**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：8209230102

姓 名：刘喜贺

**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？

函数类型决定了返回值的类型，在定义函数时，根据需要的返回值类型，给函数定义相应类型。

2. 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1中调用函数采用的是引用传递 2、4、5中采用的是值传递。

**四、算法分析，程序结果**

1.

#include<iostream>

using namespace std;

int Gcd(int &m, int &n)//定义最大公因数函数

{

int temp;//定义临时变量

for (int i = 1; i < m || i < n; i++)//设置循环

{

if (m % i == 0 && n % i == 0)//如果i作为a,b的取余数均为0 就都是a,b的因数

{

temp = i;//将目前最大的i覆盖赋值到临时变量

}

}

return temp;//返回临时变量

}

int LCM(int &m, int &n)//定义最小公倍数函数

{

int temp;//定义临时变量

for (int i = m; i <= (m \* n); i++)//设置循环

{

if (i % m == 0 && i % n == 0)//如果i作为被取余数,分别取余a,b结果为0 就是a,b的公倍数

{

temp = i;//将公倍数赋值到临时变量

break;//找到的第一个公倍数就是最小公倍数 跳出循环

}

}

return temp;//返回临时变量

}

int main()

{

int m, n;//定义m,n

cout << "请输入m的值:";//提示输入m的值

cin >> m;//输入m

cout << "请输入n的值:";//提示输入n的值

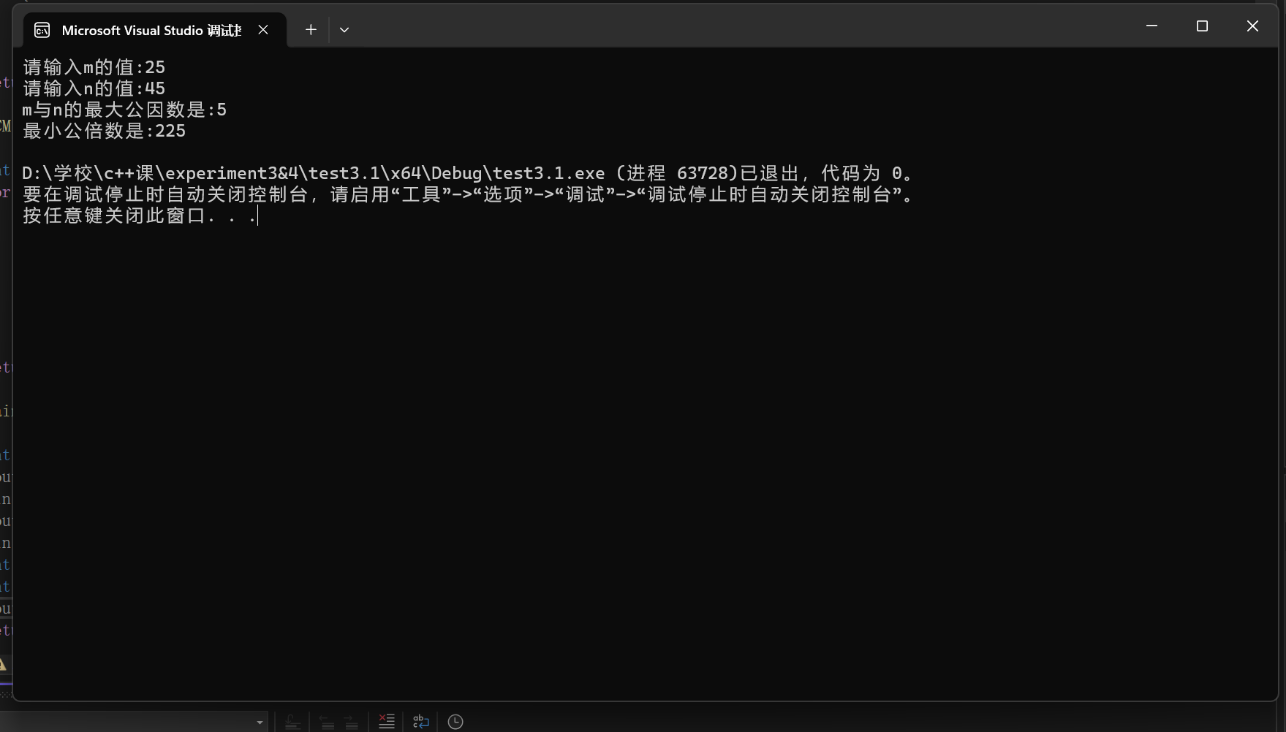
cin >> n;//输入n

int gcd = Gcd(m, n);//调用最大公因数函数

int lcm = LCM(m, n);//调用最小公倍数函数

cout << "m与n的最大公因数是:" << gcd << endl << "最小公倍数是:" << lcm << endl;//输出结果

return 0;

}

2. #include<iostream>

using namespace std;

bool is\_prime(int num)//定义判断素数的函数

{

int j = 0;//定义j=0

for (int i = 1; i <= num; i++)//设置循环枚举

{

if (num % i == 0)

{

j++;//每有一个因数,j就+1

}

}

if (j == 2)//如果j=2,即num只有两个因数,就是素数,返回true

{

return true;

}

else {//反之,返回false

return false;

}

}

int main()

{

bool prime;//定义prime用于接收函数返回值

int n=1;//定义n用来枚举

for (int j = 0; ;)//设置循环

{

n++;//n+1(从2开始枚举)

prime = is\_prime(n);//接收函数返回值

if (prime == true)//如果是素数

{

cout << n << ' ';//输出该数字

j++;//计数+1

if (j % 10 == 0)//如果已经输出了10个数字

{

cout << endl;//换行

}

}

if (j == 200)//如果输出了200个数字

{

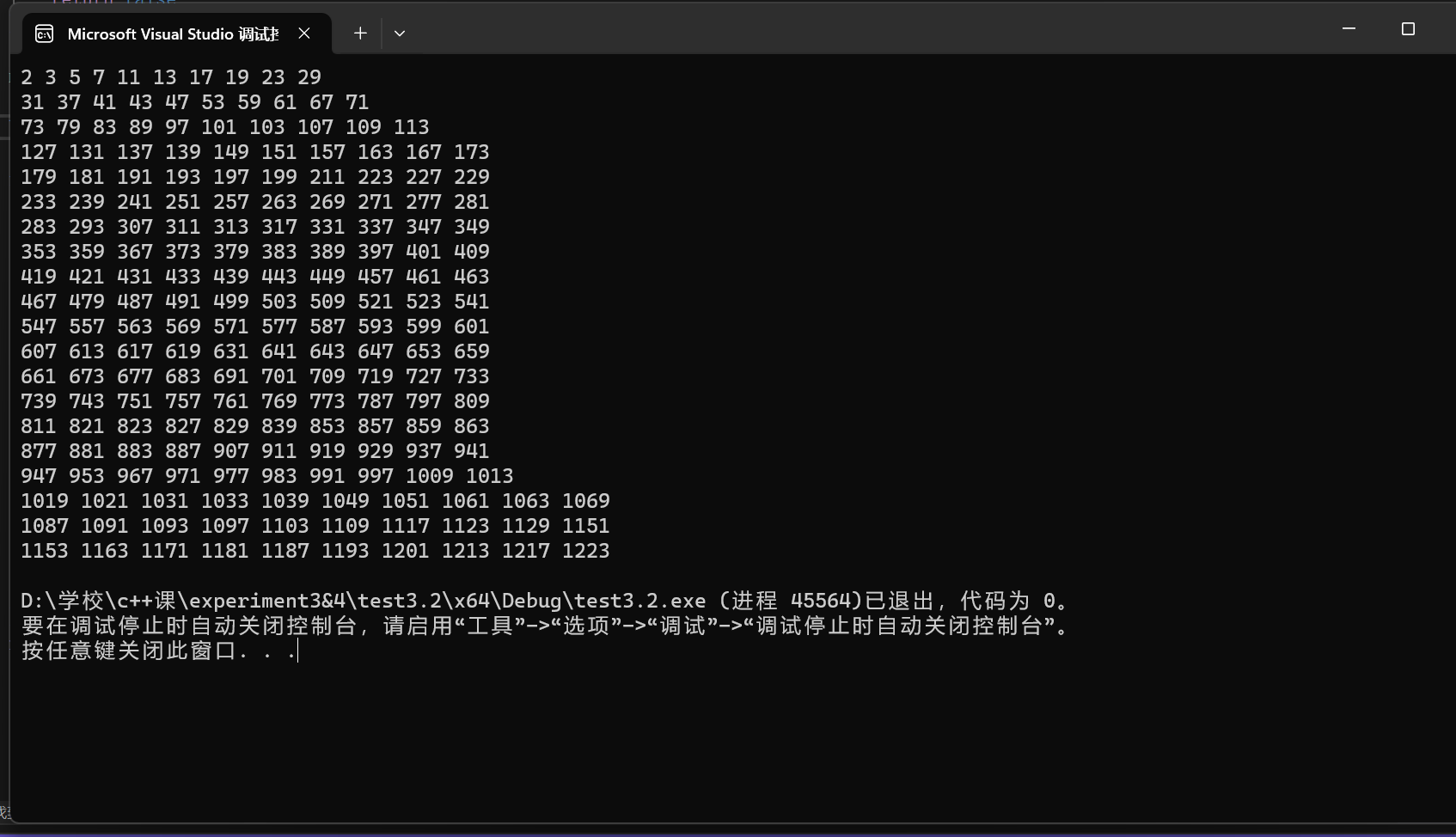
break;//跳出循环,结束程序

}

}

return 0;

}



3.

4. **mytriangle.h:**

#include<iostream>

using namespace std;

bool is\_valid(double side1, double side2, double side3);//定义判断三角形是否有效的函数

double area(double side1, double side2, double side3);//定义计算面积函数

**mytriangle.cpp:**

#include"mytriangle.h"

bool is\_valid(double side1, double side2, double side3)//判断三角形是否有效

{

if (side1 + side2 > side3 && side1 + side3 > side2 && side2 + side3 > side1)//两边之和大于第三边

{

return true;//返回true

}

else

{

return false;//返回false

}

}

double area(double side1, double side2, double side3)//计算面积

{

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

double square = sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));//面积公式

return square;//返回面积大小

}

**test3.4.cpp:**

#include"mytriangle.h"

int main()

{

int side1, side2, side3;//定义三边长

cout << "请分别输入三边边长" << endl;//提醒输入三边长

cin >> side1 >> side2 >> side3;//输入三边长

int val = is\_valid(side1, side2, side3);//调用判断三角形是否有效的函数

if (val == true)//如果有效

{

double square = area(side1, side2, side3);//调用计算面积函数

cout << "您定义的三角形面积为:" << square << endl;//输出面积

}

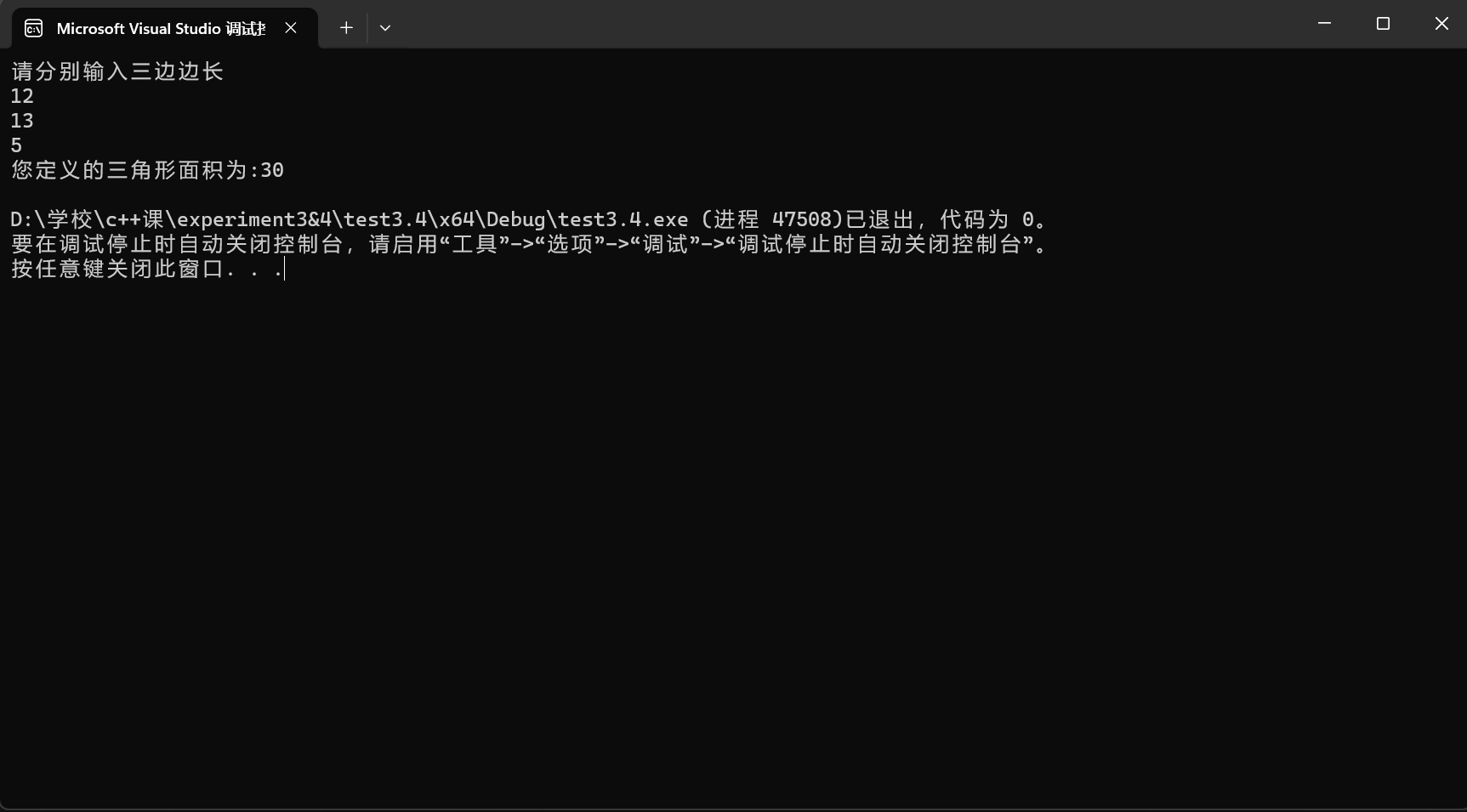
else//如果无效

{

cout << "您定义的三边长不合法" << endl;//提示无效

}

return 0;

