**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软件工程2301

学 号： 8209230105

姓 名： 袁嘉乐

**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;

int maxgys(int& a, int& b,int& n)

{

n = 1;

for (n = min(a, b); (a > 0 || b > 0); n--)

{

if ((a == 0 || b == 0))

{

break;

}

if (a % n == 0 && b % n == 0)

{

break;

}

}//找最大公约数

return 0;

}

int maxandmin(int& a, int& b,int& n)//在函数中找最大公约数和最小公倍数

{

for (n = max(a, b); (a > 0 || b > 0); n++)

{

if ((a == 0 || b == 0))

{

break;

}

if (n % a == 0 && n % b == 0)

{

break;

}

}//找最小公倍数

return 0;

}

int main()

{

cout << "请输入两个自然数：" ;

int m, n, x;

cin >> m >> n;

maxgys(m, n, x);

if ((m< 0 || n < 0))

{

cout << "您输入的数值不是正整数" << endl;

}//输出两个数字不为自然数的情况

else

{

cout << "最大公约数为：" << x << endl;

}//输出大公约数

maxandmin(m, n, x);

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

{

cout << "两个数没有最小公倍数" << endl;

}

else if (x % m == 0 && x % n == 0)

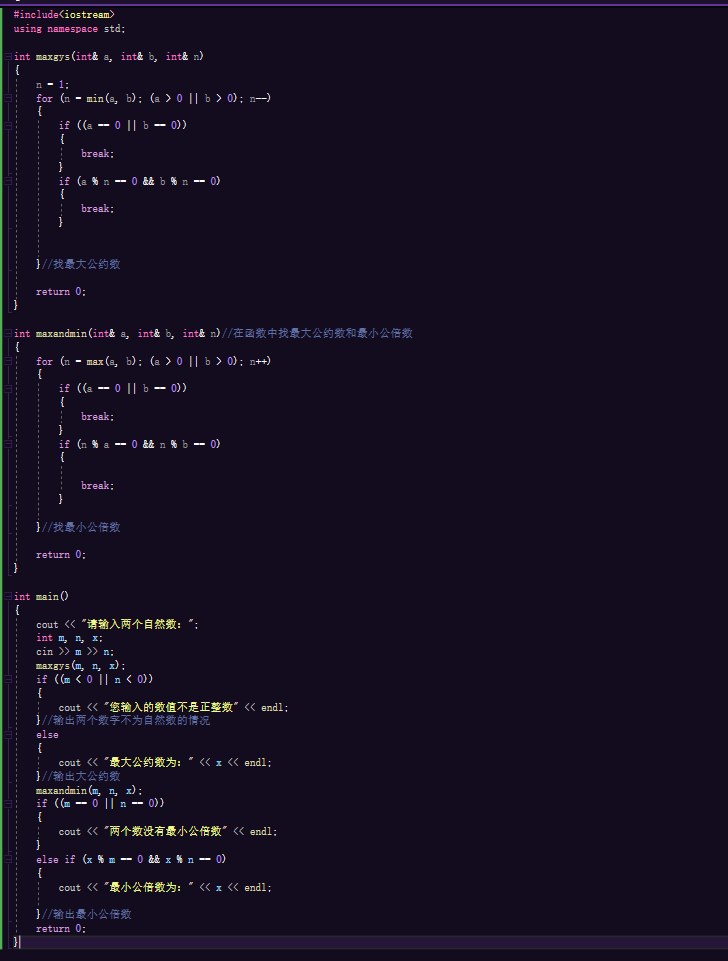
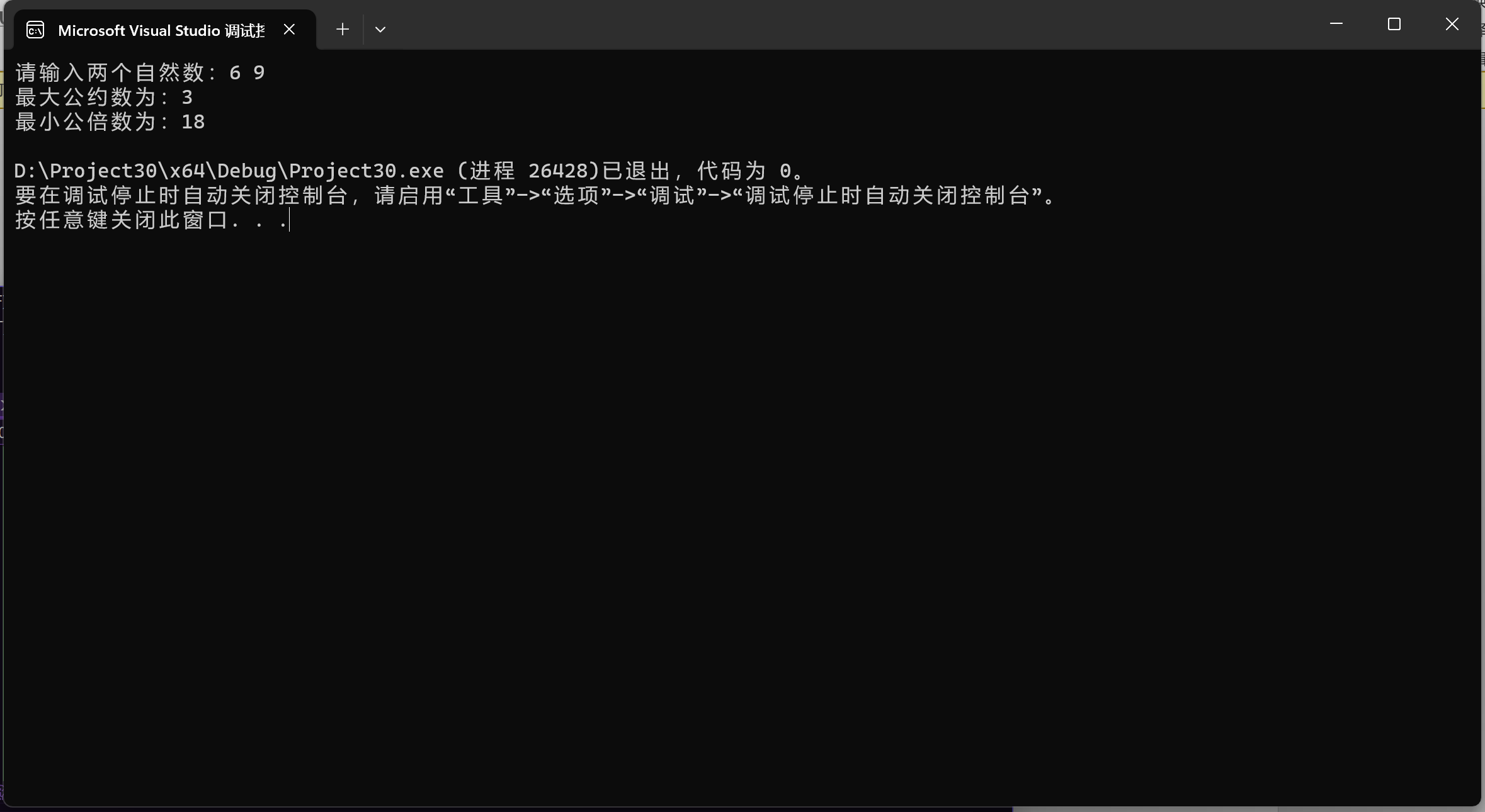
{

cout << "最小公倍数为：" << x << endl;

}//输出最小公倍数

return 0;

}

**2.**

#include<iostream>

using namespace std;

bool is\_prime(int num);//声名判断素数的函数

int main()

{

cout << "请输入一个整数：";

int a ;

cin >> a;

if (is\_prime(a) == 1)

{

cout << "是素数"<<endl;

}

else

{

cout << "不是素数"<<endl;

}//判断并输出是否为素数

cout << "前两百个素数为：" << endl;

int b = 2;

for (int m = 1; m <= 200; m++,b++)

{

for (;; b++)

{

if (is\_prime(b) == 1)

{

cout << b<<'\t';

break;

}

}

if (m % 10 == 0)

{

cout << endl;

}

}

return 0;

}//寻找前200个素数

bool is\_prime(int num)//定义判断素数的函数

{

int n = 0;

for (n = num - 1; n > 0; n--)

{

if (num % n == 0)

{

break;

}

}

if (n == 1)

