实验四

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

第一题：

#include<iostream>

using namespace std;

int main()

{

int a[10];

int i=0, j=0;

cout << "输入十个数" << endl;

while (i < 10)

{

cin >> a[i++];

}

cout << "输出结果为：" << endl;

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

{

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

{

if (a[i] == a[j])break;

}

if (i == j)

cout << a[i] << " ";

}

return 0;

}

结果：



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

第二题：

#include<iostream>

using namespace std;

void s(double list[10])

{

;

bool changed = true;

int i = 0;

int a;

do

{

changed = false;

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

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

{

a = list[j];

list[j] = list[j + 1];

list[j + 1] = a;

changed = true;

}

} while (changed);

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

{

cout << list[i] << " ";

}

;

}

int main()

{

int i = 0;

double list[10];

while (i < 10)

{

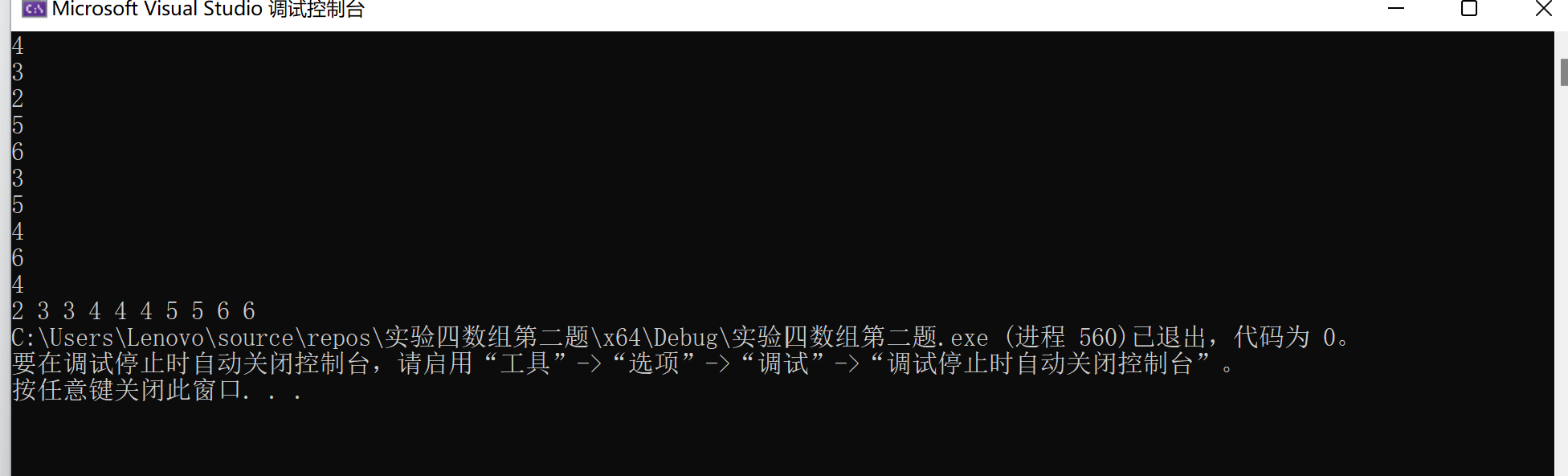
cin >> list[i++];

}

s(list);

}

结果：



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。

第三题：

#include<iostream>

using namespace std;

int main()

{

bool list[100];

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

{

list[i] = false;

}

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

{

for (int k = j; k < 100; k += j+1)

{

if (list[k])

list[k] = false;

else list[k] = true;

}

}

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

{

if (list[i])

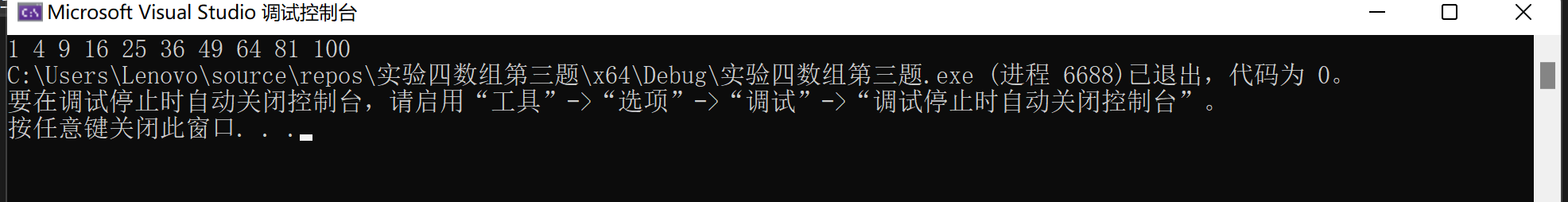
cout << i+1 << " ";