****}

5. #include<iostream>

using namespace std;

int peach\_number(int num1,int num2)//定义计算数量的函数

{

if (num2 != 0)//判断天数是否足够

{

int temp = (num1 + 1) \* 2;//计算前一天的桃子数

num1 = temp;//赋值给num1

num2--;//天数-1

return peach\_number(num1, num2);//递归

}

else//如果天数达到要求

{

return num1;//返回计算出的num1

}

}

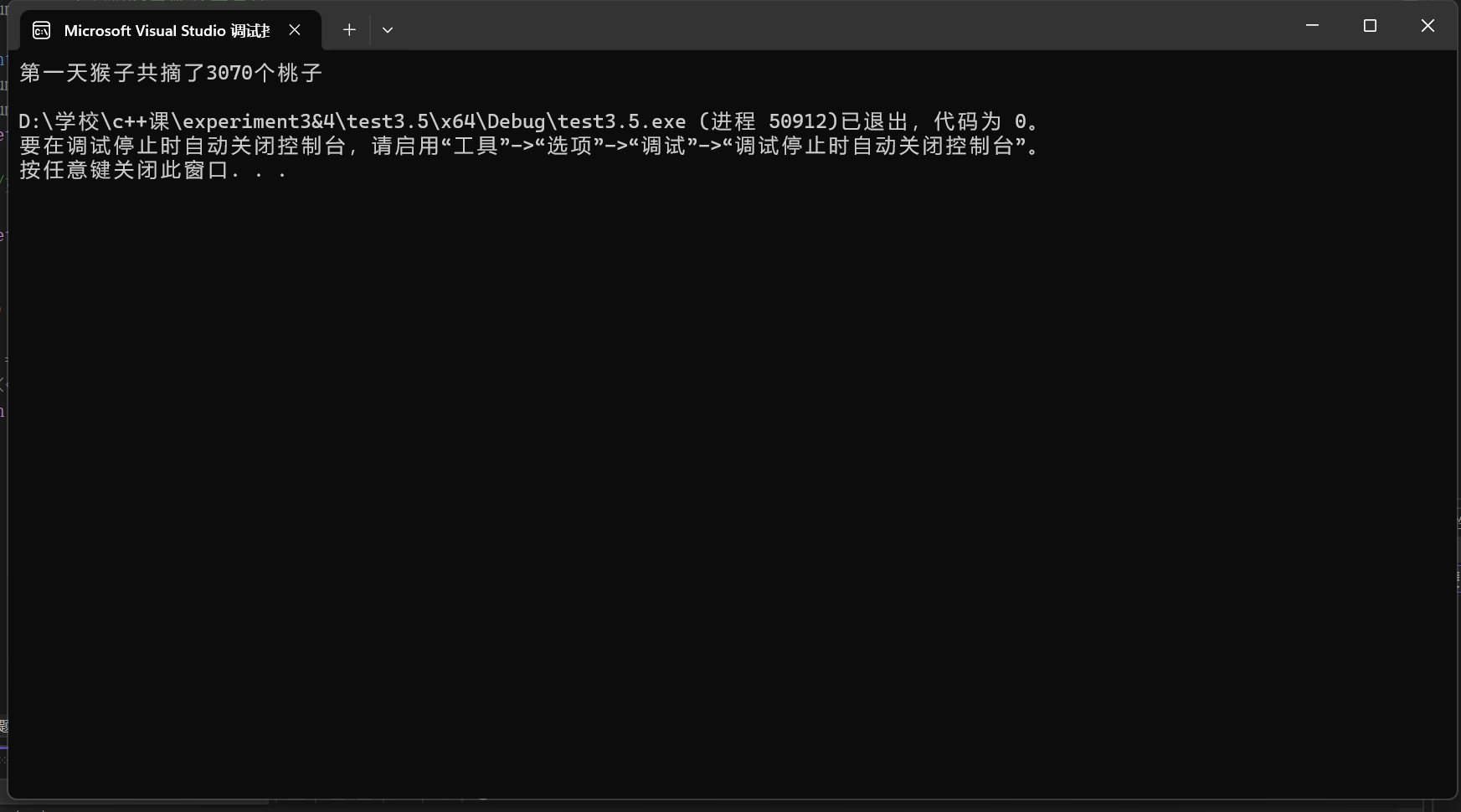
int main()

{

int n = 1;//第十天只剩一个桃子

cout << "第一天猴子共摘了" << peach\_number(n,10) << "个桃子" << endl;//调用函数,分别输入现在个数及天数

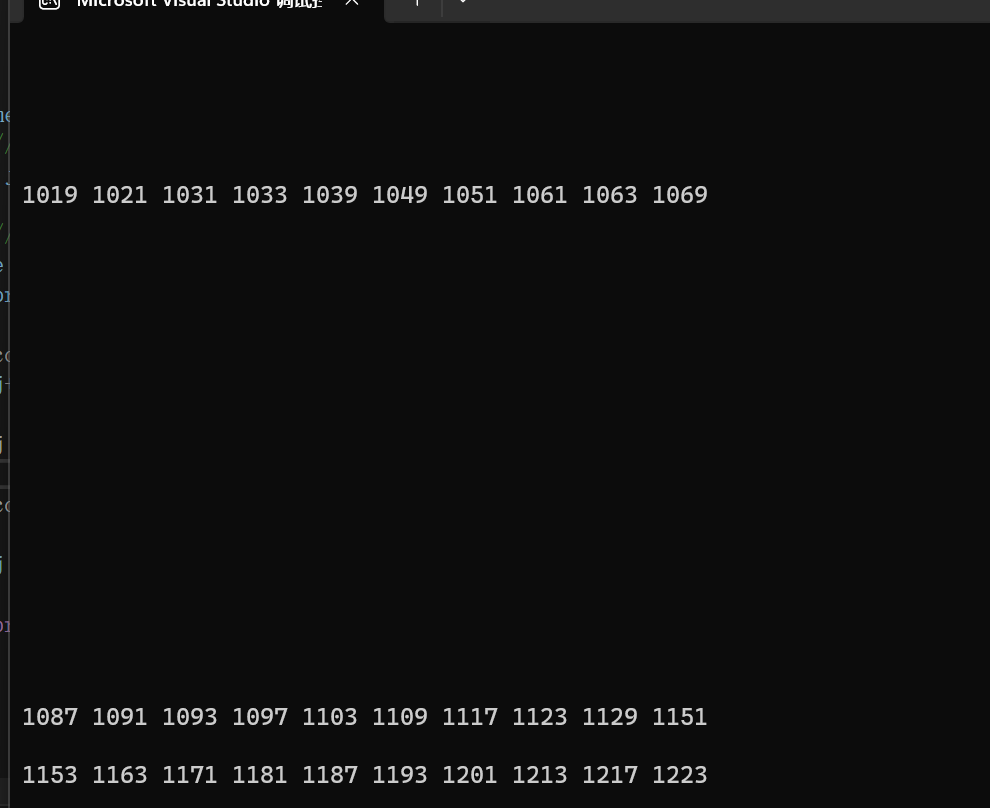
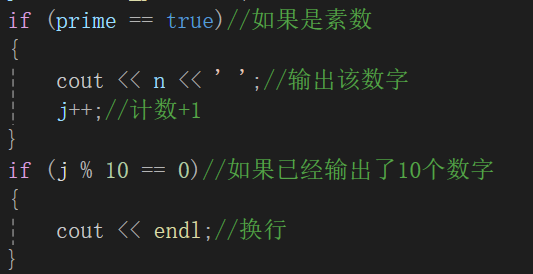
return 0;

}

**五、遇到的问题与解决方法**

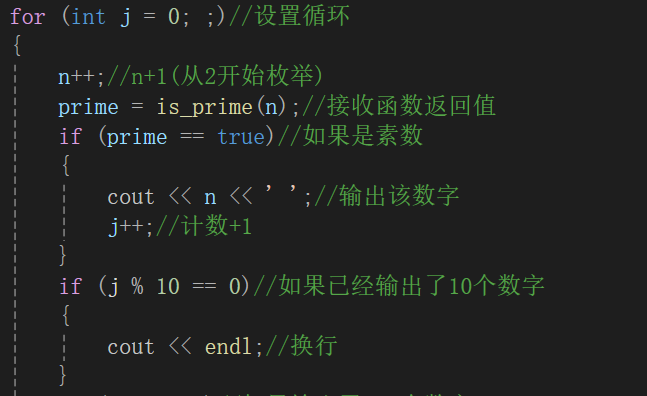
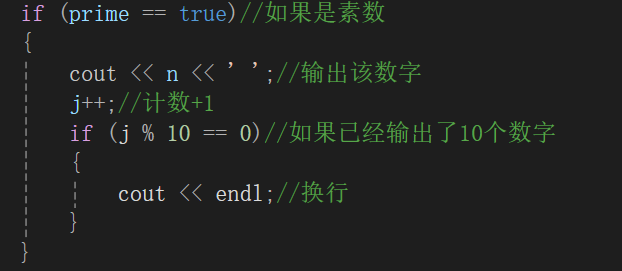
**1.在输出素数时出现问题**

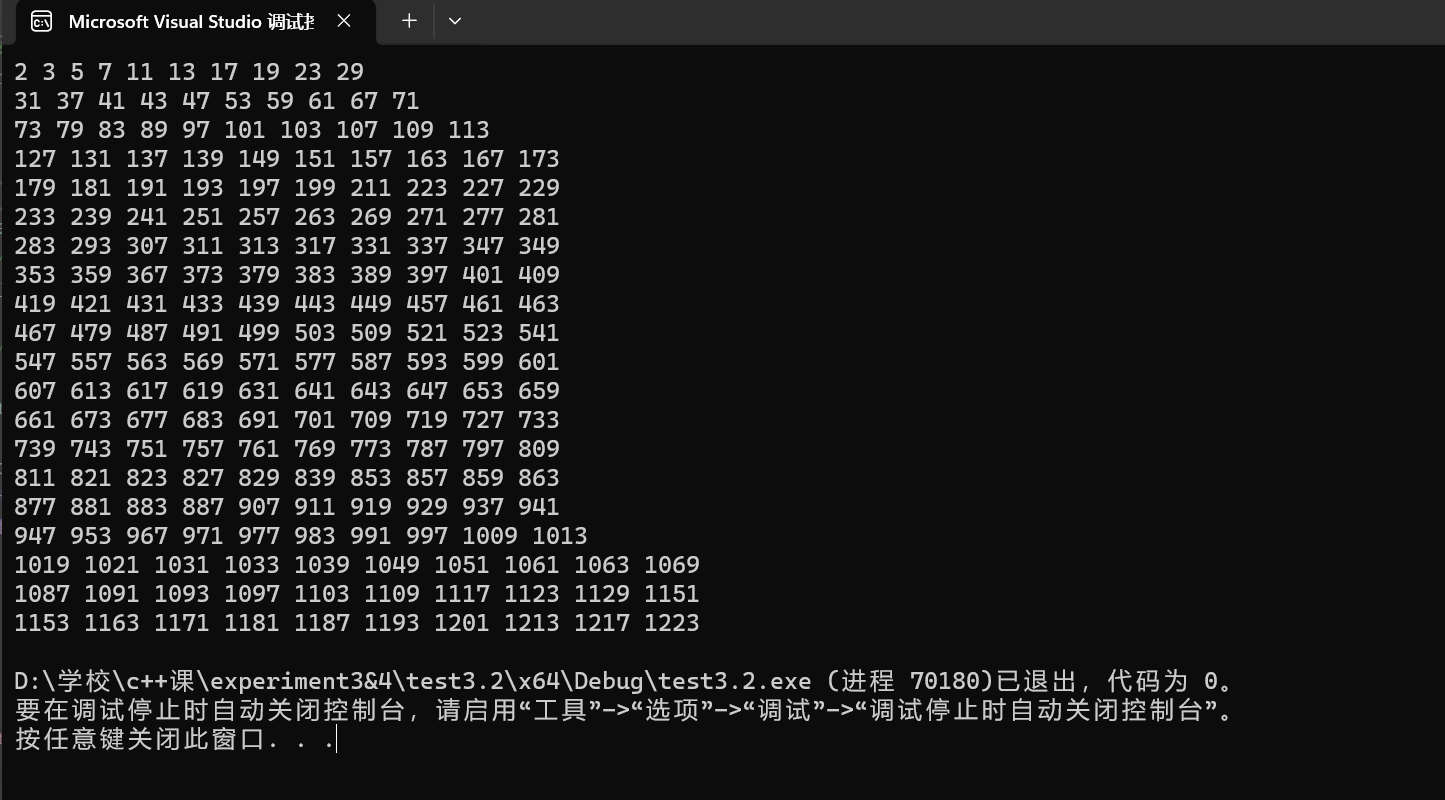
在程序设计要求中，每输出10个数据就要进行换行，但是我在最后的输出阶段，换的行数出现问题，如图所示：