{

return 1;//返回为真

}

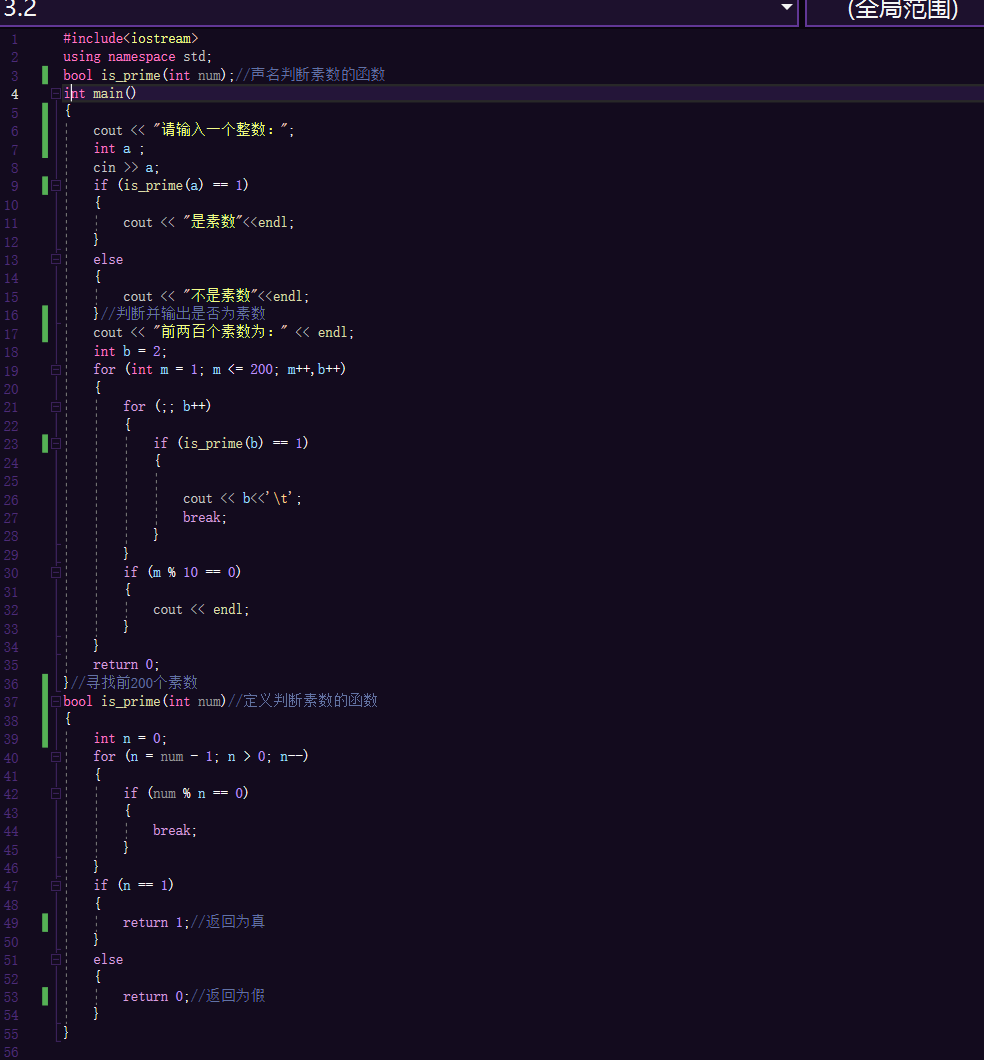
else

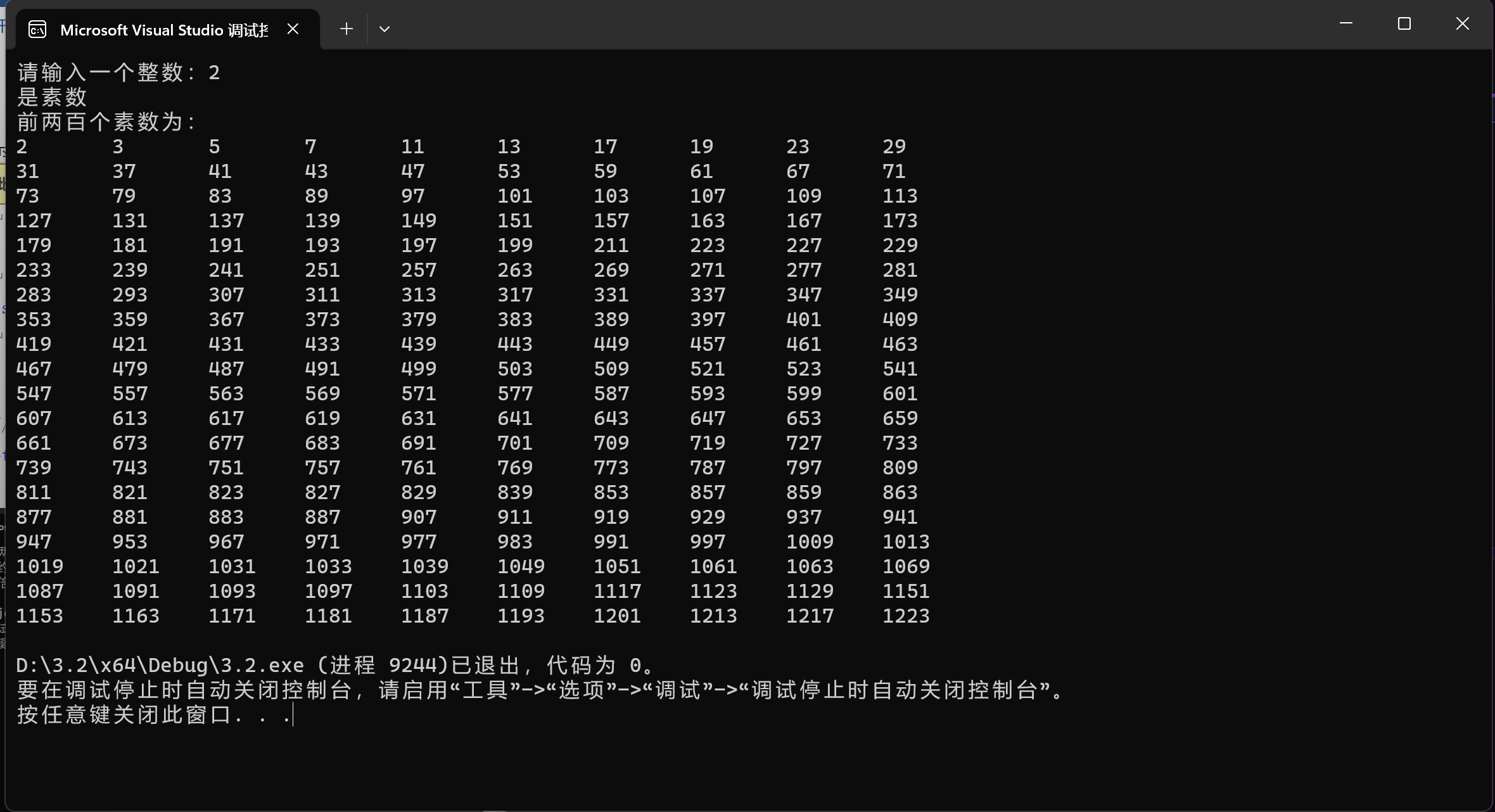
{

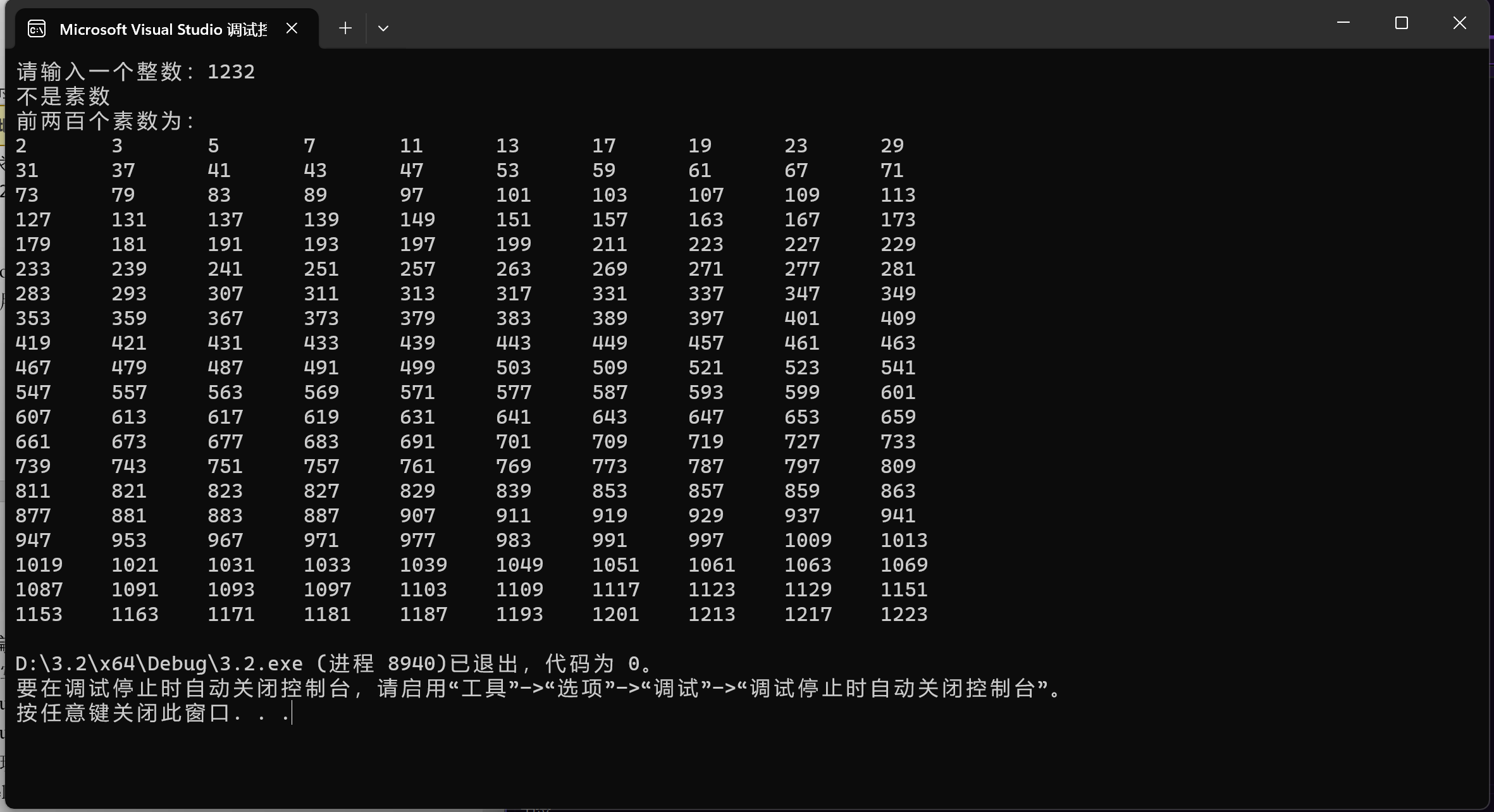
return 0;//返回为假

}

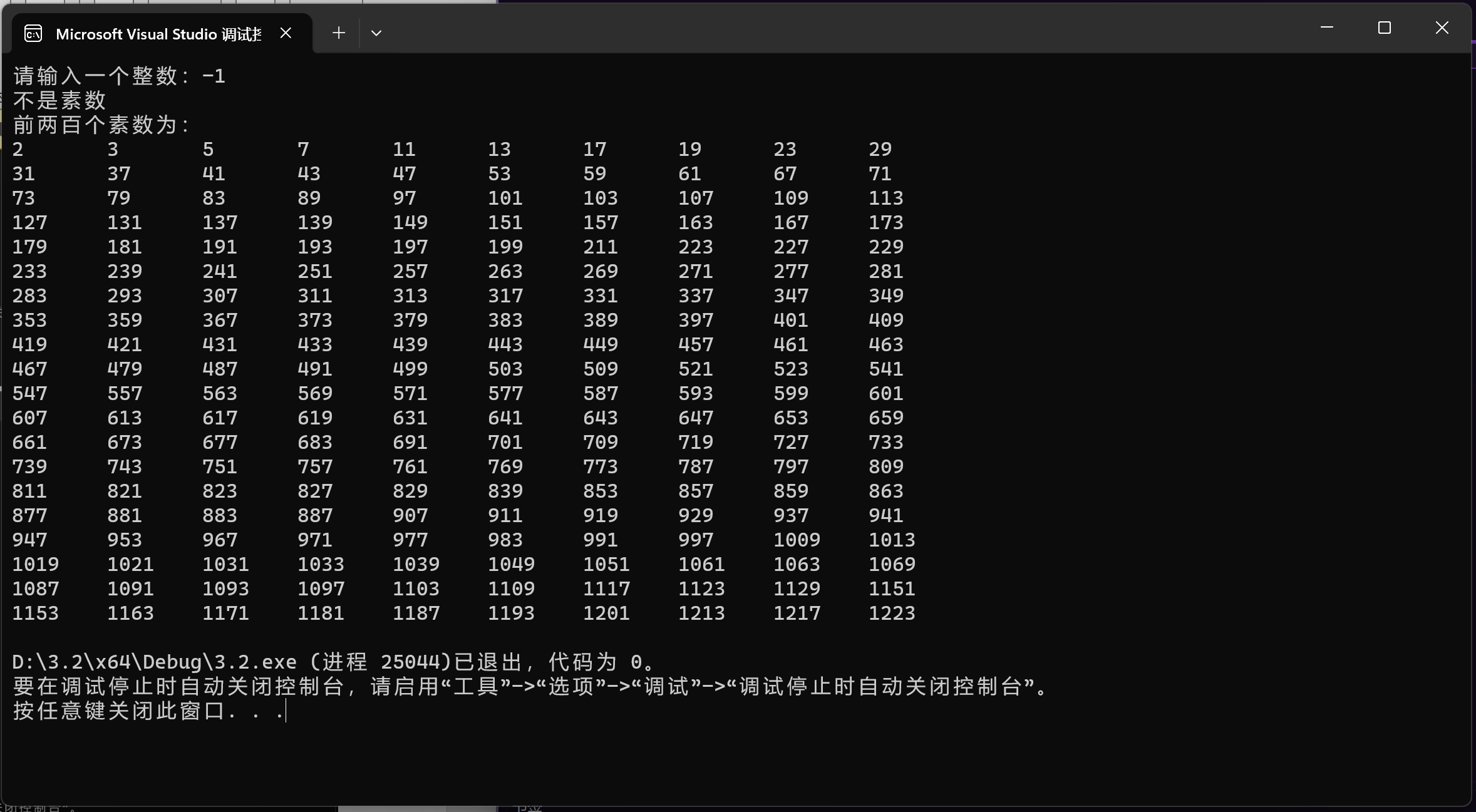
}





****

**这是输入负数的情况：**

****

**3.**

头文件部分：

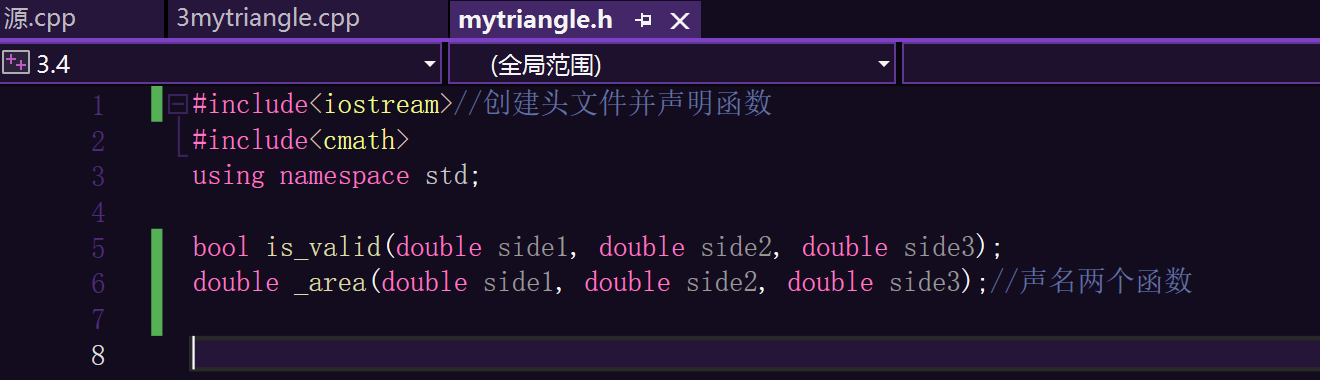
#include<iostream>//创建头文件并声明函数

#include<cmath>

using namespace std;

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

double \_area(double side1, double side2, double side3);//声名两个函数



**定义函数的源文件部分：**

#include<iostream>

#include<cmath>

using namespace std;

//定义函数

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

{

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

{

return 1;

}

else

{

cout << "无法构成三角形";

return 0;

}

}

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

{

double s, area;

