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 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

Enter

10

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 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 the first string: welcome

Enter the second string: We welcome you!

indexOf(“welcome”, “We welcome you!”) is 3

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

Enter Enter Enter Enter

Enter

11

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)

{

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

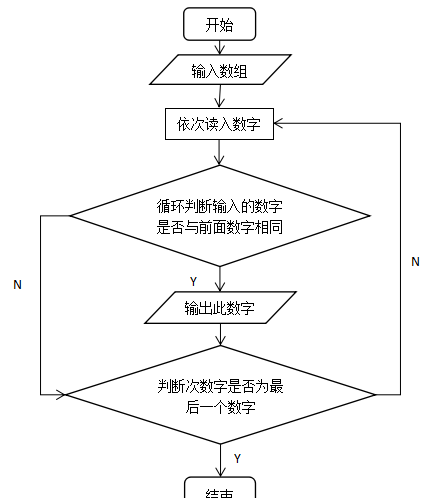
(3) 主程序中建立一动态数组（使用 new），数组元素及元素个数由键盘输入，动态调试观

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）

(一)

1.

算法如流程图所示



程序:

#include<iostream>

using namespace std;

int main()

{

const int size = 10;

int list[size];

cout << "Please enter ten numbers : " << endl;

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

cin >> list[i];

cout << list[0] << " ";

for (int m = 1; m < size; m++)

{

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

{

if (list[m] != list[j])

{

if (j == m - 1)

cout << list[m] << " ";

else

continue;

}

else

break;

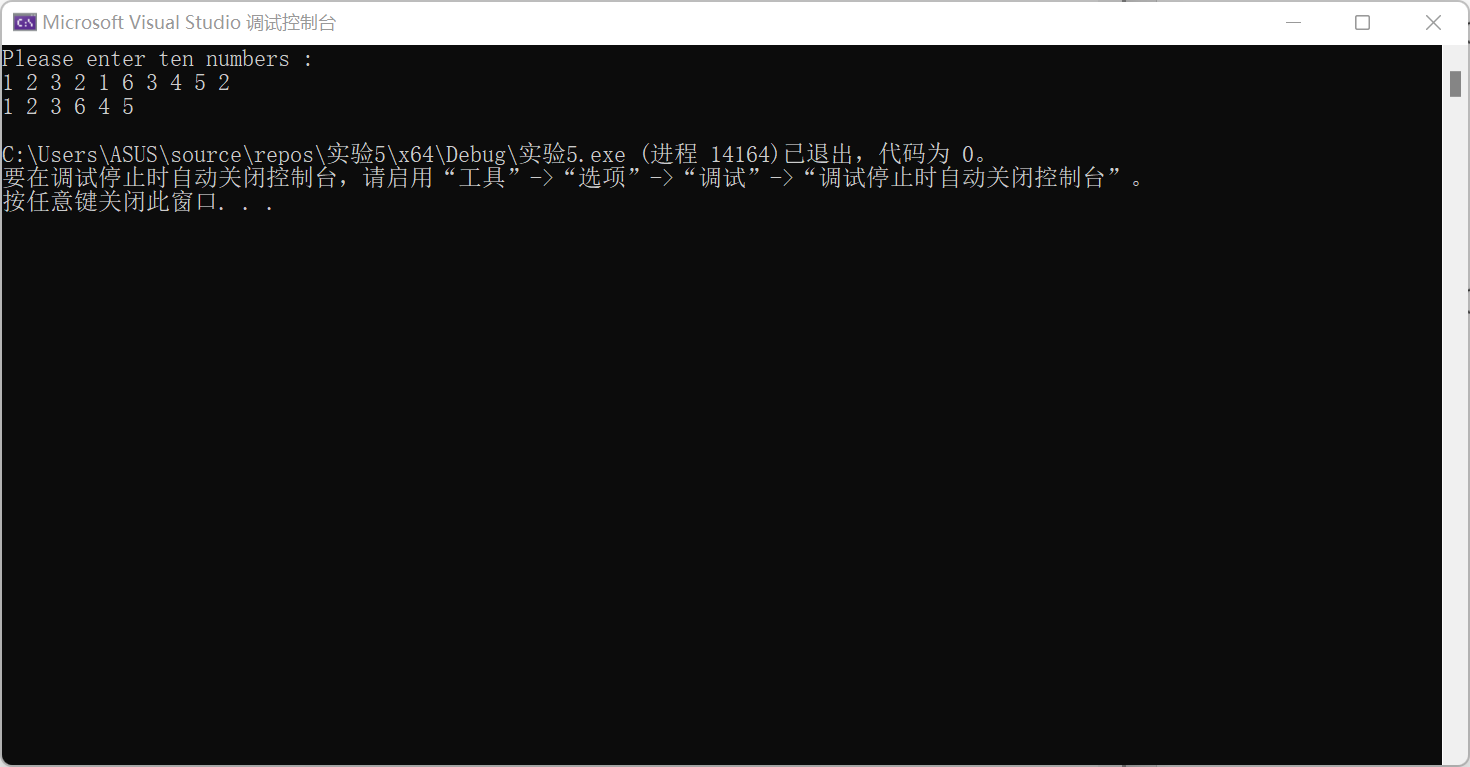
}

}

cout << endl;

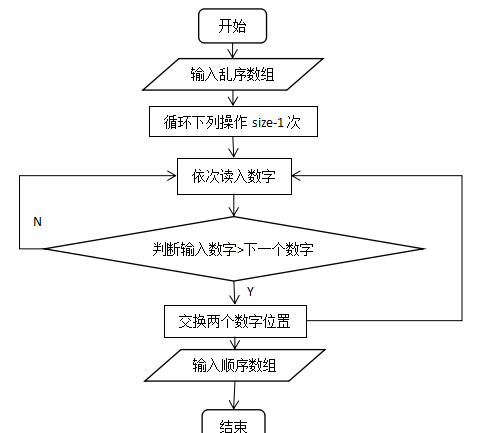
return 0;

}



2.