可以发现在某些行之间的空行非常多，考虑是endl重复执行的问题。

思考为何会出现endl重复执行的情况，想到本程序的在这一块的逻辑是：每找到一个素数，计数j就加一，当j为十的倍数时，即取余10为0时进行换行：

而在本程序中，利用循环进行了枚举，也就是说在某次找到一个素数后，满足j%10==0时，执行一次换行，然后执行下次循环，而下次循环没有找到素数，j的计数不变，也就是说，依旧满足j%10==0，所以还会进行一次换行，以此类推，在每行之间就有可能存在很多的空行。

既然空行过多是由循环引起的，考虑减少循环对换行的影响，对代码进行优化。

将换行的判断语句包含在了素数的判断语句中，也就是说，只有在找到素数后，且素数计数为10的倍数，才会执行换行，可以解决问题。以下是解决问题后的效果图：

**六、体会**

**1.要善于使用函数**

函数可以起到分割功能、简化代码的作用。在编写大型项目或者编写代码量较高的程序时，使用函数可以使自己的编程思路更加清晰，也可以提高自己代码的可读性。

在使用函数量较大时，可以分文件进行编写，在头文件中把所有函数进行定义，随后对各函数进行功能划分，把他们各自分进其他源文件中，最后在主函数所在源文件调用函数，这样编写代码可读性及逻辑性很强。

在使用函数的时候，要注意一些细节，否则可能会出现错误。如要改变参数的值的时候，要进行地址传递或者引用传递，仅仅使用值传递，只能改变形参的值，不能改变实参的值。再比如在调用函数时，可以善用const来防止自己进行错误操作。

在定义及调用函数时，也要记得使用一些技巧，减少代码量，增加程序的可读性，妥善运用函数重载以及数据传递，使自己编写的程序更上一层楼去。

**2.对于循环以及判断要细心**

在本次实验过程中，我发现，我的程序总是会在循环或者嵌套循环或者判断逻辑上出现一些问题，这些问题都是因为不细心以及思路不够清晰造成的。

例如，在编写打印前200位素数的程序时，我由于将换行判断单独放在了循环里面，导致换行的指令被重复执行，最后直接作用到了输出的内容中：行与行之间会存在大量的空行。这一问题也大大地降低了我的编程效率，在寻找这一问题以及思考怎么解决时，耗费了我大量的时间以及精力，这就是由于我在书写循环以及判断的过程中，对逻辑不够细心造成的。

实际来讲，在编程过程中出现逻辑错误是难以避免的，但是在发现出现错误后，也要能够细心发现问题所在，要争取能用最短的时间将问题解决。

**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<iostream>

using namespace std; //#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\_\_\_abcdabcaba\_\_

(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**

**（一）数组**

1. #include<iostream>

using namespace std;

int main()

{

int arr[10];//定义数组

int temparr[10];//定义临时数组

int num = 0;//定义显示数

int temp=0;//定义临时变量记录是否有数据重复

cout << "请输入数组中的10个数" << endl;//提示输入数据

for (int i = 0; i < 10; i++)//搭建循环输入数据

{

cin >> temparr[i];//输入数据

}

for (int i = 0; i < 10; i++)//搭建循环判断赋值

{

temp = 0;//将临时记录变量赋为0

for (int j = 0; j < i; j++)//创建内层循环

{

if (temparr[i] == temparr[j])//如果目前的数值和前面的数值有重复

{

temp++;//临时变量+1

}

}

if (temp == 0)//如果临时变量为0 即不存在重复

{

arr[num] = temparr[i];//将不重复的数值在数组中的最新一位赋值

num++;//显示数+1

}

}

cout << "您输入的数字是(已删去重复值)" << endl;//赋值后对数组进行打印输出

for (int i = 0; i < num; i++)//只打印在显示数内的数据

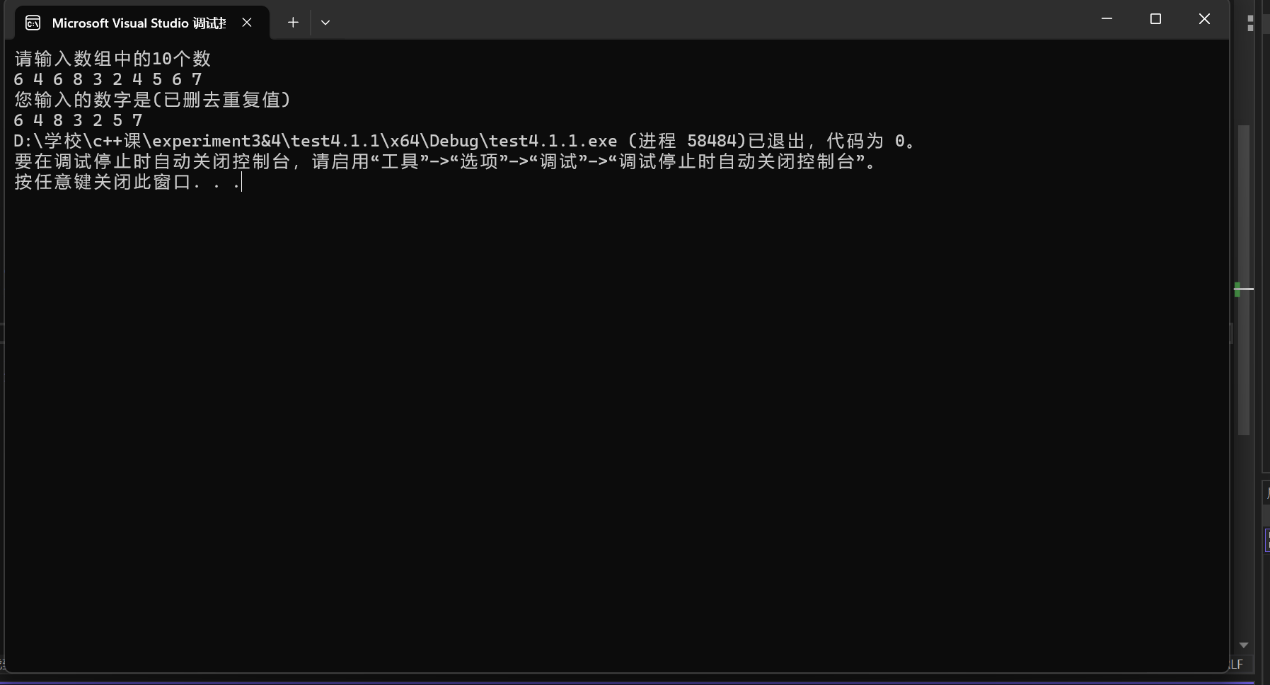
{

cout << arr[i] << ‘ ’;//打印数据

}

return 0;

}



2. #include<iostream>

using namespace std;

void bubbleSort(int\* arr, int len)//定义冒泡排序函数

{

for (int i = 0; i < len - 1; i++)//比较轮数:比较一次 可以比出 一个最大值 要将10个数排序 仅需比出9次(10-1)最大值 即可确定排序

{

for (int j = 0; j < len - 1 - i; j++)//比较的数字:第一次要10个数前后比较 比较九回 选出最大值 第二次要9个数相互比较 比较8回 选出除了第一次选出的数以外的最大值

//以此类推 比较9次 比较的回数=10-比较过的次数

{

if (arr[j] > arr[j + 1])//如果前一项 大于 后一项 调换前后两项(如果是<就是降序)

{

int temp = arr[j];//换项记得创建临时变量

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

arr[j + 1] = temp;

}

}

}

}

int main()//主程序

{

int arr1[10];//定义数组

cout << "请输入数组的数值" << endl;//提示定义数组

for (int i = 0; i < 10; i++)//循环定义数组

{

cin >> arr1[i];//输入数组

}

int num = (sizeof(arr1) / sizeof(arr1[1]));//计算数组长度

bubbleSort(arr1, num);//调用函数

cout << "排序后的数组为" << endl;//提示排序结果

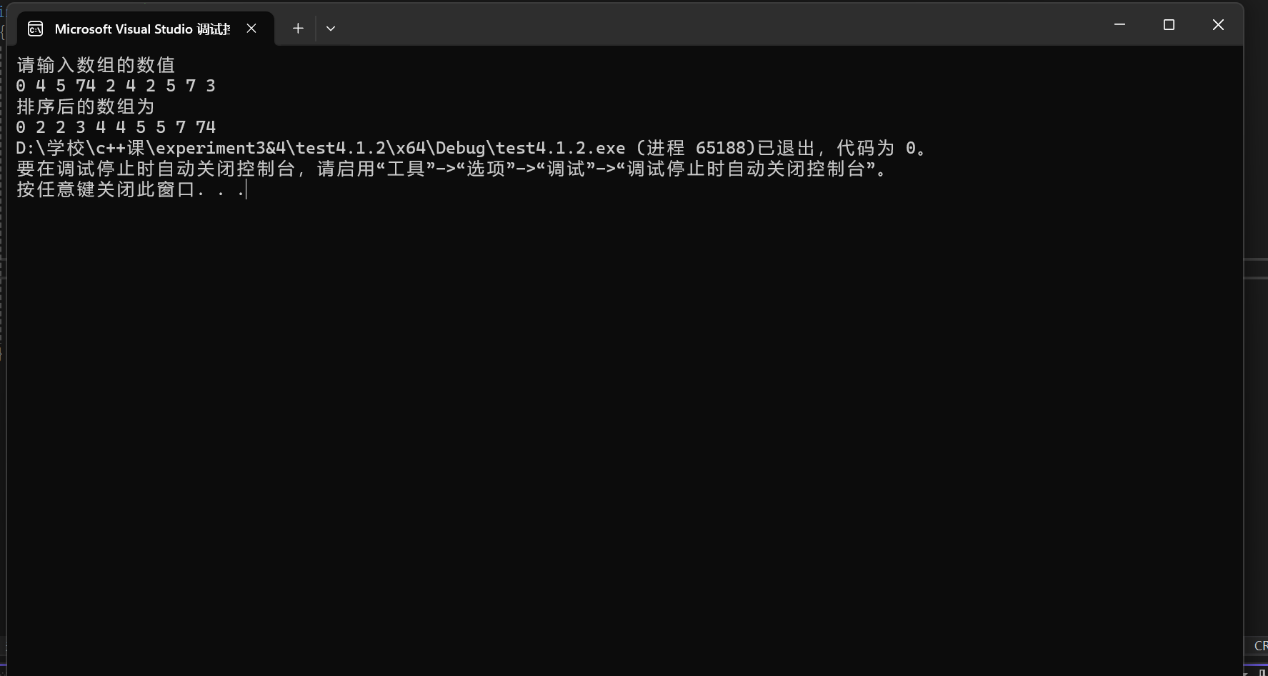
for (int i = 0; i < num; i++)

{

cout << arr1[i] << ‘ ’;//输出数组

}

return 0;

}

3.#include<iostream>

using namespace std;

void change\_state(bool\* arr, int num1)//定义改变状态函数

{

for (int i = num1; i < 100; )//从第i项开始

{

arr[i] = (!arr[i]);//第i项取反

int j = i + 1;//下标+1

j += j + 1;//隔j+1项后继续取反(因为是相隔 所以是(i+1)+1)

i = j - 1;

}

}

int main()

{

bool arr[100];//定义一个布尔类型的数组

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

{

arr[i] = true;//将数组的所有值赋为true(开)