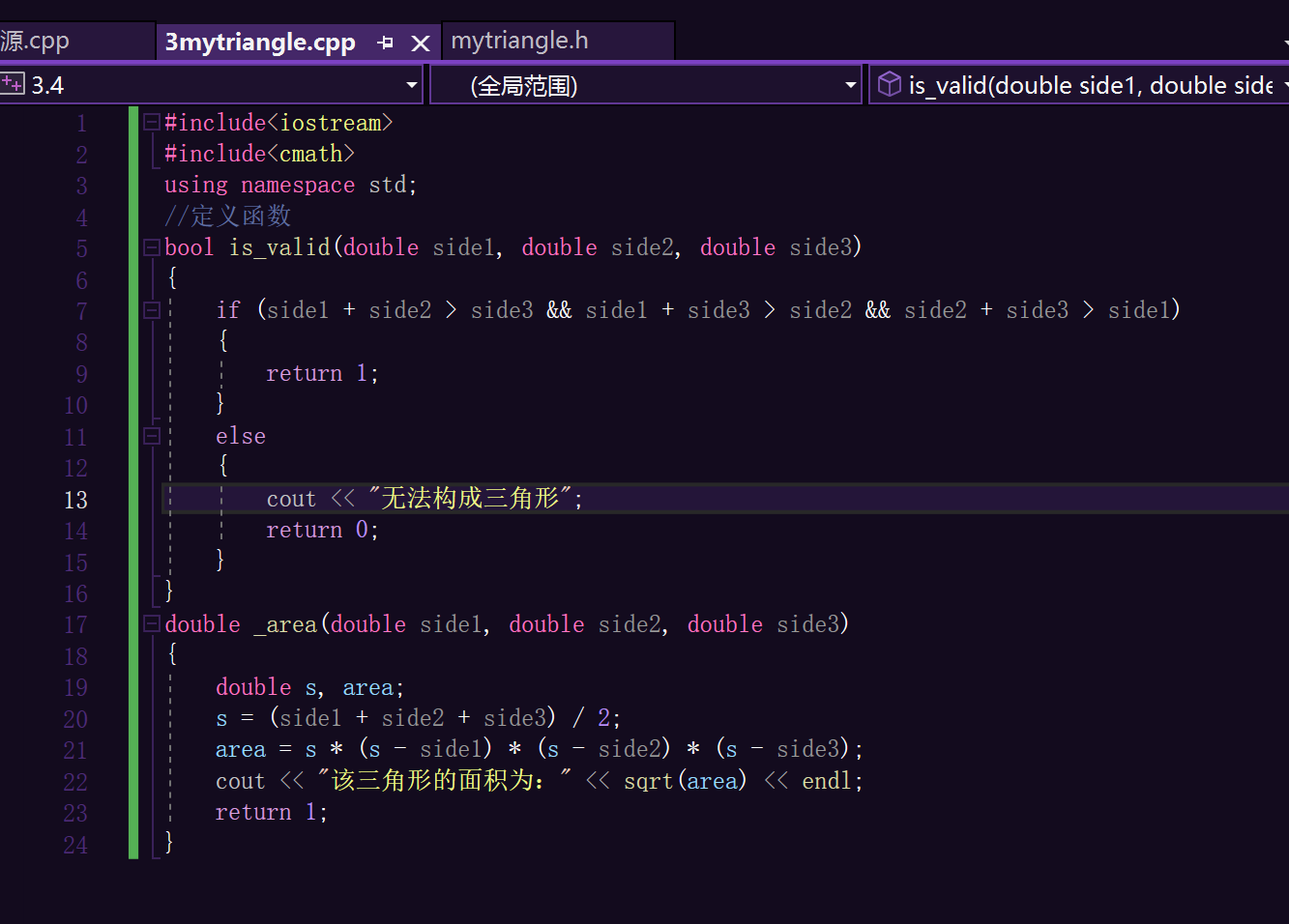
s = (side1 + side2 + side3) / 2;

area = s \* (s - side1) \* (s - side2) \* (s - side3);

cout << "该三角形的面积为：" << sqrt(area) << endl;

return 1;

}

****

**测试程序所在模块：**

#include<iostream>

#include"mytriangle.h"//引用头文件

using namespace std;

int main()

{

cout << "请输入三角形的三条边长：";

int a, b, c;

cin >> a >> b >> c;

if (is\_valid(a, b, c) == 1)

{

\_area(a, b, c);//引用函数

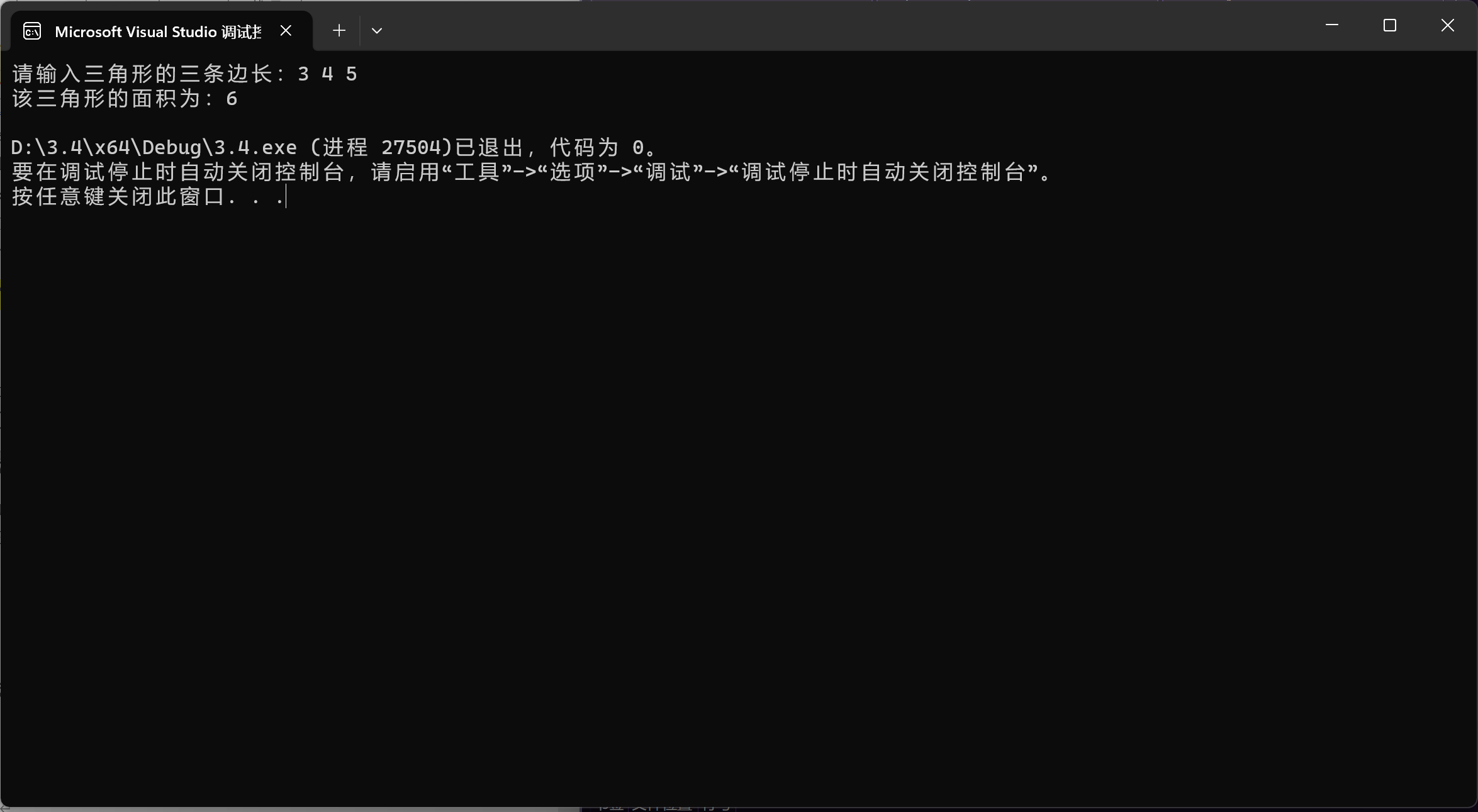
}

return 0;

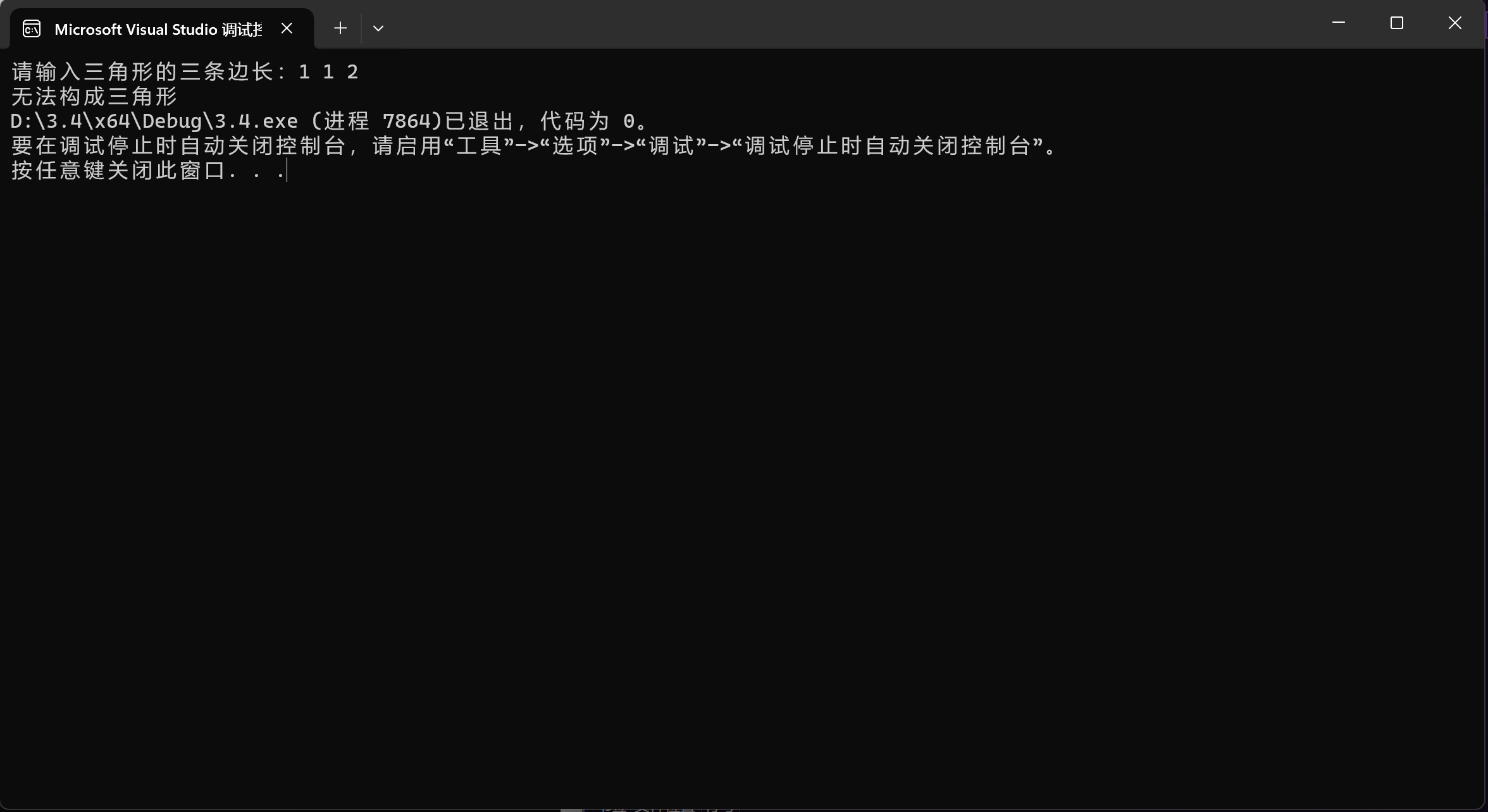
}

****

**输入合法输出三角形面积：**

****

**输入不合法输出错误信息：**

****

5.

#include<iostream>

using namespace std;

int day(int& n,int& sum)

{

sum = (sum + 1) \* 2;

n++;

if (n < 10)

{

day(n, sum);//利用递归实现计算出猴子第一天共摘桃子数目

}

return 0;

}

int main()

{

int sum = 1;

int n = 1;

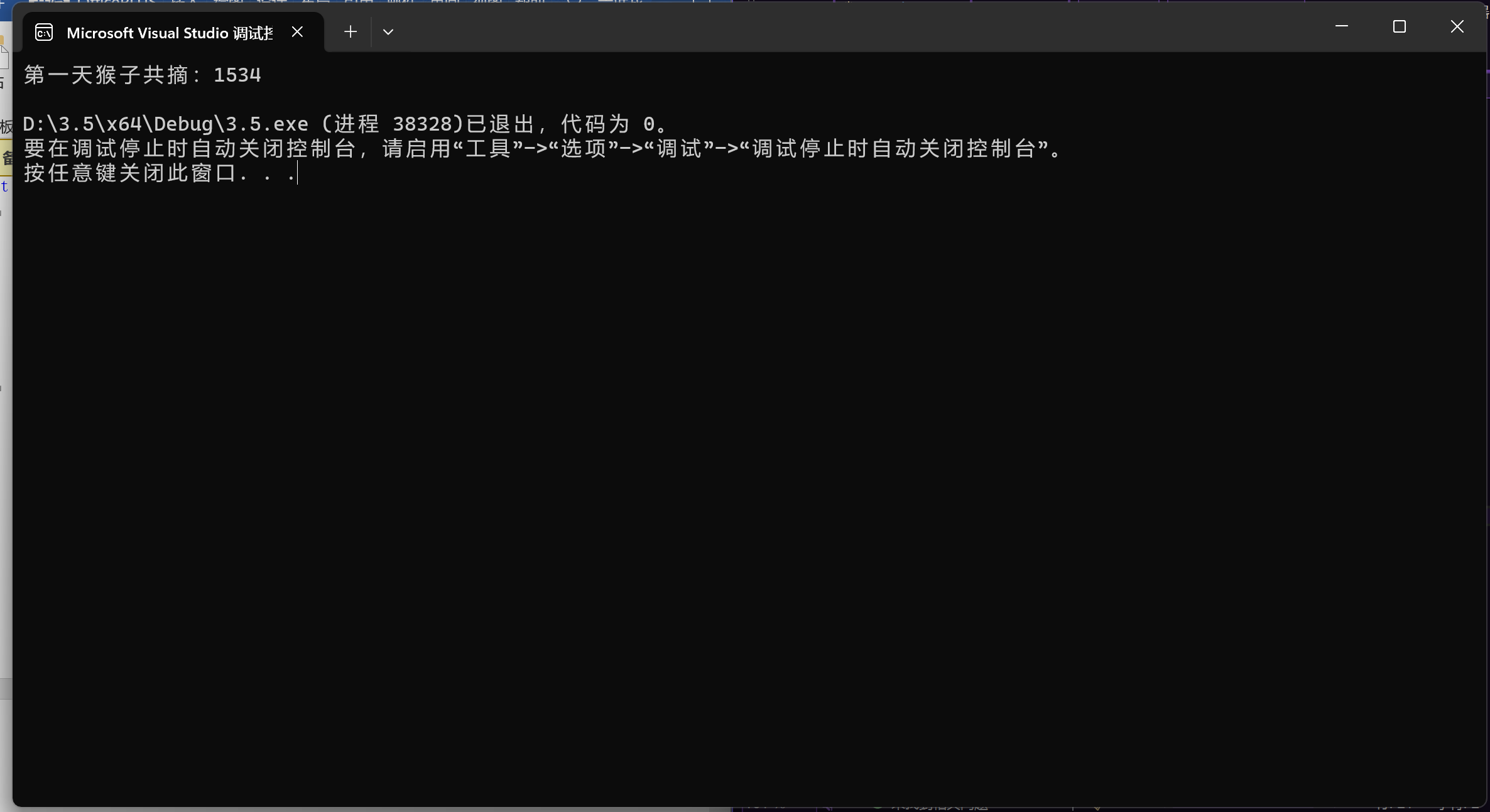
day(n, sum);

cout << "第一天猴子共摘：" << sum << endl;//调用函数输出结果

return 0;