算法流程图：



程序：

#include<iostream>

using namespace std;

int main()

{

double arr[10];

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

{

cin >> arr[i];

}

//double arr[10] = { 1,5,6,3,2,7,8,9,5,4 };

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

{

if (arr[i] >=arr[i + 1])

{

arr[i + 1] = arr[i];

}

}

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

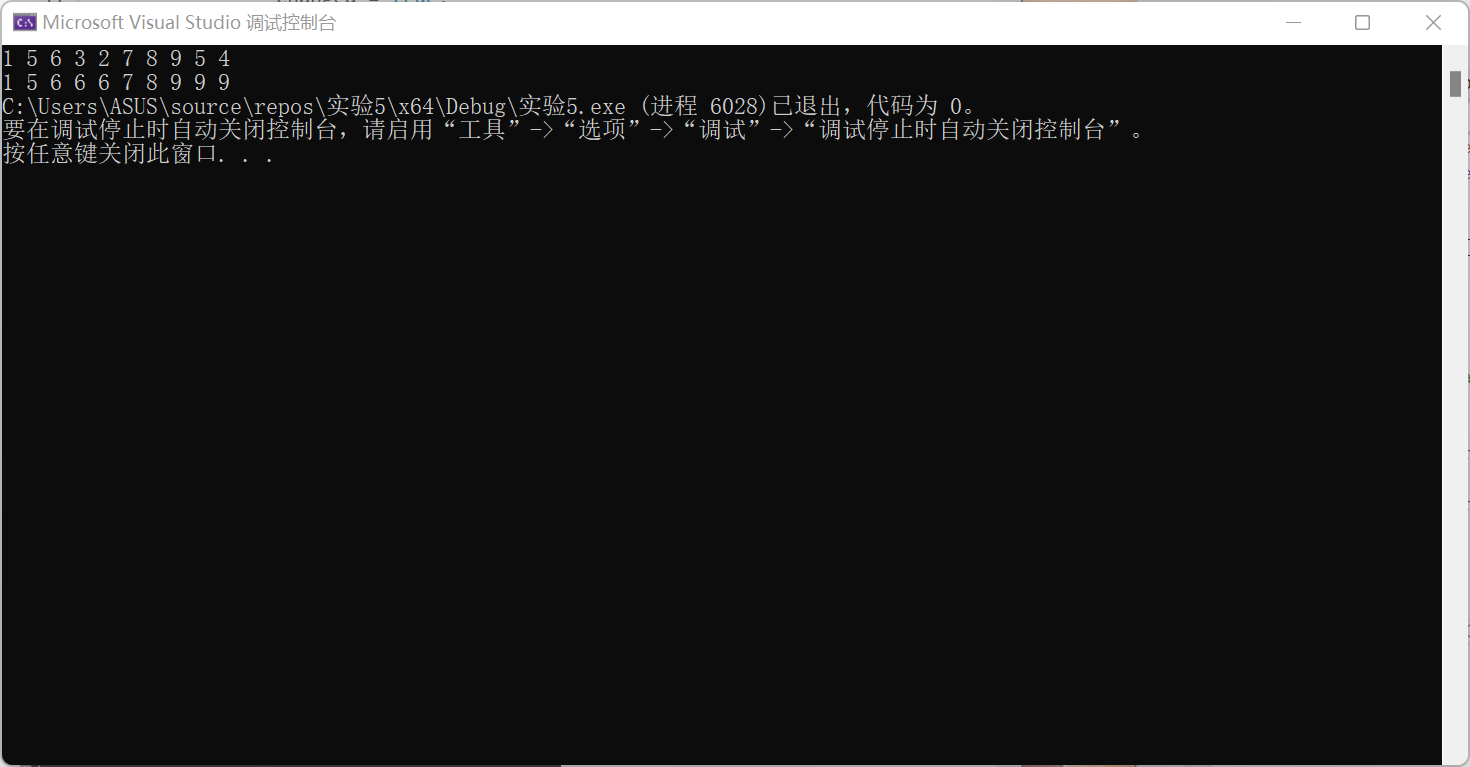
{

cout << arr[i] << “ “;

}

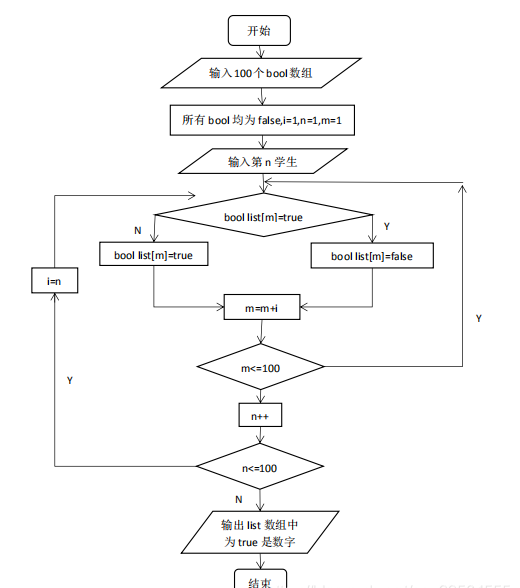
return 0;

}



3.

算法流程图：



程序：

#include<iostream>

using namespace std;

int main()

{

int arr[101];

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

{

arr[i] = 1;

}

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

{

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

{

arr[j] = 1-(arr[j]);

}

}

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

{

if (arr[i] == 1)

{

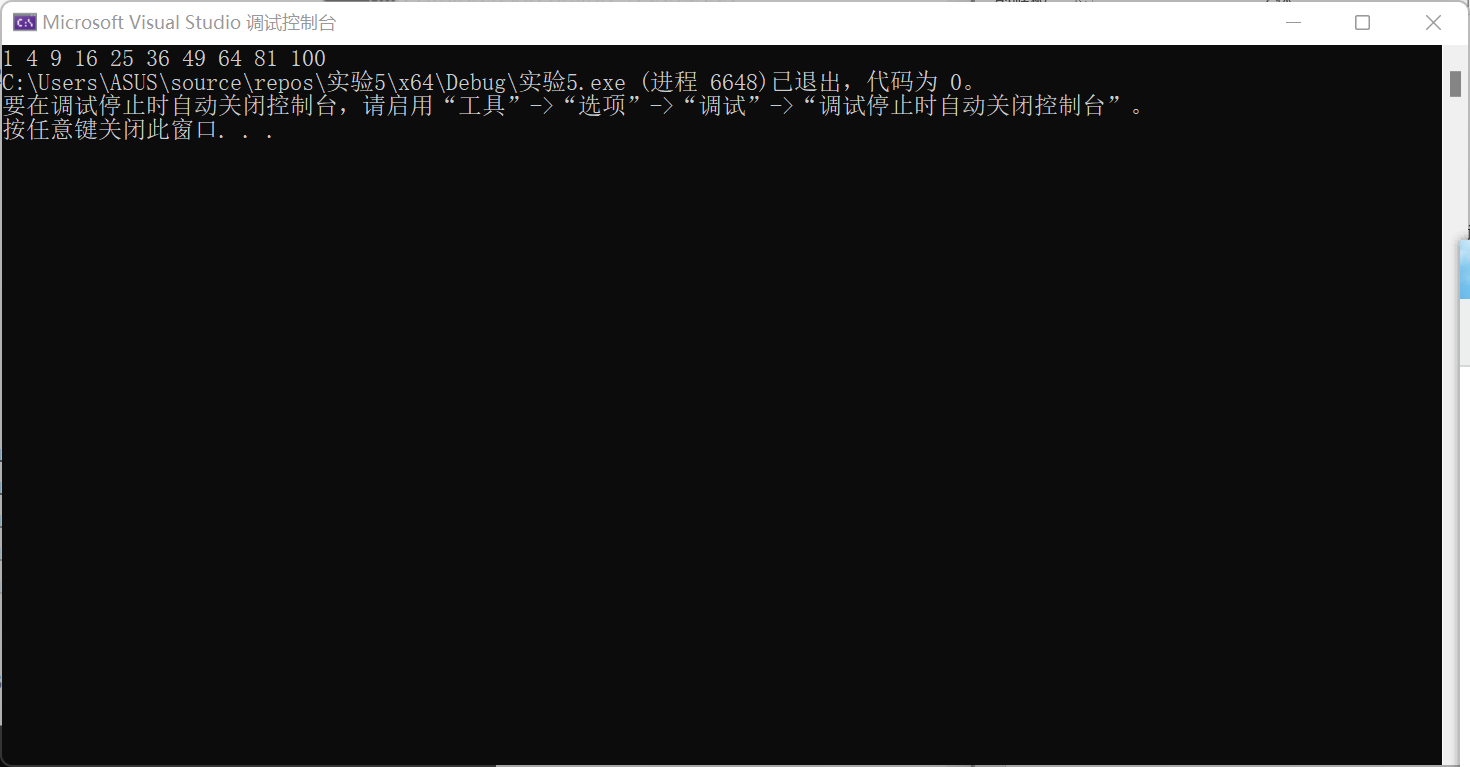
cout << i << " ";

