**计算机程序设计基础（C++)**

**实验报告**

专业班级：软工2204班

学 号：8209220409

姓 名：陈思汗

**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。

（2）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：

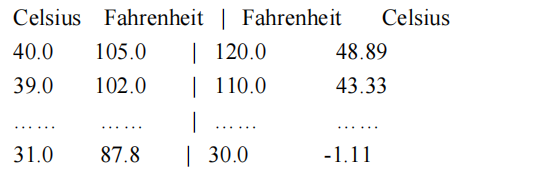
2 3 5 7 11 13 17 19 23 29

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：

（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）。

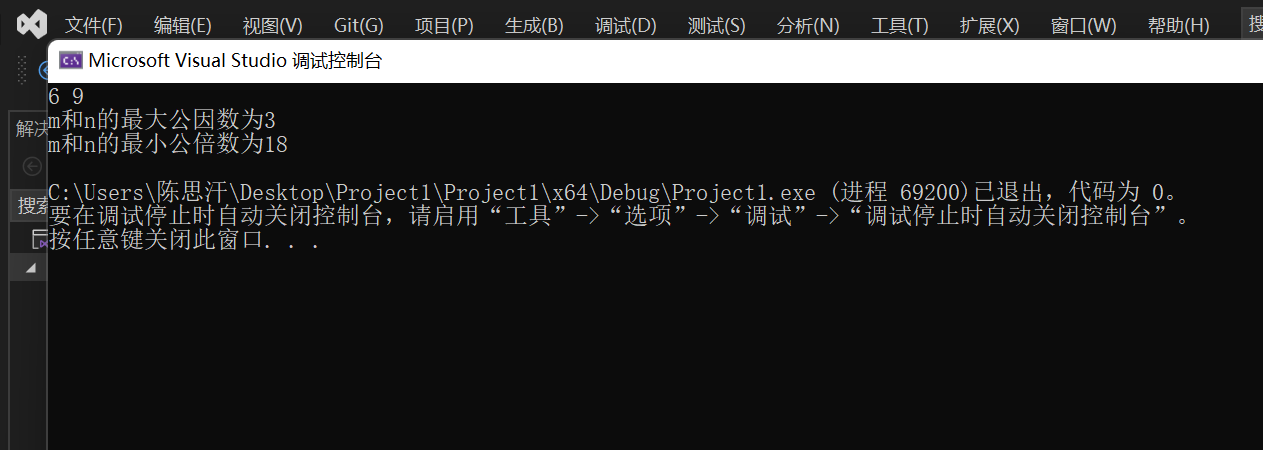
**三、实验思考题**

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 GMF(int a,int b)

{

int gmf,min;

min = a < b ? a : b;

for (min; min > 1; min--)

{

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

break;

}

return min;

}

int LCM(int a, int b)

{

int lcm, max;

max = a > b ? a : b;

for (max; max < (a\*b); max++)

{

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

break;

}

return max;

}

int main()

{

int m, n;

cin >> m >> n;

cout << "m和n的最大公因数为" << GMF(m, n) << endl;

cout << "m和n的最小公倍数为" << LCM(m, n) << endl;

return 0;

}

2.#include<iostream>

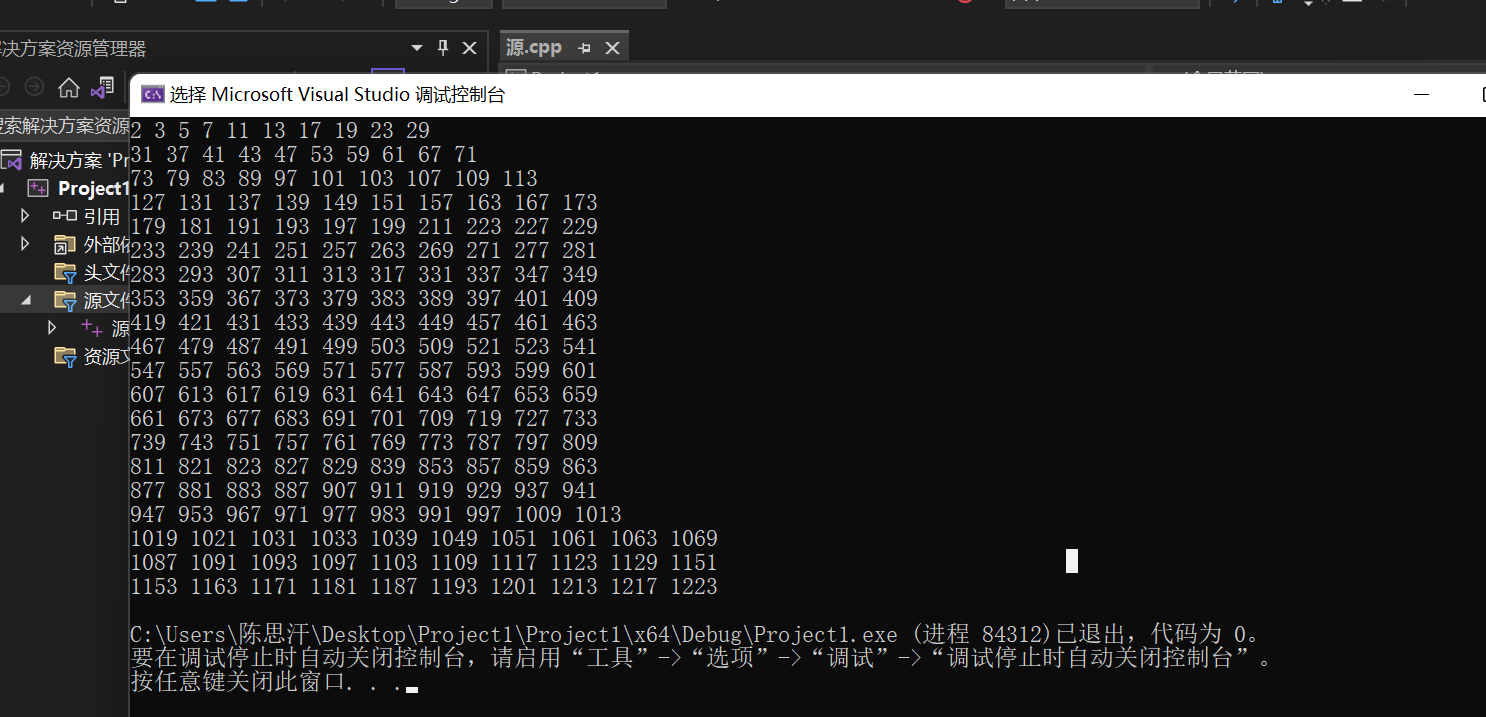
using namespace std;

bool is\_prime(int num);

int main()

{

int m=2,n=0;

 while(n<200)

{

for (int p=0; ; )

{

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

{

n += 1; p += 1;

cout << m << ' ';

}

m++;

if (p == 10)

break;

}

cout << endl;

}

return 0;

}

bool is\_prime(int num)

{

int i = num-1;

for (i; i > 1; i--)

{

if (num % i == 0)

break;

}

if (i == 1)

return true;

else

return false;

}

3.

