**计算机程序设计基础（C++）**

**实验报告**

专业班级 软件工程2206班

学号 8209220608

姓名 夏铎恺

**实验报告成绩：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验** | **实验二** | **实验三** | **实验四** | **实验五** | **总评** |
| **成绩** |  |  |  |  |  |

**批阅老师:\_\_\_\_\_\_\_\_**

**实验四 指针与数组**

1. **实验目的与要求**

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。

1. **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）。

Sample in 1 2 3 2 1 6 3 4 5 2

Sample out 1 2 3 6 4 5

2.起泡排序：

编写测试程序，读入一个含有 10 个双精度数字的数组，调用bubble\_sort函数并显示排列后的数字。

参见 ：<https://oi-wiki.org/basic/bubble-sort/>

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。

假定数组大小不超过 80。

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

Sample in:

Enter list1: 5 1 5 16 61 111

Enter list1: 4 2 4 5 6

Sample out:

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[])

Sample in:

welcome

We welcome you!

Sample out:

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

Sample in :

Welcome

We invite you!

Sample out:

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。

Sample in:

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

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）。

1. **实验步骤、算法与结果分析**
2. **数组**

**第一题：**

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

**Code：**

#include <iostream>

#include<iomanip>

using namespace std;

int main()

{

int a[10];

int i, j;

bool t = 1;

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

{

cin >> a[i];

}

for (i = 0; i < 10; i++, t = 1)

{

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

{

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

t = 0;