}

return 0;

}

结果：



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

第四题：

#include<iostream>

using namespace std;

void merge(int size1, int list1[], int size2, int list2[], int list3[], int size3)

{

int m;

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

{

list3[i] = list1[i];

}

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

{

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

}

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

{

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

{

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

m = list3[b];

list3[b] = list3[b + 1];

list3[b + 1] = m;

}

}

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

{

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

}

}

int main()

{

int size1, size2, size3;

cout << "Enter size1: ";

cin >> size1;

cout << "Enter list1: ";

int list1[999];

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

cin >> list1[i];

cout << "Enter size2: ";

cin >> size2;

cout << "Enter list2: ";

int list2[999];

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

cin >> list2[i];

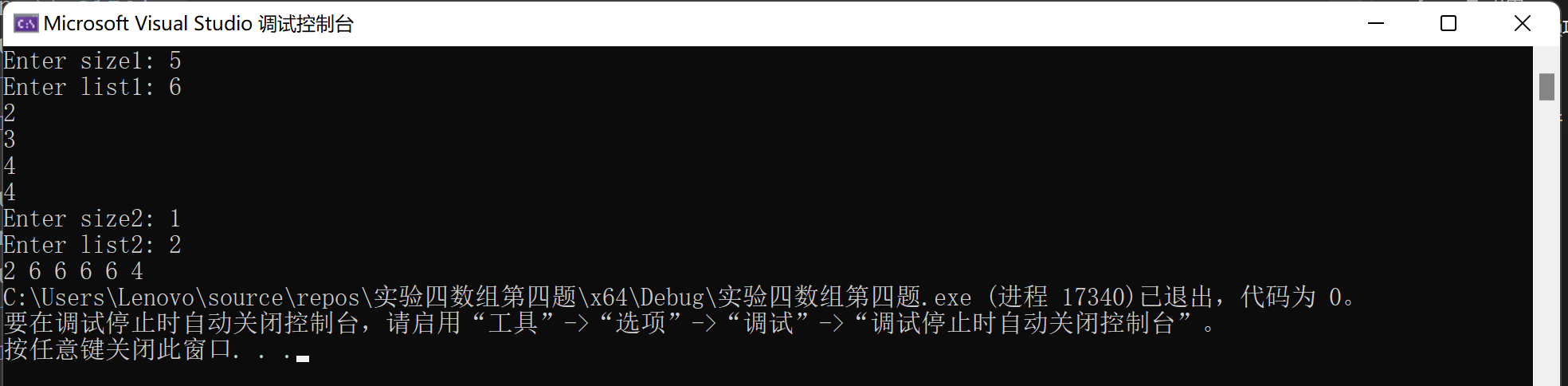
size3 = size1 + size2;

int list3[999];

merge( size1, list1,size2, list2, list3,size3);

}

结果：



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

第五题：

#include<iostream>

using namespace std;

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

{

int a, b,n;

a = strlen(s1);

b = strlen(s2);

bool \*s3 = new bool[a];

for (int i = 0; i < b - a + 1; i++)

{

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

{

s3[m] = false;

}

for (int j = 0, k = i; (j < a) && (k < i + a); j++, k++)

{

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

{

n = i;

s3[j] = true;

}

}

int h = 0;

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

{

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

{

return -1;

}

if (s3[q])

{

h++;

}

}

if (h == a)

{

return n + 1;

}

}

delete[]s3;

}

int main()

{

int size = 999;

char s1[999];

char s2[999];

cout << "输入第一个字符串" << endl;

cin.getline(s1, size);

cout << "输入第二个字符串" << endl;