Mytemperature.h:

#pragma once

#include<iostream>

using namespace std;

double celsius\_to\_fah(double cel)

{

double fah;

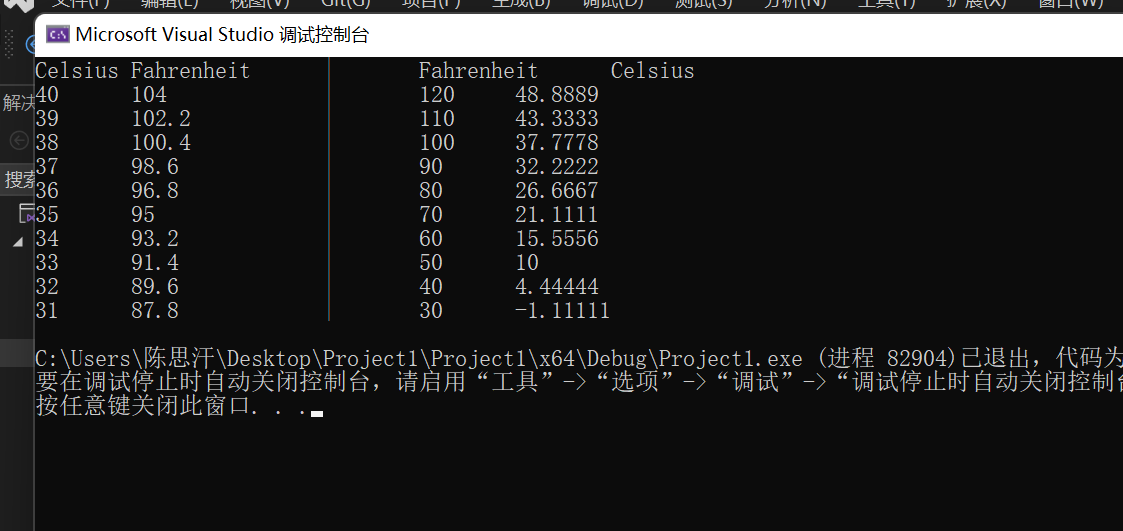
fah = cel \* 9 / 5 + 32;

return fah;

}

double fshrenheit\_to\_cels(double fah)

{

 double cel;

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

return cel;

}

源.cpp:

#include<iostream>

#include"mytemperature.h"

using namespace std;

int main()

{

cout << "Celsius" << '\t' << "Fahrenheit" << '\t' << "|" << '\t' << "Fahrenheit" << '\t' << "Celsius" << endl;

cout << 40.0 << '\t' << celsius\_to\_fah(40.0) << '\t' <<'\t' << "|" << '\t' << 120.0 << '\t' << fshrenheit\_to\_cels(120.0) << endl;

cout << 39.0 << '\t' << celsius\_to\_fah(39.0) << '\t' <<'\t' << "|" << '\t' << 110.0 << '\t' << fshrenheit\_to\_cels(110.0) << endl;

cout << 38.0 << '\t' << celsius\_to\_fah(38.0) << '\t' <<'\t' << "|" << '\t' << 100.0 << '\t' << fshrenheit\_to\_cels(100.0) << endl;

cout << 37.0 << '\t' << celsius\_to\_fah(37.0) << '\t' <<'\t' << "|" << '\t' << 90.0 << '\t' << fshrenheit\_to\_cels(90.0) << endl;

cout << 36.0 << '\t' << celsius\_to\_fah(36.0) << '\t' <<'\t' << "|" << '\t' << 80.0 << '\t' << fshrenheit\_to\_cels(80.0) << endl;

cout << 35.0 << '\t' << celsius\_to\_fah(35.0) << '\t' <<'\t' << "|" << '\t' << 70.0 << '\t' << fshrenheit\_to\_cels(70.0) << endl;

cout << 34.0 << '\t' << celsius\_to\_fah(34.0) << '\t' <<'\t' << "|" << '\t' << 60.0 << '\t' << fshrenheit\_to\_cels(60.0) << endl;

cout << 33.0 << '\t' << celsius\_to\_fah(33.0) << '\t' <<'\t' << "|" << '\t' << 50.0 << '\t' << fshrenheit\_to\_cels(50.0) << endl;

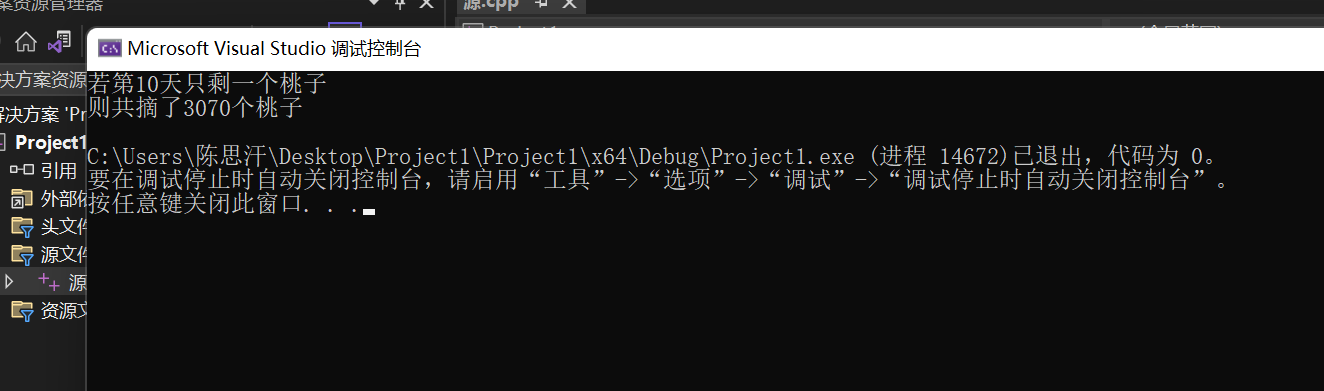
cout << 32.0 << '\t' << celsius\_to\_fah(32.0) << '\t' <<'\t' << "|" << '\t' << 40.0 << '\t' << fshrenheit\_to\_cels(40.0) << endl;

cout << 31.0 << '\t' << celsius\_to\_fah(31.0) << '\t' <<'\t' << "|" << '\t' << 30.0 << '\t' << fshrenheit\_to\_cels(30.0) << endl;

return 0;

}

5.

#include<iostream>

using namespace std;

int sum(int a)

{

int n;

if (a == 1)

n = 4;

else

n = 2 \* (sum(a - 1) + 1);

return n;

}

int main()

{

cout << "若第10天只剩一个桃子" << endl;

cout << "则共摘了" << sum(10) << "个桃子" << endl;

}

思考题一：

因为最开始定义的函数类型就是定义了函数返回值的类型，即便返回值类型与此不同也会被强制转化为定义的类型。

思考题二：

主函数调用函数时传递方式应为传递函数代码的形式。

**四、遇到的问题与解决方法**

第二题中一开始没有想到好的的方法去实现每行十个输出，经过几次尝试以后才想到应该要通过计数的方式来计算这一行已经输入了多少个，从而引入了p来计数，最终解决了这个问题。

**五、体会**

实验三还是比较基础，可能有关函数的递归会比较绕，但实际上多分析一下就可以想出解决方案。

**实验四、数组与指针**

**一、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。

**二、实验内容**

**（一）数组**

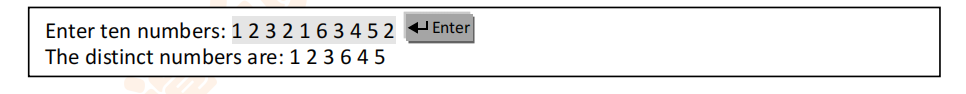
1、打印不同的数：

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：



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

Enter10

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。



5、检验子串：

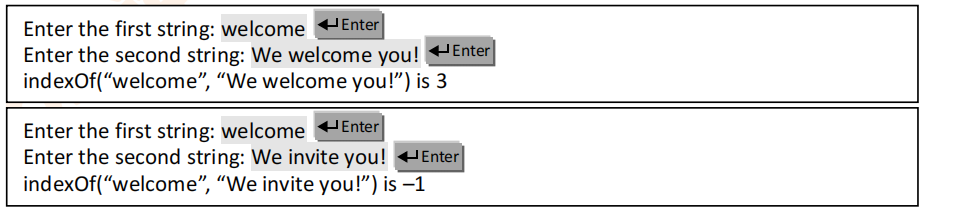
编写如下函数，检验 C 字符串 s1 是否是 C 字符串 s2 的子串。如果匹配，返回 s1 在 s2 中的