}

}

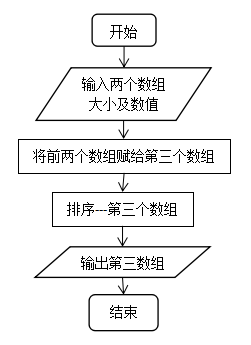
return 0;

}



4.

算法流程图：



程序：

#include<iostream>

using namespace std;

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

{

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

{

list3[i] = list1[i];

}

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

{

list3[size1 + i] = list2[i];

}

for (int b = 0; b < size1 + size2-1; b++)//冒泡排序

{

for (int i = 0; i < (size1 + size2 - 1); i++)

{

if (list3[i] >= list3[i + 1])

{

int t = 0;

t = list3[i];

list3[i] = list3[i + 1];

list3[i + 1] = t;

}

}

}

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

{

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

}

}

int main()

{

int size1, size2, size3;

cout << "enter size1: ";

cin >> size1;

cout << "enter list1: ";

int list1[80];

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

cin >> list1[i];

cout << "enter size2: ";

cin >> size2;

cout << "enter list2: ";

int list2[80];

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

cin >> list2[i];

size3 = size1 + size2;

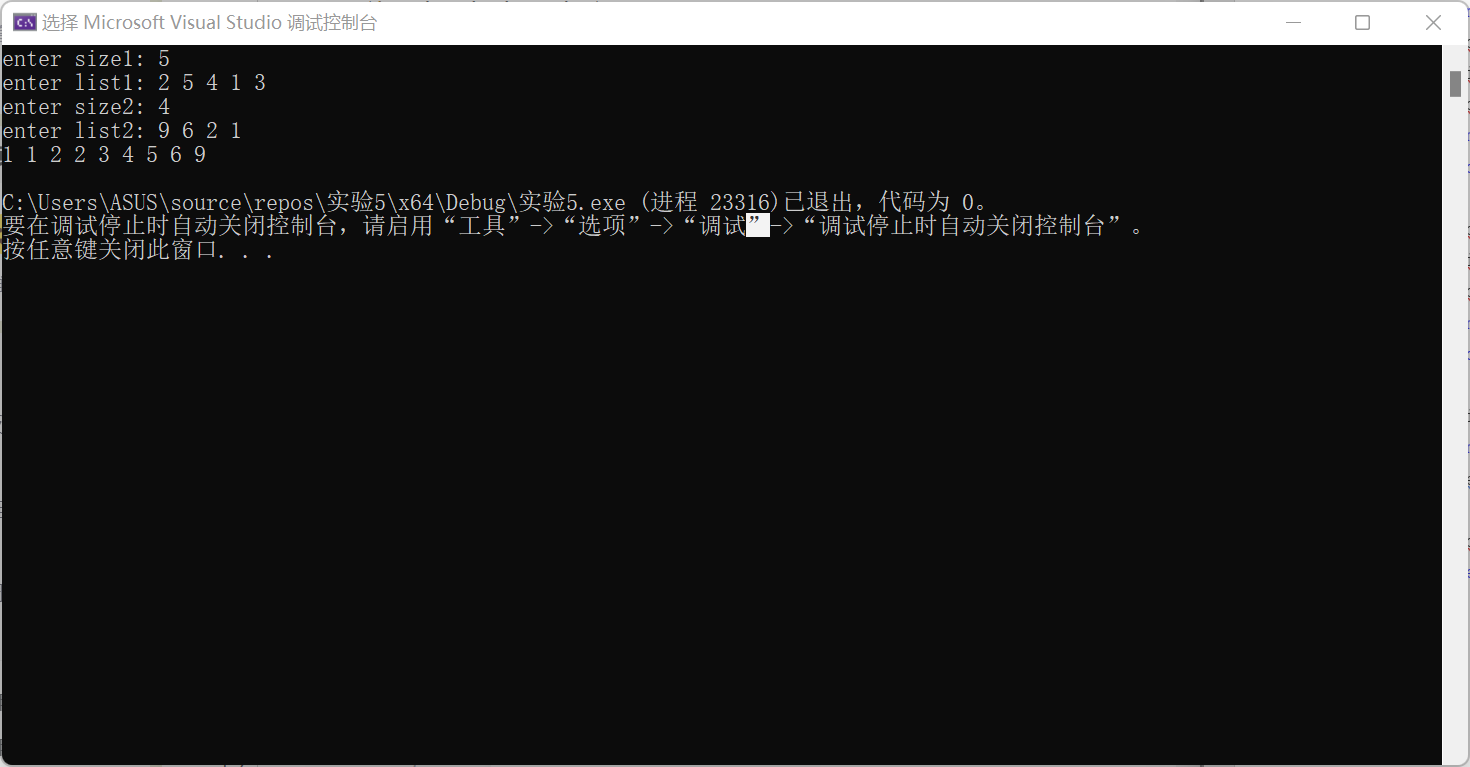
int list3[80];

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

cout << endl;

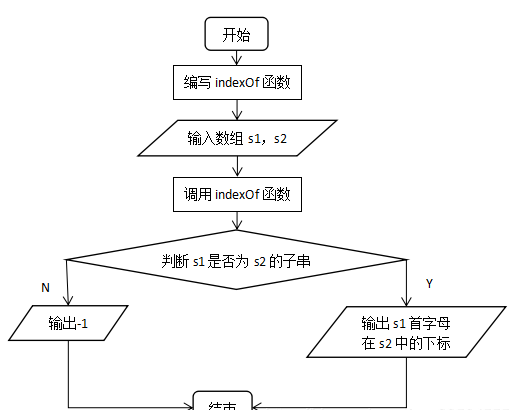
return 0;

}



5.

