**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软工2403

学 号： 8209240324

姓 名： 刘丹阳

**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**

一 本实验中函数中返回的值为什么与函数类型一致？

1. 函数定义和类型规则

- 在C++中，函数在定义时就明确了其返回类型。这个返回类型规定了函数结束时应该返回的数据类型。例如，当定义一个函数`int add(int a, int b)`时，函数头部的`int`表示这个函数返回的是一个整数类型的值。这是C++语言的语法规则，编译器会根据这个规则来检查函数的返回值是否符合要求。

- 从语义上来说，函数的返回类型是一种约定，它告诉调用者这个函数执行完毕后会返回何种类型的数据。就像一个契约，调用者可以根据这个契约来正确地使用函数的返回值。例如，如果一个函数被声明为返回一个`double`类型的值，调用者就可以在一个需要`double`类型的表达式中使用这个返回值，如用于计算或者赋值给一个`double`类型的变量。

2. 类型安全和正确的程序行为

- 要求函数返回值与函数类型一致是为了保证类型安全。如果函数可以随意返回与声明类型不一致的值，可能会导致程序出现难以预料的错误。例如，假设一个函数声明为返回`int`类型，但是却返回了一个指针类型的值。当调用者试图将这个返回值当作整数来使用，比如进行算术运算时，程序就会出现错误，可能会导致程序崩溃或者产生错误的计算结果。

- 类型一致的要求也有助于编译器进行优化。编译器可以根据函数的返回类型对返回值进行合理的处理，例如，如果函数返回一个整数，编译器可以将这个整数直接存储在一个合适的寄存器中，以便后续的计算使用。如果返回值类型不固定，编译器就很难进行这样有效的优化。

3. 函数调用的上下文匹配

- 函数返回值的类型与函数类型一致，使得函数能够很好地融入调用它的上下文。例如，在一个表达式中调用函数，如`int result = add(3, 5);`，这里`add`函数的返回值类型是`int`，正好可以赋值给`result`这个`int`类型的变量。如果函数返回值类型与函数声明不符，就会破坏这种赋值的合法性，导致程序无法正确编译或者运行。

- 而且在复杂的表达式中，函数返回值的类型也需要与其他操作数的类型相匹配。例如，`double value = add(3, 5)/2.0;`，`add`函数返回的整数可以与`2.0`这个`double`类型的值进行除法运算，因为编译器会根据类型规则将整数隐式转换为`double`类型，从而保证整个表达式能够正确求值。

二 本实验中主函数调用函数时采用的是何种传递方式？

使用的是值传递的方式，传递的形参，对形参的操作不会营销形参。

**四、算法分析，程序结果**

**1.代码如下**

#include <iostream>

using namespace std;

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

int c = 0;

if (\*b > \*a) {

c = \*b;

\*b = \*a;

\*a = c;

}

}

int 最大公约数(int a, int b) {

int c = 0;

while (a != 0) {

c = a;

a = b % a;

b = c;

}

return c;

}

void 公倍数和公因数(int a,int b,int &c,int &d) {

c = 最大公约数(a,b);

d = a \* b / c;

}

int main()

{

int a, b;

cout << "请输入第一个数\t";

cin >> a ;

cout << "请输入第二个数\t";

cin >> b;

swap(&a, &b);

int c, d;

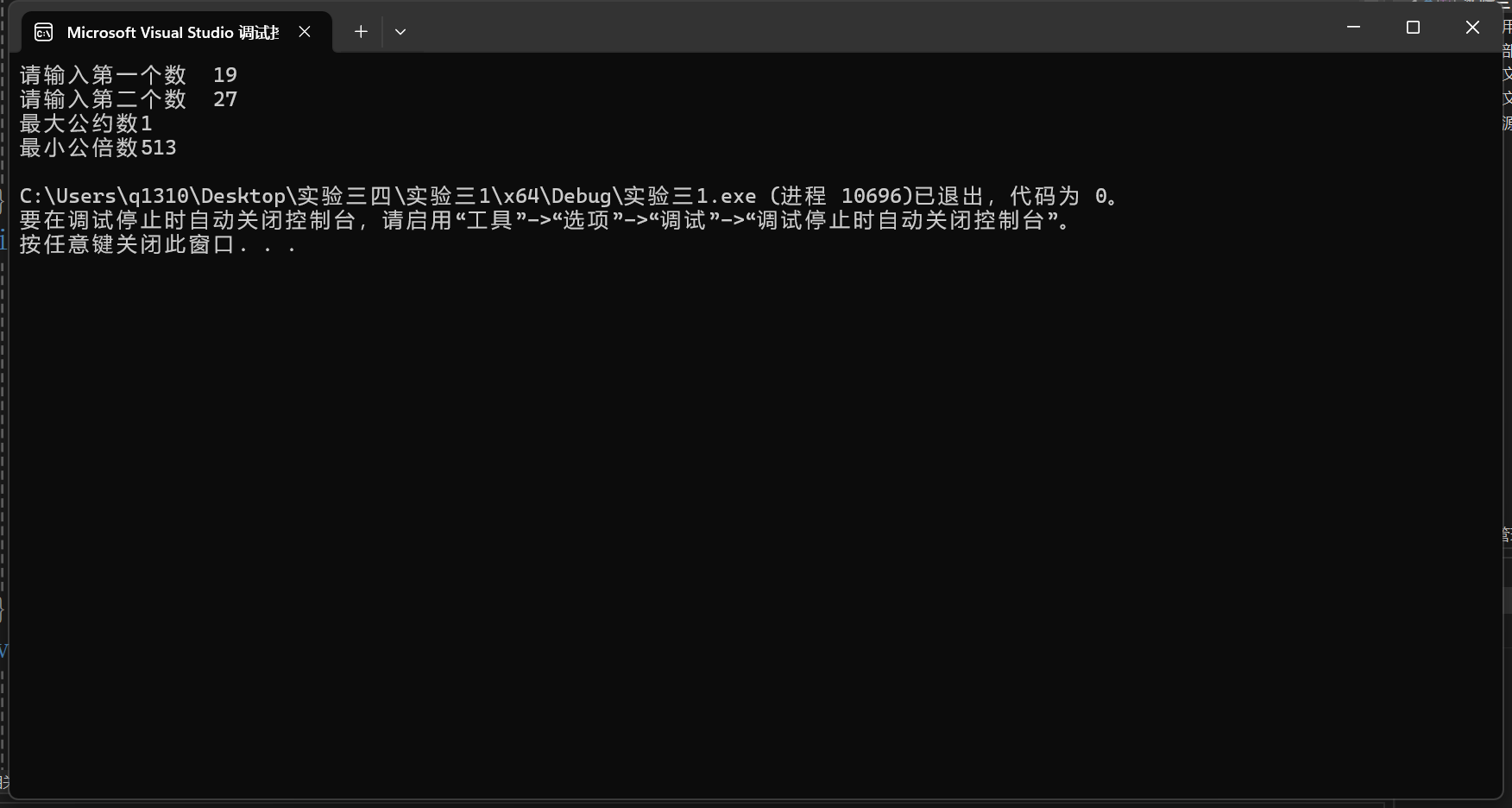
公倍数和公因数(a, b, c, d);

cout<<"最大公约数"<<c<<endl;

cout << "最小公倍数" << d << endl;

}

**实验结果：**

1. **代码如下**

#include<iostream>

using namespace std;

bool is\_prime(int num) {

int a,temp=1;

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

{

temp = num % a;

if (temp == 0) {

return false;

break;

}

}

return true;

}

int main() {

int a = 0;

int b[200];

int num = 2;

while (a < 200) {

if (is\_prime(num)) {

cout << num<<"\t";

a++;

if (a % 10 == 0) {

cout << endl;