}

for (int i = 1; i < 100; i++)//从学生2到学生100改变状态

{

change\_state(arr, i);//调用改变状态函数

}

cout << "开着的门有:" << endl;//打印开着的门

for (int i = 0; i < 100; i++)//创建循环

{

if (arr[i] == true)//如果开着

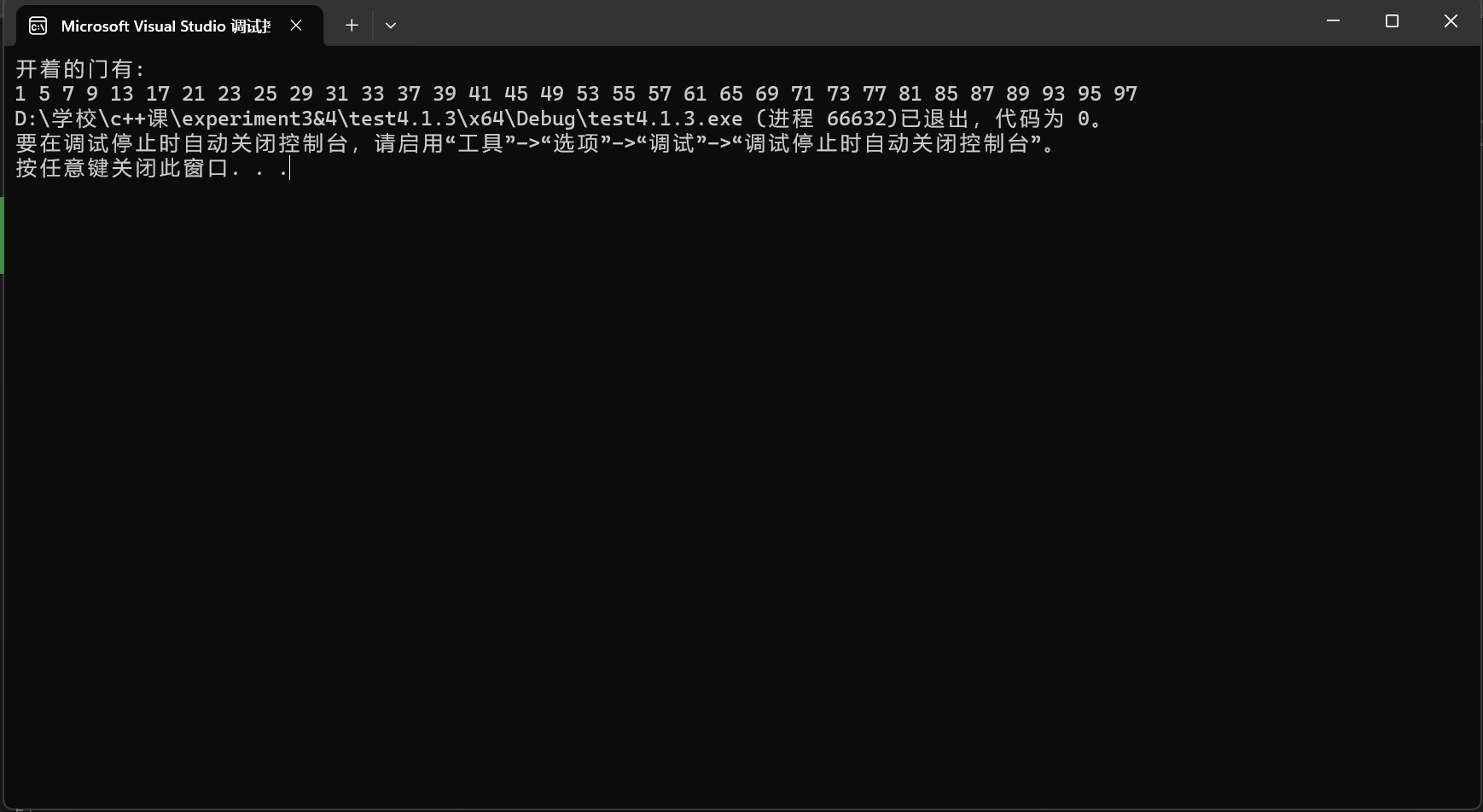
{

cout << (i+1) << ' ';//打印编号

}

}

return 0;

}

4. #include<iostream>

using namespace std;

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

{

int size3 = size1 + size2;//list3的长度

for (int i = 0; i < size3; i++)//建立循环 对list3赋值

{

if (i < size1)//如果i<list1的长度 将list1中的数值依次赋给list3

list3[i] = list1[i];

else//i超出了list1的长度 将list2中的数值依次赋给list3

list3[i] = list2[i - size1];//从list2[0]开始赋值

}

for (int i = 0; i < size3 - 1; i++)//冒泡排序

{

for (int j = 0; j < size3 - i - 1; j++)

{

if (list3[j] > list3[j + 1])

{

int temp;

temp = list3[j];

list3[j] = list3[j + 1];

list3[j + 1] = temp;

}

}

}

cout << "合并后排序的数组是:";//提示合并后的结果

for (int i = 0; i < size3; i++)

{

cout << list3[i] << ' ';//输出合并后的结果

}

}

int main()

{

int size1, size2;//定义数组长度

cout << "请输入list1的长度:";//提示输入size1

cin >> size1;//输入size1

int\* list1 = new int[size1];//使用new 用常量赋值数组大小

cout << "请输入list1的数值:";//提示输入list1数值

for (int i = 0; i < size1; i++)//建立循环

{

cin >> list1[i];//输入数值

}

cout << "请输入list2的长度:";//提示输入size2

cin >> size2;//输入size2

int\* list2 = new int[size2];//使用new 用常量赋值数组大小

cout << "请输入list2的数值:";//提示输入list2数值

for (int i = 0; i < size2; i++)//建立循环

{

cin >> list2[i];//输入数值

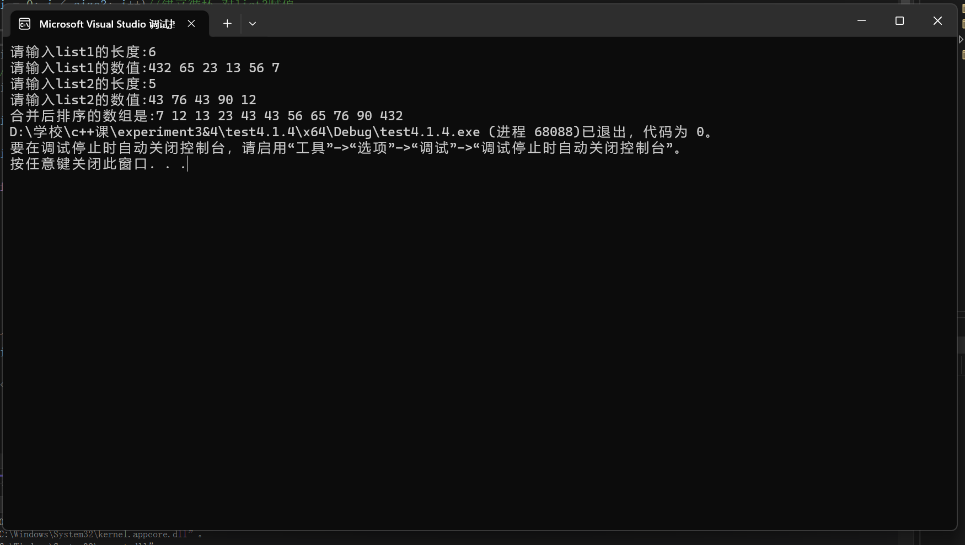
}

int\* list3 = new int[size1 + size2];//使用new 定义数组list3

merge(list1, size1, list2, size2, list3);//调用函数

return 0;

}



5. #include<iostream>

using namespace std;

int indexOf(const char s1[], const char s2[])//定义寻找子串

{

int len\_s1 = strlen(s1);//求s1长度

int len\_s2 = strlen(s2);//求s2长度

int equal = 0;//计数=0

for (int i = 0; i < len\_s2; i++)

{

equal = 0;//计数=0

if (s1[0] == s2[i] && len\_s1 <= (len\_s2 - i + 1))//如果子串的第一项与母串的某一项相等 且子串长度小于母串剩余长度

{

equal++;//计数+1

for (int j = 1; j < len\_s1; j++)//建立循环

{

i++;//i+1

if (s1[j] == s2[i])//如果子串下一项和母串下一项相等

{

equal++;//计数+1

}

else//有不相等

{

break;//跳出循环

}

}

if (equal == len\_s1)//如果相等数和s1长度相等

{

return (i - len\_s1 + 1);//返回子串下标

}

else

{

continue;//否则进行下次循环

}

}

}

return -1;//不是子串关系 返回-1

}

int main()

{

char s1[100], s2[100];//定义两个字符串

cout << "请输入string1(子串):";//输入字符串s1

cin.getline(s1, 100);

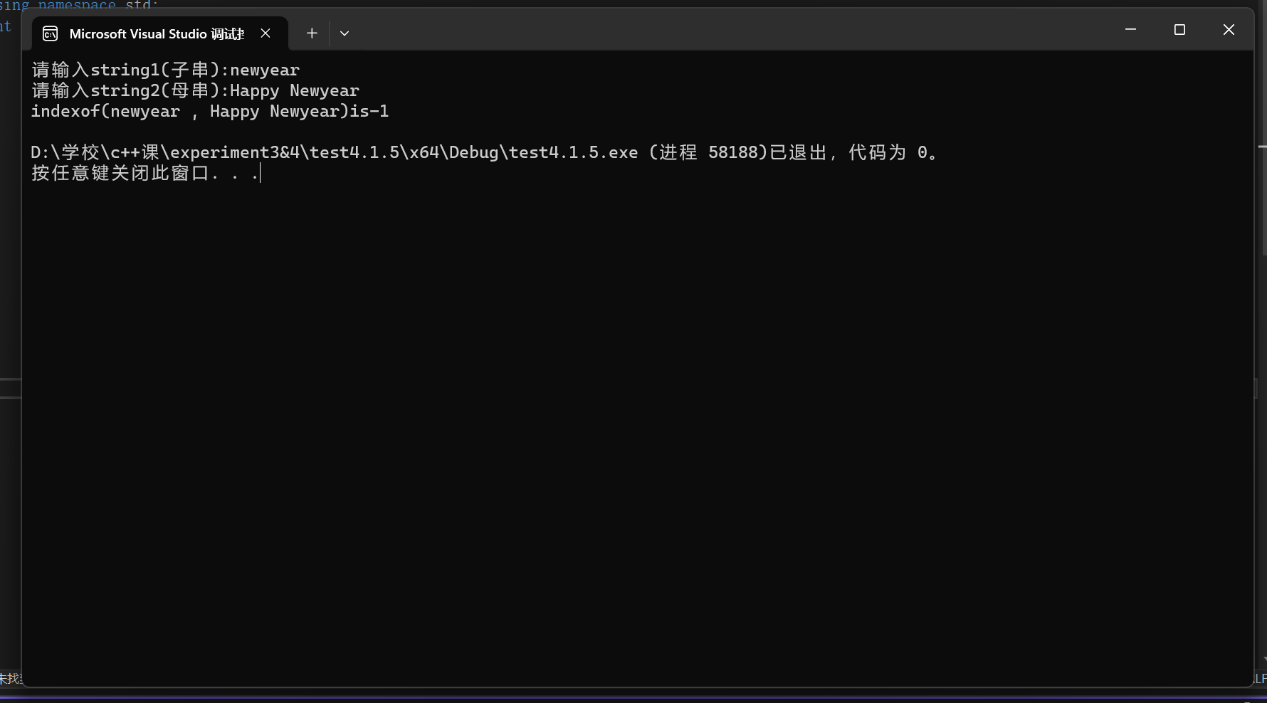
cout << "请输入string2(母串):";//输入字符串s2

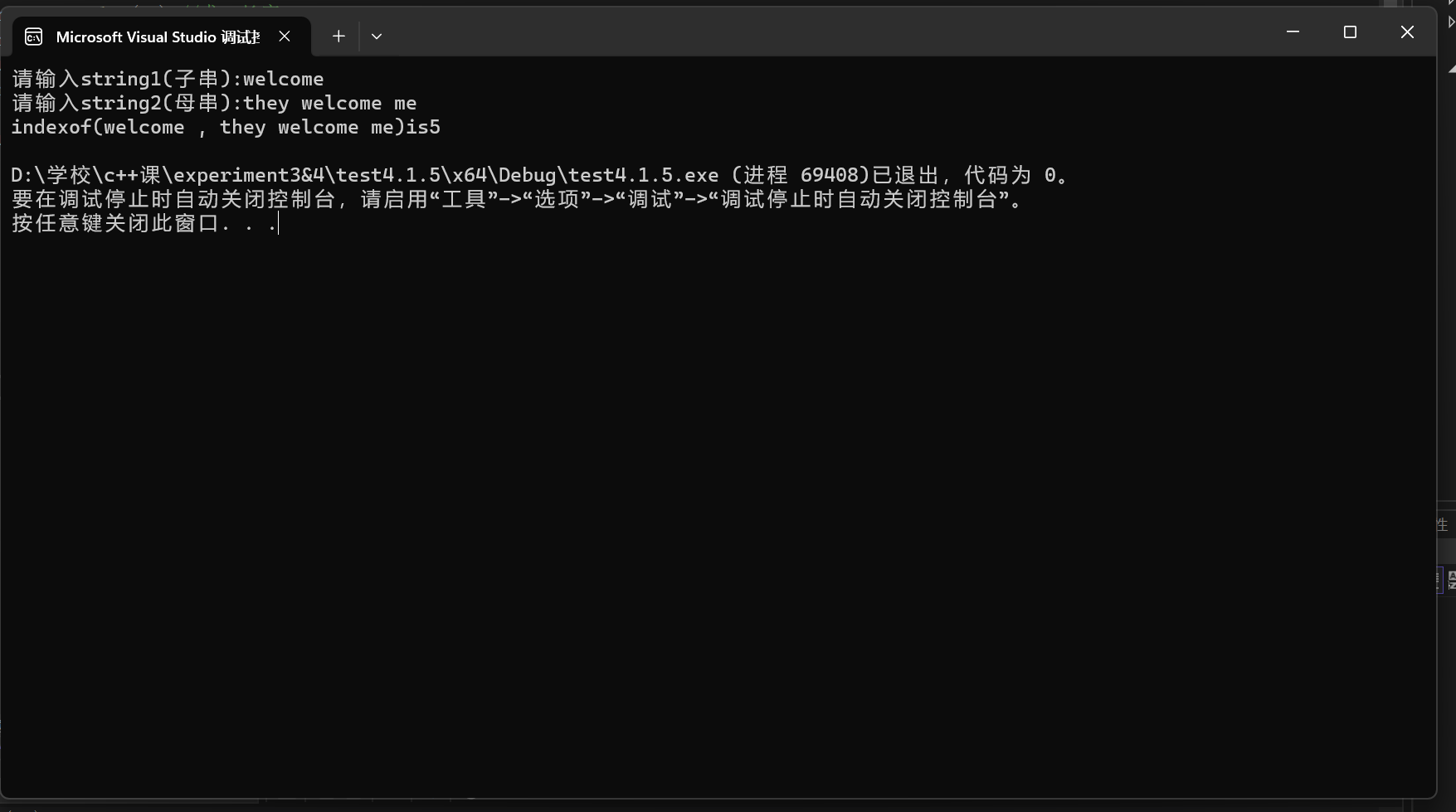
cin.getline(s2, 100);

int indexof=indexOf(s1, s2);//调用函数

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

return 0;