算法流程图：



程序：

#include<iostream>

#include<cstring>

using namespace std;

int indexOf(char\* s1, char\* s2);

int main()

{

int size = 999;

char s1[999], s2[999];

cout << "Enter the first string : ";

cin.getline(s1, size);

cout << "Enter the second string: ";

cin.getline(s2, size);

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

return 0;

}

int indexOf(char\* s1, char\* s2)

{

int a, b, index, j, k;

a = strlen(s1);

b = strlen(s2);

bool\* s3=new bool[a];

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

s3[m] = false;

for (int i = 0; i < b - a + 1; i++) //控制s2的首字母

{

for (j = 0, k = i; (j < a) && (k < i + a); j++, k++) //控制s2的总输出数

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

{

index = i;

s3[j] = true;

}

int h = 0;

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

{

if (s3[n] == false && i == b - a)

{

return -1; break;

}

if (s3[n])

h++;

}

if (h == a)

{

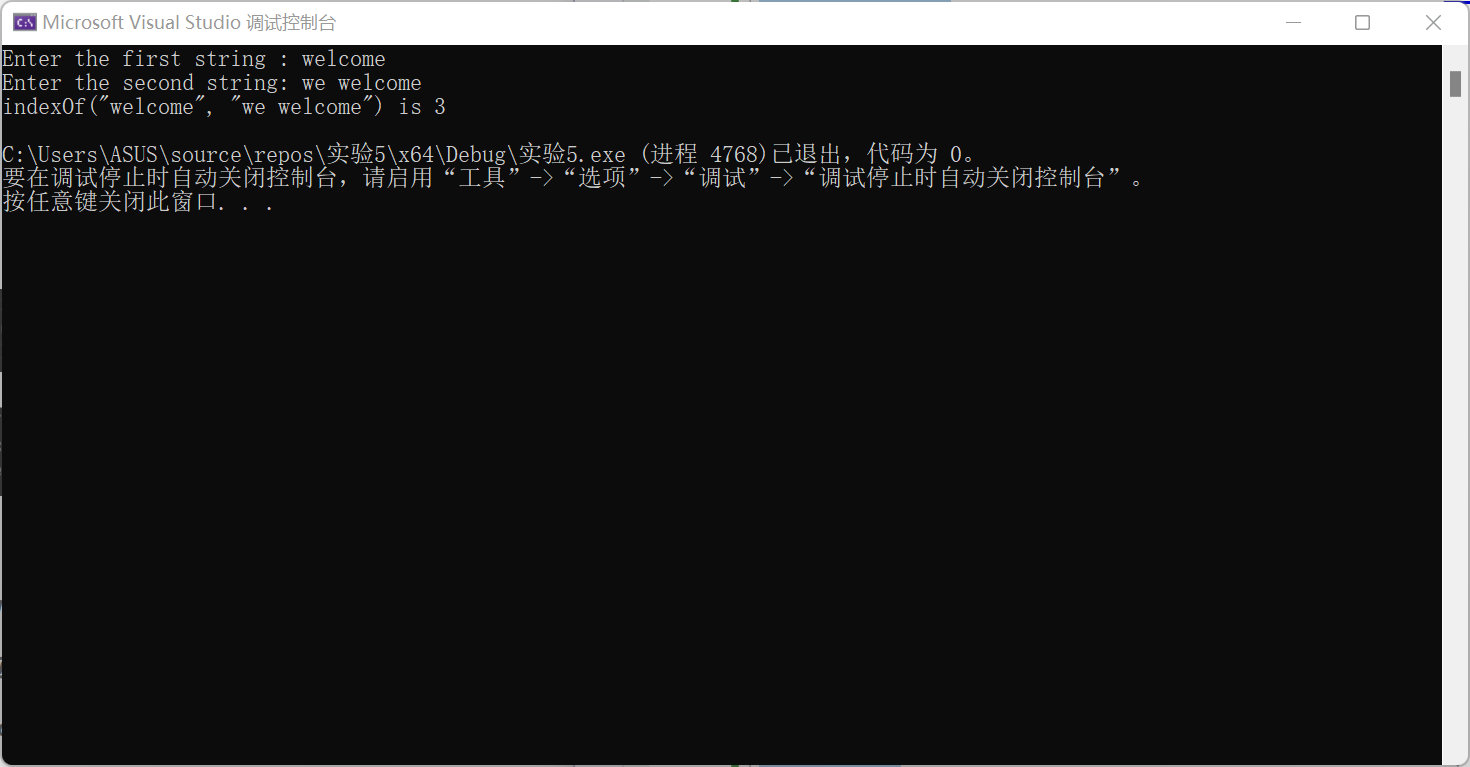
return index;