}

****

****

**五、遇到的问题与解决方法**

**对递归函数的使用不熟练，经过挖掘课本，查找资料，我熟悉了递归函数的使用方法**

**六、体会**

**通过此次实验，我对函数的定义，声名以及调用有了更深刻的认识，同时我也学会了函数的分文件编写，对代码的简明性，可读性，以及美观性有了初步的认识，我对c++的学习更加深入，编写代码更加熟练。**

**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.

//单独拿出来一个数字，相当于垃圾桶，重复的数字都赋值为这个数字，

//输出的时候如果这个数组中的元素等于这个数字，那么便不会输出，

//如果输入了这个数字，那么只需要把这个数字输出一遍即可

#include<iostream>

using namespace std;

int main()

{

int a[10];

int n,e=0;

cout << "Enter/； ten numbers: ";

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

{

cin >> a[n];

if (a[n] == -1000)

{

e = -1000;

}//单独拿出一个数字

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

{

if (a[n] == a[i])

{

a[n] = -1000;

}

}

}//将与之前相同的数字翻进垃圾桶

cout << "The distinct numbers are:";

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

{

if (a[n] !=-1000)

{

cout << a[n] << " ";

}

}//输出出现的数字

if (e==-1000)

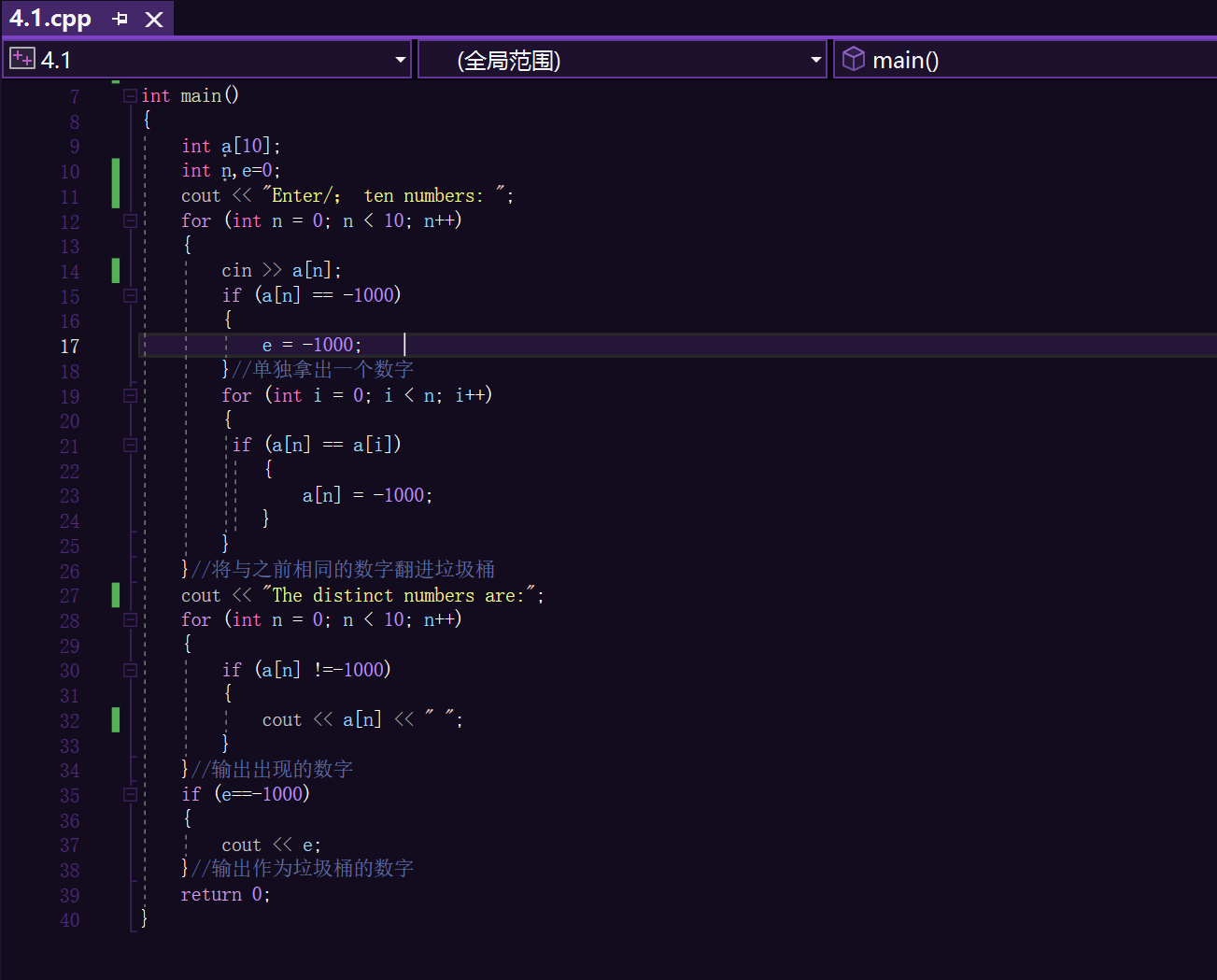
{

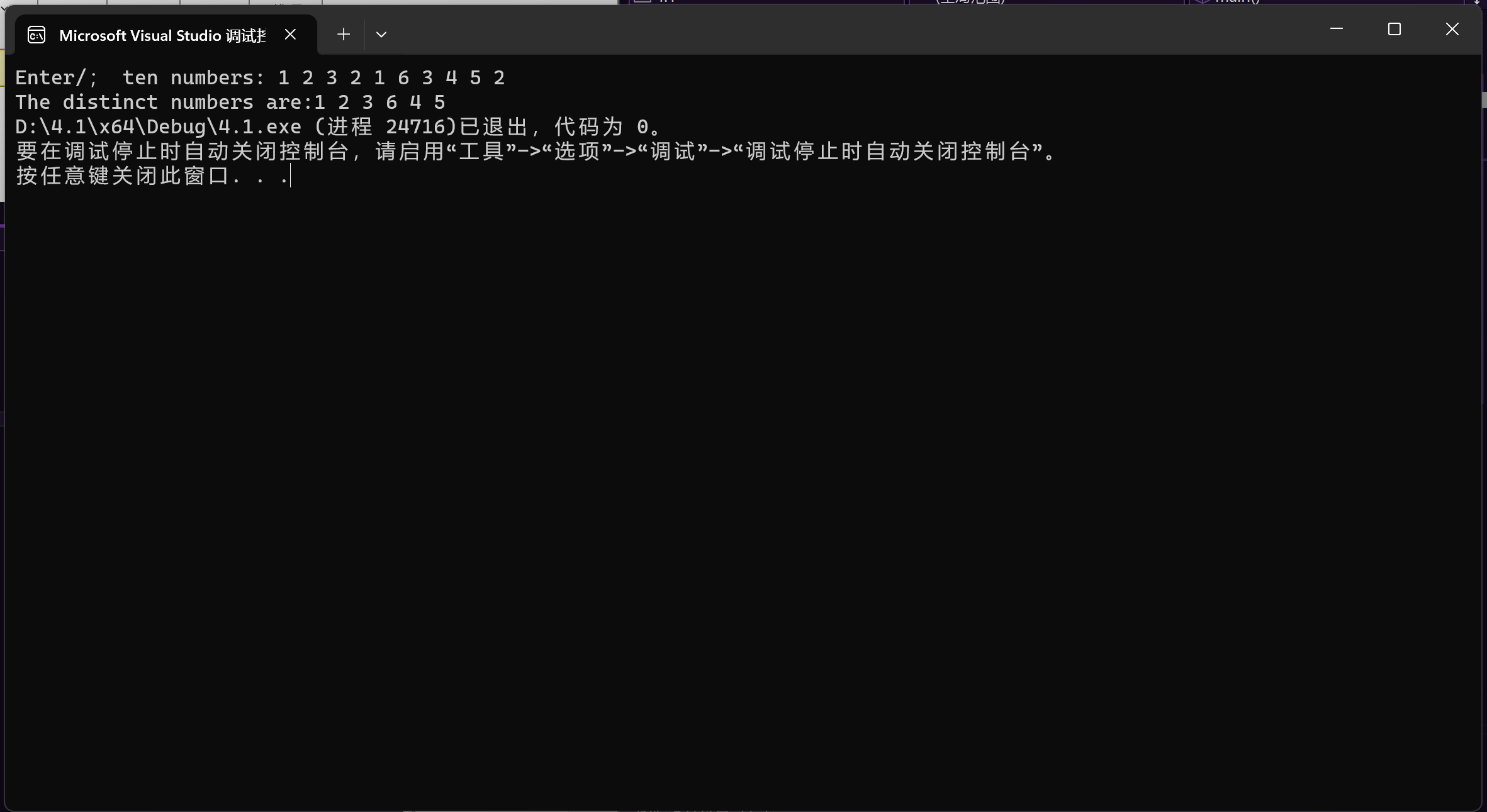
cout << e;

}//输出作为垃圾桶的数字

return 0;

}



****

**2.**

#include<iostream>

using namespace std;

double swap(double& a, double& b)//交换函数

{

double t = a;

a = b;

b = t;

return 0;

}

double bubble(double list[10])//起泡排序函数

{

bool changed = true;

do

{

changed = false;

for (int j = 0; j<10-1; j++)

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

{

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

changed = true;

}

} while (changed);

return 1.1;

}

int main()

{

cout << "请输入一个含有10个双精度数字的数组:";

double list[10];

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

{

cin >> list[n];

}

bubble (list);

cout << "该数组升序排列为：";

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

{

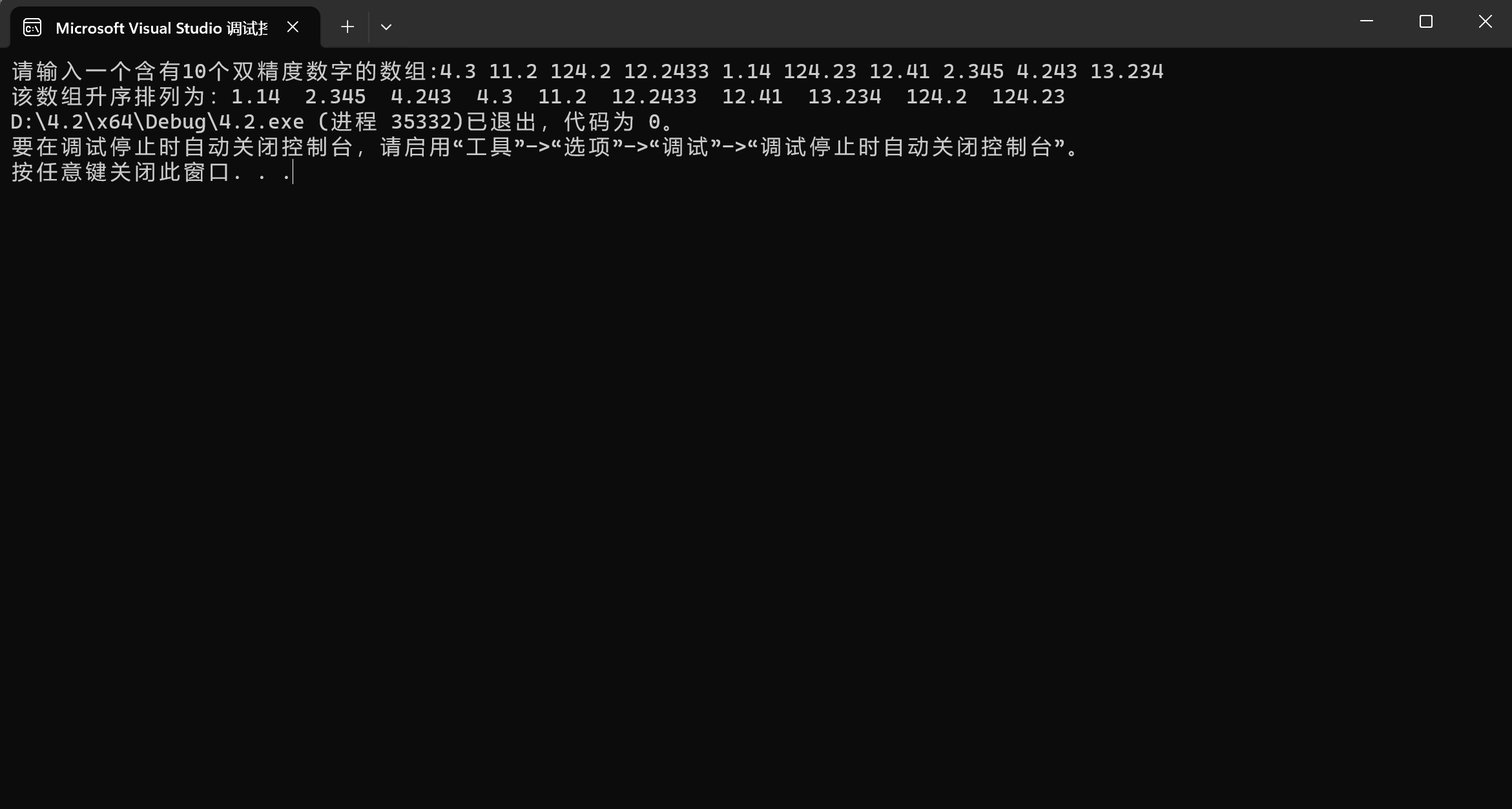
cout<< list[n]<<" ";