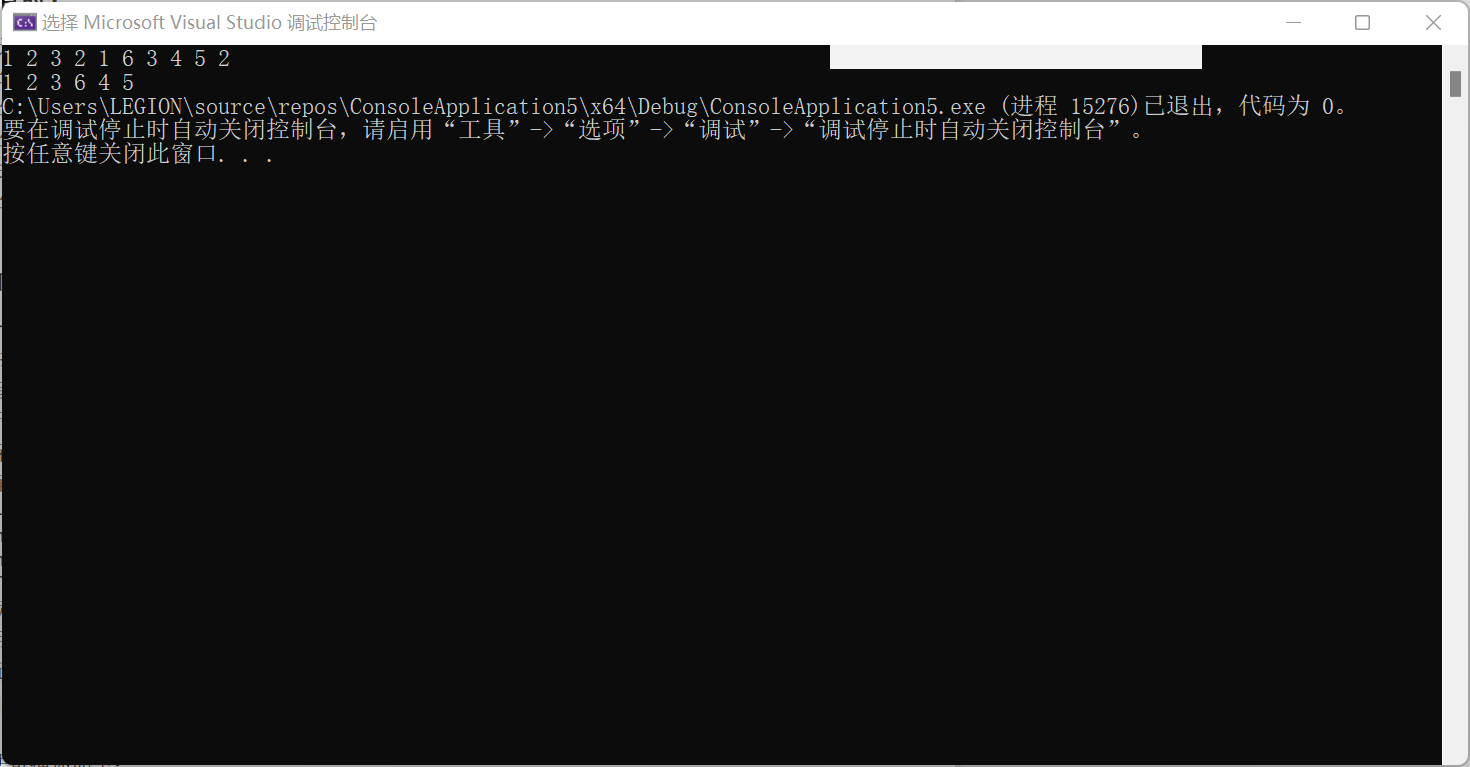
}

if (t == 1) cout << a[i] << setw(2);

}

}

**Result：**



**第二题：**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。

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

**Code：**

#include <iostream>

using namespace std;

void bubblesort(double a[], int size)

{

int i, j; double t;

for (i = 0; i < size - 1; i++)

{

for (j = i + 1; j < size; j++)

{

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

{

t = a[j]; a[j] = a[i]; a[i] = t;

}

}

}

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

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

}

int main()

{

double a[10] = { 2.01,5.02,6.03,6.98,7.64,1.23,6.14,9.36,1.36,4.95 };

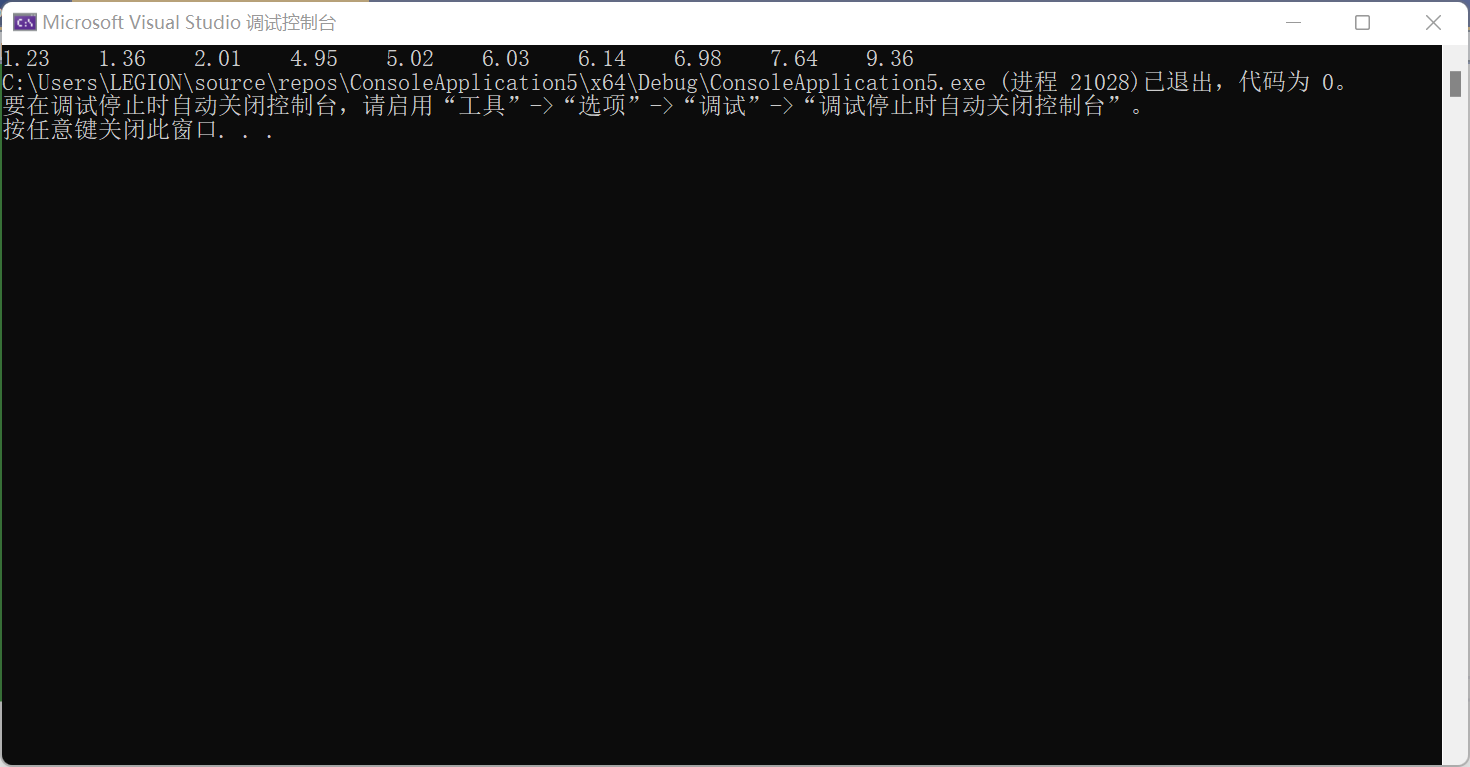
int size = sizeof(a) / sizeof(a[0]);