}

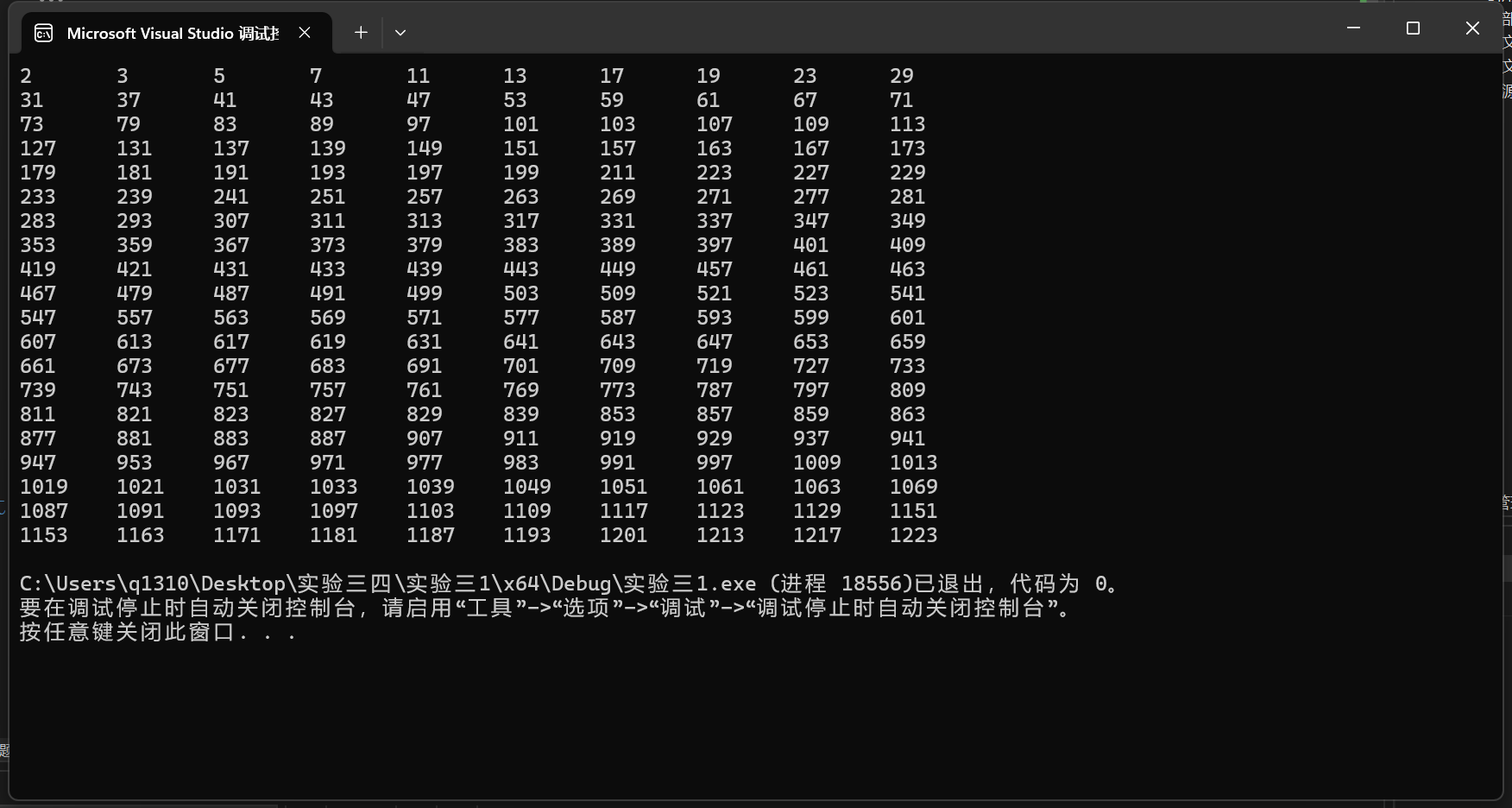
}

num++;

}

}

**结果如下**

****

1. **代码如下**

#include"mytemperature.h"

double celsius\_to\_fah(double cel) {

double fah = cel \* 1.8 + 32;

return fah;

}

double fahrenheit\_to\_cels(double fah) {

double cel = (fah - 32) / 1.8;

return cel;

}

#pragma once

//摄氏到华氏温度

double celsius\_to\_fah(double cel);

//华氏温度到摄氏

double fahrenheit\_to\_cels(double fah);

#include"mytemperature.h"

double celsius\_to\_fah(double cel) {

double fah = cel \* 1.8 + 32;

return fah;

}

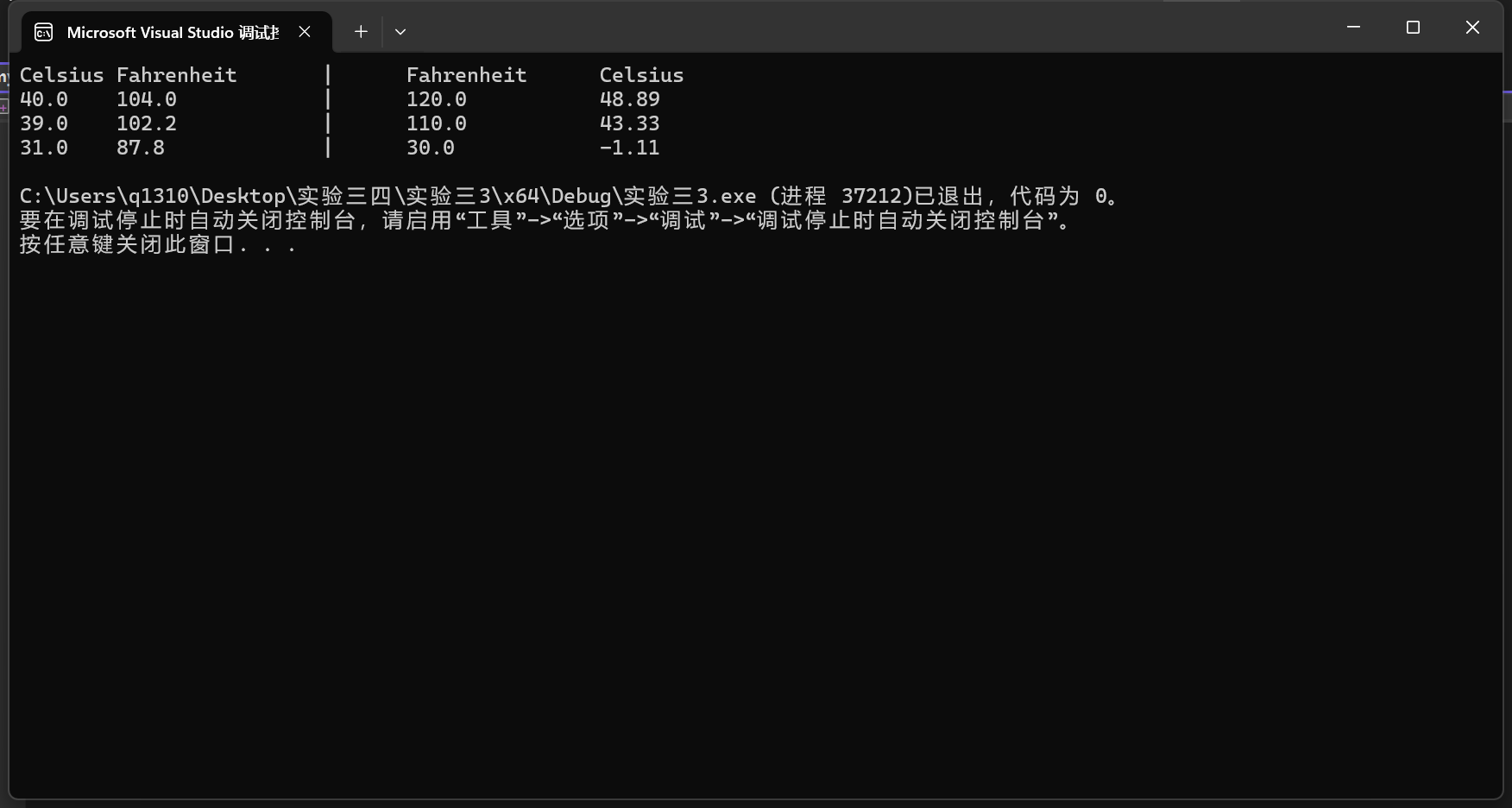
double fahrenheit\_to\_cels(double fah) {

double cel = (fah - 32) / 1.8;

return cel;

}

结果如下：



1. **代码如下**

#include <iostream>

using namespace std;

int taozi10(int day,int taozi)

{

if (day == 10 ) {

return taozi;

}

else {

return taozi10(day + 1, (taozi \* 2 + 2));