}//排列数字并显示

return 0;

}

****

****

**3．**

#include<iostream>

using namespace std;

int main()

{

bool a[100];

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

{

a[m] = 1;

}

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

{

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

{

if (a[n])

{

a[n] = false;

}

else

{

a[n] = true;

}

}

}//根据返回值为真算出开着的柜子序号

cout << "开着的柜子序号为：";

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

{

if(a[m])

{

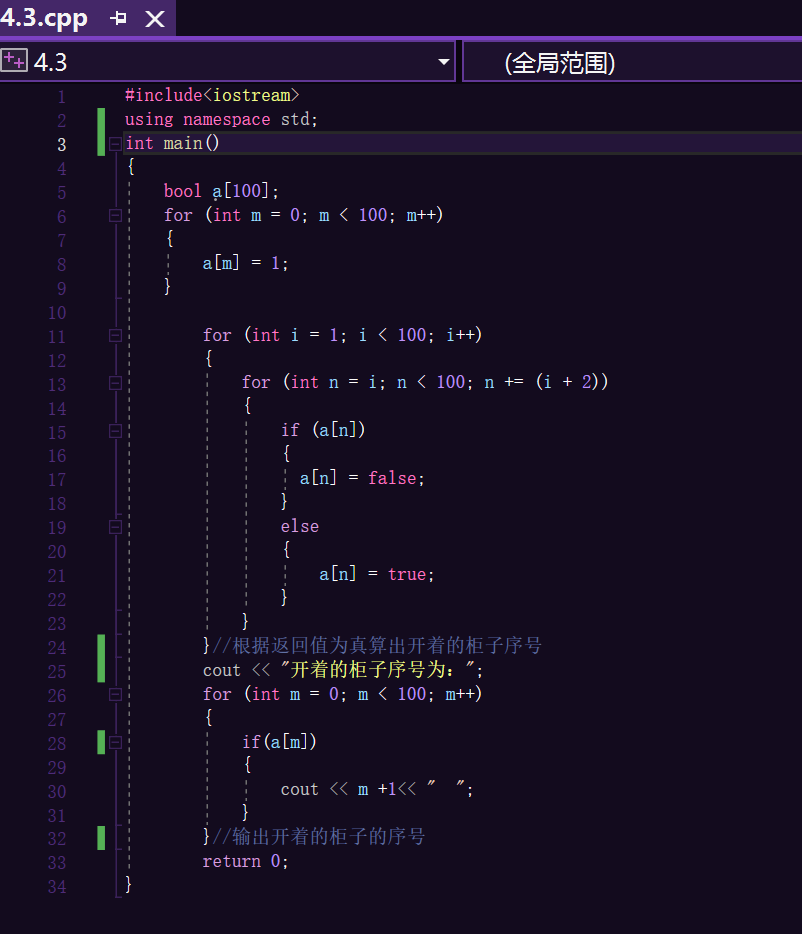
cout << m +1<< " ";

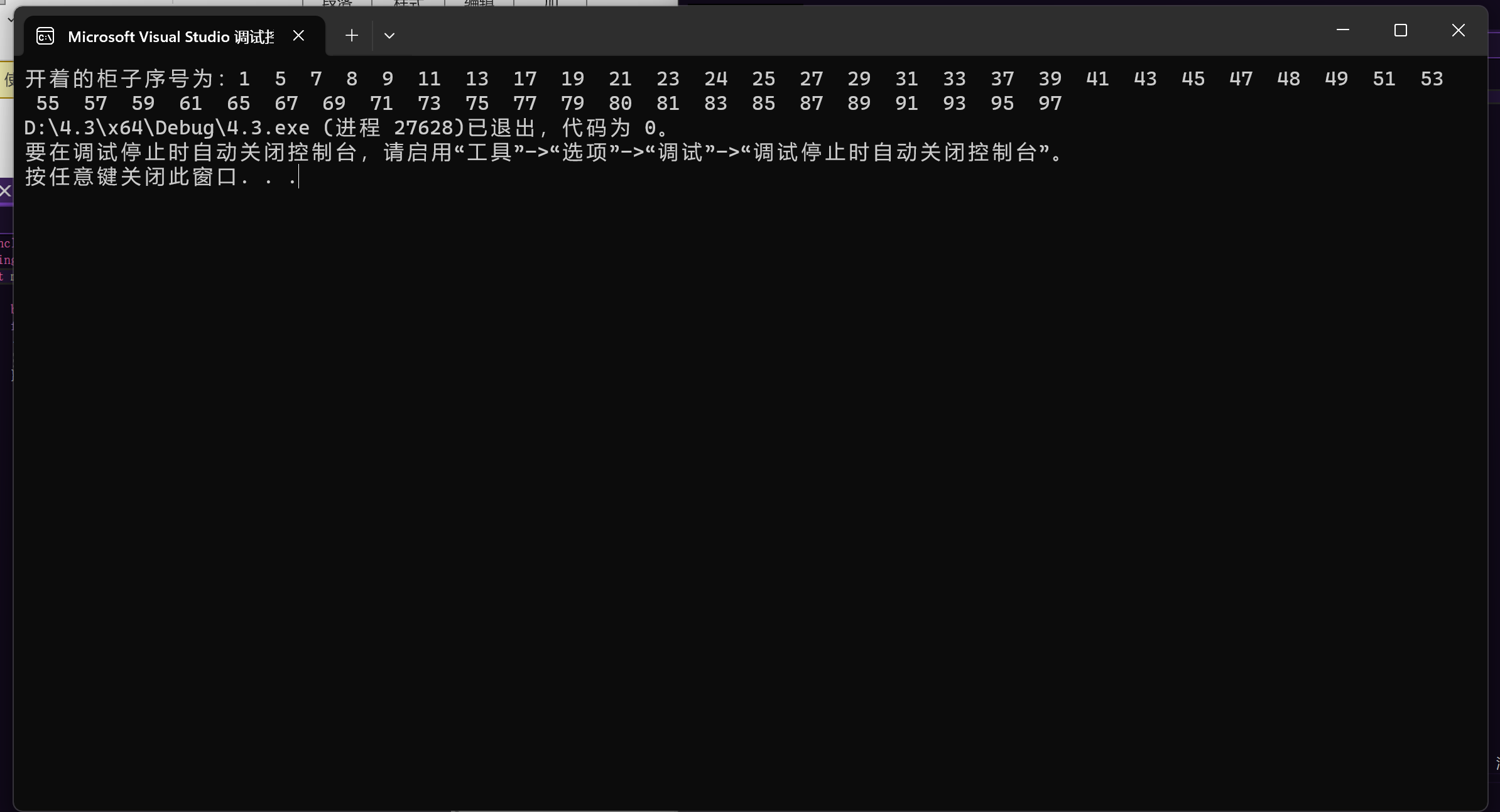
}

}//输出开着的柜子的序号

return 0;

}

****

****

**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[j]) {

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

}

else {

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

}

}

while (i < size1) {

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

}

while (j < size2) {

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

}

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

{

cout << list3[n]<<" ";

}

}//排序与合并同时进行的函数

int main()

{

cout << "Enter list1:";

int size1;

cin >> size1;

int list1[80];

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

{

cin >> list1[n];

}

cout << "Enter list2:";

int size2;

cin >> size2;

int list2[80];

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

{

cin >> list2[n];

}

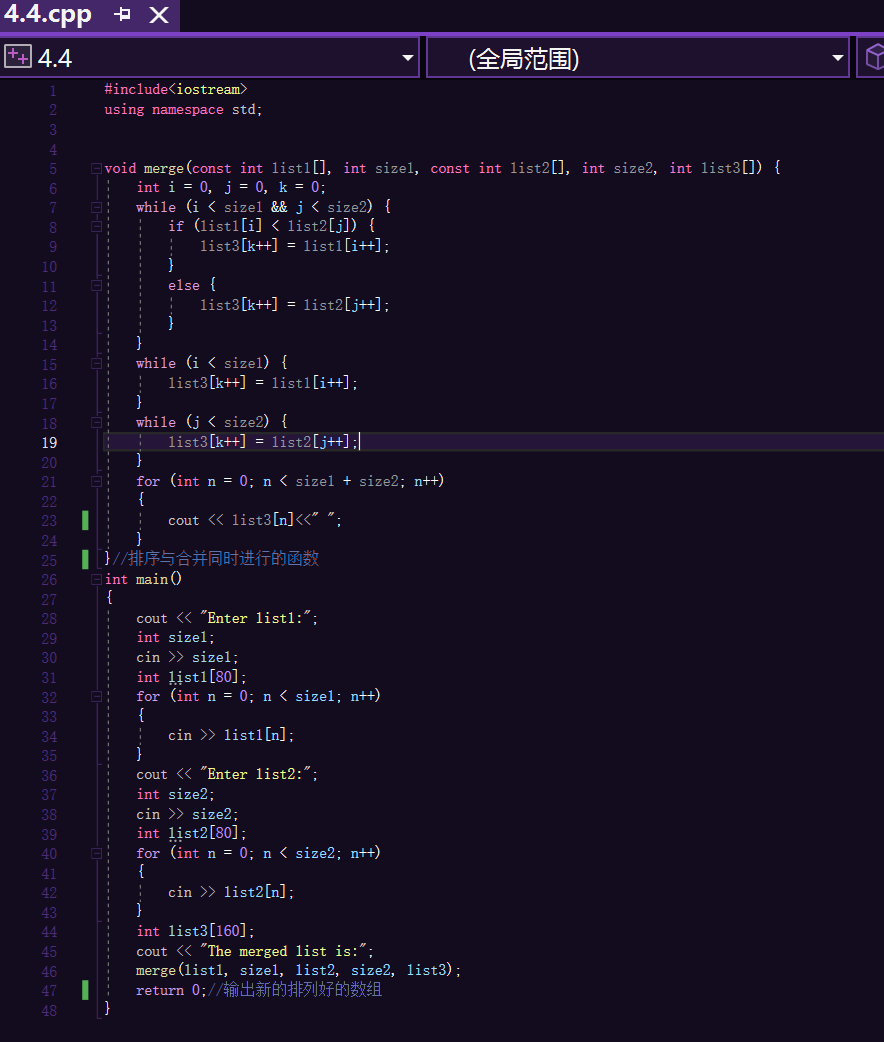
int list3[160];

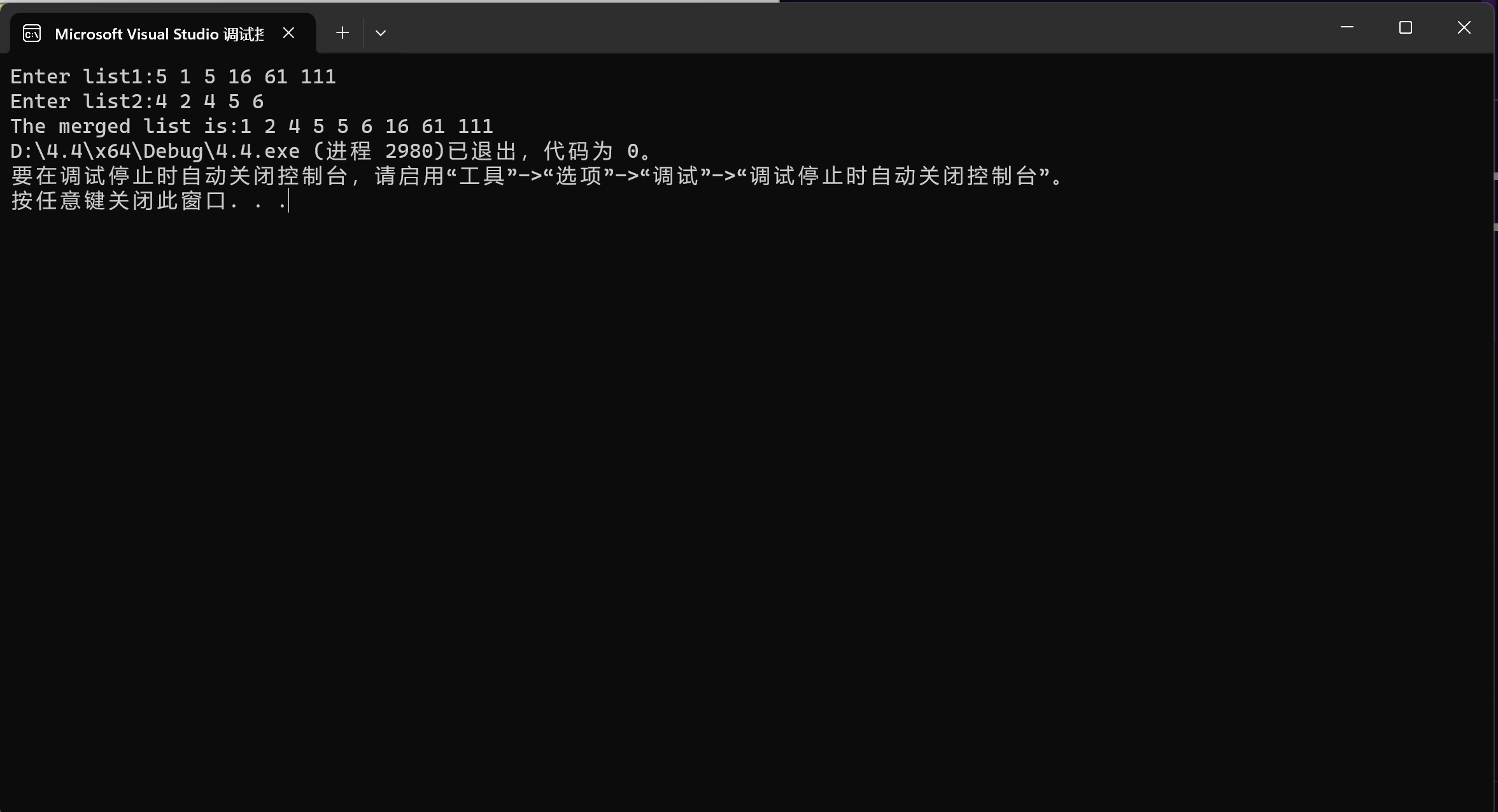
cout << "The merged list is:";