bubblesort(a, size);

return 0;

}

**Result：**



**第三题：**、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

**Code：**

#include <iostream>

using namespace std;

int main()

{

bool a[100];

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

{

a[i] = 1;

}

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

{

for (int j = 0; j < 100; j = j + n + 1)

{

if (a[j] == 0)

a[j] = 1;

else a[j] = 0;

}

}

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

{

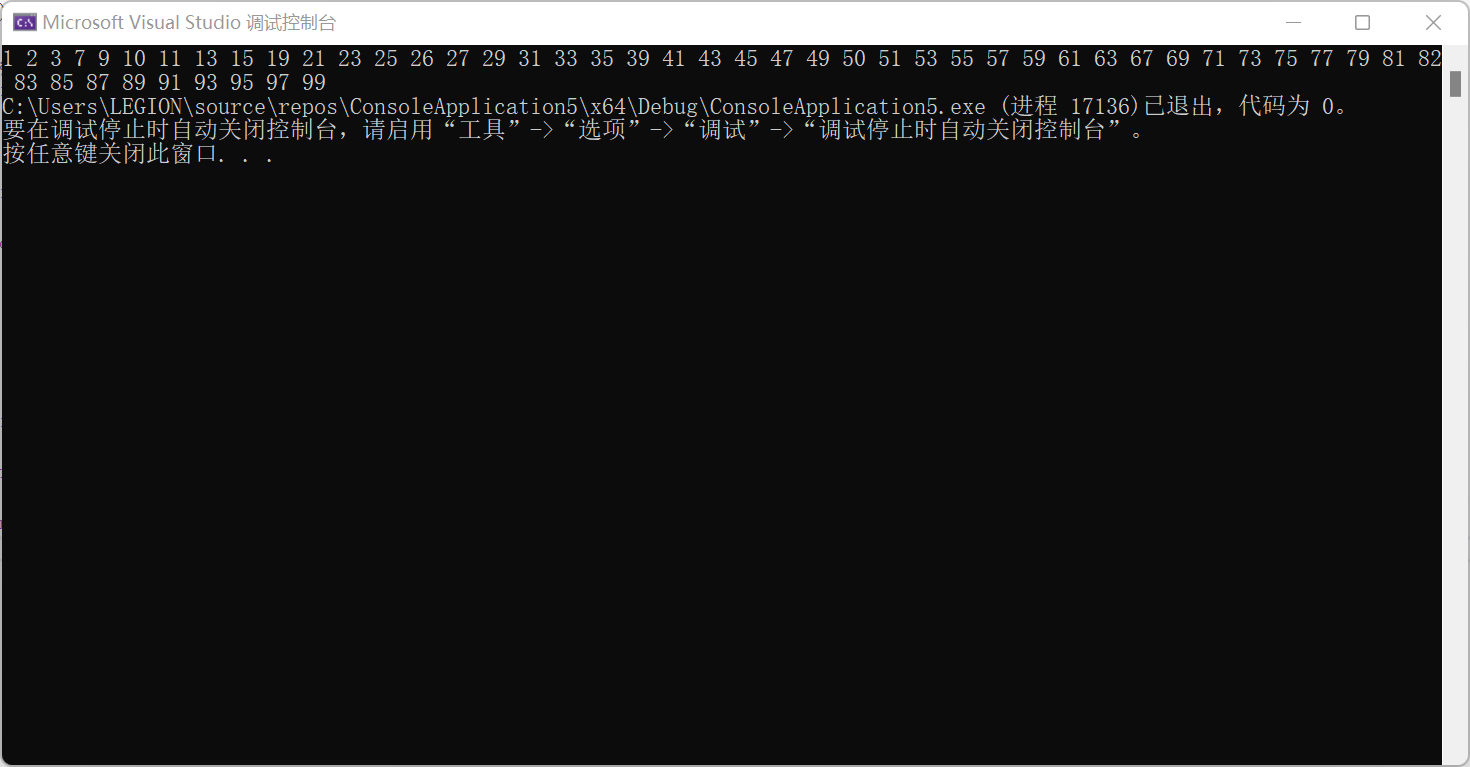
if (a[k] == 1) cout << k + 1 << " ";