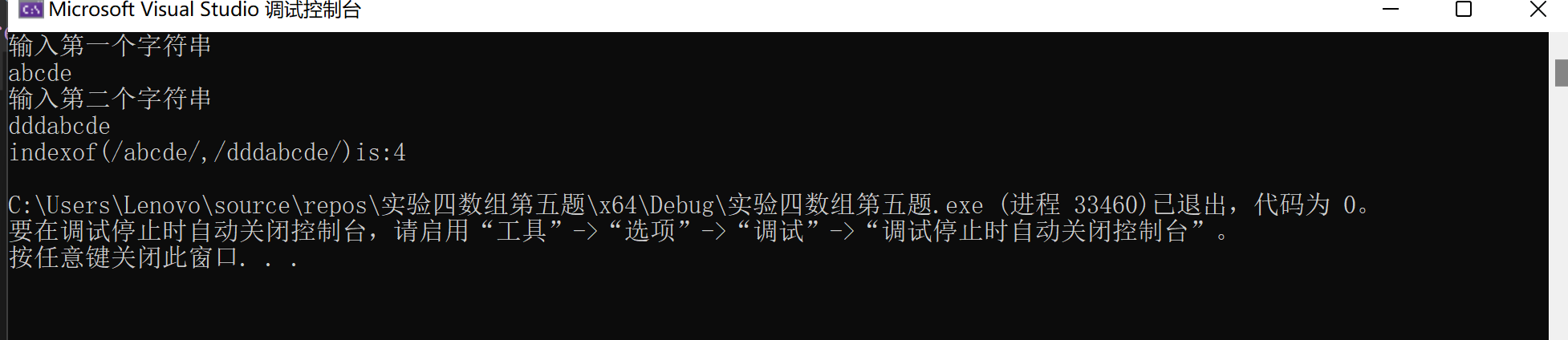
cin.getline(s2, size);

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

return 0;

}

结果：



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

Enter11

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

第六题：

#include <iostream>

using namespace std;

const int num1 = 26;

const int num2 = 999;

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

{

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

{

counts[m] = 0;

}

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

{

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

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

}

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

{

if(counts[a]!=0)

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

}

}

int main()

{

char list[num2];

int counts[num1];

cout << "输入字符串" << endl;

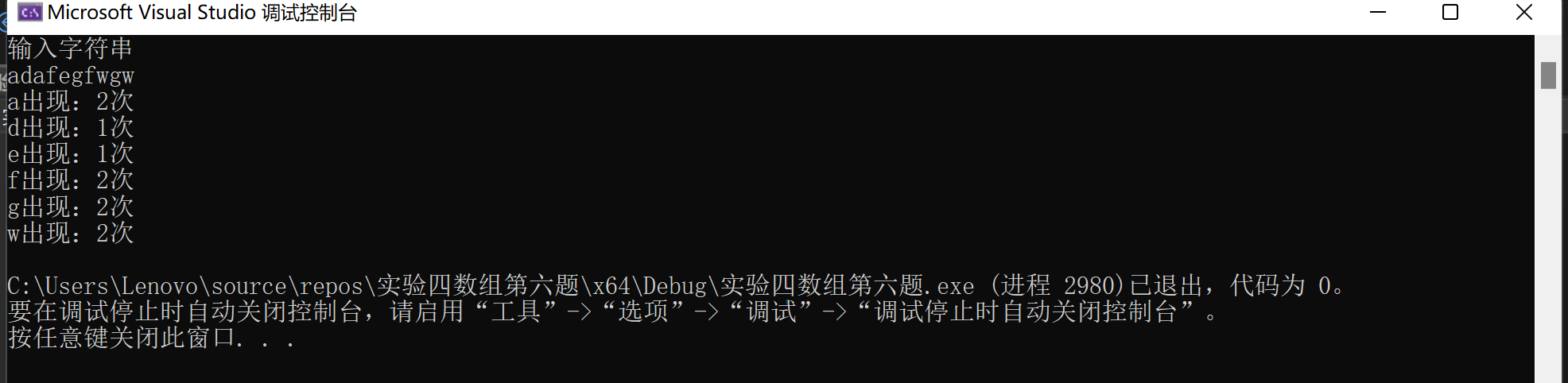
cin.getline(list, num2);

count(list, counts);

return 0;

}

结果：



(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);

第一题：

#include<iostream>

using namespace std;

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

{

int a, b,n;

a = strlen(s1);

b = strlen(s2);

bool \*s3 = new bool[a];

for (int i = 0; i < b - a + 1; i++)

{

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

{

s3[m] = false;

}

for (int j = 0, k = i; (j < a) && (k < i + a); j++, k++)

{

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

{

n = i;

s3[j] = true;

}

}

int h = 0;

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

{

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

{

return -1;

}

if (s3[q])

{

h++;

}

}

if (h == a)

{

return n + 1;

}

}

delete[]s3;

}

int main()

{

int size = 999;

char s1[999];

char s2[999];

cout << "输入第一个字符串" << endl;

cin.getline(s1, size);

cout << "输入第二个字符串" << endl;

