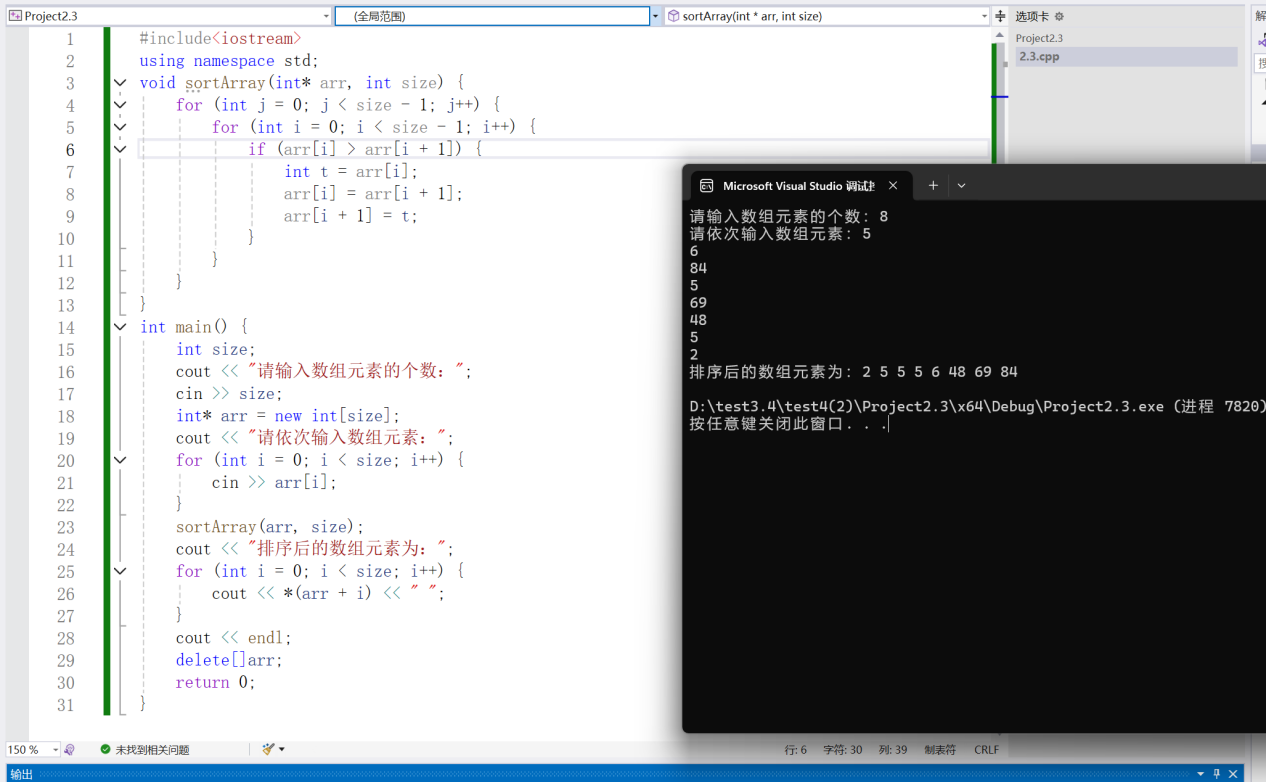
****

**2.2.1**

**2.2.2**

****

**2.2.3**

****

**四、遇到的问题与解决方法**

1. 问题：在编写打印不同数的程序时，一开始判断新值存入数组的逻辑有误，导致有些重复数字还是被存入了数组。

解决方法：仔细梳理判断流程，通过设置标志变量来记录数字是否已存在于数组中，遍历已存数字进行比对，修正逻辑后成功实现只保留不同数字。

2. 问题：在编写检验子串函数  indexOf  时，字符串遍历匹配逻辑复杂，出现漏检及下标计算错误。

解决方法：参考经典字符串匹配算法，采用双指针分别遍历两个字符串，严谨记录匹配位置，反复检查指针移动及下标赋值语句，修复漏洞得到正确结果。

3. 问题：用指针实现十六进制转十进制函数  parseHex  时，对十六进制字符与数值转换关系处理不当，导致转换结果错误。

解决方法：梳理十六进制位权规则，构建字符 - 数值映射表，利用指针遍历字符串时按位计算累加，纠正转换算法，成功得到准确十进制数。

**五、体会**

这次实验对数组和指针的理解与运用达到了新高度。数组操作里，从去重、排序到合并数组等任务，让我深刻领会到灵活运用数组索引、循环结构来处理批量数据的精妙之处，也明白了精准把控边界条件是程序正确性的关键。排序算法不再是抽象理论，亲手实现起泡排序让我洞悉其逐步优化数据顺序的过程，对算法效率有了直观认知。

指针部分极具挑战却也收获满满。初始理解指针存储地址及间接访问数据时迷雾重重，但随着不断实践，逐渐明晰其与数组千丝万缕的联系，像  a[i]  与  \*(a + i)  的等价关系打开了新视野，感受到代码简洁高效之美。动态内存分配更是拓展了编程边界， new  和  delete  的熟练运用赋予程序灵活掌控资源的能力，不过也深知内存管理稍有不慎便会陷入隐患泥潭，培养了严谨编程习惯。

函数编写贯穿始终，从子串检验到进制转换，参数传递、返回值设计与功能实现环环相扣，每一次成功运行都是思维缜密组织的成果，极大提升了模块化编程思维，深切体会到将复杂问题拆解为函数模块逐步击破的高效性，为今后大型程序开发筑牢根基，编程信心也随之水涨船高。