}

return 0;

}

**Result：**



**第四题：**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

**Code：**

#include <iostream>

using namespace std;

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

{

int z = size1 + size2;

int\* z1 = new int[z];

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

{

z1[d] = list1[d];

}

for (int d = 0; d < z; d++, size1++)

{

z1[size1] = list2[d];

}

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

{

for (int c = b + 1; c < z; c++)

{

if (z1[b] > z1[c])

{

int t;

t = z1[b];

z1[b] = z1[c];

z1[c] = t;

}

}

}

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

cout << z1[b] << " ";

delete[]list1;

delete[]list2;

delete[]z1;

}

int main()

{

int n, m;

cout << "第一个数组元素的个数：";

cin >> n;

cout << "第二个数组元素的个数：";

cin >> m;

int\* p = new int[n];

int\* q = new int[m];

cout << "输入第一个数组元素:";

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

cin >> p[i];

cout << "输入第二个数组元素:";

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

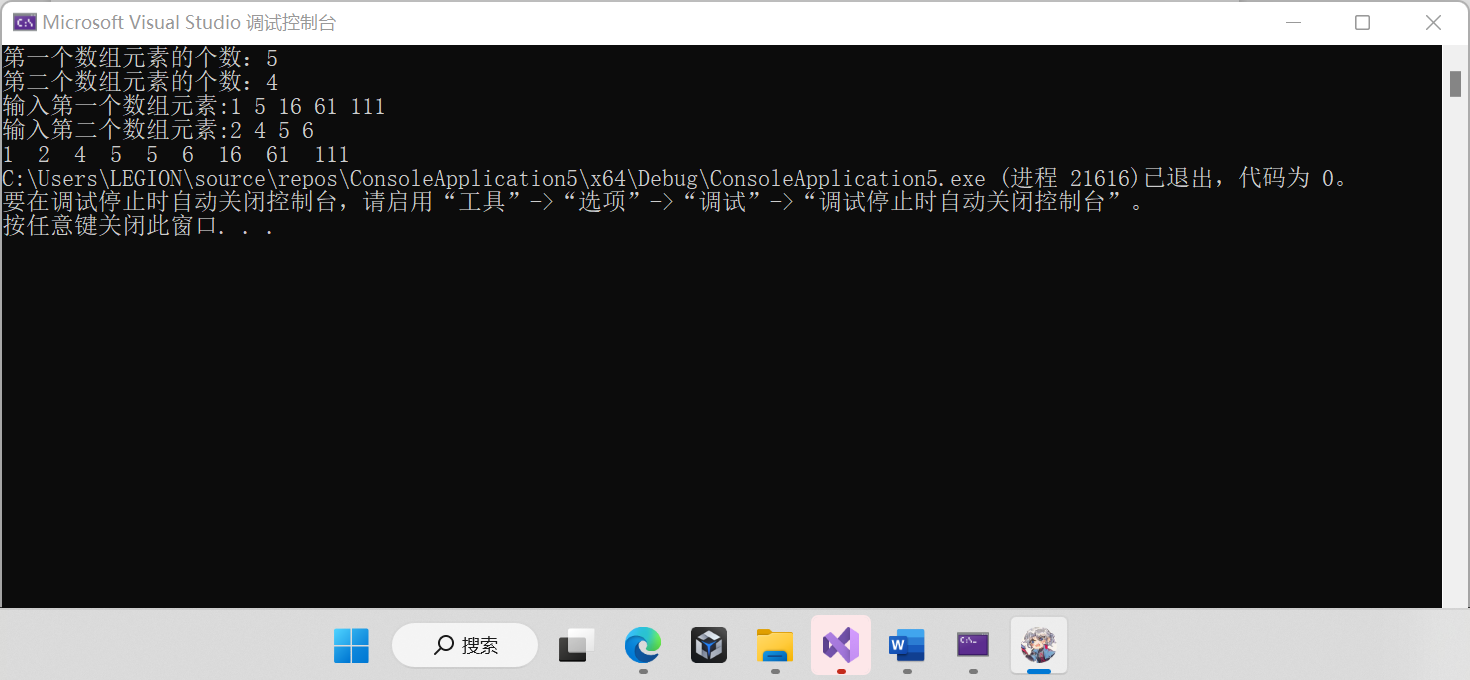
cin >> q[j];