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：



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。

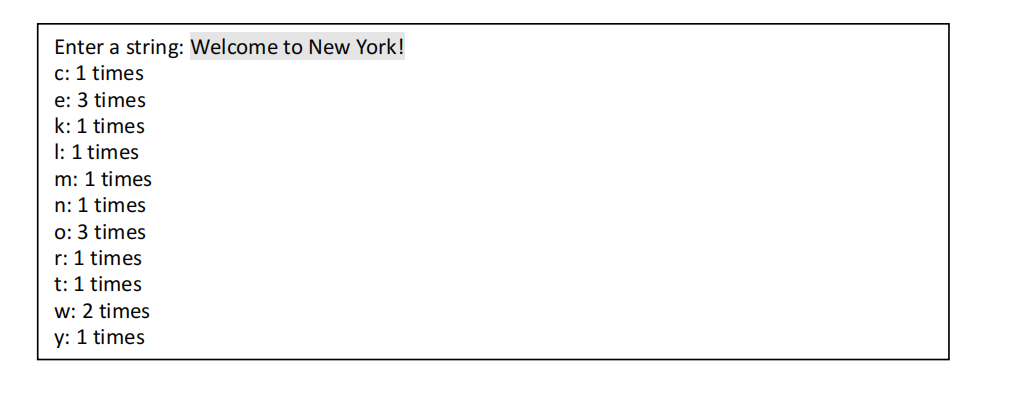
void count(const char s[], int counts[])

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：



**（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)

{12

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

(3) 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）。

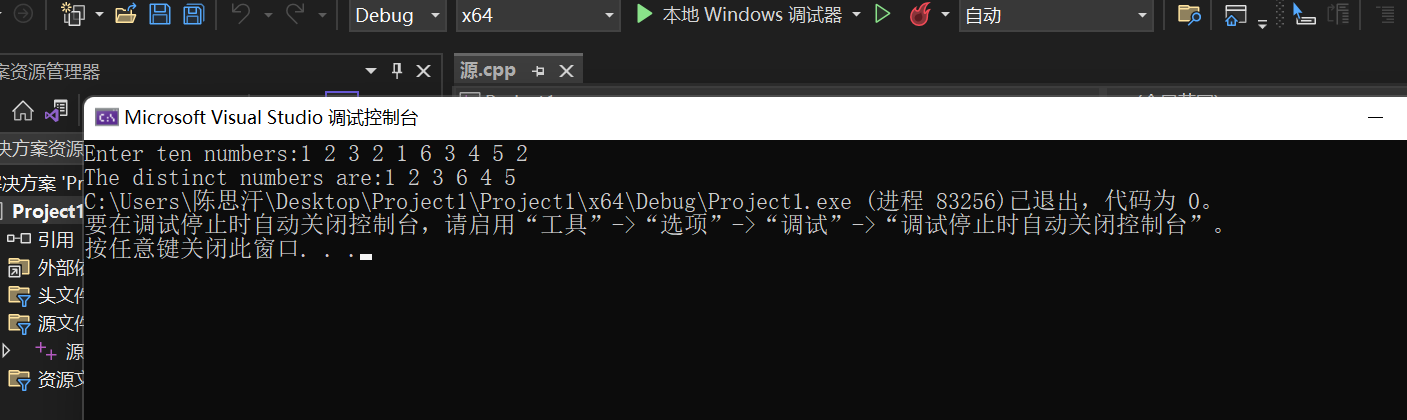
**三、程序设计**

**（一）数组：**

1.#include<iostream>

using namespace std;

int main()

{

int a[10];

cout << "Enter ten numbers:";

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

{

cin >> a[i];

}

cout << "The distinct numbers are:";

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

{

int n=0;

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

{

if (a[j] == a[k])

n++;

}

if (n == 0)

cout << a[j]<<' ';

}

return 0;

}

2.#include<iostream>

using namespace std;

int main()

{

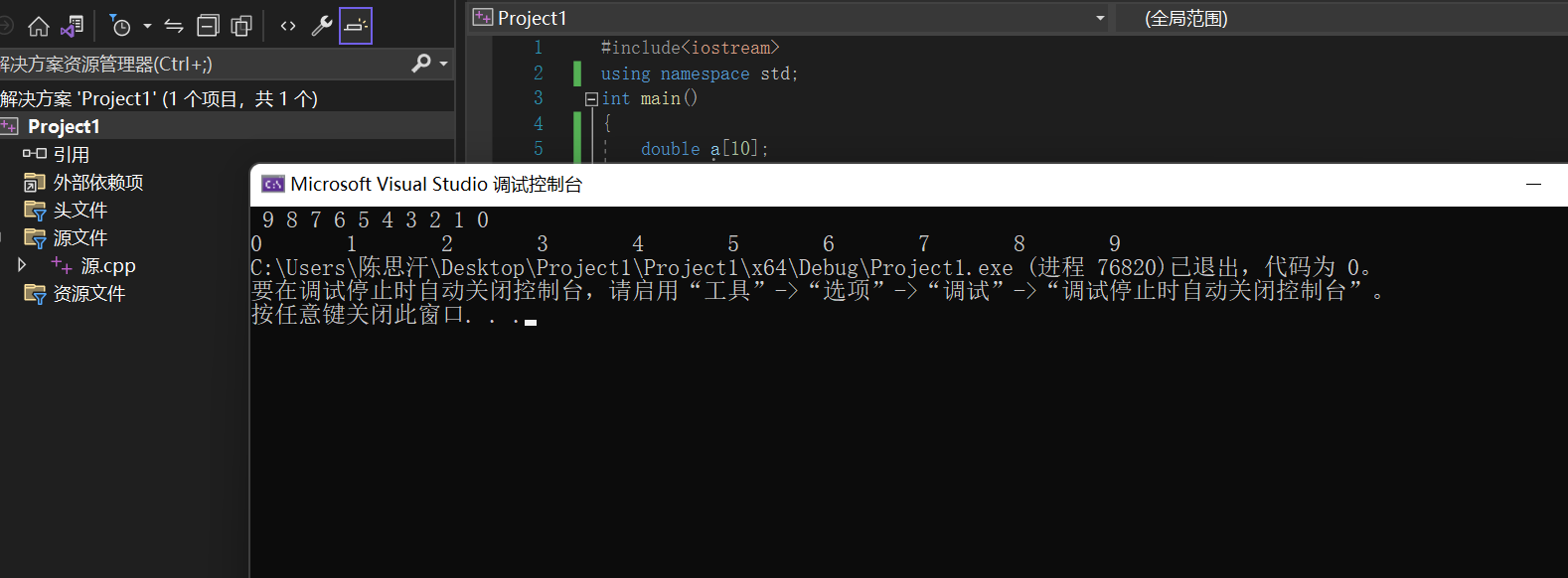
double a[10];

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

{

cin >> a[i];

}

 bool changed = true;

do

{

changed = false;

for (int j = 0; j < 9; j++)

{

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

{

int t;

t = a[j];

a[j] = a[j + 1];

a[j + 1] = t;

changed = true;

}

}

} while (changed);

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

{

cout << a[i] << '\t';

}

return 0;

}

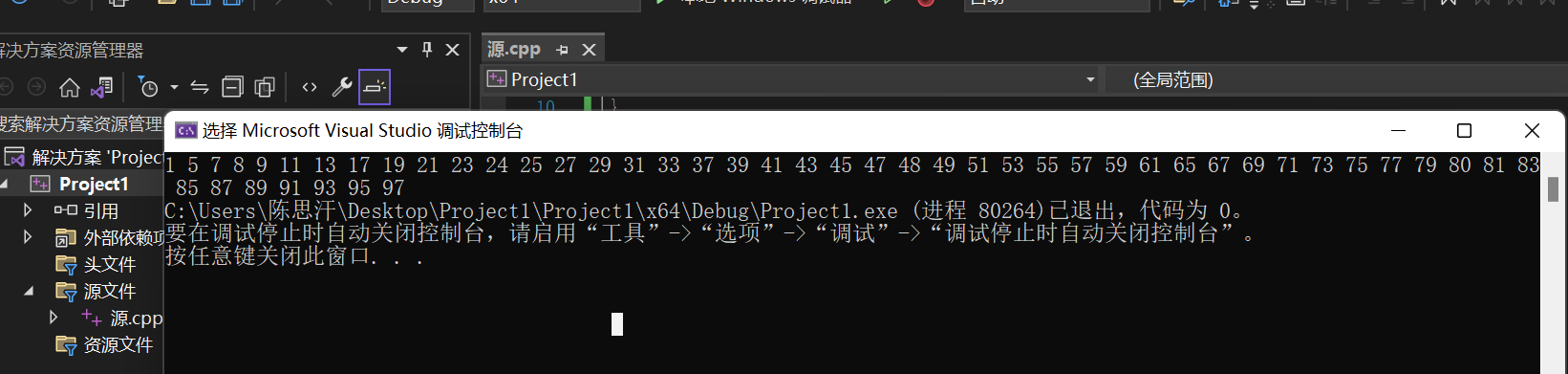
3.#include<iostream>

using namespace std;

void turn(bool &a)