}



6. #include<iostream>

using namespace std;

void count(const char s[], int counts[])//定义计数函数

{

int len\_s = strlen(s);//定义字符串长度

for (int i = 0; i < len\_s; i++)//创立循环

{

int m = int(s[i]) - 65;//在m与ASCII码之间建立关系

if (m >= 0 && m < 26)//m为大写字母

{

counts[m]++;//相应字母计数+1

}

if (m >= 32 && m < 58)//m为小写字母

{

counts[m - 32]++;//相应字母计数+1

}

else//不是字母

{

continue;//进入下次循环

}

}

}

int main()

{

char s[100];//定义字符数组

cout << "请输入字符串:";//输入字符串

cin.getline(s, 100);

int counts[26];//定义整数数组

for (int i = 0; i < 26; i++)//建立循环

{

counts[i] = 0;//给每个值都赋为0

}

count(s, counts);//调用函数

for (int i = 0; i < 26; i++)//建立循环

{

if (counts[i] != 0)//打印有计数的字母

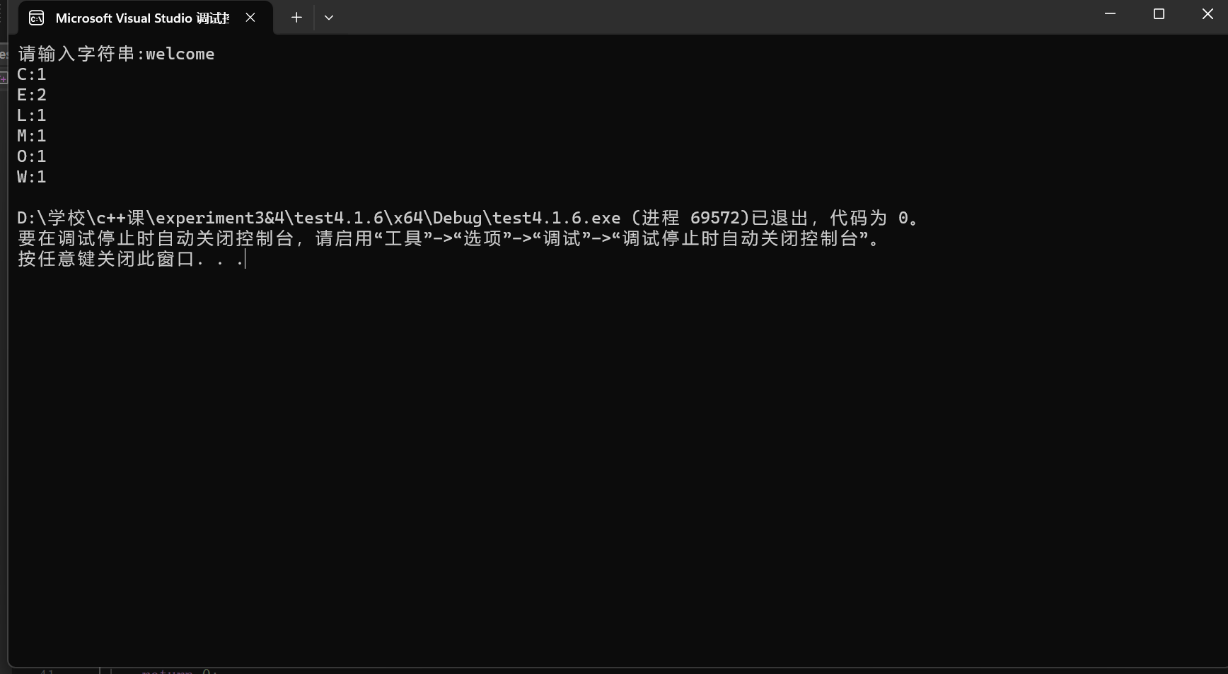
{

cout << char(i + 65) << ":" << counts[i] << endl;//打印字母及其计数

}

}

return 0;

}

**（二）指针**

**1、上机验证下列程序的运行结果**

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;

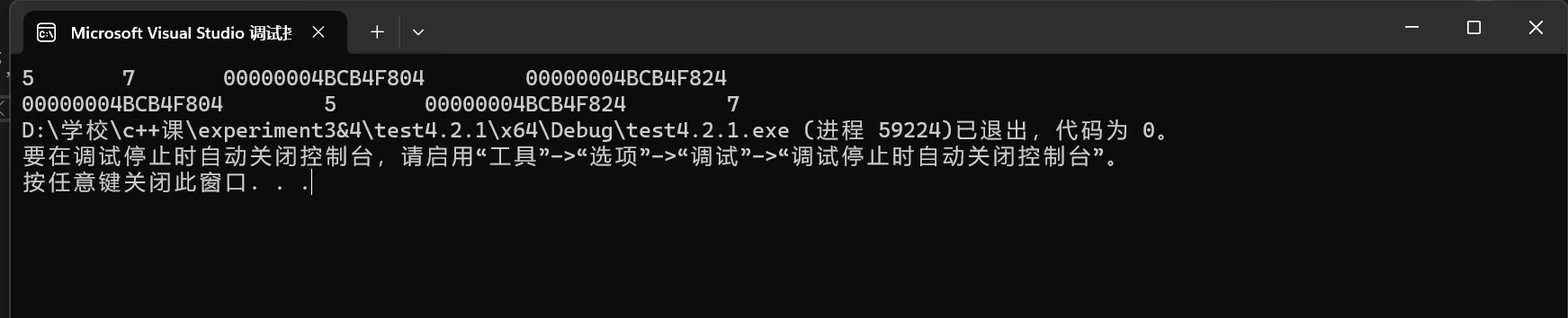
pj = &j;

i = 5; j = 7;

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

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

}



2. #include<iostream>

using namespace std;

int main() //C语言程序，要了解

{

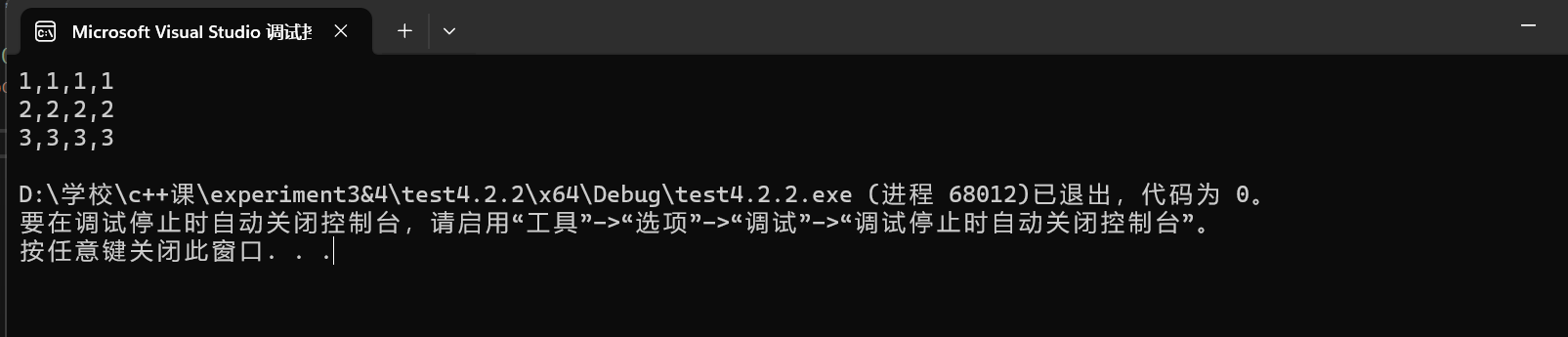
int a[] = { 1,2,3 };

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

}

3. #include<iostream>

using namespace std;

//#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);

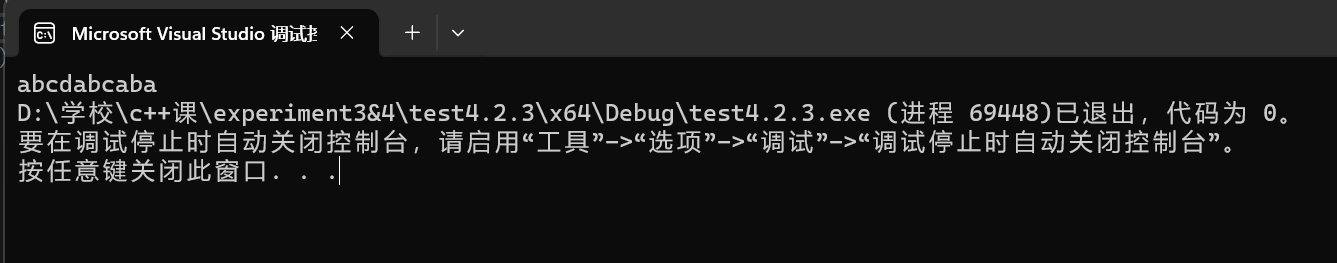
}

void main()

{

char st[] = "abcd";

f(st, 4);

}

4. #include<iostream>

using namespace std;

int\* f()

{

int\* list = new int[4];

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

list[i] = i + 1;

return list;

}

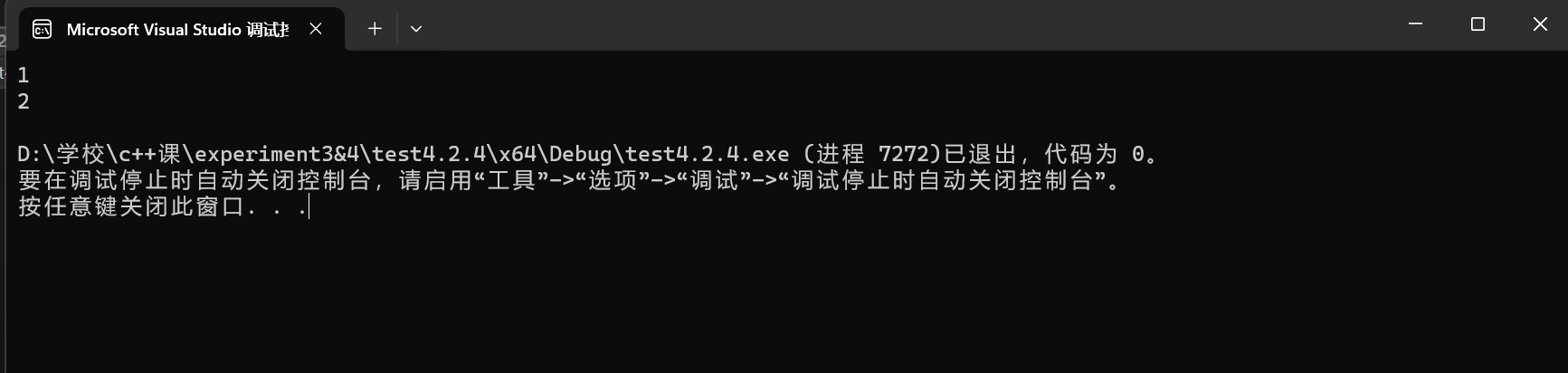
void main()

{

int\* p = f();

cout << p[0] << endl;

cout << p[1] << endl;

}

**2、程序设计**

1. #include<iostream>

using namespace std;

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

{

int len\_s1 = strlen(s1);//求s1长度

int len\_s2 = strlen(s2);//求s2长度

int equal = 0;//计数=0

for (int i = 0; i < len\_s2; i++)

{

equal = 0;//计数=0

if (s1[0] == s2[i] && len\_s1 <= (len\_s2 - i + 1))//如果子串的第一项与母串的某一项相等 且子串长度小于母串剩余长度

{

equal++;//计数+1

for (int j = 1; j < len\_s1; j++)//建立循环

{

i++;//i+1

if (s1[j] == s2[i])//如果子串下一项和母串下一项相等

{

equal++;//计数+1

}

else//有不相等

{

break;//跳出循环

}

}

if (equal == len\_s1)//如果相等数和s1长度相等

{

return (i - len\_s1 + 1);//返回子串下标

}

else

{

continue;//否则进行下次循环

}

}

}

return -1;//不是子串关系 返回-1

}

int main()