}

}

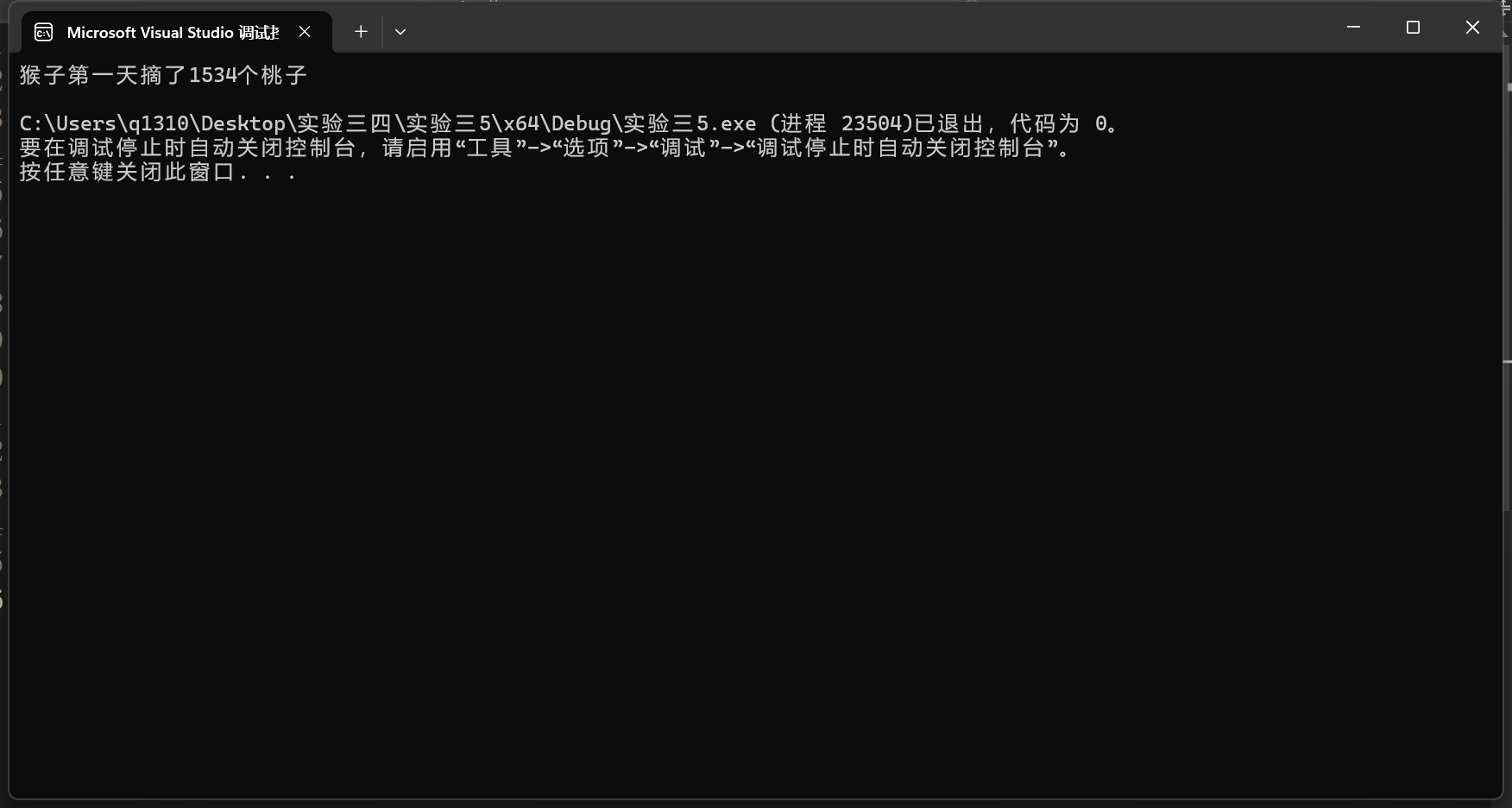
int main()

{

cout << "猴子第一天摘了" << taozi10(1,1)<<"个桃子" << endl;

}

结果如下：



1. **遇到的问题与解决方法**

**六、体会**

**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**

1. **算法分析，程序结果**

**(一)**

**1程序设计：**

#include <iostream>

#include <set>

using namespace std;

int main() {

set<int> a;

int b;

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

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

cin >> b;

a.insert(b);

}

cout << "The distict numbers are:" << endl;

for (const auto& c : a) {

cout << c <<"\t";

}

cout << endl;

return 0;

}

**结果：**



1. **程序设计**

#include <iostream>

using namespace std;

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

double t = a;

a = b;

b = t;

}

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);

}

int main() {

const int arraySize = 10;

double numbers[arraySize];

cout << "请输入10个双精度数字：" << endl;

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

cin >> numbers[i];

}

bubbleSort(numbers, arraySize);

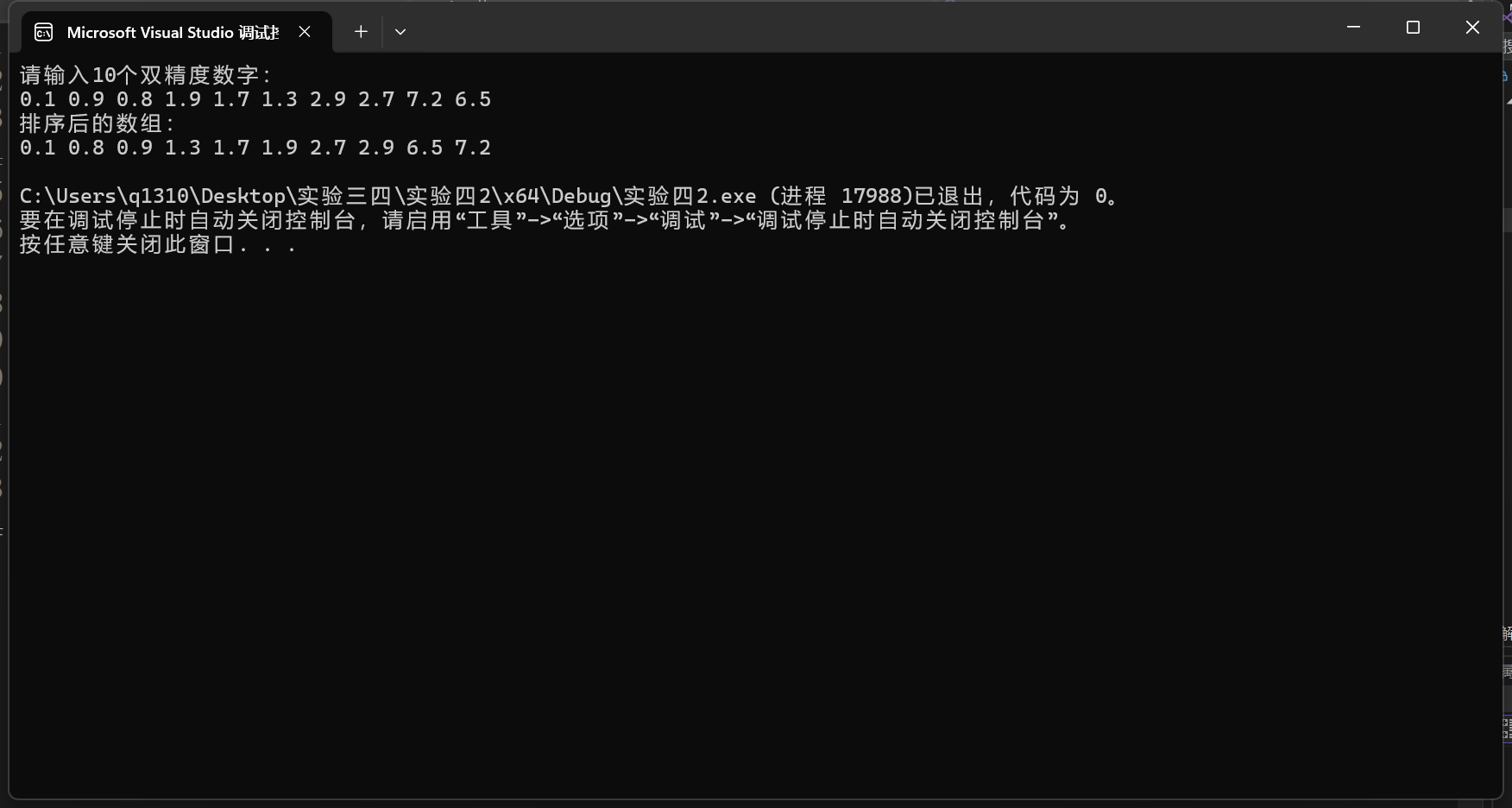
cout << "排序后的数组：" << endl;