{

if (a == true)

 a = false;

else

a = true;

}

int main()

{

int num[100];

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

{

num[i] = i + 1;

}//给每个柜子编号

bool L[100] ;

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

{

L[i] = false;

}//设定初始情况

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

{

turn(L[i]);

}//第一个人打开了所有柜子

for (int j = 2; j <100; j++)

{

int max = 100 / (j + 1);

for (int k = 0; k < max; k++)

{

turn(L[j+k \* (j + 1)-1]);

}

}//从第二个人到第九十九个人打开柜子的情况

turn(L[99]);//第一百个人的打开柜子情况

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

{

if (L[m] == 1)

cout << num[m] << ' ';

}

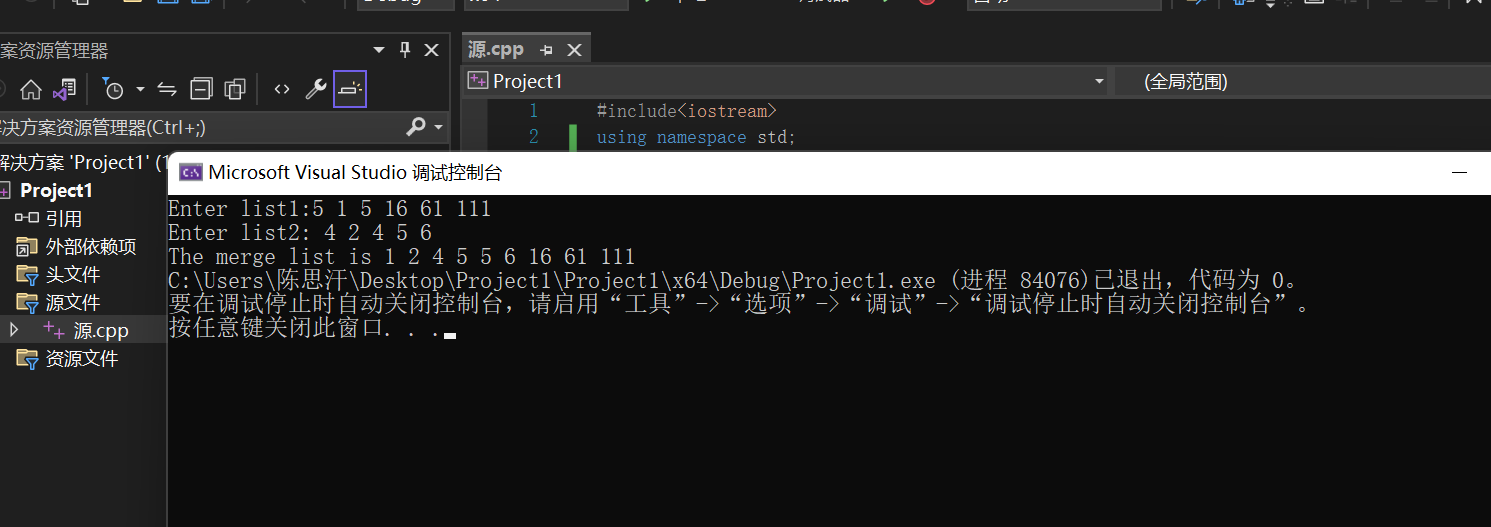
return 0;

}

4.#include<iostream>

using namespace std;

void merge(const int list1[], int size1, const int list2[], int size2, int list3[])

{

int j = 0, i = 0, m = 0;

while (j < size2 && i < size1)

{

while (list2[j] > list1[i]&&i<size1)

{

i++;

}//判断第二个数组应该插到第一个数组的哪个位置

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

for (int k = m + j; k < i + j; k++)

{

list3[k] = list1[k - j];

}

m = i;//m用于表示上次的i从而可以计算出两次插入第二个数组之间的数之间的距离并在中间补齐第一个数组的数

j++;

}

if(j==size2)

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

{

list3[i + j + n] = list1[i + n];

}//如果第二个数组已经全部插完，则剩下的全部用第一个数组后面的数来排

else

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

{

list3[i + j + n] = list2[j + n];

}//否则说明已经达到第一个数组的最后一个，剩下的全部用第二个数组来排

}

int main()

{

int size1, size2;

int list1[80], list2[80], 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 merge list is"<<' ';

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

{

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

}

return 0;

}

5.#include<iostream>

using namespace std;

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

{

int size1 = strlen(s1), size2 = strlen(s2), j = 0, num = 0;//用于计数

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

{

if (s1[0] == s2[j])

{

if (size1 > (size2 - j))

break;//此时由数组长度可判断出已无法成立

else

{

for (int k = 1; k < size1; k++)

{

if (s1[k] == s2[j + k])

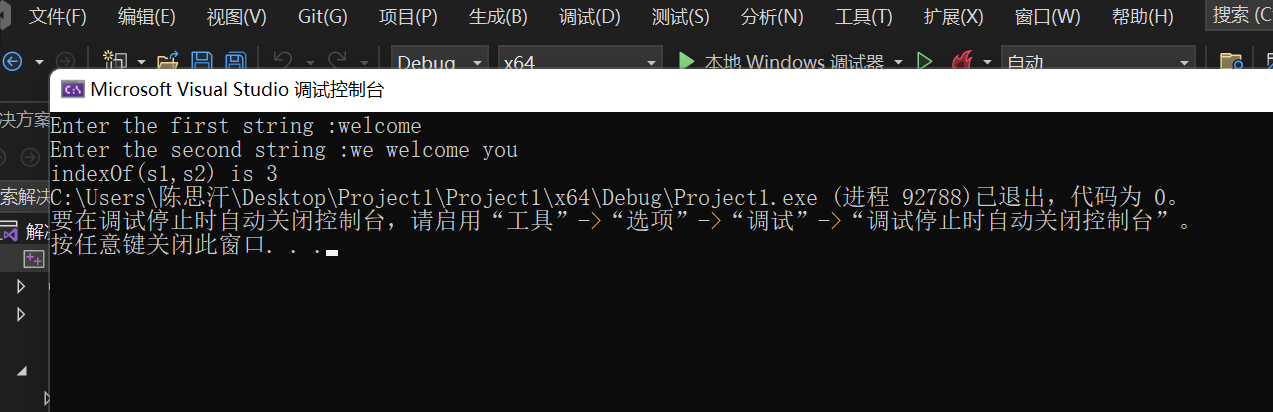
num++;

}

}

}

if (num == size1 - 1)//如果相等，则说明从第二个字母开始到最后一个字母都相等

 break;

else

num = 0;

}

if (num == size1 - 1)

return j;

else

 return -1;

}

int main()

{

char s1[1000],s2[10000];

cout << "Enter the first string :";

gets\_s(s1);

cout << "Enter the second string :";

gets\_s(s2);

cout << "indexOf(s1,s2) is" << ' ' << indexOf(s1, s2);

}

6.#include<iostream>

using namespace std;

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

{

int size = strlen(s);