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

return 0;//输出新的排列好的数组

}



****

**5.**

#include<iostream>

using namespace std;

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

{

int length1 = strlen(s1);

int length2 = strlen(s2);

for (int n = 0,m=0; n < length2; n++)

{

if (s2[n] ==32)

{

m++;

}//计算空格数量

if (strncmp(s1, s2 + n, length1) == 0)

{

return n - m + 1;

}//检索对应的下标，减去空格的数量，得到除去空格的下标

}

return -1; //未检索到，返回-1

}

int main()

{

char s1[520], s2[520];

cout << "Enter the first string:";

cin.getline(s1,520);

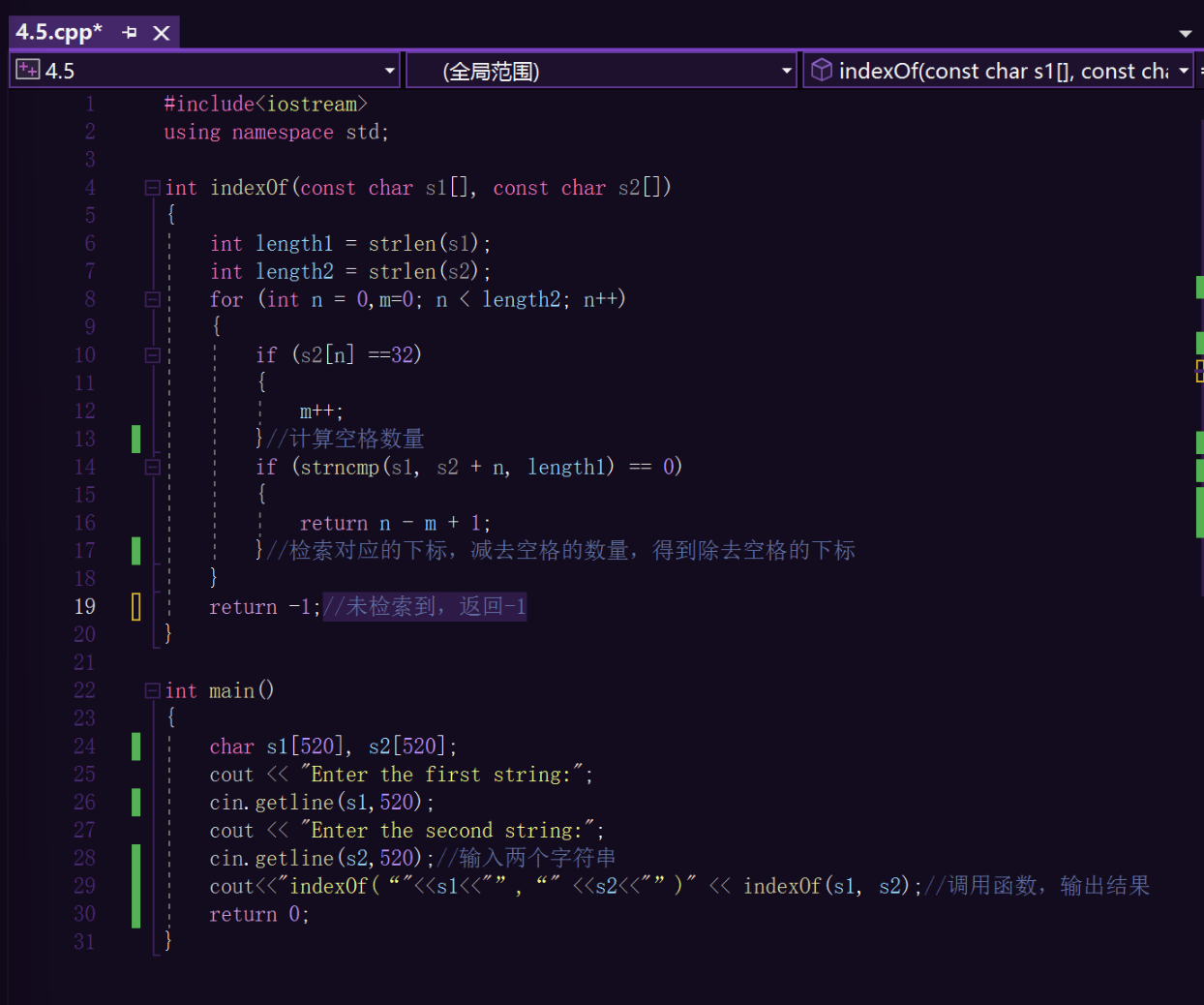
cout << "Enter the second string:";

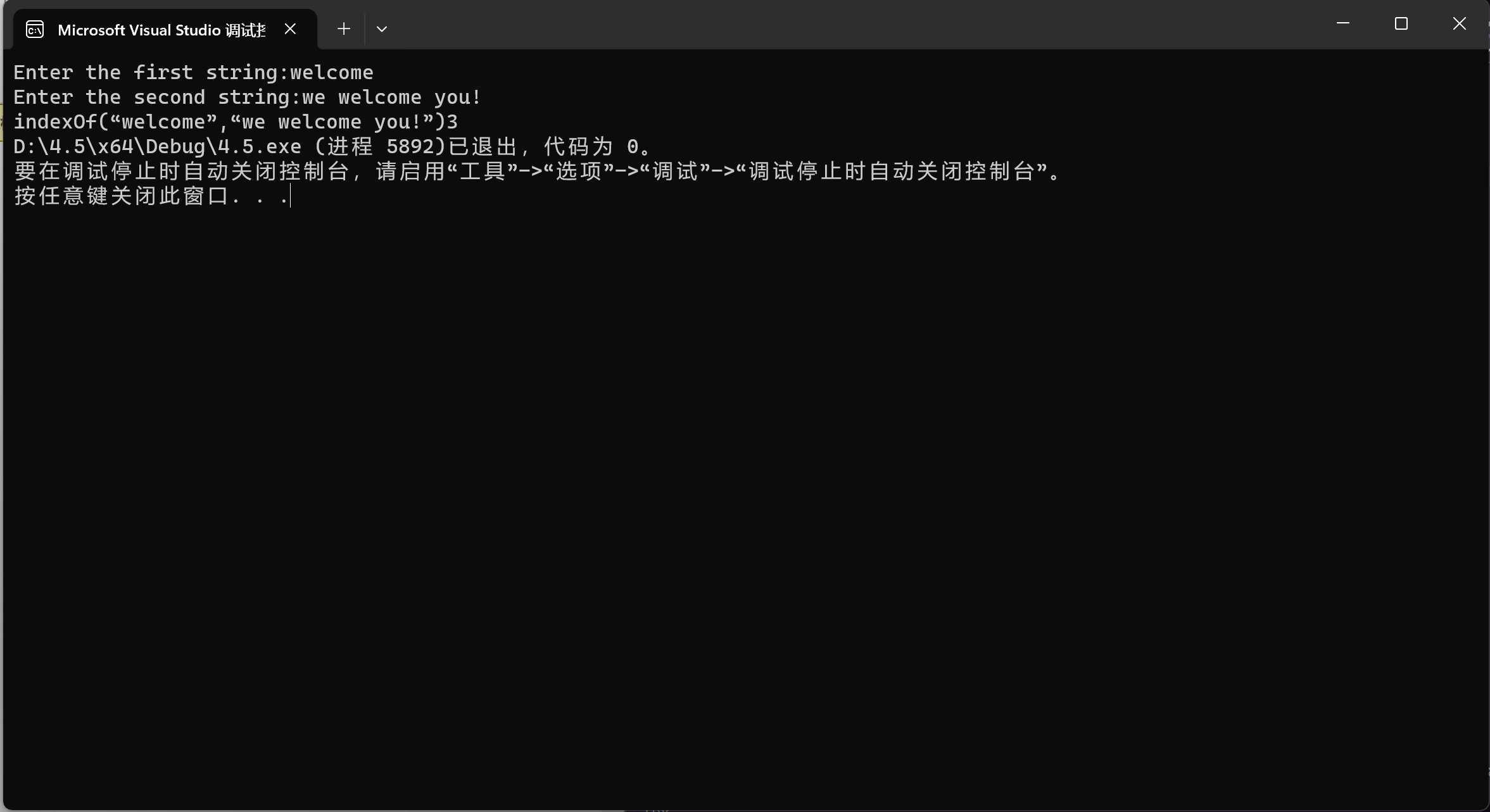
cin.getline(s2,520);//输入两个字符串

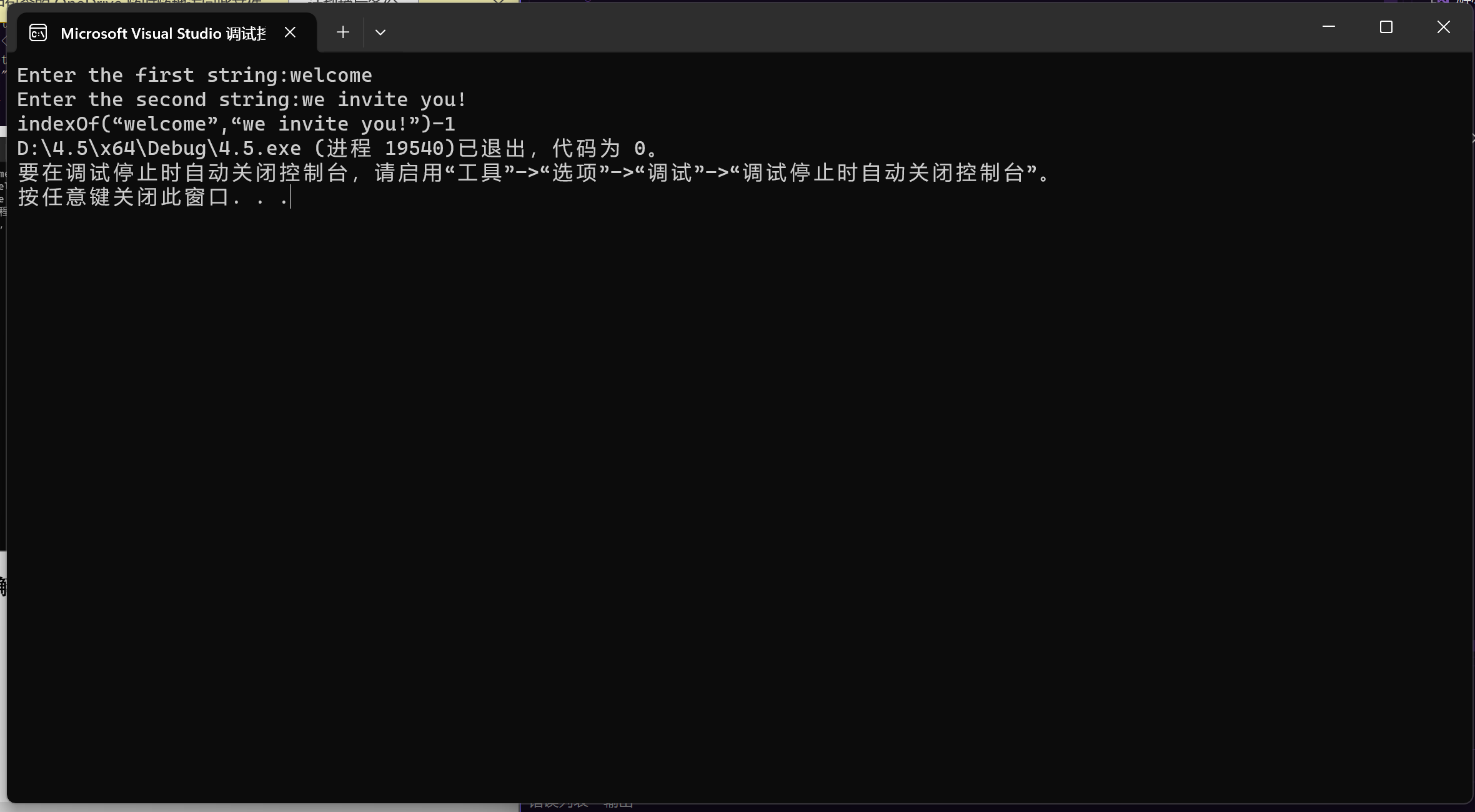
cout<<"indexOf(“"<<s1<<"”,“" <<s2<<"”)" << indexOf(s1, s2);//调用函数，输出结果

return 0;