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

cout << numbers[i] << " ";

}

cout << endl;

**结果：**

1. **程序设计**

#include <iostream>

using namespace std;

int main() {

bool lockers[100] = { false };

for (int student = 1; student <= 100; student++) {

for (int locker = student - 1; locker < 100; locker += student) {

lockers[locker] = !lockers[locker];

}

}

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

if (lockers[i]) {

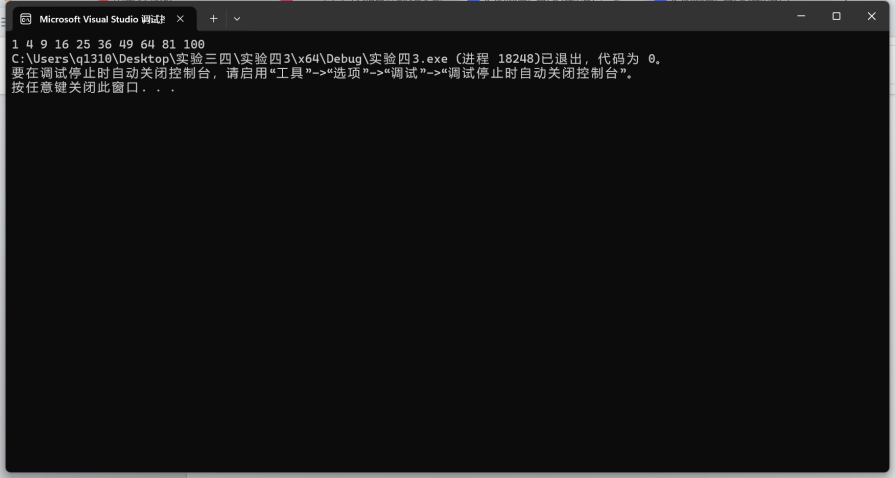
cout << (i + 1) << " ";

}

}

return 0;

}

**结果**

1. **程序设计**

#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++];

}

}

int main() {

int list1[80], size1;

int list2[80], size2;

int list3[160];

cout << "Enter list1: ";

cin >> size1;

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

cin >> list1[i];

}

cout << "Enter list2: ";

cin >> size2;

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

cin >> list2[i];

}

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

cout << "The merged list is ";

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

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

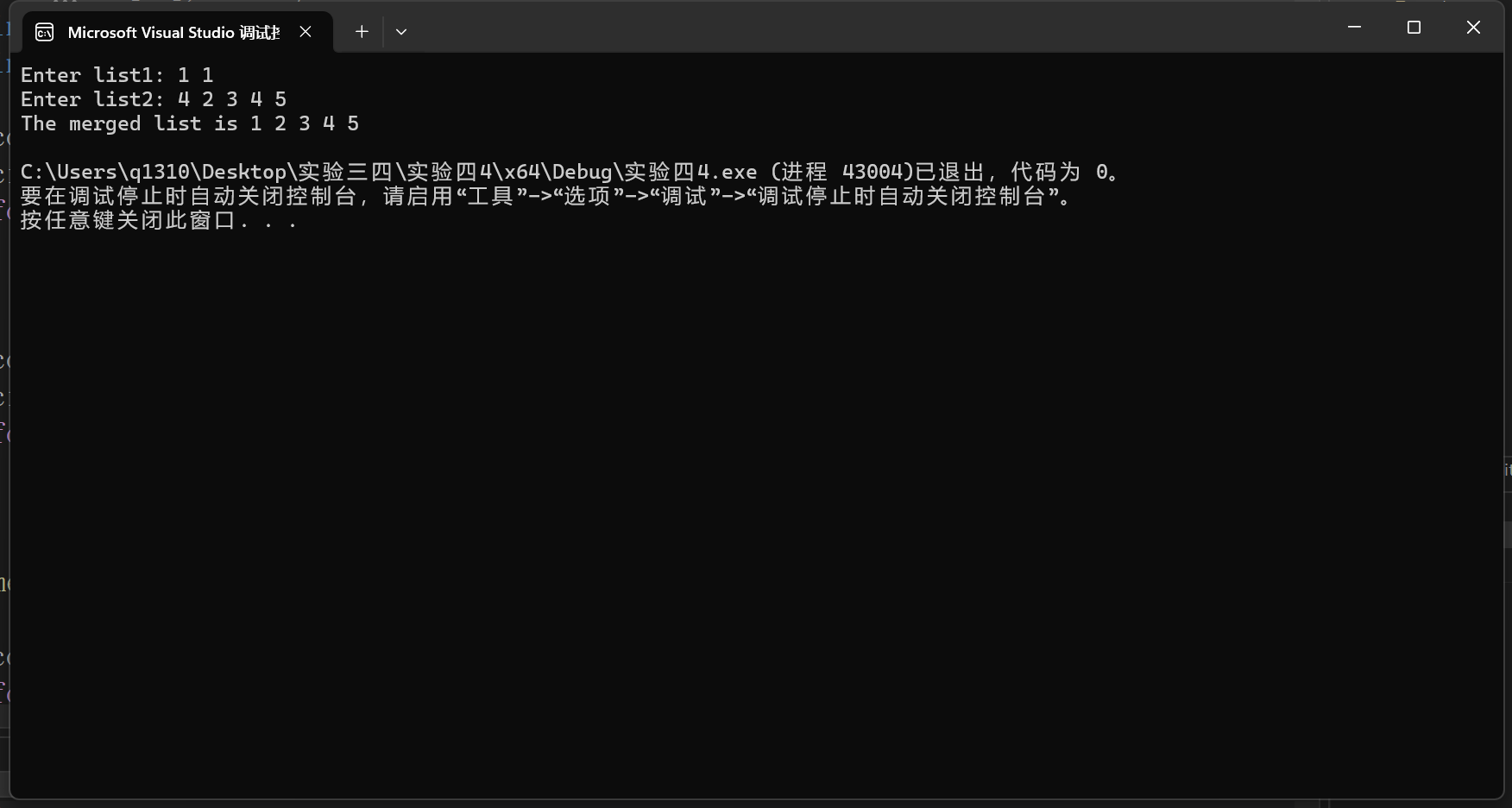
}

cout << endl;

return 0;

}

结果展示：

1. **程序设计**

#include <iostream>

#include <string>

using namespace std;

int main() {

string s1, s2;

cout << "Enter the first string: ";

cin >> s1;

cout << "Enter the second string: ";

cin >> s2;

int result = s2.find(s1);