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

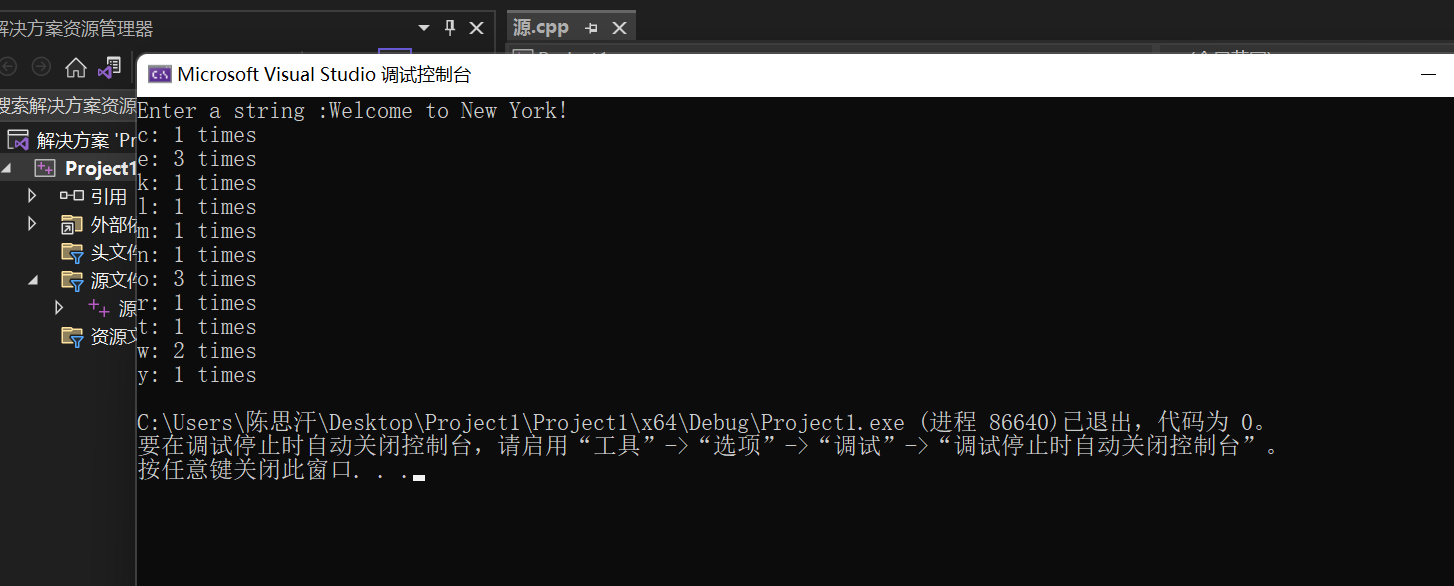
{

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

{

if ((s[i] + 0) == (65 + j) || (s[i] + 0) == (97 + j))//用ASCII码来判断

counts[j]++;

 }

}

}

int main()

{

char s[10000];

int counts[26] = { 0 };

cout << "Enter a string :";

gets\_s(s);

count(s, counts);

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

{

if (counts[i] != 0)

{

char letter = i + 97;

cout << letter << ':' << ' ' << counts[i] << ' ' << "times" << endl;

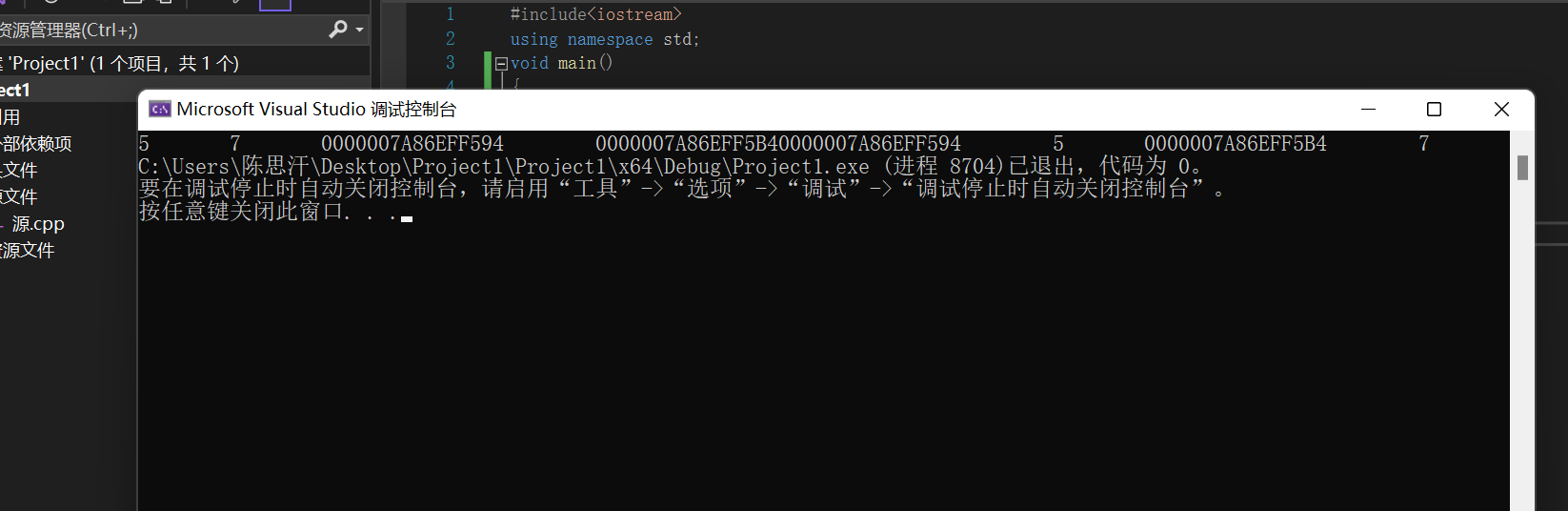
}

}

return 0;

}

**（二）指针：**

1.

（1）#include<iostream>

using namespace std;

void main()

{

int i, j, \* pi, \* pj;

pi = &i;

pj = &j;

i = 5; j = 7;

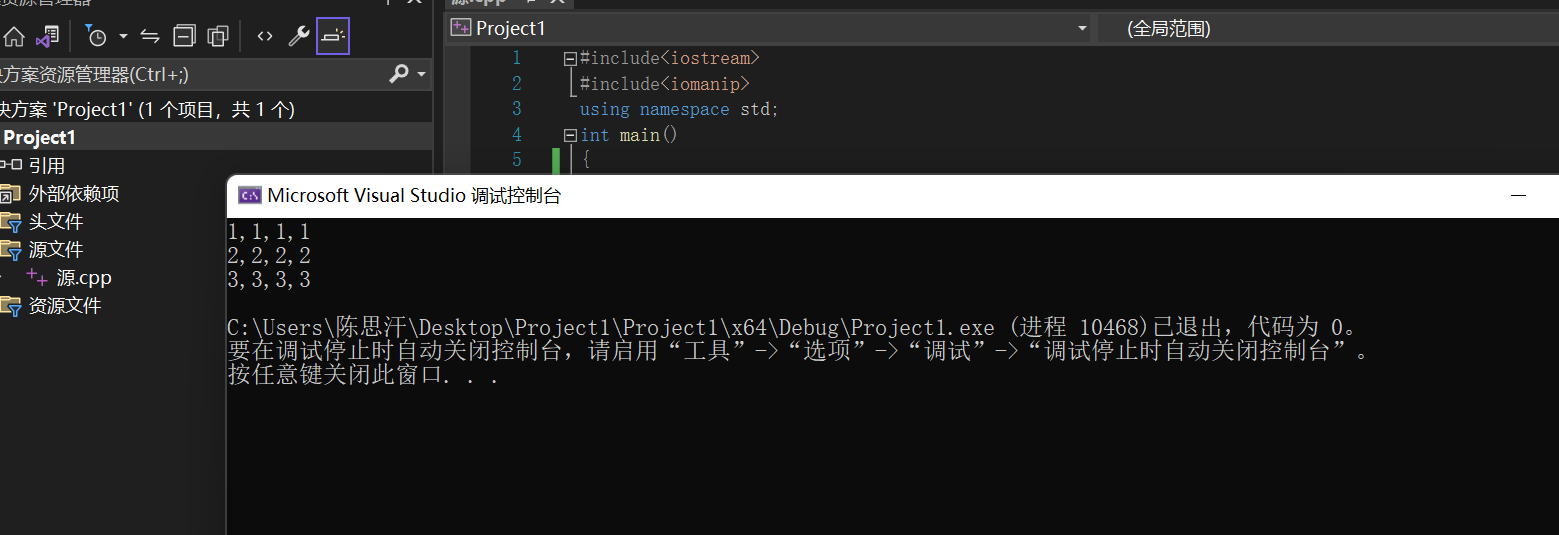
cout << i << '\t' << j << '\t' << pi << '\t' << pj;