{

char s1[100], s2[100];//定义两个字符串

cout << "请输入string1(子串):";//输入字符串s1

cin.getline(s1, 100);

cout << "请输入string2(母串):";//输入字符串s2

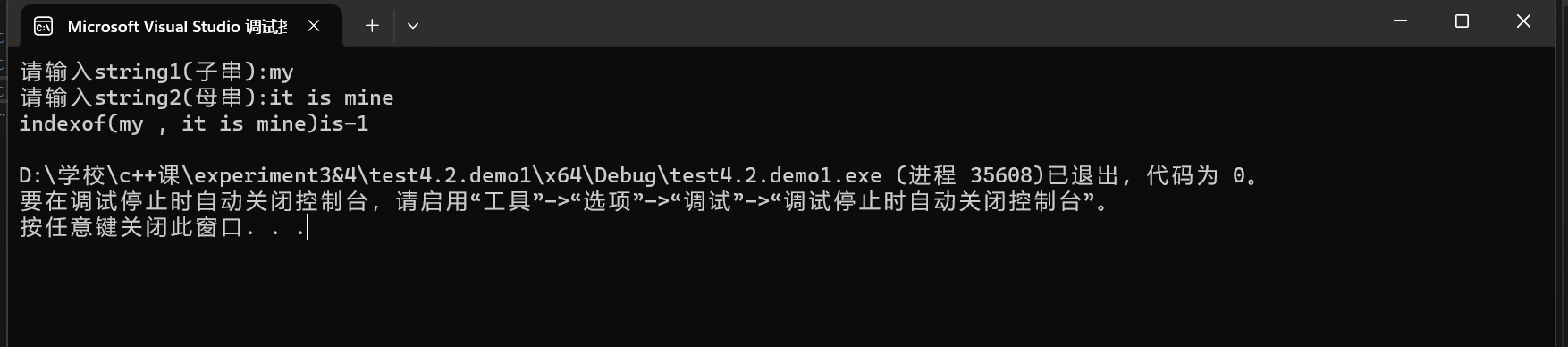
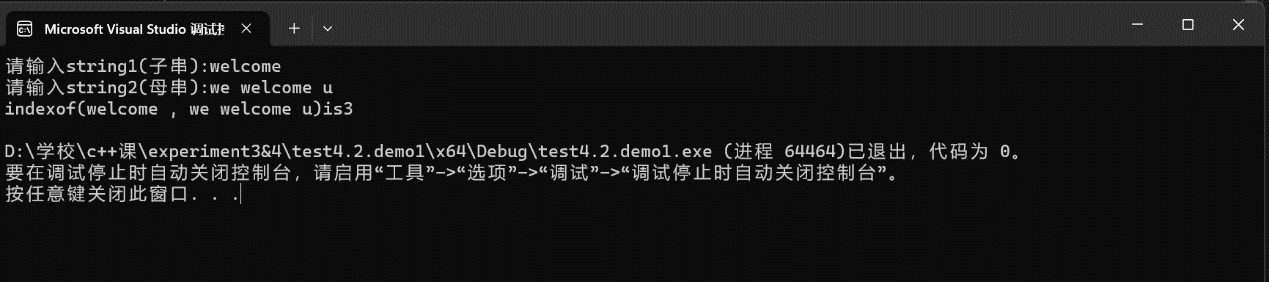
cin.getline(s2, 100);

int index = indexof(s1, s2);//调用函数

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

return 0;

}



2. #include<iostream>

#include<cmath>

using namespace std;

int parseHex(const char\* const hexString)//定义函数

{

int s\_len = strlen(hexString);//定义数组长度

int num = 0;//定义计数=0

for (int i = 0; i < s\_len; i++)//建立循环

{

if (hexString[i] >= '0' && hexString[i] <= '9')//对于0-9的数值

{

num = num + (((int(hexString[i]))-48) \* pow(16, s\_len - i - 1));//转换成int类型的0-9 根据相应位数乘上16的x-1次方

}

else if (hexString[i] >= 'A' && hexString[i] <= 'F')//对于A-F的数值

{

num = num +(((int(hexString[i])) - 55) \* pow(16, s\_len - i - 1));//转换为int类型的10-15 根据相应位数乘上16的x-1次方

}

else if (hexString[i] >= 'a' && hexString[i] <= 'f')//对于a-f的小写数值

{

num = num +(((int(hexString[i])) - 87) \* pow(16, s\_len - i - 1));//转换为int类型的10-15 根据相应位数乘上16的x-1次方

}

else {//对于其他字符

cout << "数值不是16进制" << endl;//提示字符串不合法

return 0;//返回错误值

}

}

return num;

}

int main()

{

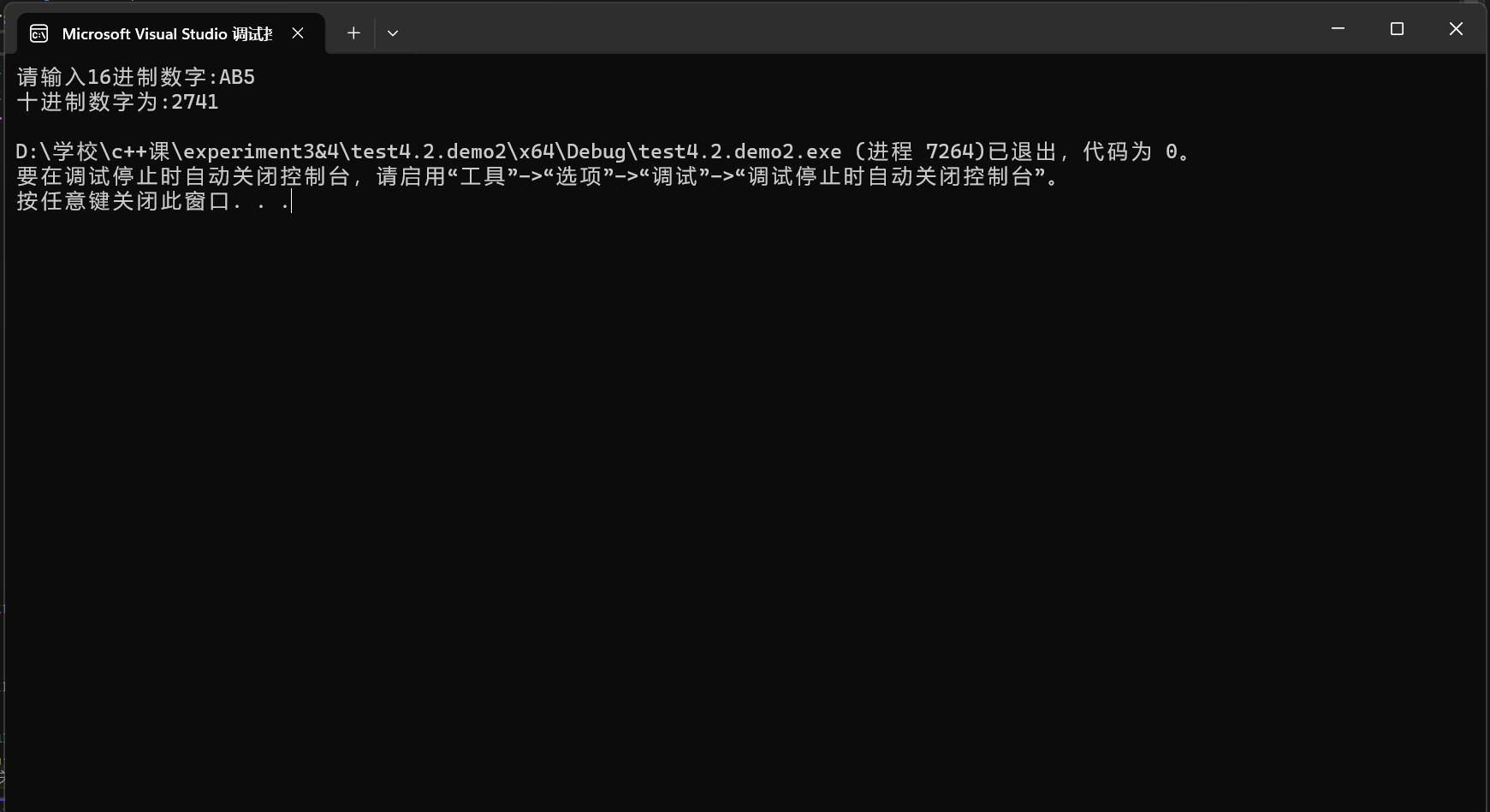
char s[50];//定义数组

cout << "请输入16进制数字:";//输入数组

cin.getline(s, 50);

int dec=parseHex(s);//调用函数

cout << "十进制数字为:" << dec << endl;//输出十进制数字

}

3. #include<iostream>

using namespace std;

void bubble\_sort(int\* arr, const int\* size)//定义排序函数

{

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 size;//定义数组大小

cout << "请输入数组大小:";//输入数组大小

cin >> size;

int\* arr = new int[size];//用new定义动态数组

cout << "请输入数组数值:";//输入数组数值

for (int i = 0; i < size; i++)//建立循环输入数值

{

cin >> arr[i];

}

bubble\_sort(arr, &size);//调用排序函数

cout << "排序后的数组为:";//输出排序后的函数

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

{

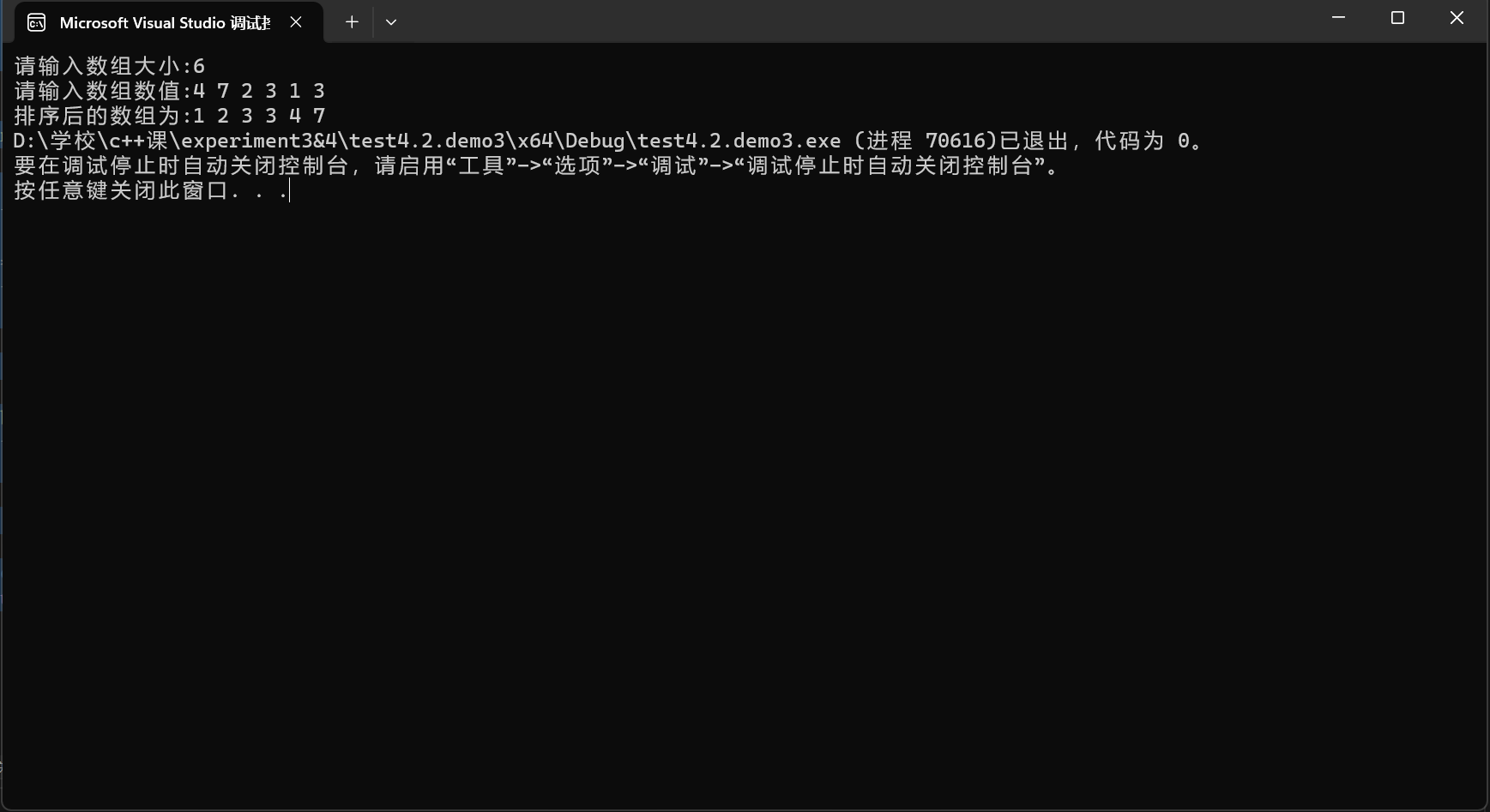
cout << \*&arr[i] << ' ';//使用指针输出结果

}

delete[]arr;//删除创建的动态数组

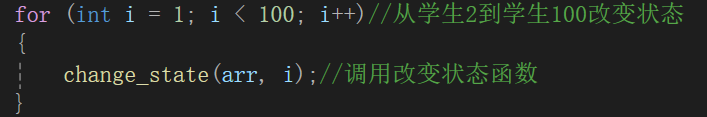
return 0;