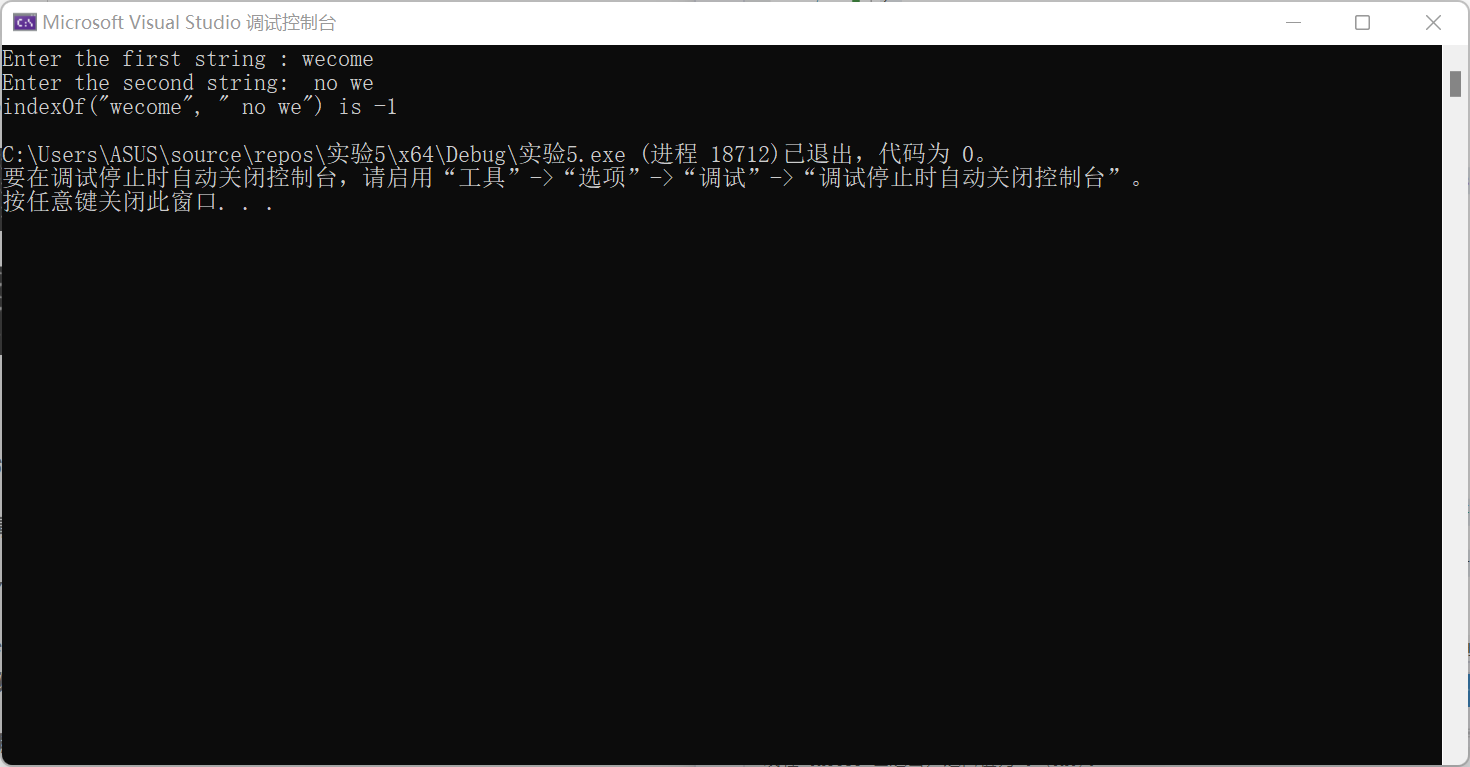
}

}

delete[]s3;

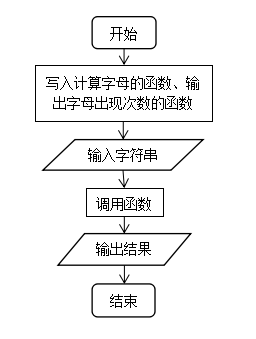
}





6.

算法流程图：

、

程序：

//#include<iostream>

//using namespace std;

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

//{

// int n;

// n = strlen(s);

//

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

// {

// // if (s[i] >= 65 && s[i] <= 90 || s[i] >= 97 && s[i] <= 122)

// // {

//

// //}

// for (int j = '65'; j <= '90'; j++)

// {

// if (s[i] == j)

// {

// counts[j - 'A']++;

// }

// }

// for (int k = '97'; k <= '122'; k++)

// {

// if (s[i] == k)

// {

// counts[k - 'a']++;

// }

// }

// for (int t = 0; t <= 25; t++)

// {

//

// if (counts[t] != 0)

// {

// cout <<char(t+'a') << counts[t] << " times" << endl;

// }

// }

// }

//}

//int main()

//{

// int counts[26];

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

// {

// counts[i] = 0;

// }

// char s[50];

// cin >> s;

// void count(const char s, int counts);

// return 0;

//}

#include <iostream>

using namespace std;

const int num1 = 26; //num1=letters

const int num2 = 99; //num2=string

void countLetters(const char list[], int counts[])

{

for (int i = 0; i < num1; i++)

counts[i] = 0;

for (int j = 0; j < num2; j++)

{

counts[list[j] - 'a']++;

counts[list[j] - 'A']++;

}

}

void displayCounts(const int counts[])

{

for (int i = 0; i < num1; i++) {

if (counts[i] != 0)

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

}

}

int main()

{

char list[num2];

int counts[num1];

cout << "Enter a string : ";

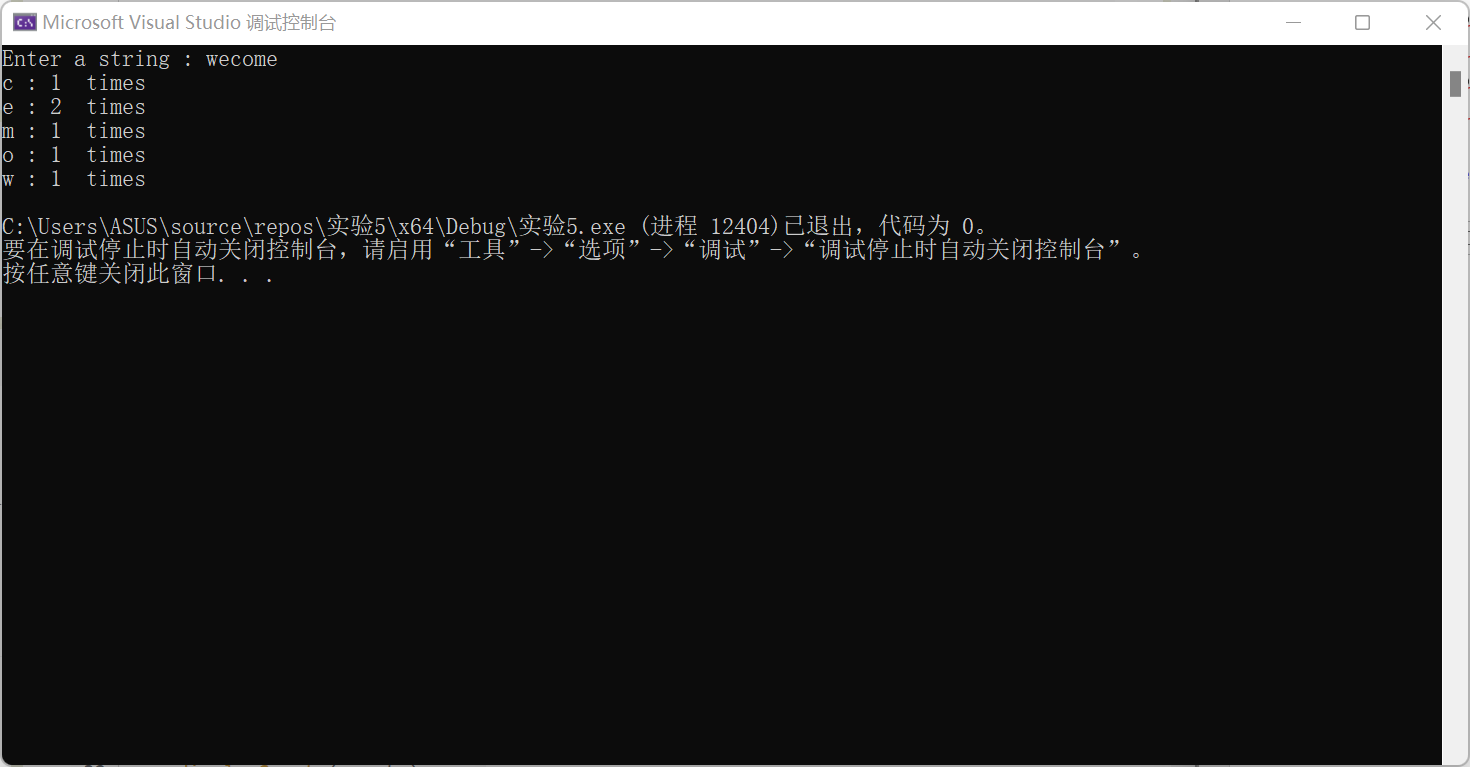
cin.getline(list, num2);