}

****

****

****

**6.**

#include<iostream>

using namespace std;

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

{

int length = strlen(s);

for (int n = 0; n < length; n++)

{

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

{

if (s[n] == i+65)

{

counts[i]=counts[i]+1;

}//利用长度为26的数组数出字母出现的个数

}

}

}

int main()

{

cout << "Enter a string: ";

char s[100];

cin.getline(s,100);//获取数组

int length = strlen(s);

for (int n = 0; n < length; n++)

{

s[n] = toupper(s[n]);//将小写字母转化成大写字母便于计数

}

int counts[26];

for (int n = 0; n < 26; n++)

{

counts[n] = 0;

}//定义并初始化计数数组

count(s, counts);//调用函数，实现计数

for (int n = 0; n < 26; n++)

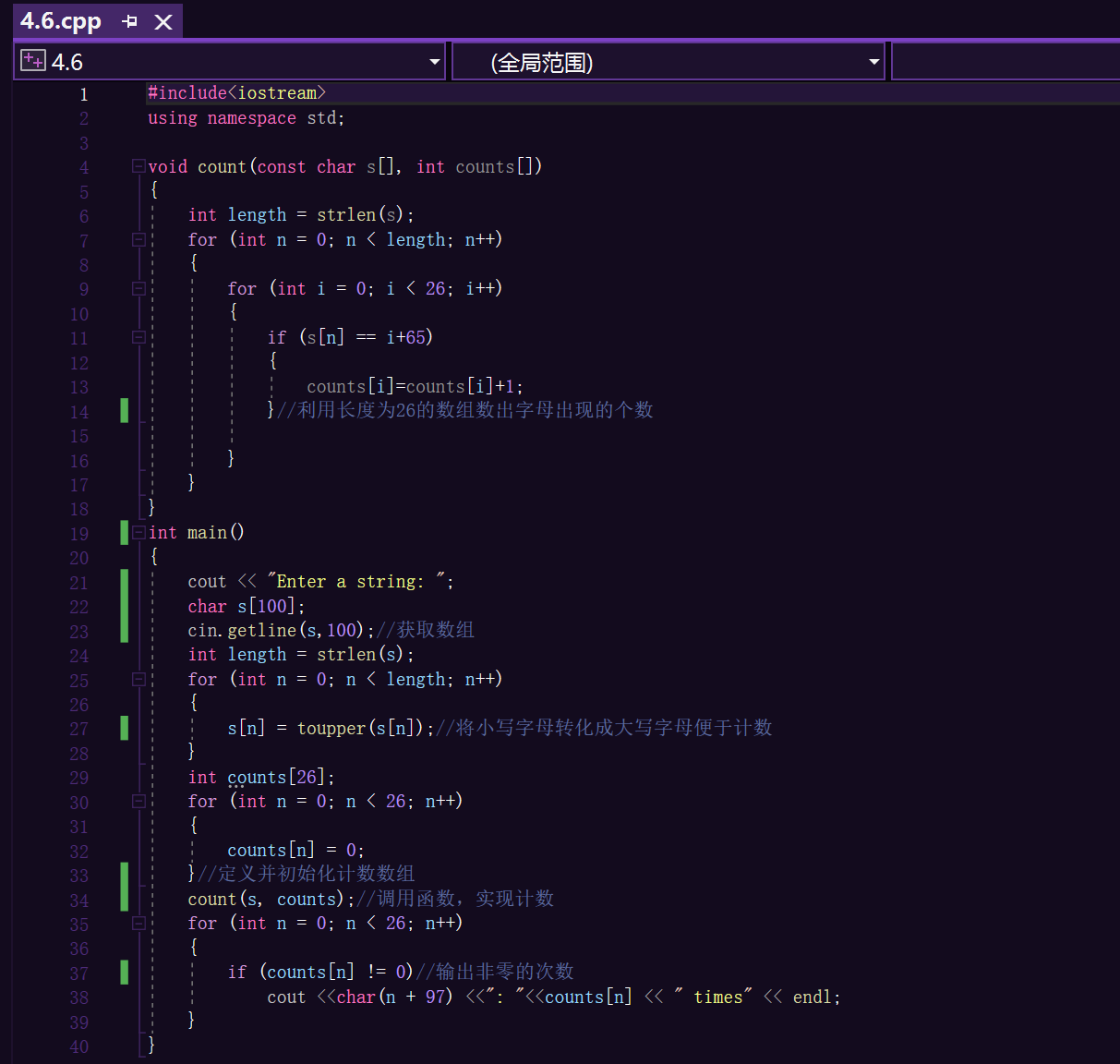
{

if (counts[n] != 0)//输出非零的次数

cout <<char(n + 97) <<": "<<counts[n] << " times" << endl;

}

}

****

****

**4.（2）.2.1**

#include<iostream>

using namespace std;

int indexOf(const char \*s1, const char \*s2)//将4.5中的函数改为含有指针的函数原型

{

int length1 = strlen(s1);

int length2 = strlen(s2);

for (int n = 0, m = 0; n < length2; n++)

{

if (s2[n] == 32)

{

m++;

}

if (strncmp(s1, s2 + n, length1) == 0)

{

return n - m + 1;

}

}

return -1;

}

int main()

{

char s1[100], s2[100];

cout << "Enter the first string:";

cin.getline(s1, 100);

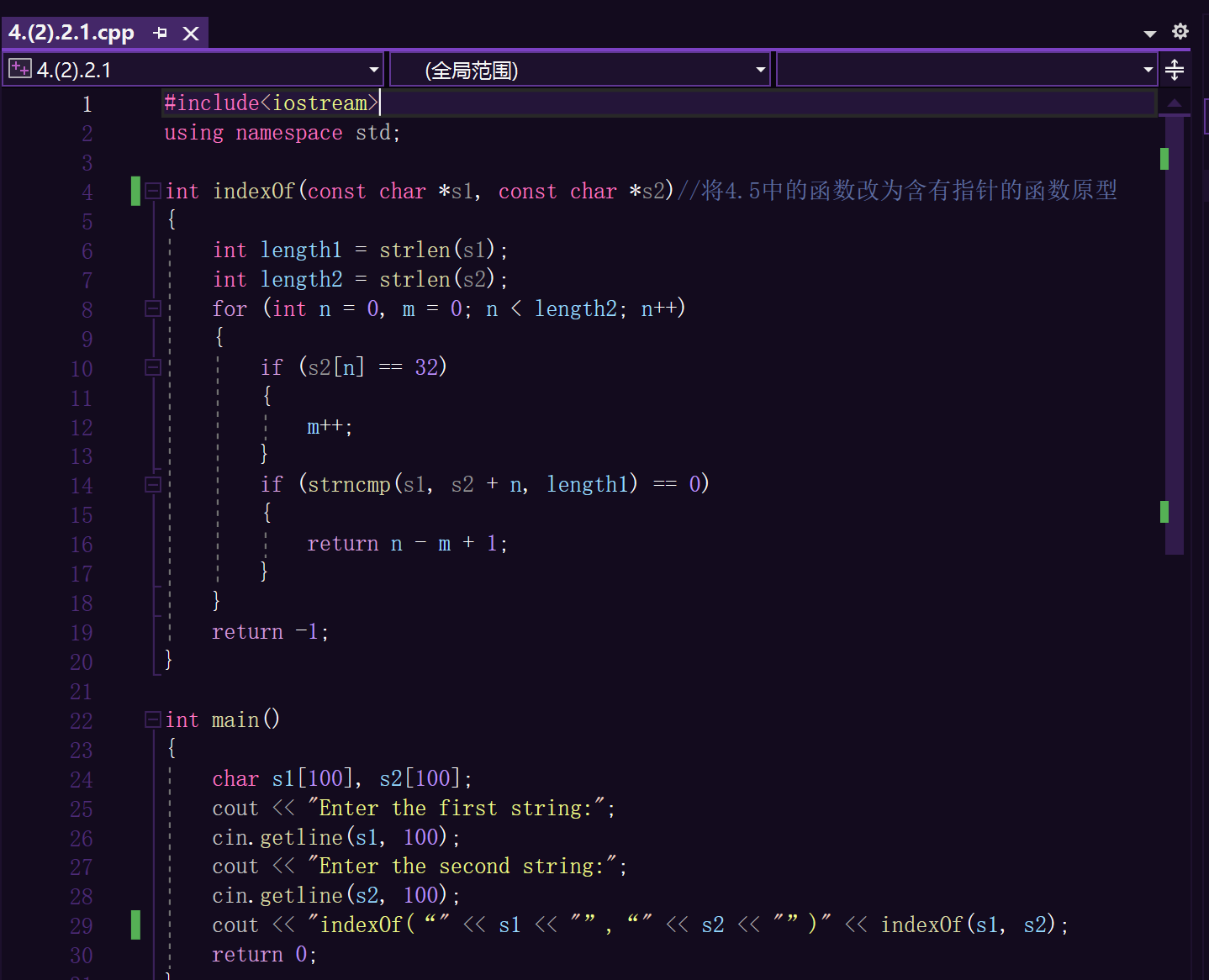
cout << "Enter the second string:";

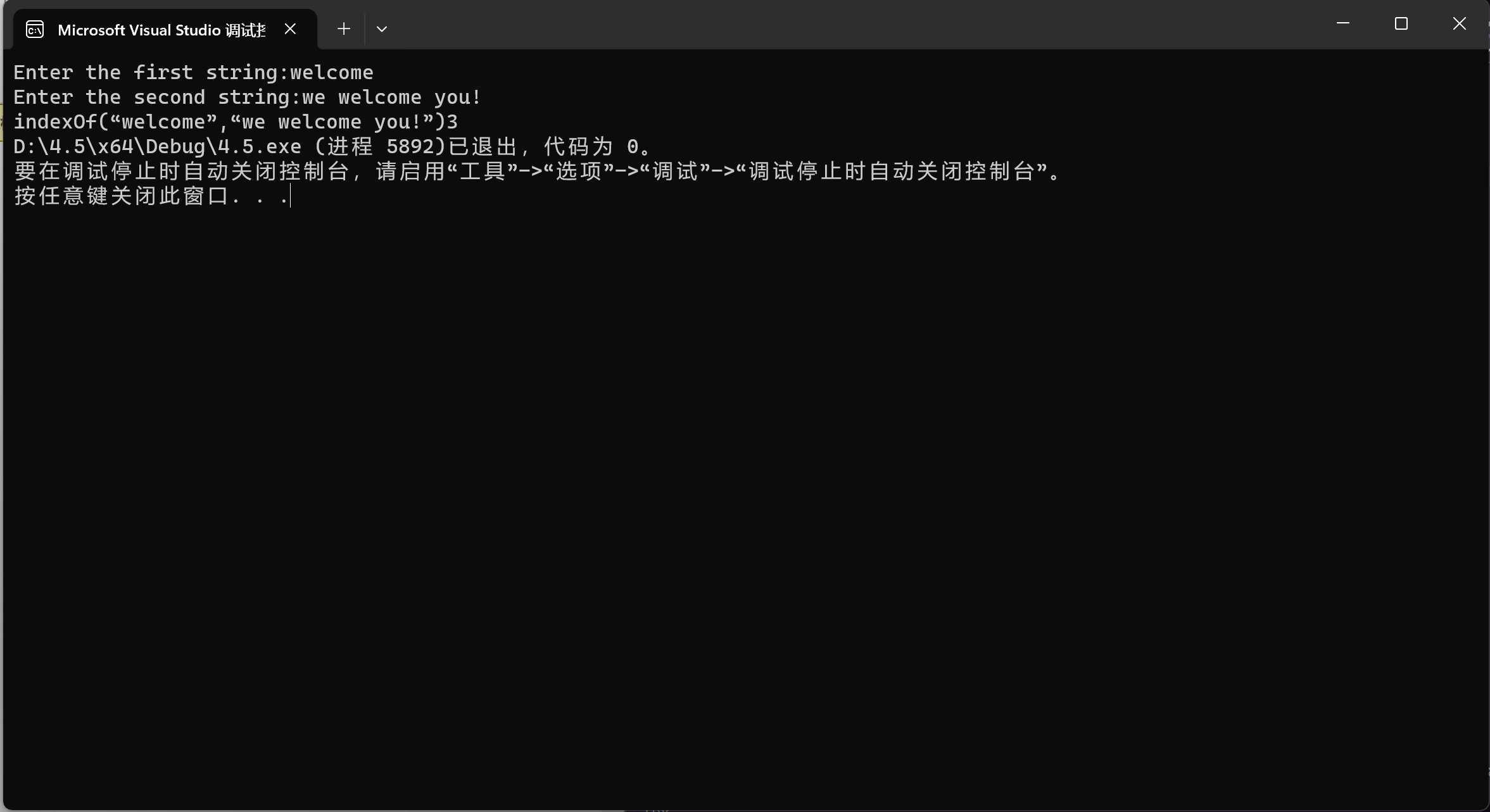
cin.getline(s2, 100);