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

}

（2）#include<iostream>

#include<iomanip>

using namespace std;

int main()

{

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

int\* p,i;

p = a;

for (i = 0; i < 3; i++)

{

printf("%d,%d,%d,%d\n", a[i],p[i], \* (p + i), \*(a + i));

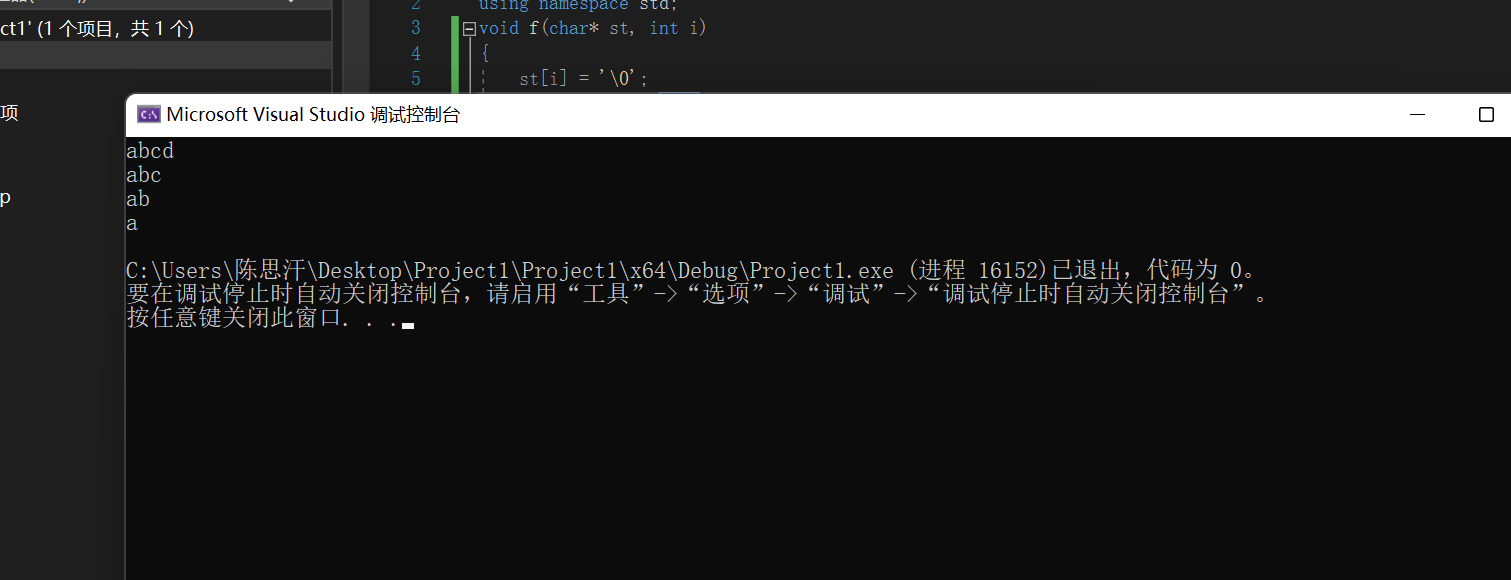
}

return 0;

}

（3）#include<iostream>

using namespace std;

void f(char\* st, int i)

{

st[i] = '\0';

cout << st << endl;

if (i > 1)

f(st, i - 1);

}

void main()

{

char st[] = "abcd";

f(st, 4);

}

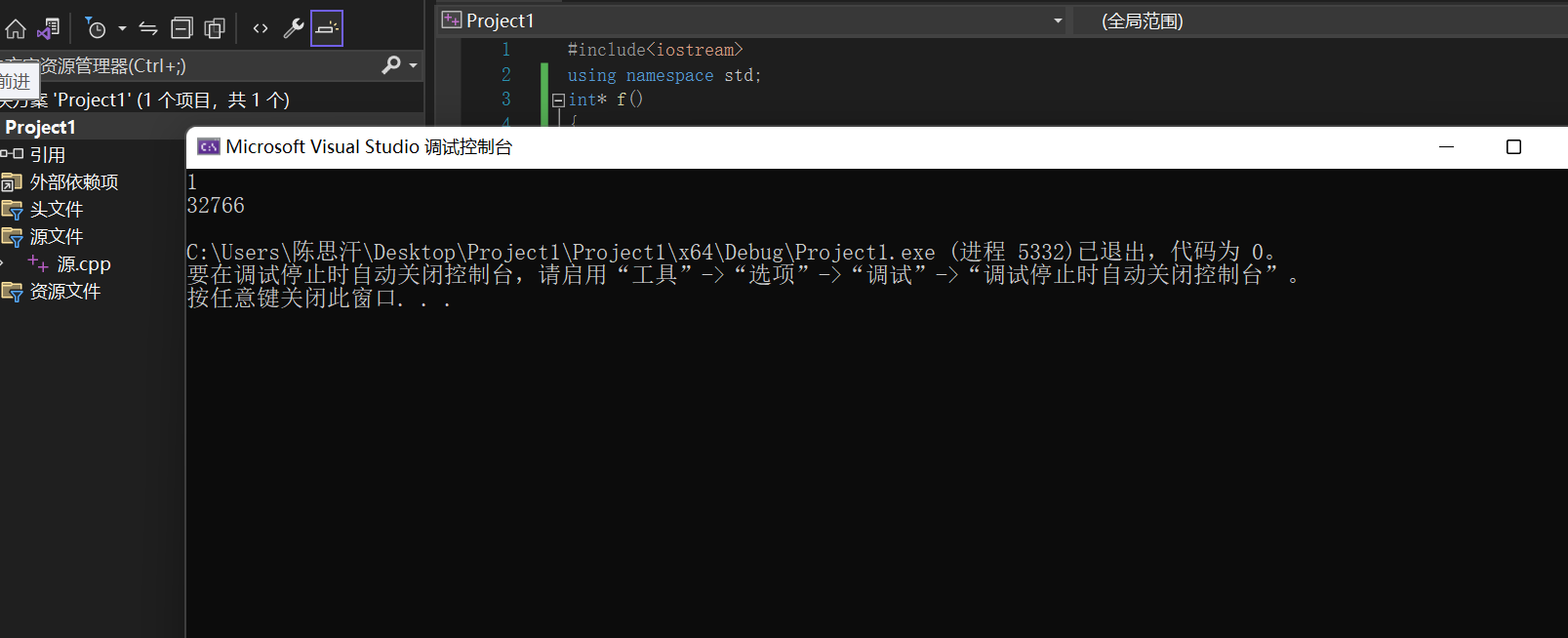
（4）

修改前：

#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;

}

修改后：

#include<iostream>

using namespace std;

int\* f()

{

int\* q;