countLetters(list, counts);

displayCounts(counts);

return 0;

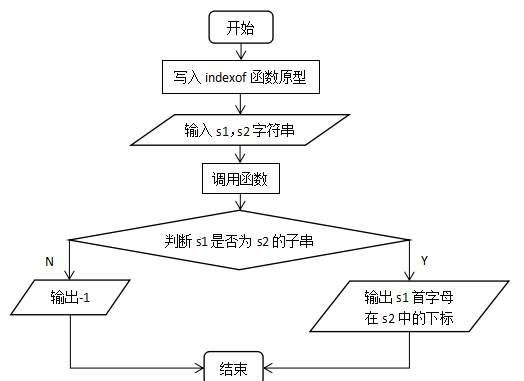
}



（二）

1.

算法流程图：



程序：

#include<iostream>

#include<cstring>

using namespace std;

int indexOf(char\* s1, char\* s2);

int main()

{

int size = 999;

char s1[999], s2[999];

cout << "Enter the first string : ";

cin.getline(s1, size);

cout << "Enter the second string: ";

cin.getline(s2, size);

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

return 0;

}

int indexOf(char\* s1, char\* s2)

{

int a, b, index, j, k;

a = strlen(s1);

b = strlen(s2);

bool\* s3=new bool[a];

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

s3[m] = false;

for (int i = 0; i < b - a + 1; i++) //控制s2的首字母

{

for (j = 0, k = i; (j < a) && (k < i + a); j++, k++) //控制s2的总输出数

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

{

index = i;

s3[j] = true;

}

int h = 0;

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

{

if (s3[n] == false && i == b - a)

{

return -1; break;

}

if (s3[n])

h++;

}

if (h == a)

{

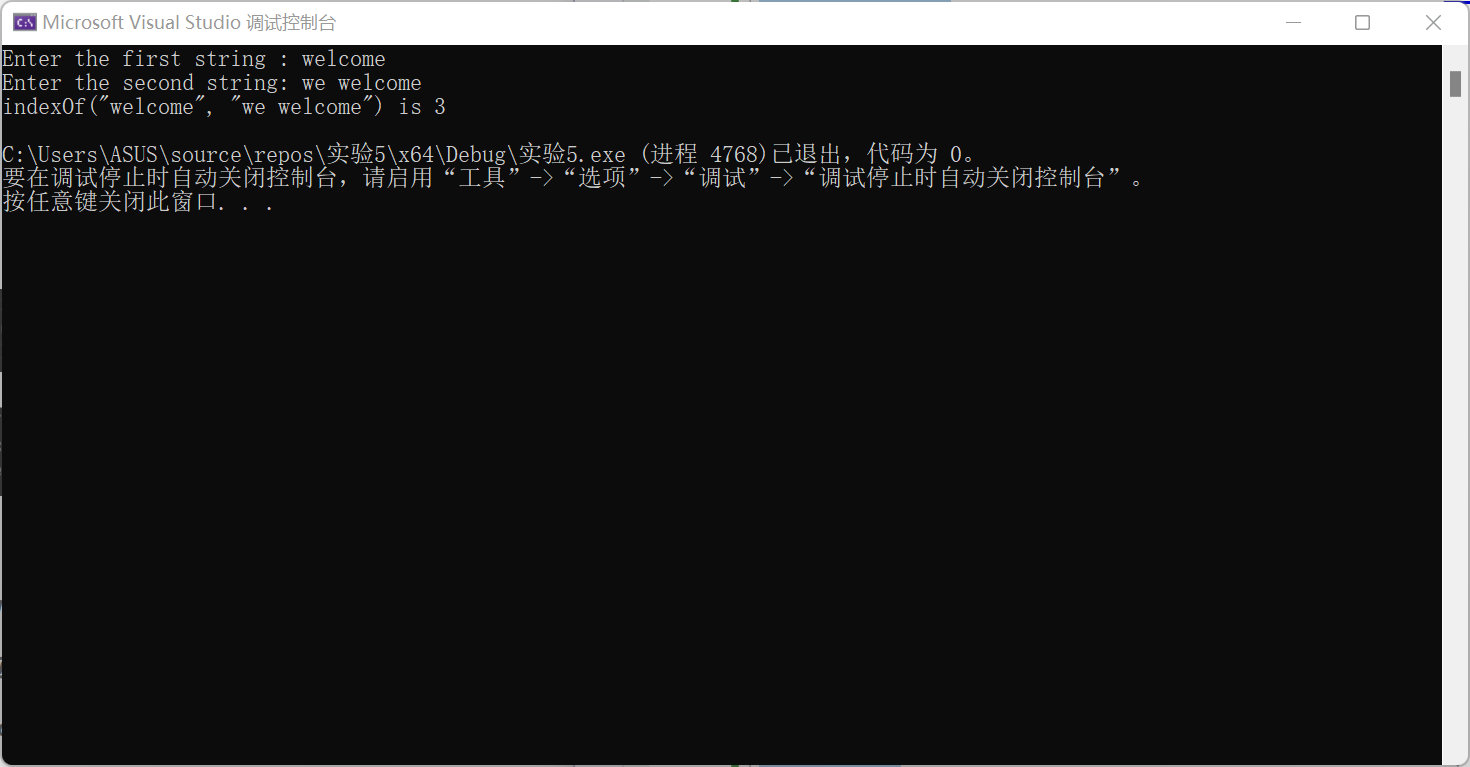
return index;