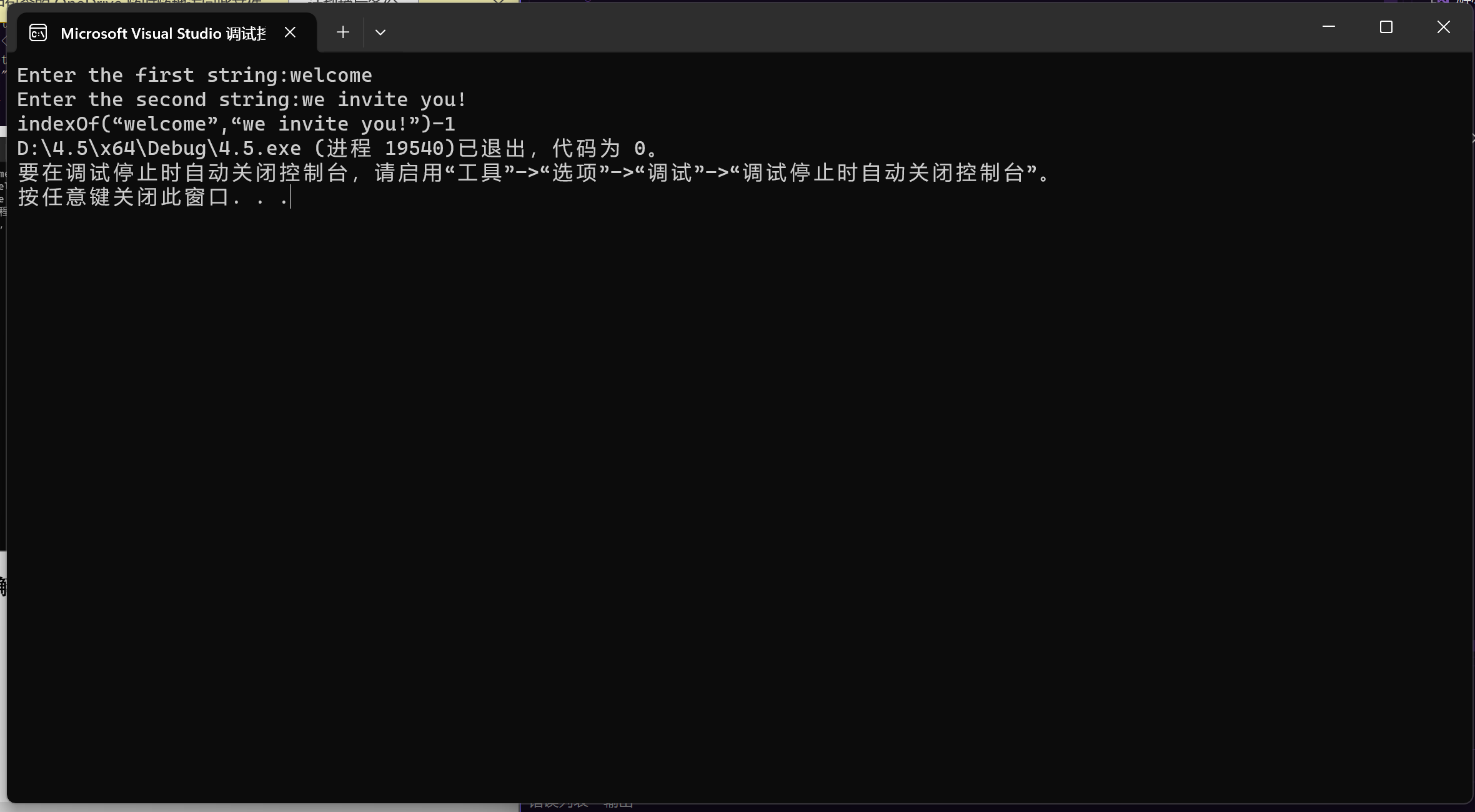
cout << "indexOf(“" << s1 << "”,“" << s2 << "”)" << indexOf(s1, s2);

return 0;

}

****

****

****

**4.（2）.2.2**

#include <iostream>

using namespace std;

int hex(int a,int b){

if (a == 0)

{

return 1;

}

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

{

a \*= 16;

}

return a;//实现十六的n次方的函数

}

int parseHex(const char\* const hexString){

int length = strlen(hexString);

int tem=0, fin=0;

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

{

int zhi = length - (i+1);

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

{

tem = hexString[i] - 'A' + 10;//将大写十六进制数字转化为十进制

fin += hex(tem, zhi);

}

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

{

tem = hexString[i] - 'a' + 10;//将小写十六进制数字转化为十进制

fin += hex(tem, zhi);

}

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

{

tem = hexString[i]-'0';

fin += hex(tem, zhi);//将不同位数上的数字加起来，得到最终数字

}

}

return fin;

}

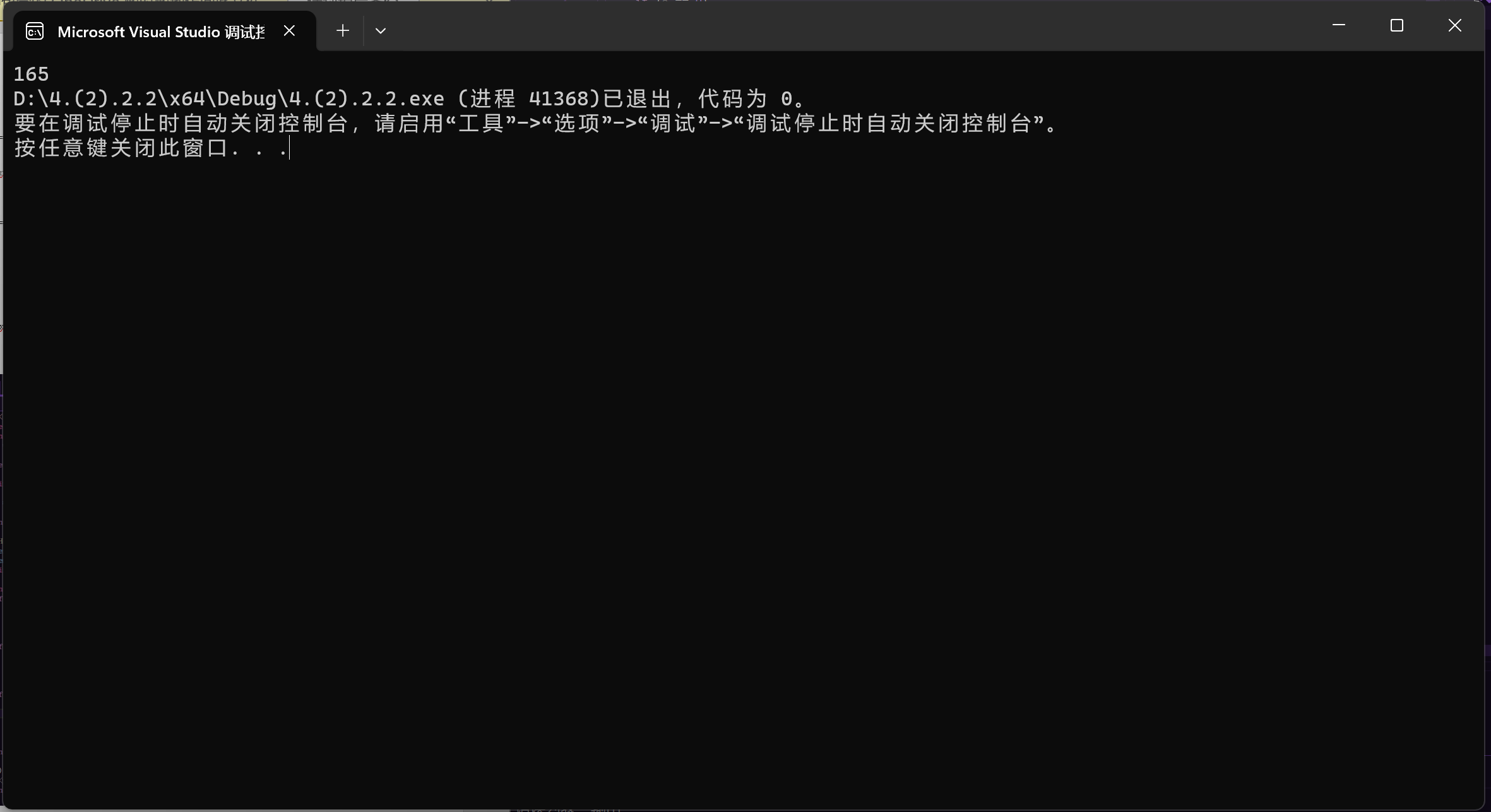
int main() {

cout<<parseHex("A5");//调用函数输出转化结果

return 0;

}

****

****

**4.（2）.2.3**

#include<iostream>

using namespace std;

int swap(int& a, int& b)

{

int t = a;

a = b;

b = t;

return 0;

}//交换函数

int bubble(int\* list,int size)

{

bool changed = true;

do

{

changed = false;

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

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

{

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

changed = true;

}

} while (changed);

return 1;

}//排序函数

int main()

{

int n;

cout << "请输入数组元素个数：";

cin >> n;

int\* arr = new int[n];

cout << "请输入一个数组：";

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

{

cin >> arr[i];

}//建立动态数组并输入数组元素个数和数组元素

bubble(arr,n);

cout << "排序后的结果为：";

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

{

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

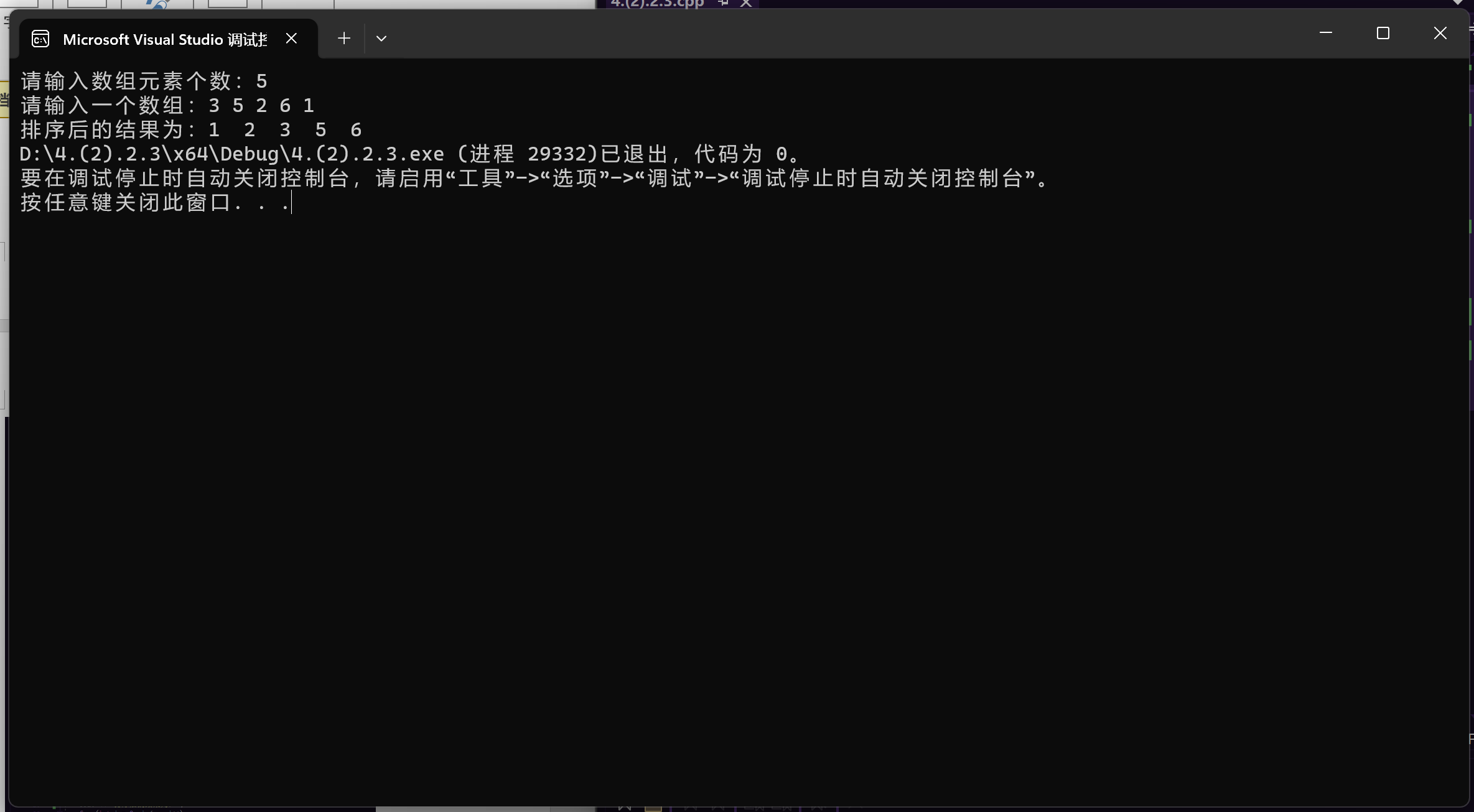
}//用指针的方式输出数组元素

delete[]arr;//释放数组内存

return 0;

}

****

****

**四、遇到的问题与解决方法**

**有时使用for循环显得代码繁杂，通过会议上课内容，我运用了课本上有关处理字符及字符串的函数，大大减少了代码行数，代码的·可读性变得更高**

**五、体会**

通过此次实验，我的编程思维更上一层楼，对复杂问题的思考方式也更为成熟，对数组可以实现的功能有了更深入的认识。我掌握了有关数组和指针的内容，，能够运用动态内存分配，对c++的语言特点有了更深入的了解，对函数的使用也更为熟练。同时我也明白了，做事情戒骄戒躁才会取得成功。