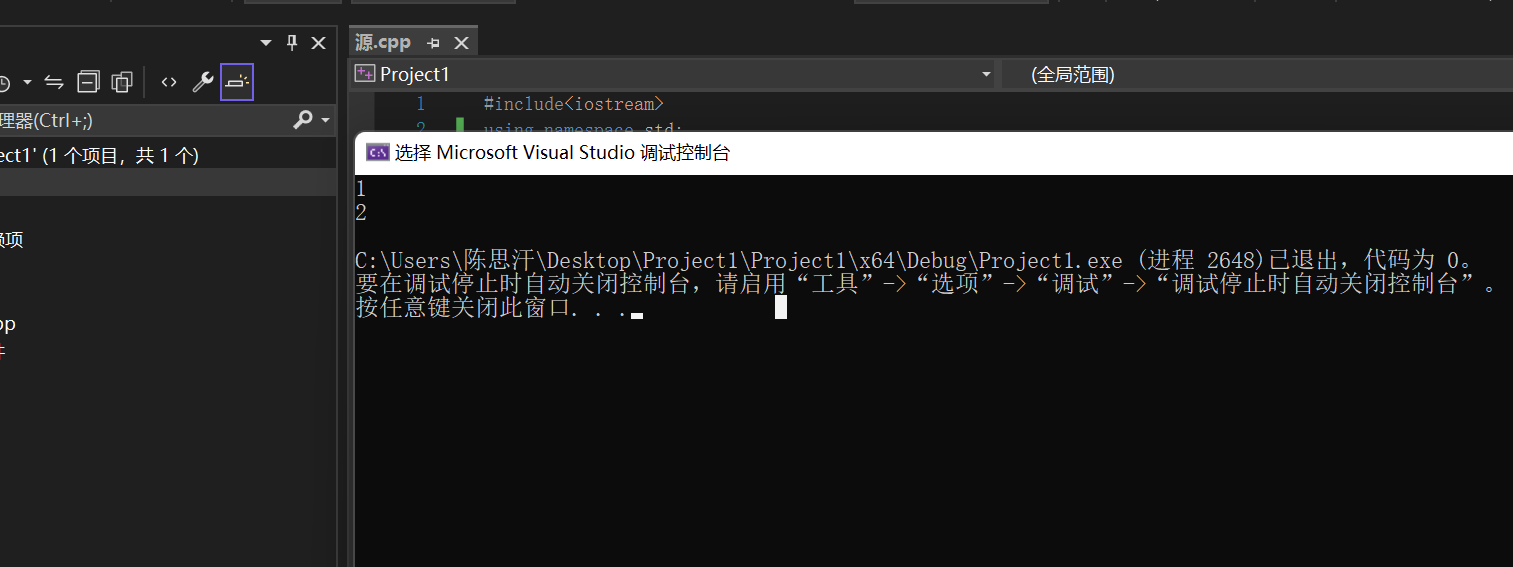
q=new int [4];

for (int i = 0; i < 4; i++)

{

q[i] = i + 1;

}

 return q;

}

void main()

{

int\* p = f();

cout << p[0] << endl;

cout << p[1] << endl;

delete[]p;

}

2.#include<iostream>

using namespace std;

int indexof(const char\* s1,const char\*s2)

{

int size1 = strlen(s1), size2 = strlen(s2),j=0,num=0;//用于计数

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

{

if (s1[0] == s2[j])

{

if ((size1 - 1) > (size2 - j))

break;//此时由数组大小可判断出已不能成立

else

for (int i = 1; i < size1; i++)

{

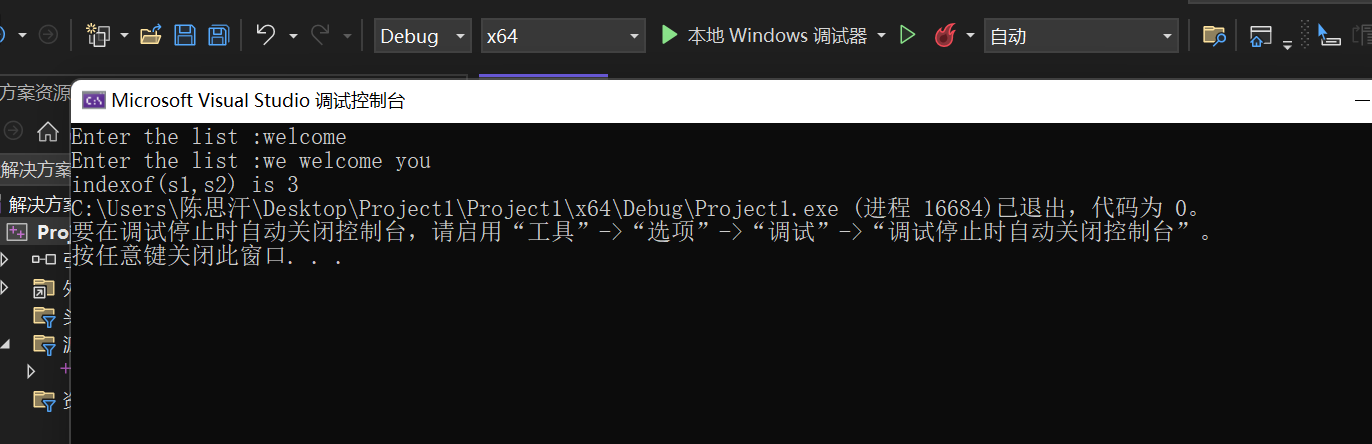
if (s1[i] == s2[j + i])

num++;

}

}

if (num == size1 - 1)//如果相等，则说明从第一个字母到最后一个字母都相等

 break;

else

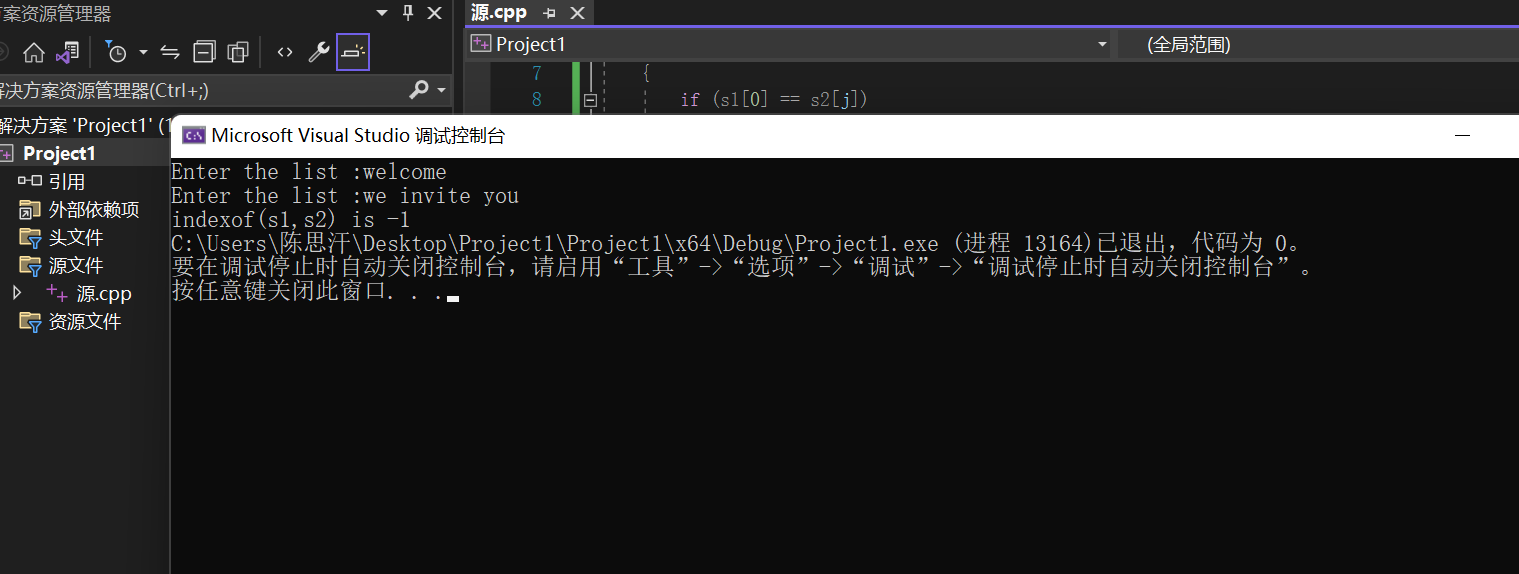
num = 0;

}

if (num == (size1 - 1))

return j;

else

 return -1;

}

int main()

{

char s1[1000],s2[10000];

cout << "Enter the list :";

gets\_s(s1);

cout << "Enter the list :";

gets\_s(s2);