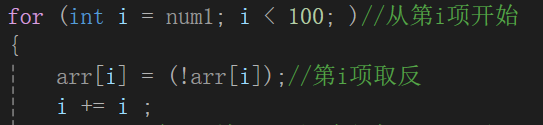
}

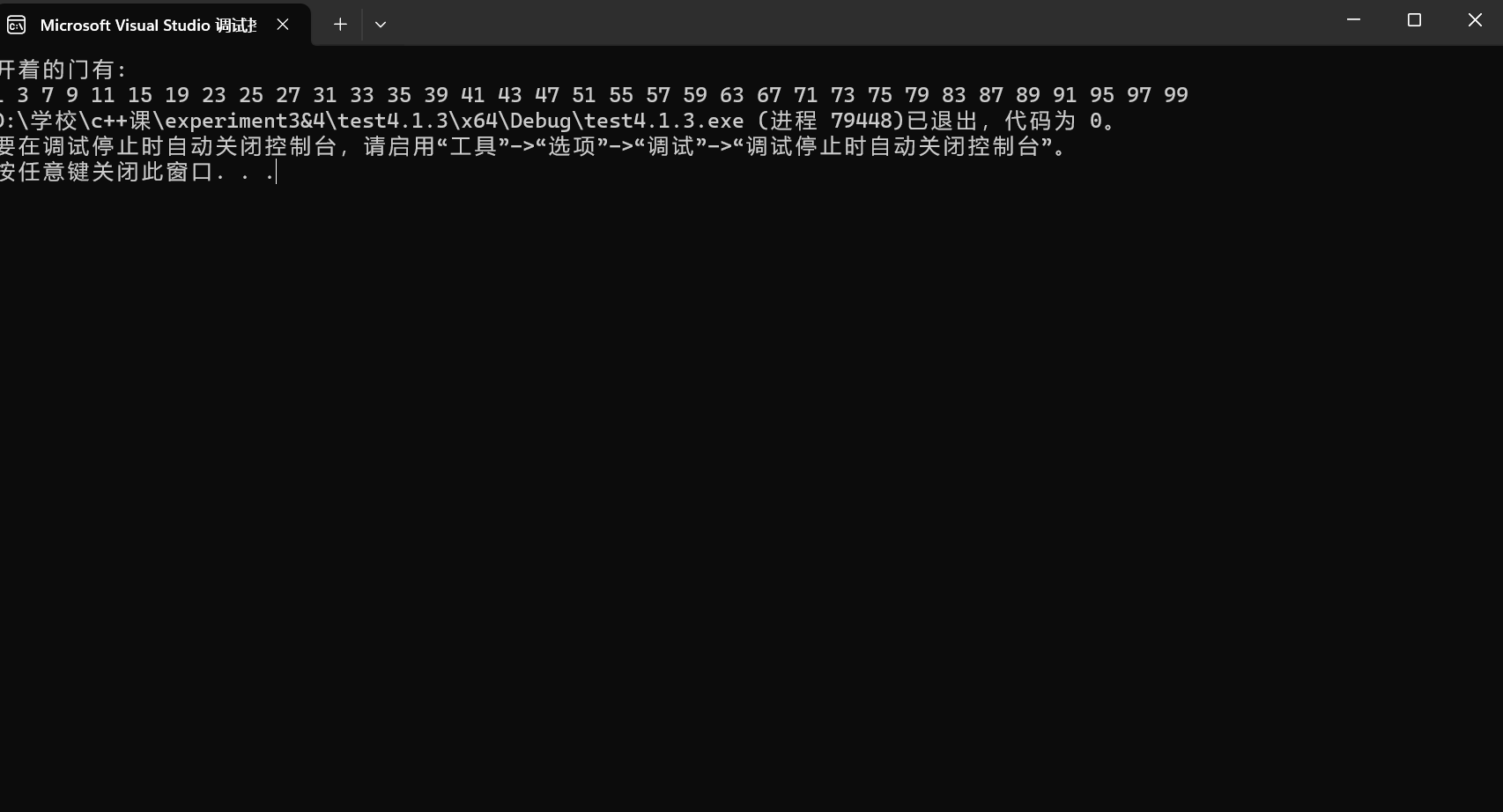


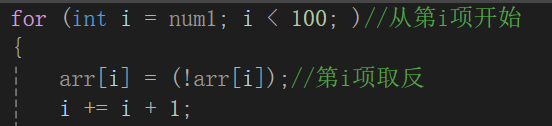
**四、遇到的问题与解决方法**

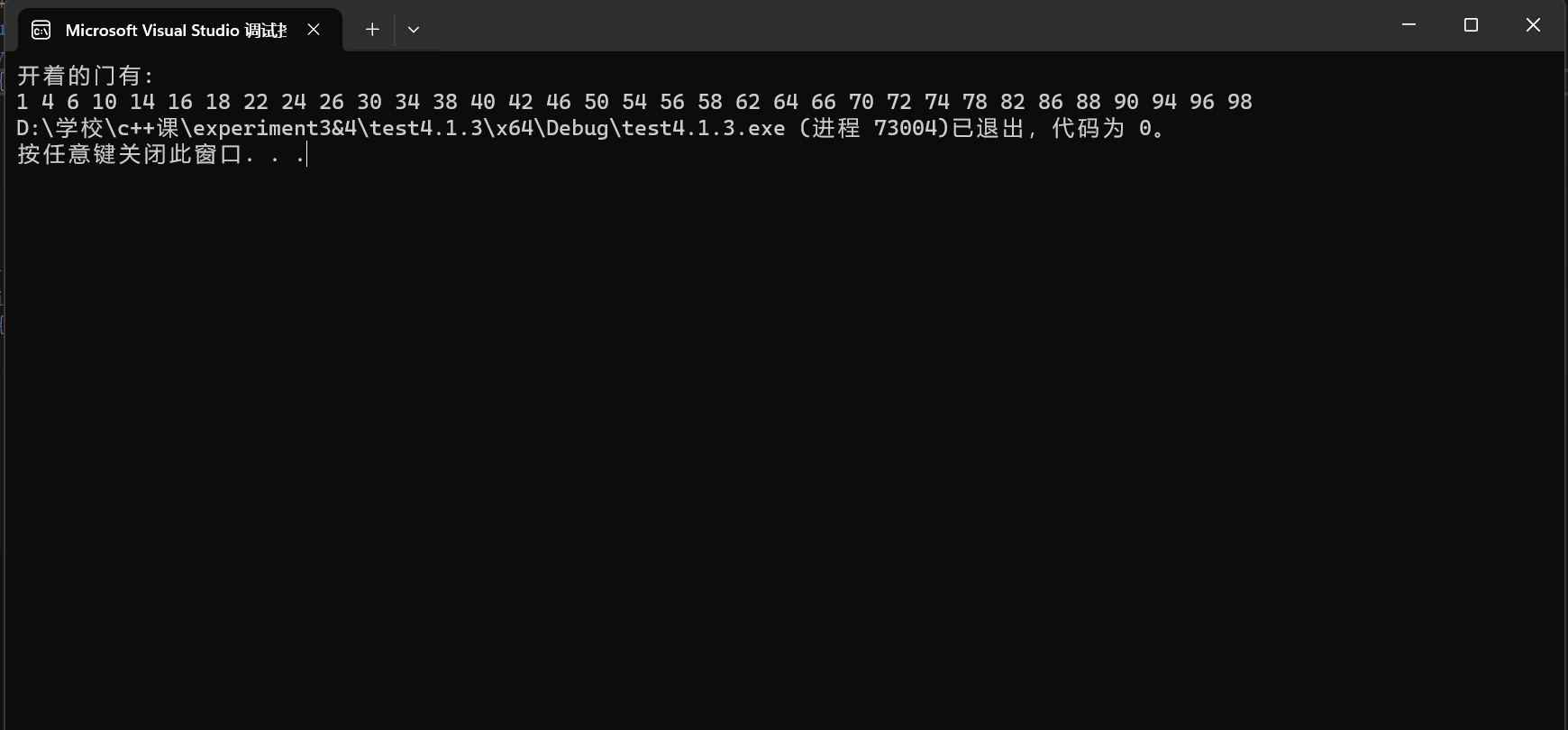
**1.在解决翻柜子的问题时，难以确定计算表达式**

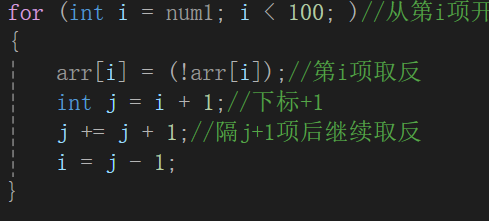
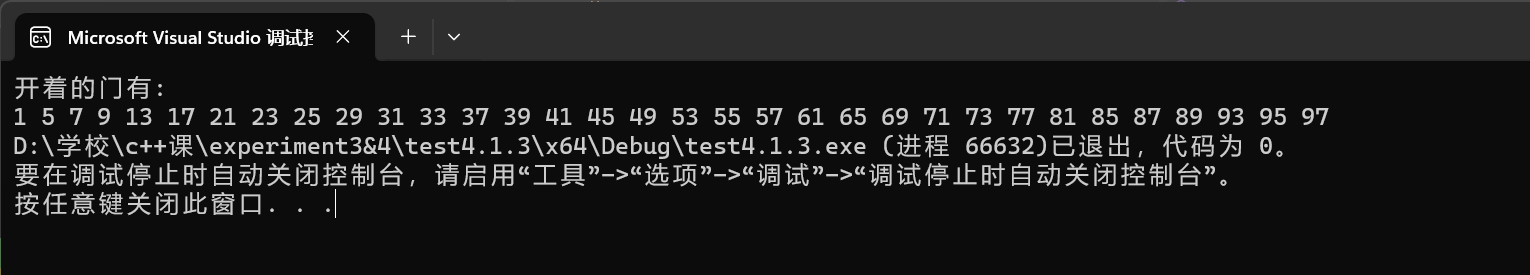
 在最开始调用函数时，为了后续复制方便，直接传递了下标的值（这也就导致了后续计算困难）

 对于题目的第一理解是，每隔i个柜子就改变状态，所以在循环中列出i+=i即可。

 但是根据此代码运行出来的结果与现实中粗略计算得到的结果完全不同。

 随后便开始重新审视表达式，注意到，题中所给的是“到第x个同学，从第x个开始，每隔x个改变状态”，也就是说，中间隔得是x项。根据简单计算递推，2和5之间隔2个数，5=2+2+1，便在表达式后+1。