merge(p, n, q, m);

return 0;

}

**Result：**



**第五题：**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[]

**Code：**

#include <iostream>

using namespace std;

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

{

int a, b, m = 0, location, i, j;

a = strlen(s1);

b = strlen(s2);

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

{

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

{

if (s1[j] == s2[j + i] && (j == a - 1))

{

return i;

}

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

{

break;

}

}

}

return -1;

}

int main()

{

char s1[100], s2[100];

cout << "请输入第一行字符串：" << endl;

cin.getline(s1, 100);

cout << "请输入第二行字符串：" << endl;

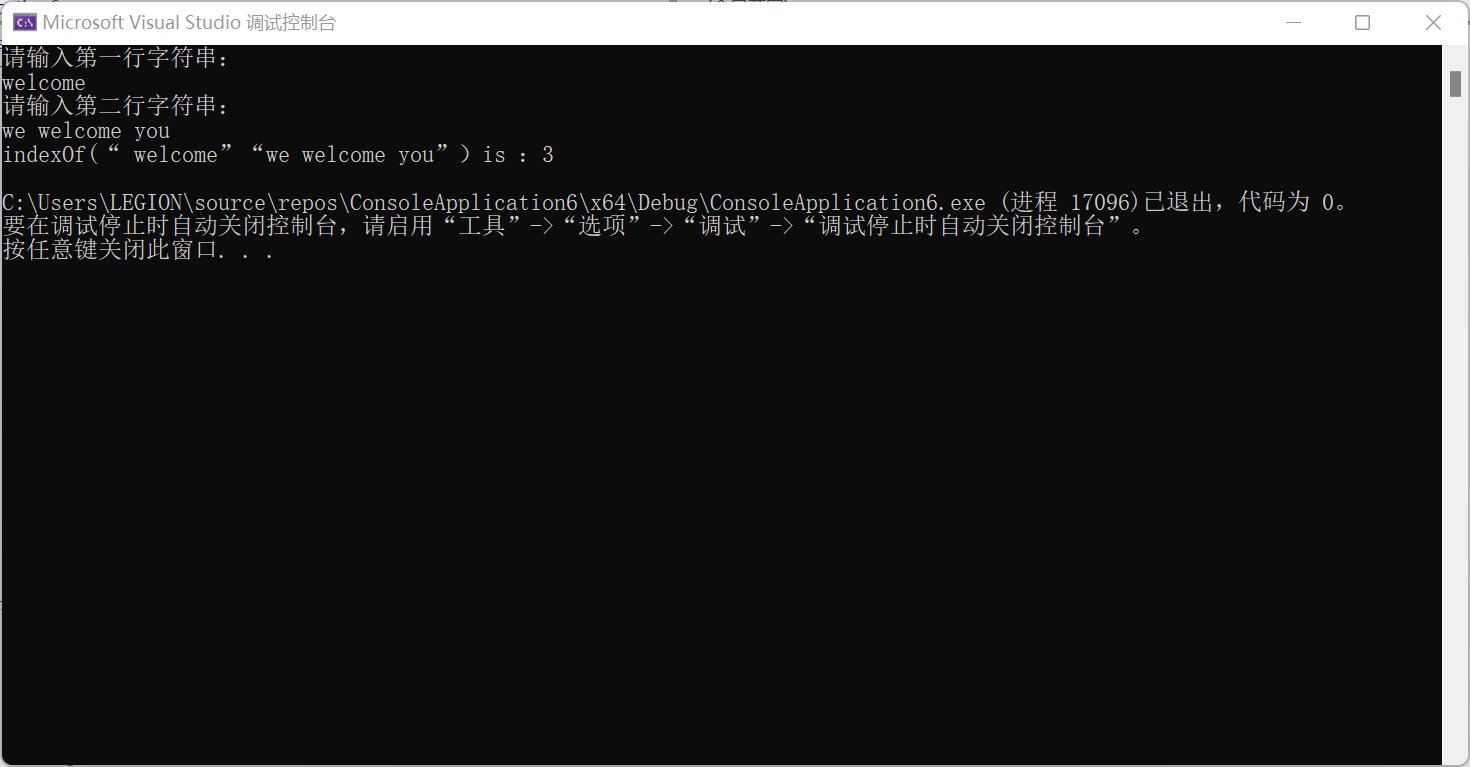
cin.getline(s2, 100);