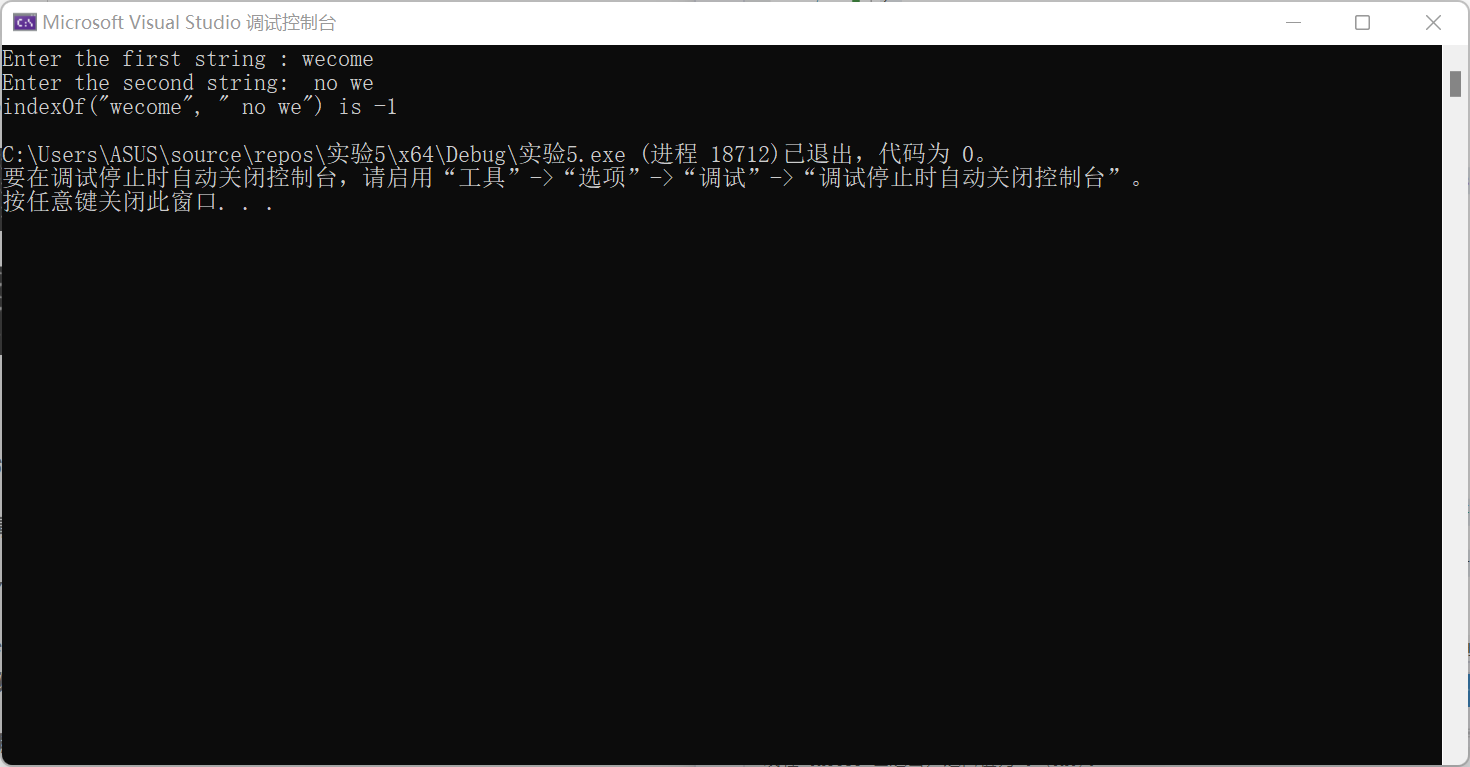
}

}

delete[]s3;

}





2.

程序：

#include<iostream>

#include<string>

using namespace std;

int parseHex(const char\* const hexString)

{

int n = strlen(hexString);

int num = 0;

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

{

//cout << hexString[i]<<endl ;

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

{

num = num + (hexString[i] - '0');

}

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

{

num += (hexString[i] - 'a' + 10);

}

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

{

num += (hexString[i] - 'A' + 10);

}

if (i < n - 1)

{

num = num \* 16;

}

}

//cout << n<<endl;

return num;

}

int main()

{

int size = 999;

char s[999];

cout << "Please enter a number in hexChar : \n";

//cin.getline(s, size);

cin >> s;

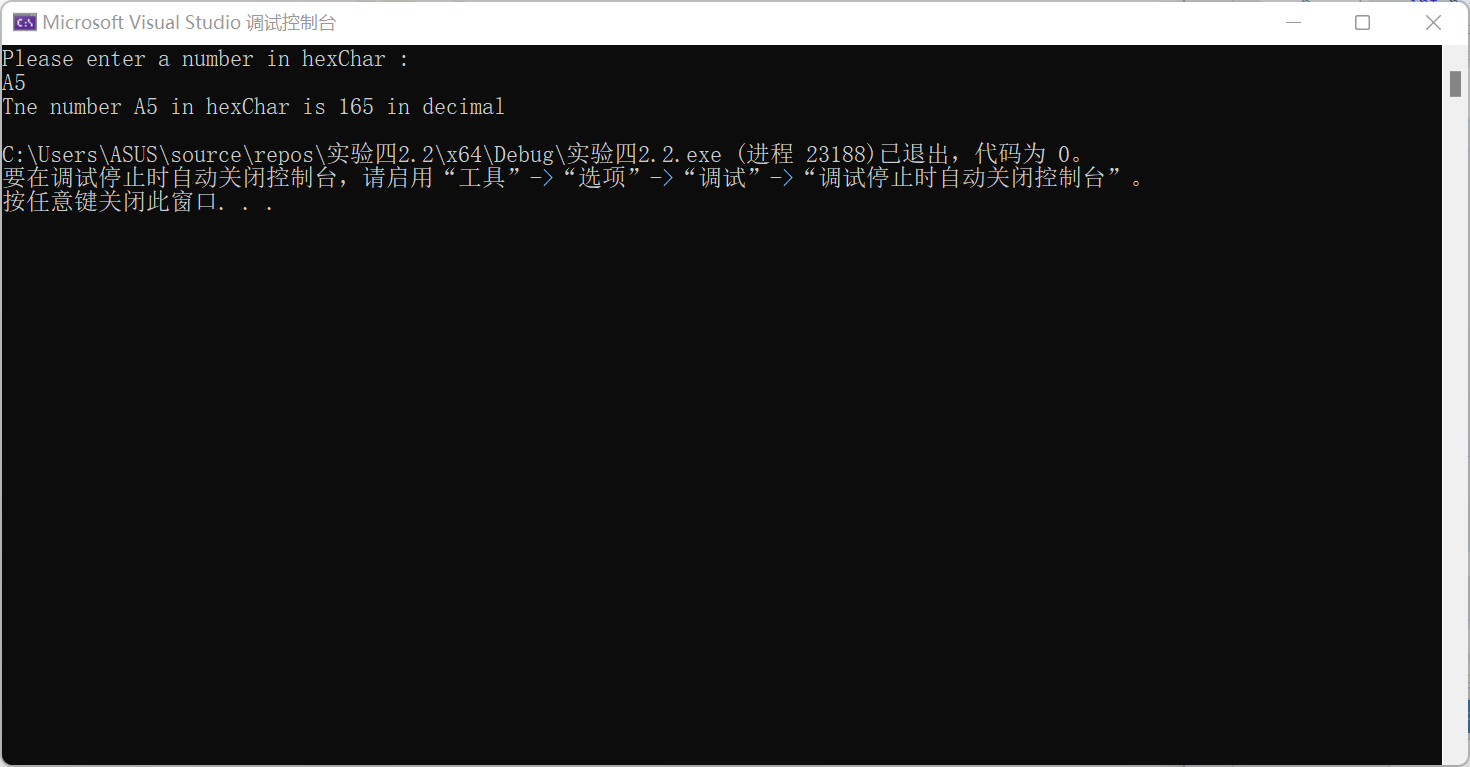
//for (int i = 0; i < 1; i++) { cin >> s[i]; }