 但是修改后的代码与现实中的估算仍有不符。

 再次重新审视代码中的错误，发现最开始传递的i值是数组中的下标，这会导致每项会小一个，修改该问题。

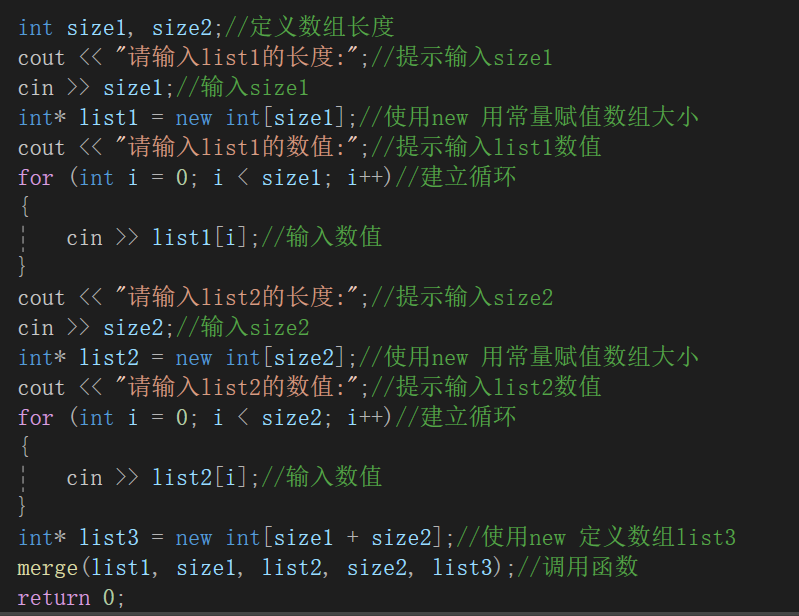
程序运行结果正确。

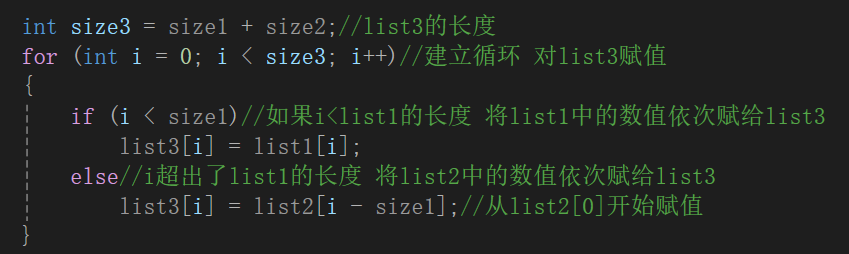
**2.在合并数组时无法合并**

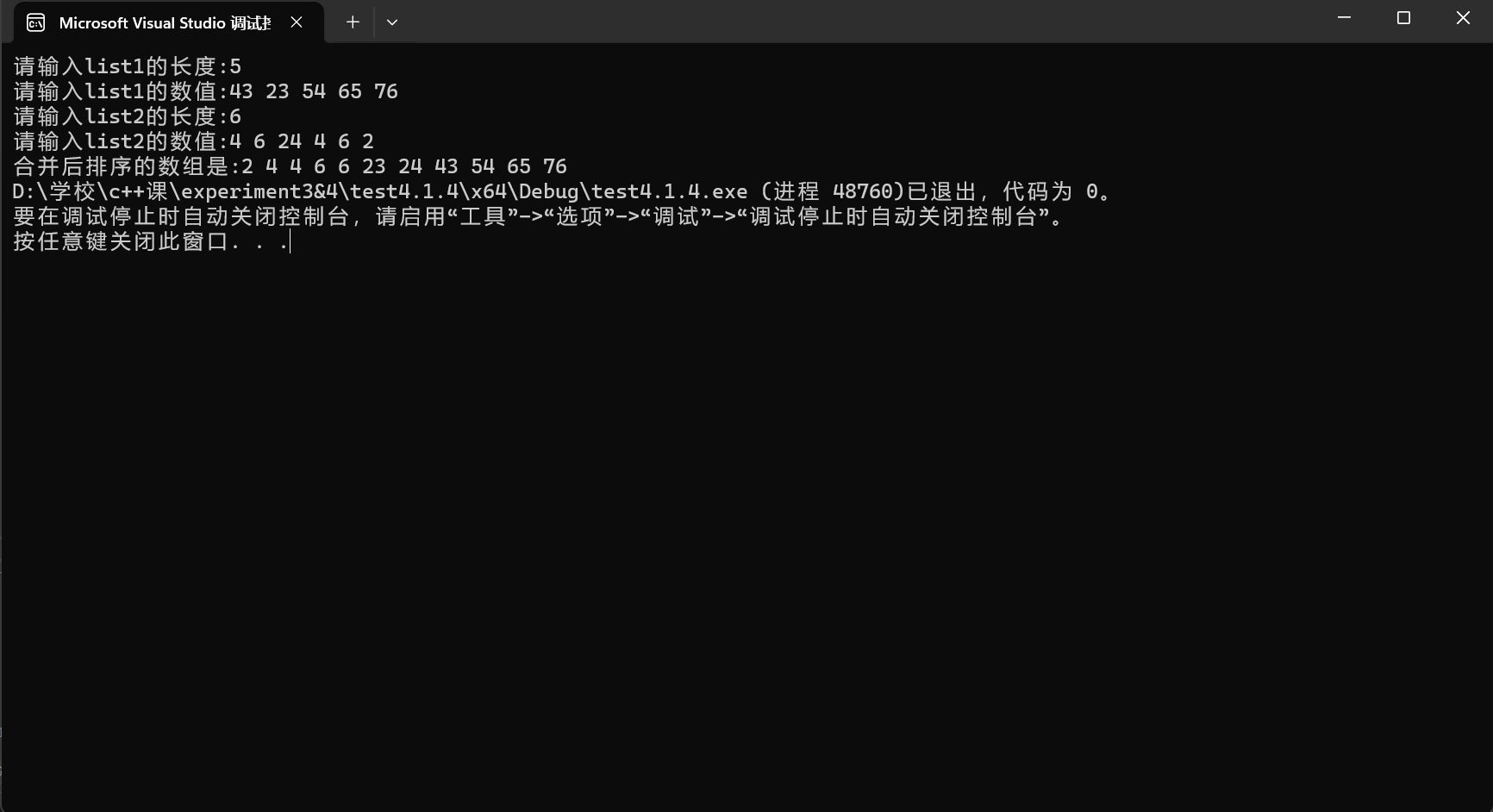
在最开始合并数组时，想通过基本数组定义size的方式实现，但是在定义过程中发现变量不能定义数组的大小，程序会报错，便考虑使用其他方法。

接下来考虑将数组大小定义为一个常数，使用计算元素的方法，每有一个合法元素便计数+1，最后根据计数来操作数组内的合法元素。

但是在实际运行的过程中，这种代码比较难以实现，且我编写的程序在合并数组后还是会出现赋值错误。

 因此便考虑引入动态数组。动态数组大小固定，比较利于计算赋值。

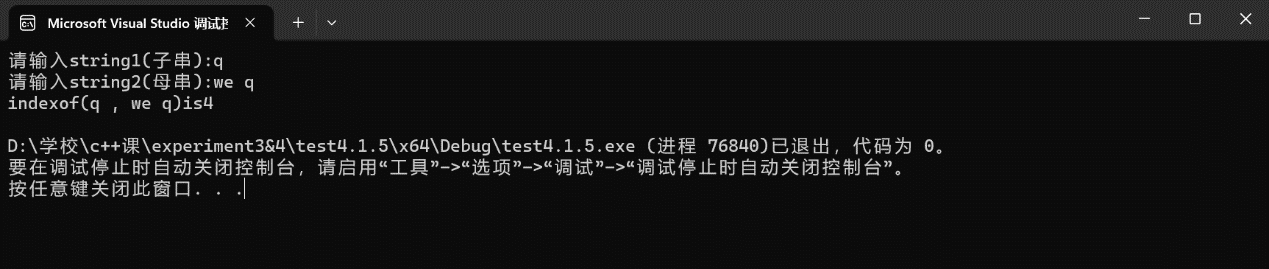
 因为动态数组合法数值的数量和数组大小相同，便利于表达式的计算，实现起来比较方便。

 最后数组合并结果正常，程序编写成功。

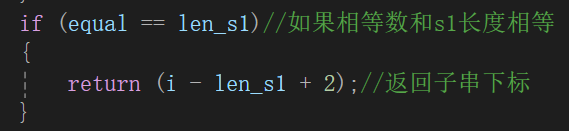
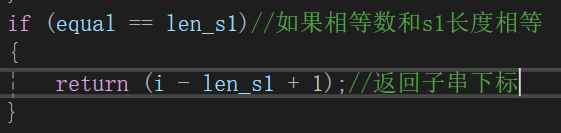
**3.检测子串功能不完全**

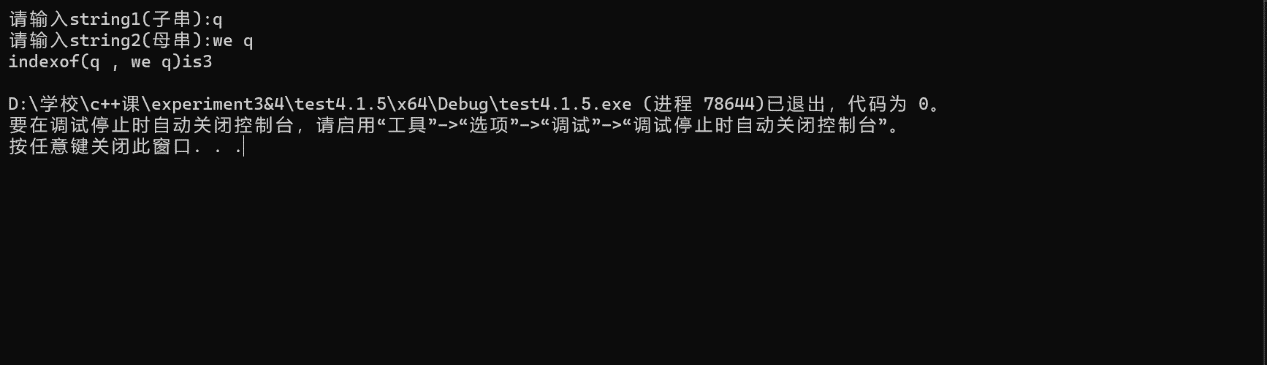
在最开始编写的一版程序中，我可以完成子串的检查，但是只能检测连续的字符串。

考虑到是因为我使用cin输入字符数组的值，中间不能带有空格，便将cin替换为了cin.getline（），在替换后，程序就可以正常带有空格了。

 但是在运行程序时，发现我的运行结果比题中所给的运行结果多了一位。

在询问他人后发现，我误把下标理解成了位次，所以会比题中多一个。

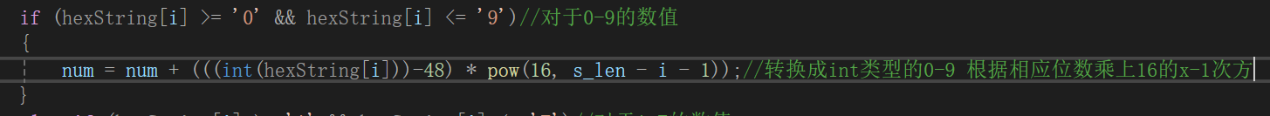
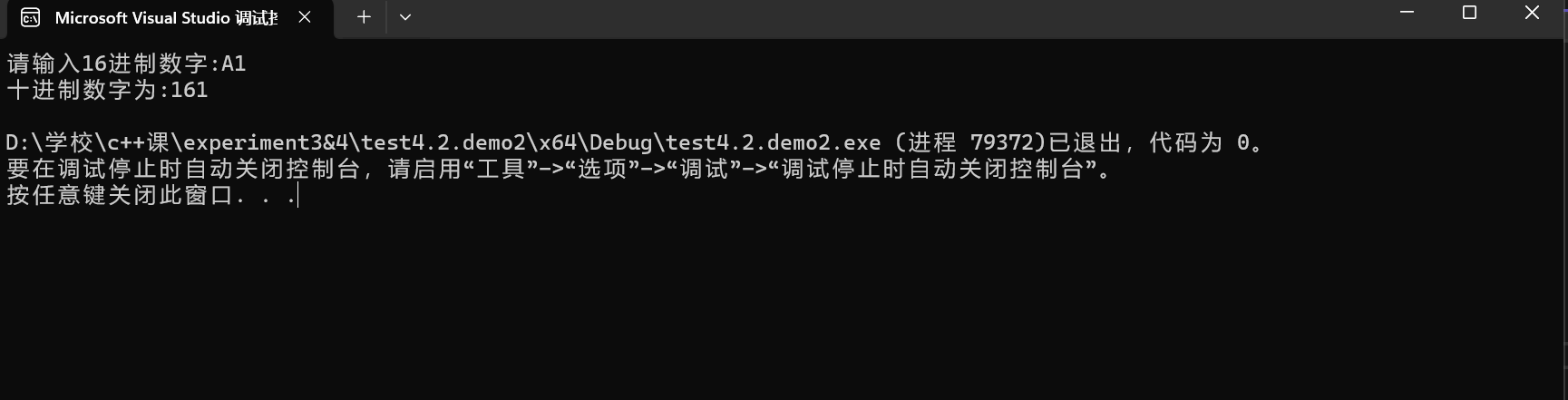
修改程序。

重新运行程序，程序优化成果。

**4.16进制转为10进制时计算出现错误**

我在编写进制转换代码时，底层原理是依次根据数组中数值所在位次及数据大小，使用对应的进制转换计算方法。

因此程序中有一个步骤是对char类型数据进行转化。但是在转化过程中，忽略的数字1-9的ASCII码转化，直接把相应的字符转化成了int，导致最后计算出的结果偏大。

 将1-9的ASCII码进行转化，程序得到优化，计算结果准确。

**五、体会**

**1.数组较以前学习的内容更加复杂，要更加切实的掌握数组有关知识**

数组可以和函数、循环相结合，其自身的底层逻辑也相对比较复杂，其可以实现的功能也比较丰富。但是如果想要实现相对比较复杂、更多的功能，就要掌握并且理解更多的有关于数组的知识。

例如，在进行本次相关实验后，我发现已会的使用常量定义数组大小的方式在一些情况下比较难以使用，需要引入动态数组。我在此前已经掌握了new的具体用法，在了解了动态数组的原理以及定义方式后，可以很快的掌握动态数组的定义以及使用，在使用了新的动态数组的知识后，运算过程、代码编写的方式都变得更加方便。但是，在使用动态数组时，也要记得在数组使用完毕后通过delete将数组数据删除。

数组的知识体系是比较复杂的，且数组在进行计算的时候，需要更加细心准确地书写表达式；要想更加自如地运用数组的功能，就要巩固数组相关知识，通过实践、练习来切实掌握数组知识。

**2.指针在灵活使用的情况下，可以实现更多功能**

与普通定义变量不同的是，指针指向的是储存变量的内存地址，这一特点决定了指针的运用情况可以是多样的。

在调用函数时，通过指针传递函数值，进行地址传递，函数在运行时，形参修饰了实参，形参在修改后，也可以修改实参的值，这是普通值传递无法做到的。虽然引用在一定程度上可以替代指针实现功能，但指针在一些情况下依旧有不可替代的作用。

此外，通过在定义函数时，将变量设定成const的方法，也可以防止指针的值或者指向地址被修改，如果所定义的函数不需要修改指针的值或者地址，可以通过这一方法来避免编写程序时出现错误修改的问题。

在本次实验中，针对指针更多的是分辨指针相关的内容，如\*p、&p、\*&p之间的区别等，在对这些有了比较充分的认知后，在以后编写指针时便可以避免错误的发生，节省时间。

总的来说，调用指针可以为实现更多的程序功能，为函数提供了更多的可操作性，提高了代码的可读性，在接下来的一段时间中，要巩固指针相关的知识，将指针掌握，以为自己的程序提供更多的可能性。

**3.要对一些数学问题更加了解**

在进行编程的前期思路成型阶段，要想拥有一个比较完善的程序逻辑，对数学问题的了解精通是十分有必要的。如果说程序的设计需要一定数学运算或者数学方法，而不了解相应的数学方法无疑会使编程毫无思路以及进展。

如在进行翻柜子的实验以及16进制转化10进制的实验中，都是要对数学运算方法具有一定了解才能做出完善的程序。如果不了解进制之间的转化，或者不了解字符ASCII码之间的转化关系，便难以做出相应程序。

因此，能够编写出逻辑完善的代码的先要条件是要能想出相应数学问题的最佳解法，能够了解更多的编程过程中可能会用到的数学问题，便可以在编写程序时扩展更多的思路。

**4.要能够更加灵活地处理问题**

在编写程序的过程中，必然会遇到许多问题，而对于这些问题的解决方法，也必然会有很多种。

在遇到问题的时候，要善于寻找问题来源，通过f11或者分段运行程序等方式排查程序所处错误，分析栈区中的数据以更精确地找到问题所在。

在解决问题时，可以通过目前已学的知识对其进行解决，也可以额外学习一种新的更加高效的解决问题的方法，以拓宽自己的知识面，获得更多解决问题的能力，最终提高自己的综合编程能力以及学习能力。