if (result != string::npos) {

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

}

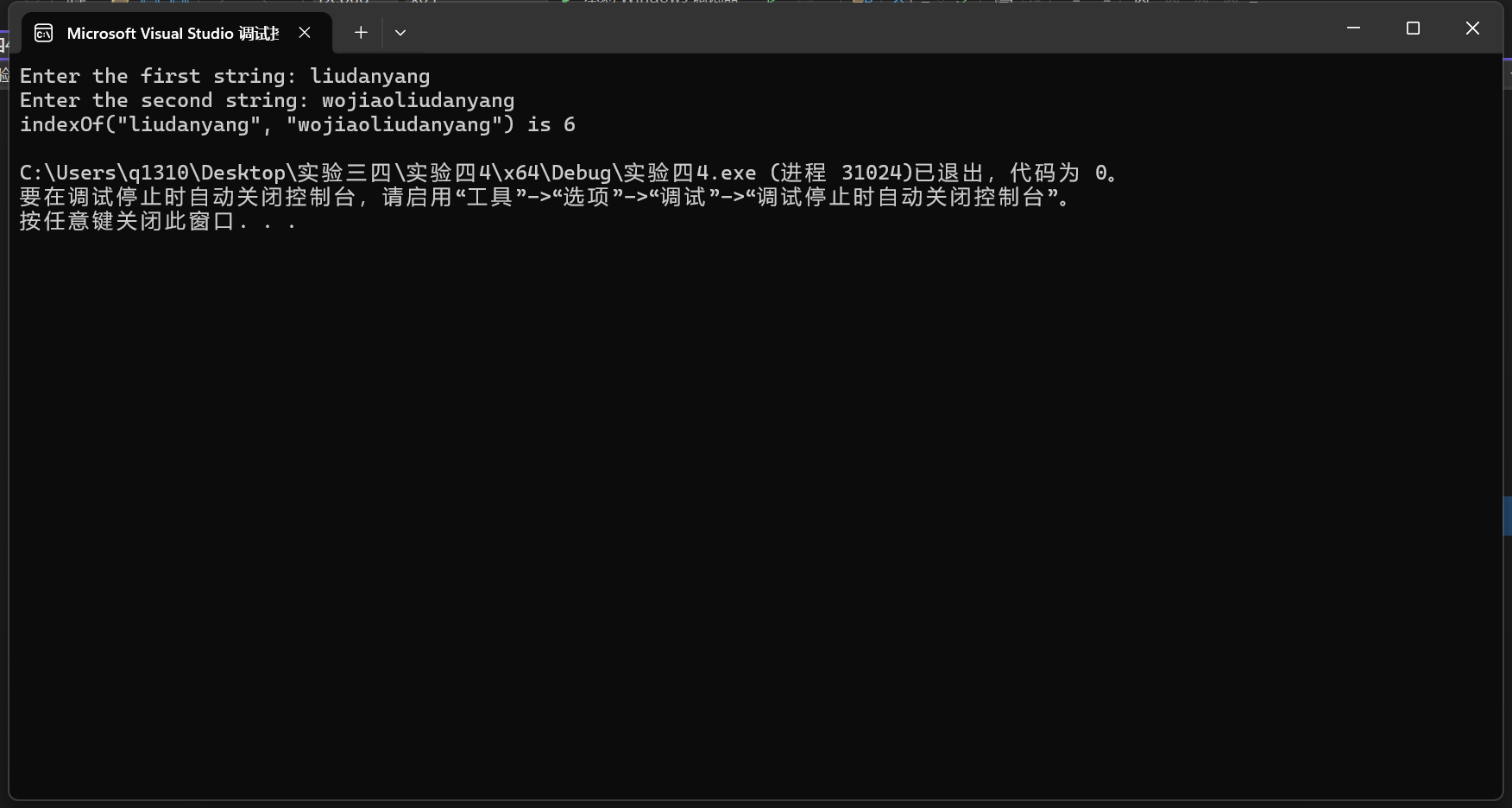
else {

cout << "indexOf(\"" << s1 << "\", \"" << s2 << "\") is " << -1 << std::endl;

}

return 0;

}

结果展示：

1. **程序设计**

#include <iostream>

#include <cctype>

using namespace std;

// 计算字符串中每个字母出现的次数

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

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

if (isalpha(s[i])) {

char lowerChar = tolower(s[i]);

counts[lowerChar - 'a']++;

}

}

}

int main() {

char str[100];

int counts[26] = { 0 };

cout << "Enter a string: ";

cin >> str;

count(str, counts);

cout << "字母出现的次数:\n";

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

if (counts[i] != 0) {

cout << static\_cast<char>('a' + i) << ": " << counts[i] << '\n';

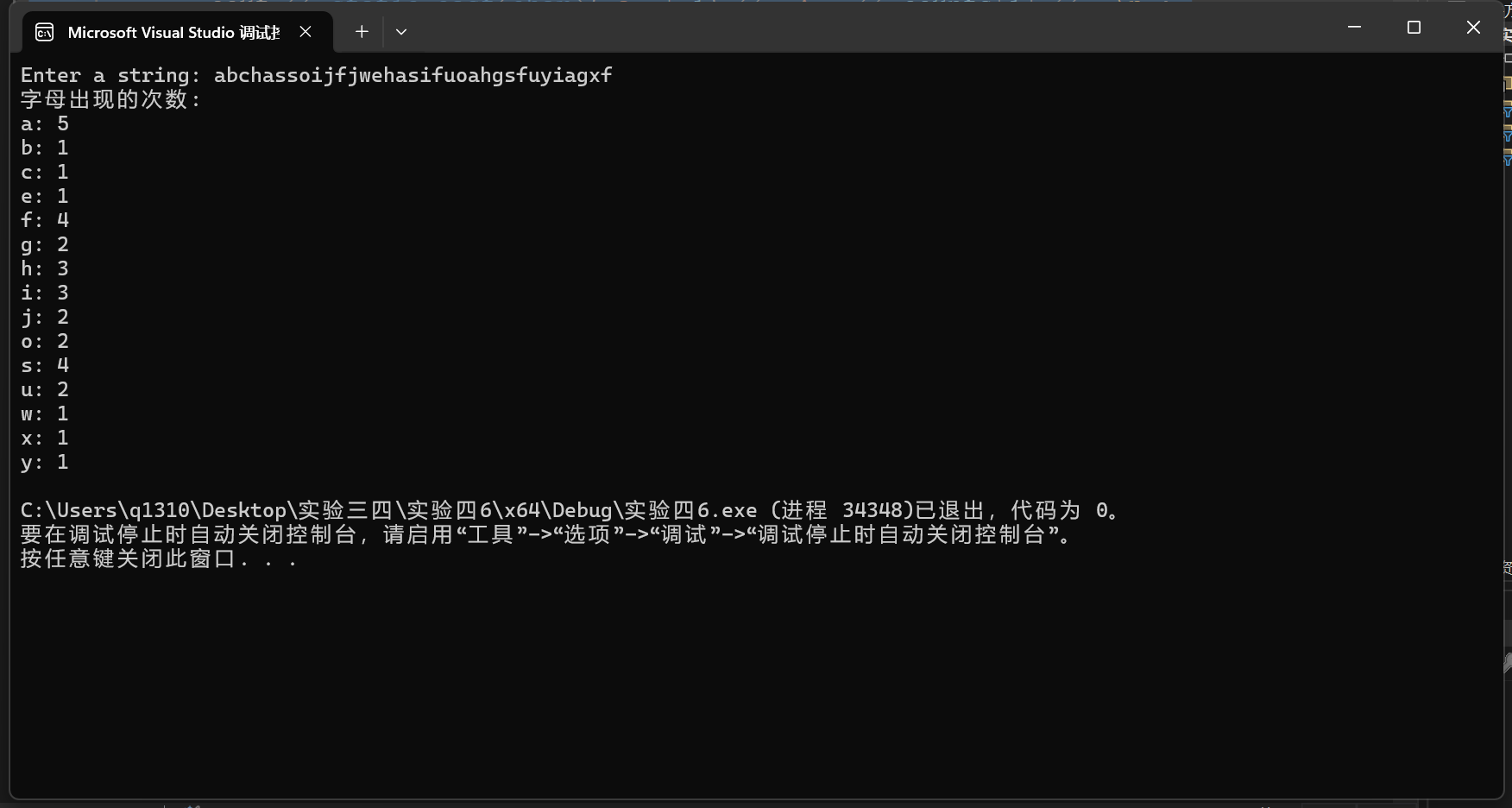
}

}

return 0;

}

结果展示



**(二)**

**（1）**

**1程序设计**

#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;