cout << "Tne number " << s << " in hexChar is " << parseHex(s) << " in decimal\n";

return 0;

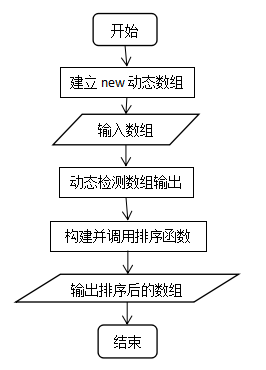
}//字符数组在部分初始化后，后面的元素自动赋值为’\0’

//(2)字符数组定义在main函数里面，(int)a[i]的值



3.

算法流程图：



程序：

#include<iostream>

using namespace std;

void arrange(int p[], int n);

int main()

{

int n;

cin >> n;

int\* p = new int[n];

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

{

cin >>\* (p+i);

}

arrange(p,n);

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

{

cout << \*(p + b) << " ";

}

delete []p;

return 0;

}

void arrange(int p[], int n)

{

int temp;

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

{

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

{

if (\*(p + j) > \*(p + j + 1))

{

temp = \*(p + j);

\*(p + j) = \*(p + j + 1);

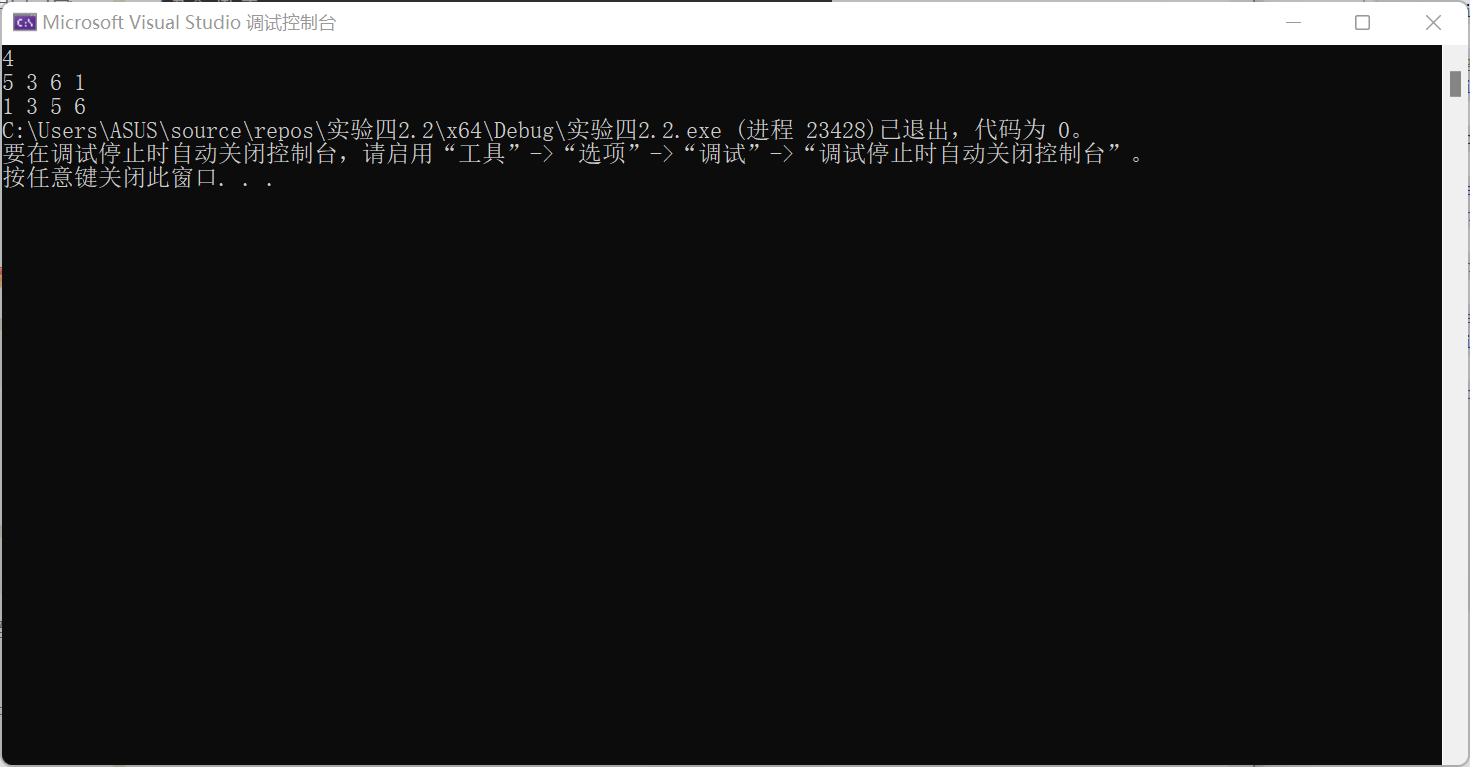
\*(p + j + 1) = temp;

}

}

}

}



遇到的问题与体会：

有时候每次代码看懂了，复制能够运行，然后自己码一遍，结果运行没问题，但就是结果出了问题，前前后后仔细对比，就是不知道哪里错了，感觉都是对的啊，每次到最后找到问题所在时，基本上都是咪咪小问题被忽略了，就造成了大问题，比如多个或少个分号啊空格，或者数据类型搞的不对、少个简单但很关键的语句等，好的习惯需要更多训练。