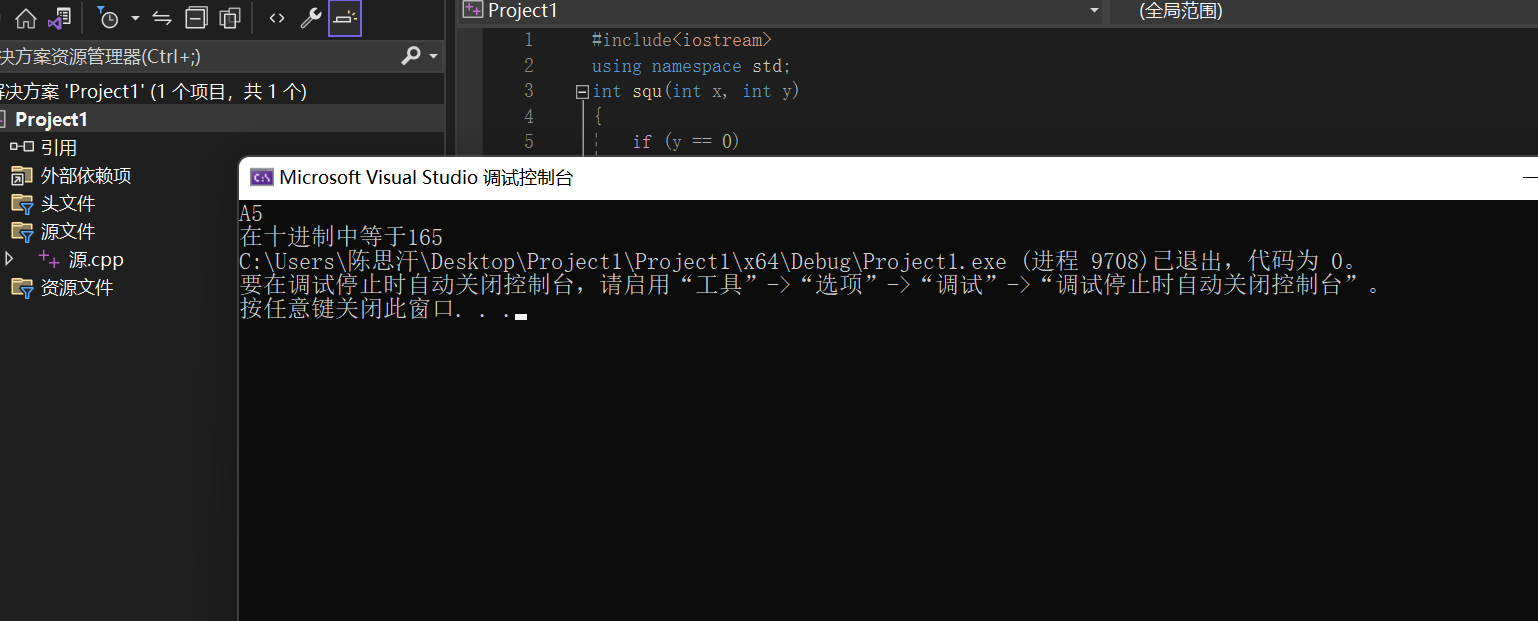
cout << "indexof(s1,s2) is" << ' ' << indexof(s1, s2);

return 0;

}

3.#include<iostream>

using namespace std;

int squ(int x, int y)

{

if (y == 0)

return 1;

else

return x \* squ(x, y - 1);

}

int parseHex(const char\* const hexstring)

{

int size = strlen(hexstring);

int\* p, answer = 0;

p= new int[size];

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

{

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

p[i] = hexstring[i] - 48;

else if ('a' <= hexstring[i] && hexstring[i] <= 'f')

p[i] = hexstring[i] - 87;

else if ('A' <= hexstring[i] && hexstring[i] <= 'f')

p[i] = hexstring[i] - 55;

answer += p[i] \* squ(16, size - i - 1);

}

delete[]p;

return answer;

}

int main()

{

char hexstring[1000];

cin >> hexstring;

int answer = parseHex(hexstring);

cout << "在十进制中等于" << answer;

return 0;

}

4.#include<iostream>

using namespace std;

void sort(int a[],int n)

{

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

{

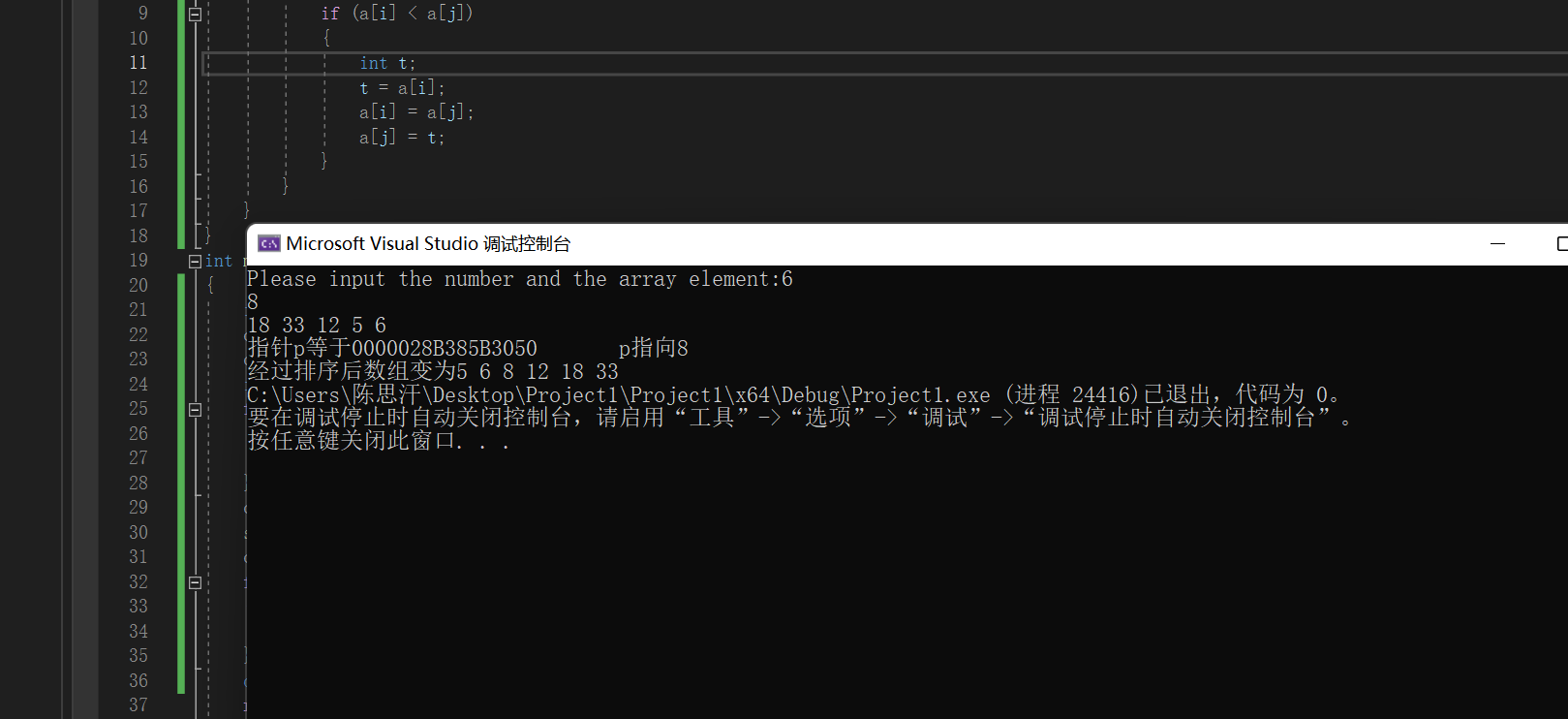
for (int j = 0; j < i; j++)

{

if (a[i] < a[j])

{

int t;

 t = a[i];

a[i] = a[j];

a[j] = t;

}

}

}

}

int main()

{

int size;

cout << "Please input the number and the array element:";

cin >> size;

int\* p = new int[size];

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

{

cin >> p[i];

}

cout << "指针p等于" << p << '\t' << "p指向" << \*p << endl;

sort(p, size);

cout << "经过排序后数组变为";

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

{

cout << \*(p + i) << ' ';

}

delete[]p;

return 0;

}