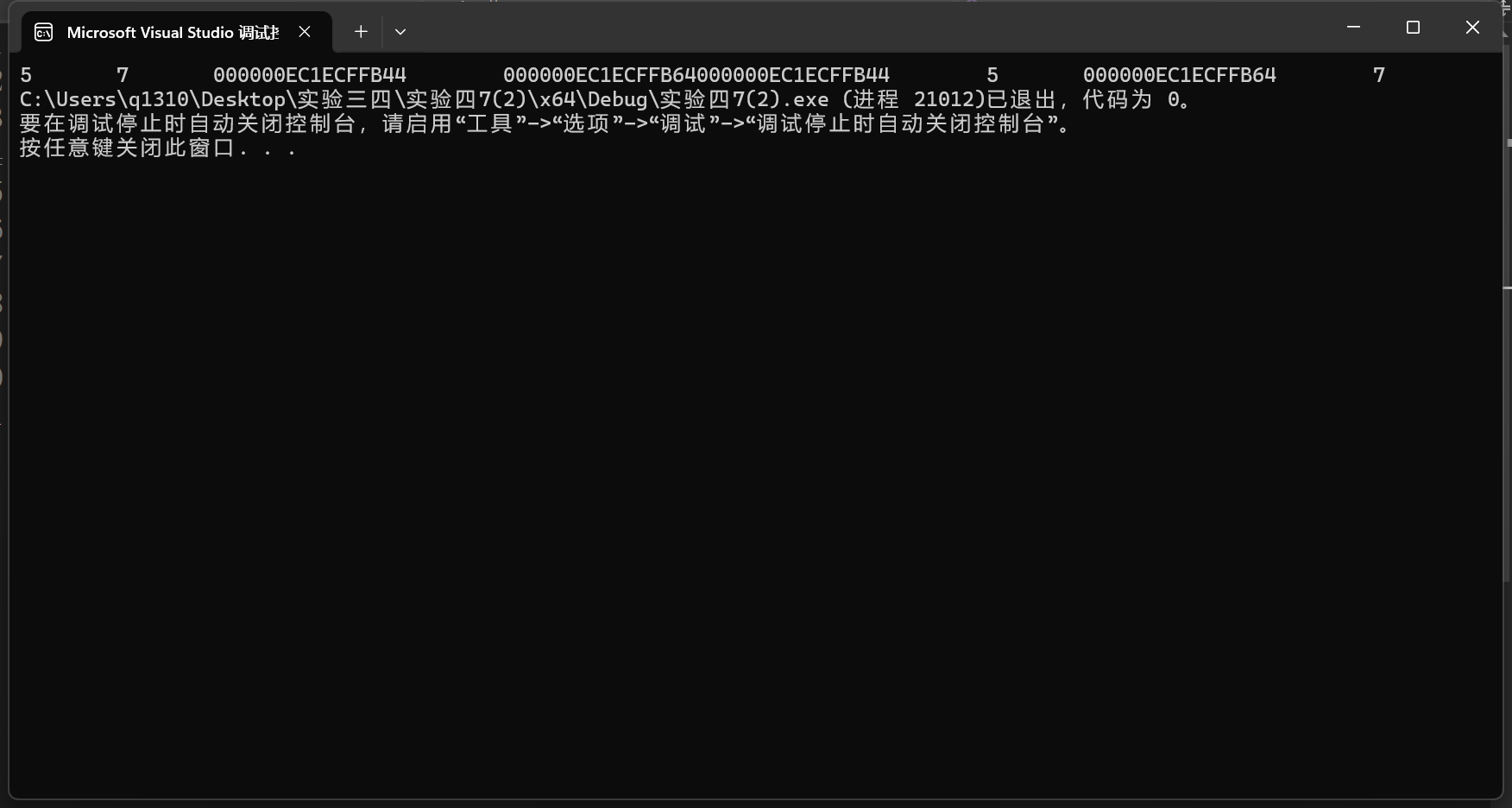
i = 5; j = 7;

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

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

}

**结果**

****

**2程序设计**

#include <iostream>

int main() //C语言程序，要了解

{

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

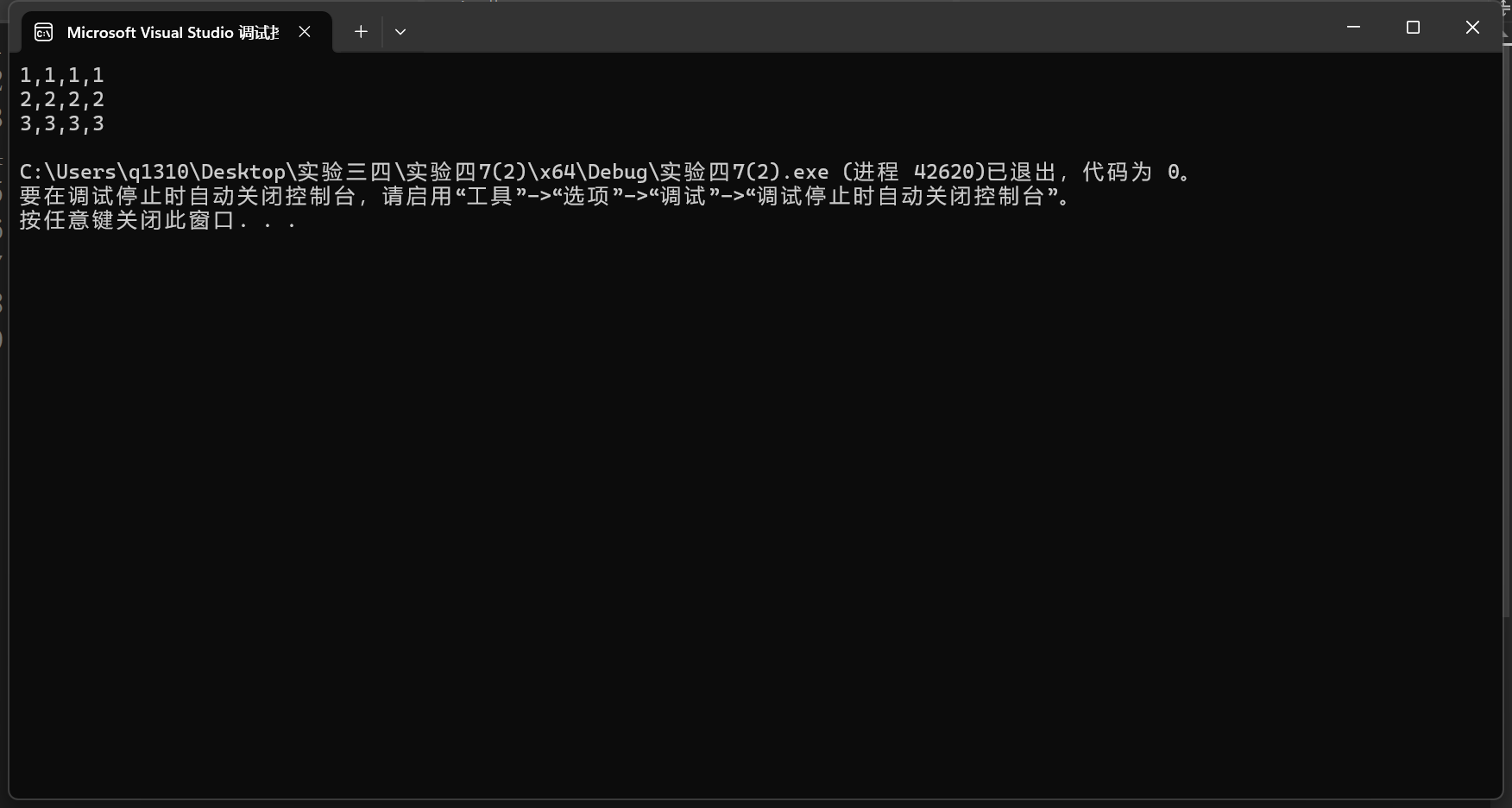
int\* p, i;

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

**结果**

****

**3程序设计**

#include <iostream>

#include <cstring>

void f(char\* st, int i)

{

st[i] = '\0';

std::cout << st << std::endl;

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

}

int main()