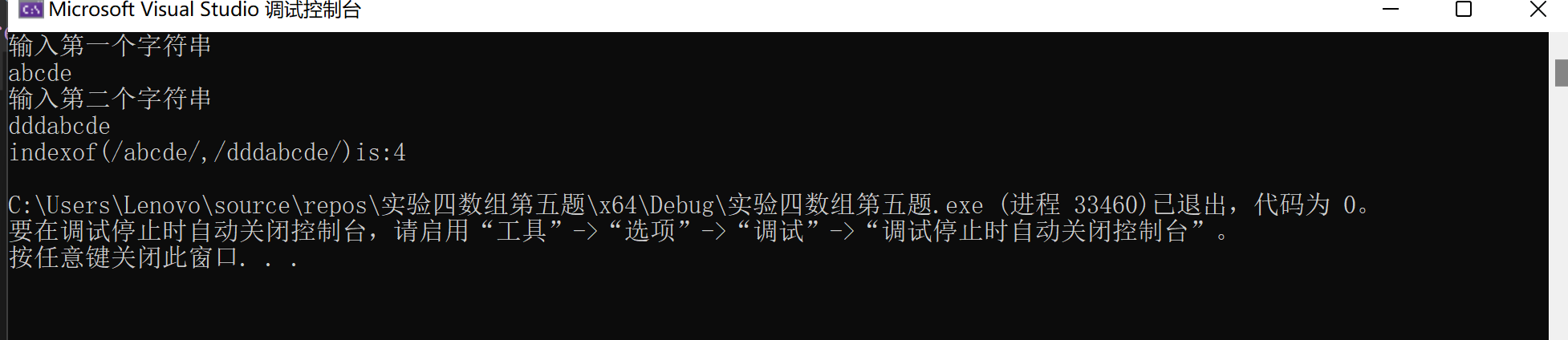
cin.getline(s2, size);

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

return 0;

}

结果：



(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

第二题：

#include<iostream>

using namespace std;

int parseHex(const char\* const hexString)

{

int n;

int num=0;

n = strlen(hexString);

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

{

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

{

num = num + (hexString[i] - '0') \* pow(16, n - 1 - i);

}

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

{

num = num + (hexString[i] - 'a') \* pow(16, n - 1 - i);

}

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

{

num = num + (hexString[i] - 'A') \* pow(16, n - 1 - i);

}

}

return num;

}

int main()

{

int a;

const int size = 10;

char hexString[size];

cout << "输入十六进制" << endl;

cin.getline(hexString, size);

cout << "十进制为：";

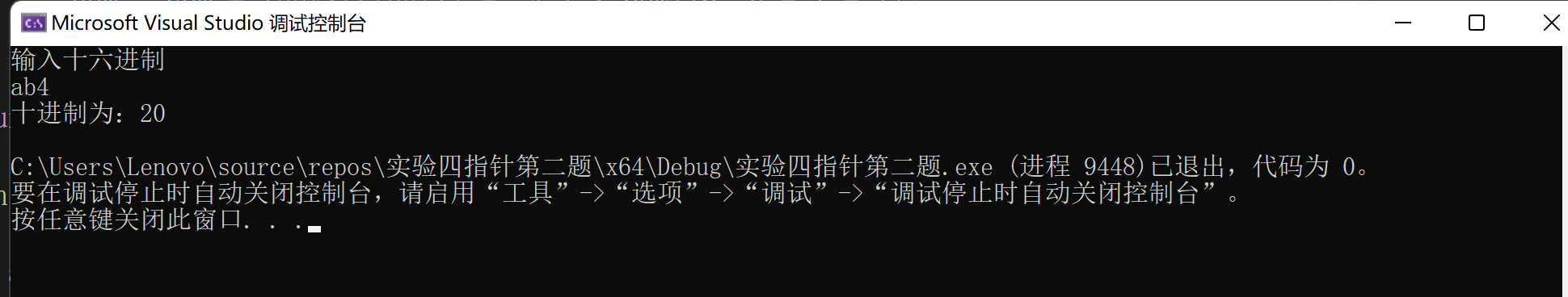
a=parseHex(hexString);

cout << a << endl;

return 0;

}

结果：



(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）

第三题：

#include<iostream>

using namespace std;

void t(int\* a, int num1)

{

int m;

for (int i = 0; i < num1-1; i++)

{

for (int k = 0; k < num1 - 1; k++)

{

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

{

m = a[k];

a[k] = a[k + 1];

a[k + 1] = m;

}

}

}

}

int main()

{

int n;

cout << "输入数组元素的个数" << endl;

cin >> n;

int\* p = new int[n];

cout << "输入数组" << endl;

int i = 0;

while (i < n)

{

cin >> p[i++];

}

t(p, n);

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

{

cout << p[t] << " ";

}

delete[]p;

return 0;

}

结果：