{

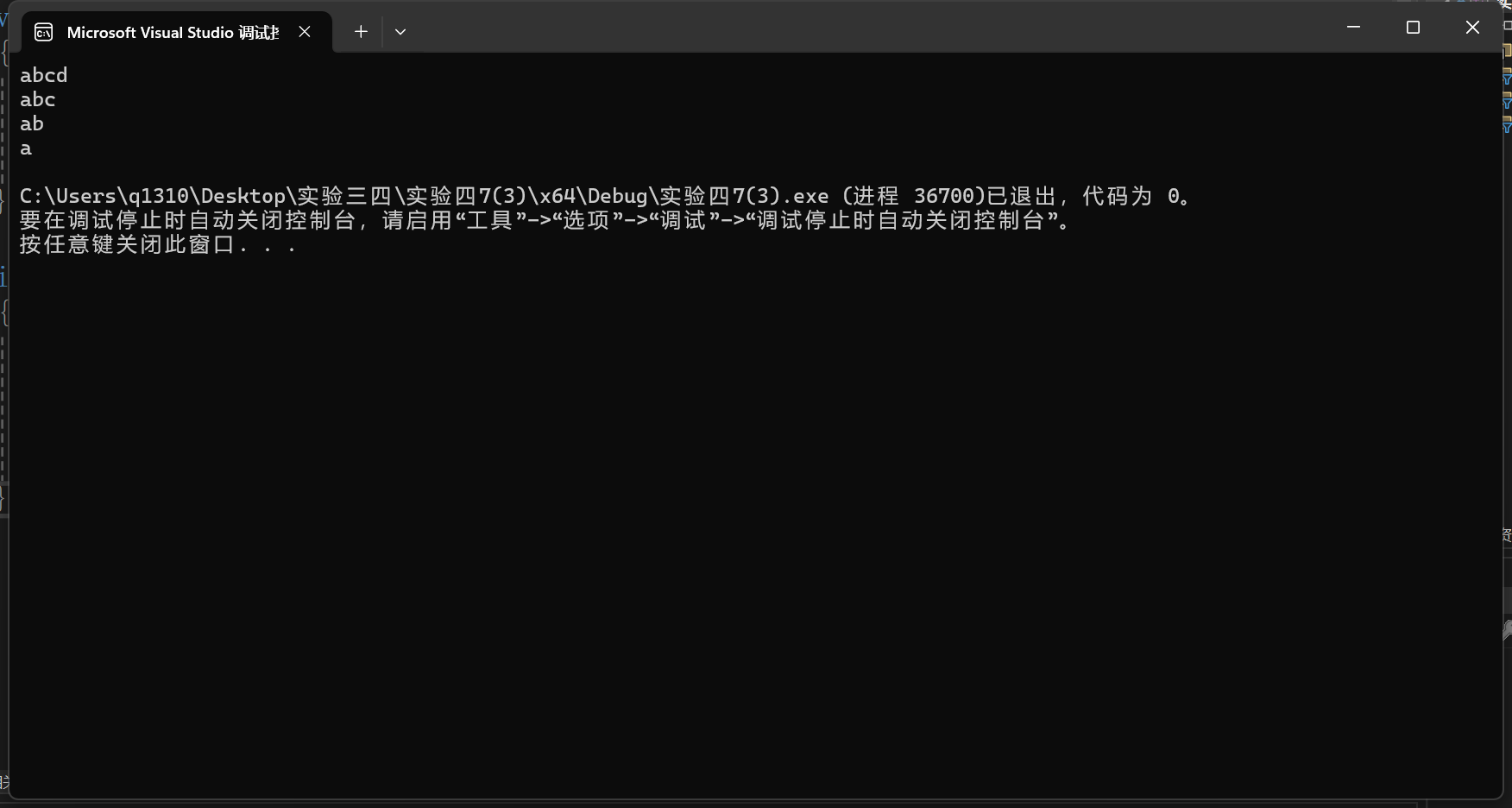
char st[] = "abcd";

f(st, 4);

return 0;

}

结果：



4程序设计

#include<iostream>

using namespace std;

int\* f()

{

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

return list;

}

int main()

{

int\* p = f();

cout << p[0] << endl;

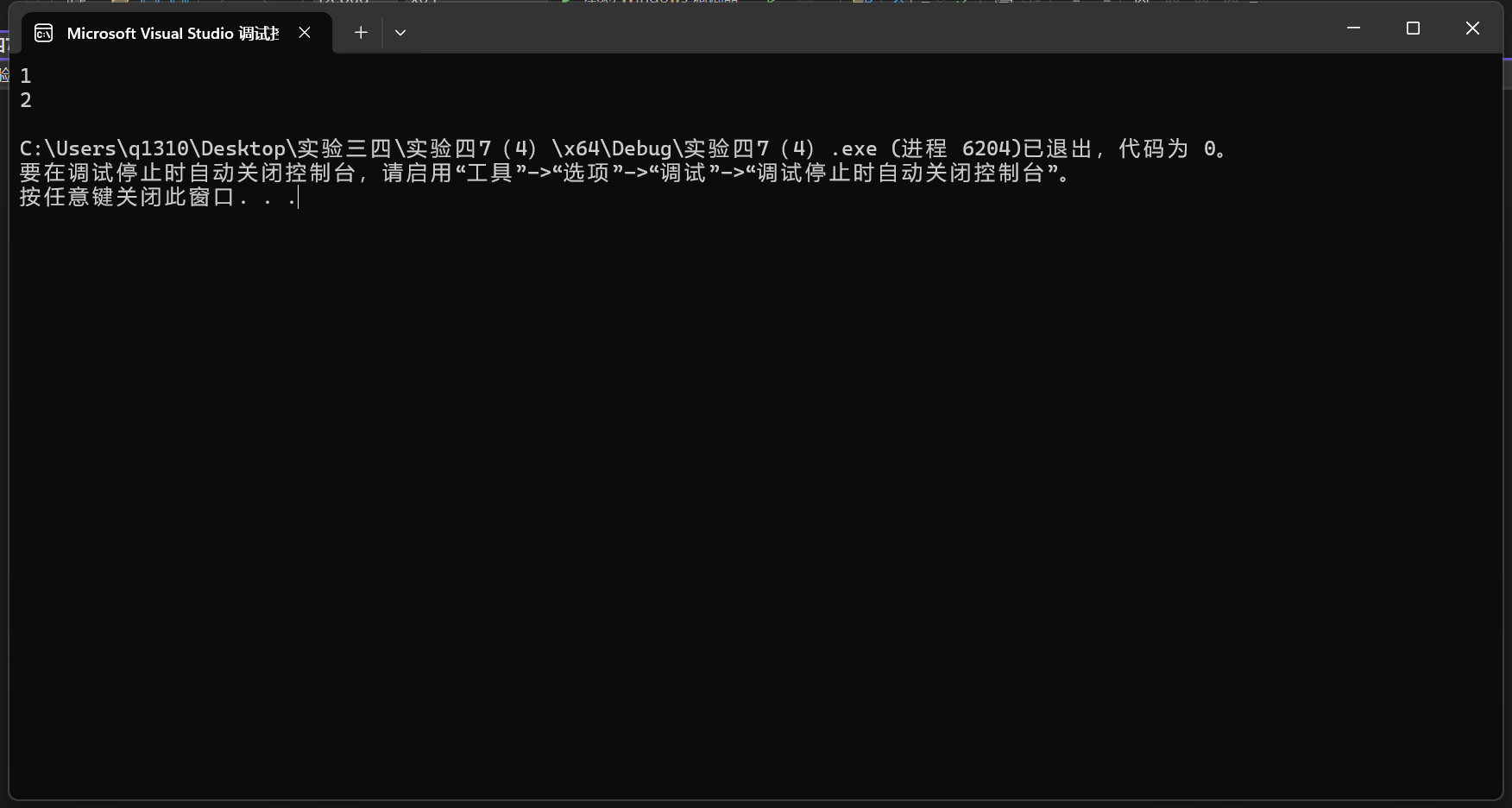
cout << p[1] << endl;

delete[] p;

return 0;

}

结果



（2）

1程序设计

#include <iostream>

#include <cstring>

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];

char s2[100];

cout << "请输入字符串s1：";

cin.getline(s1, 100);

cout << "请输入字符串s2：";

cin.getline(s2, 100);

int result = indexof(s1, s2);

if (result != -1) {

cout << "字符串s1是字符串s2的子串，第一次匹配的下标为：" << result << endl;

}

else {

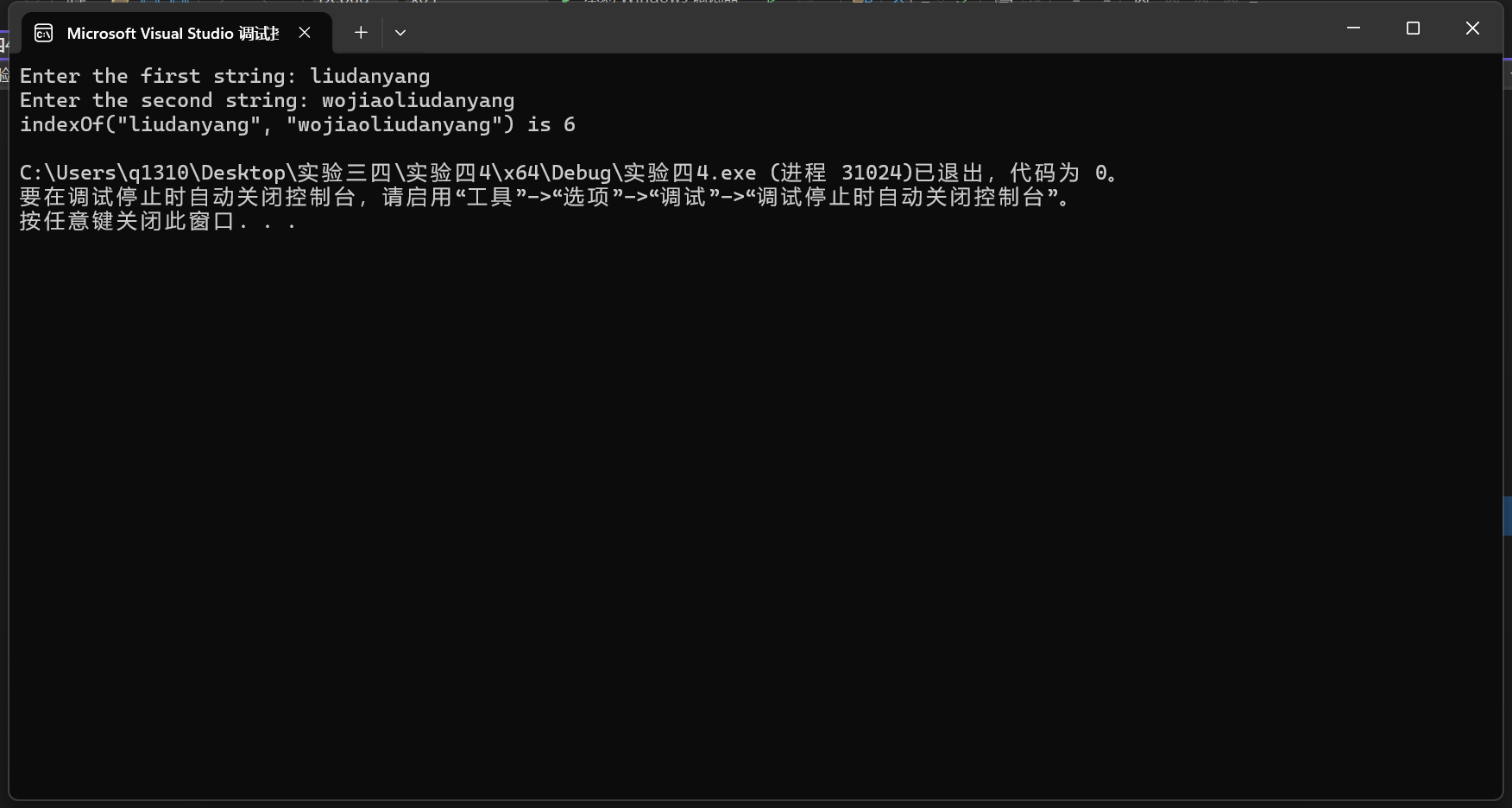
cout << "字符串s1不是字符串s2的子串" << endl;

}

return 0;

}

结果：



2

#include <iostream>

#include <cctype>

#include <string>

using namespace std;

int parseHex(const char\* const hexString) {

int result = 0;

int len = strlen(hexString);

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

char ch = hexString[len - 1 - i];

int digit;

if (isdigit(ch)) {

digit = ch - '0';

}

else if (isalpha(ch)) {

ch = toupper(ch);

digit = ch - 'A' + 10;

}

else {

return 0;

}

result += digit \* (1 << (4 \* i));

}

return result;

}

int main() {

char hexInput[100];

cout << "请输入十六进制数：";

cin >> hexInput;

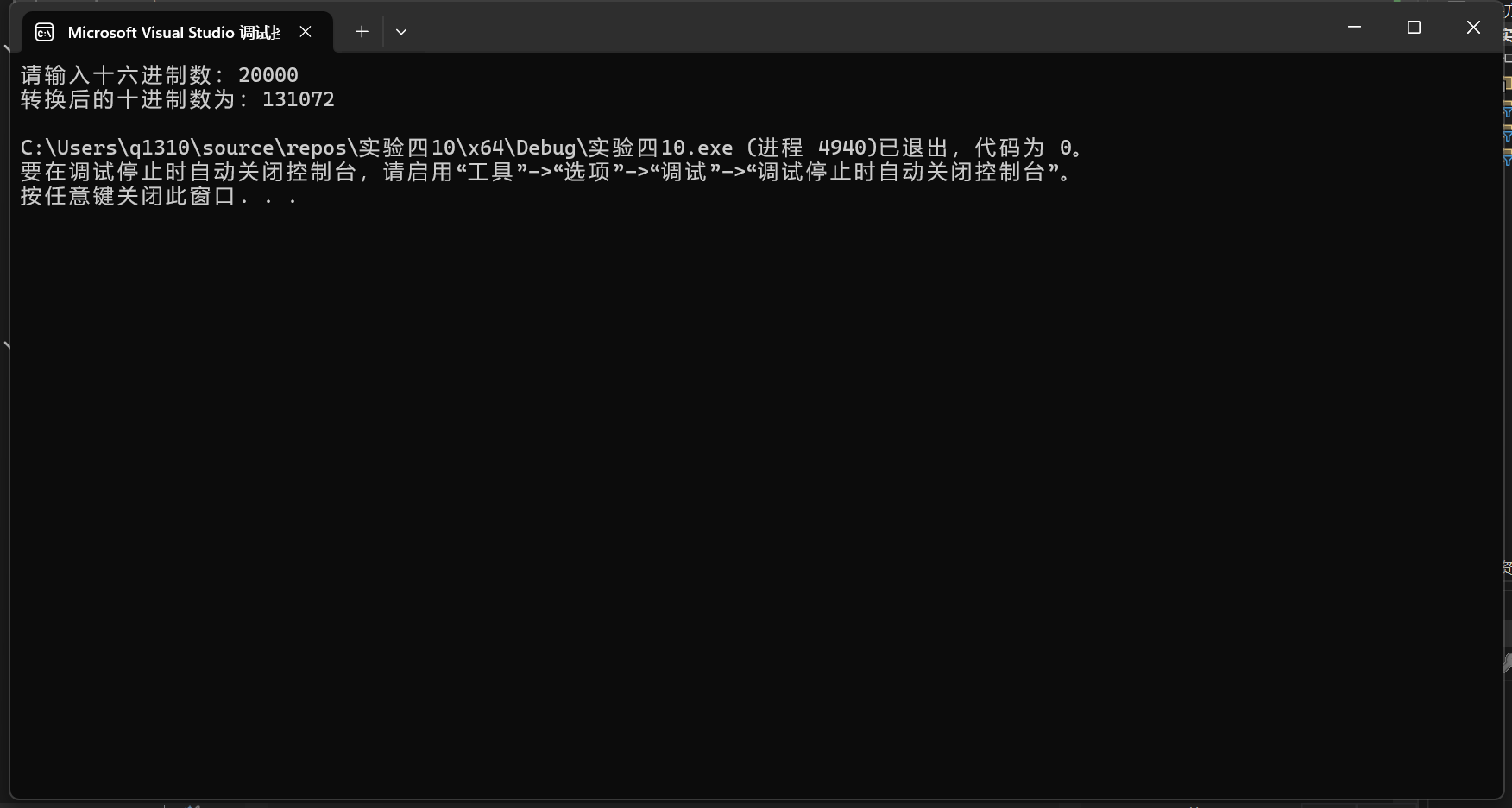
int decimal = parseHex(hexInput);

cout << "转换后的十进制数为：" << decimal << endl;

return 0;

}

结果



3

#include <iostream>

using namespace std;

void bubbleSort(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]) {

int temp = arr[j];

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

arr[j + 1] = temp;

}

}

}

}

int main() {

int numElements;

cout << "请输入数组元素个数: ";

cin >> numElements;

int\* dynamicArray = new int[numElements];

cout << "请依次输入数组元素: ";

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

cin >> dynamicArray[i];

}

bubbleSort(dynamicArray, numElements);

cout << "排序后的数组元素为: ";

for (int\* p = dynamicArray; p < dynamicArray + numElements; p++) {

cout << \*p << " ";

}

cout << endl;

delete[] dynamicArray;

return 0;

}

结果;



1. **遇到的问题与解决方法**

**五、体会**