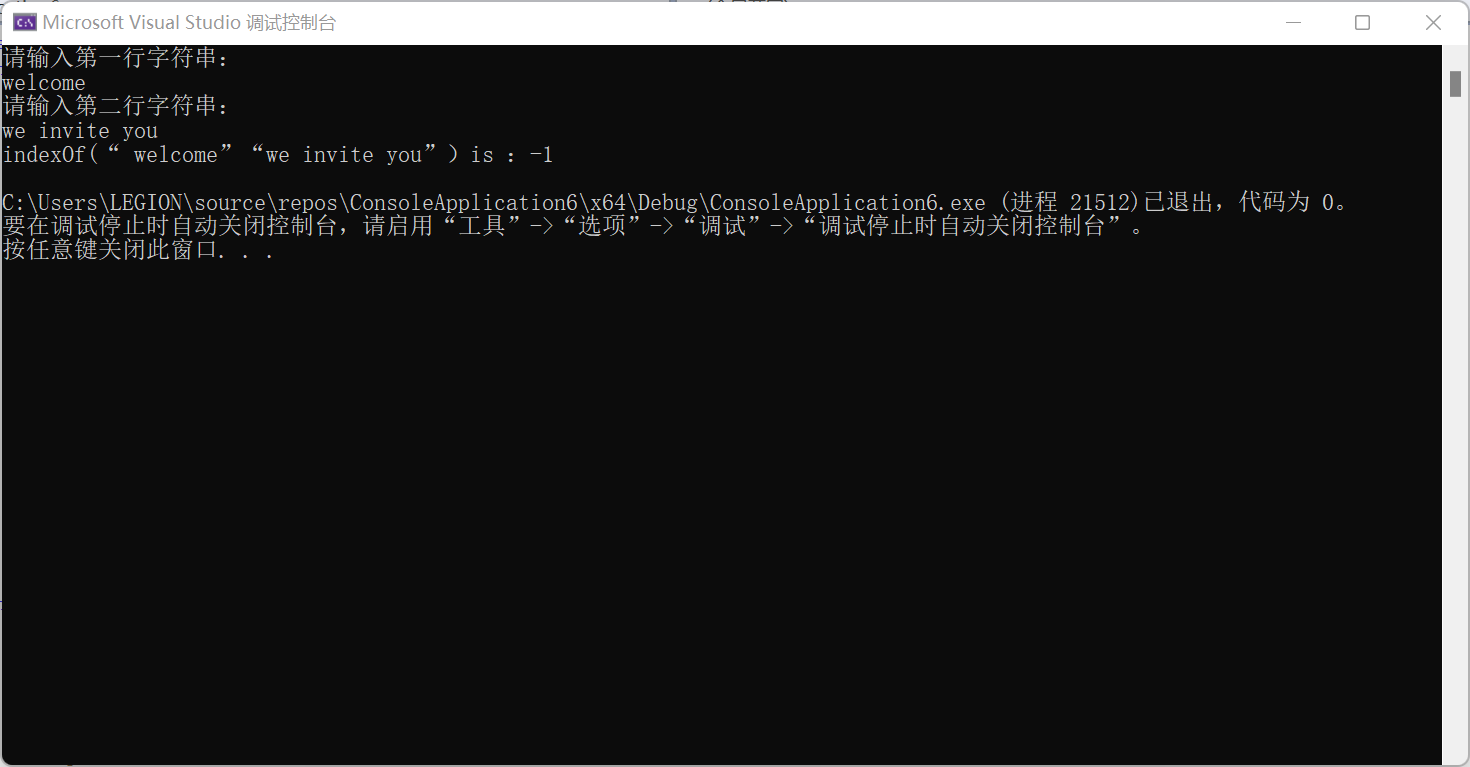
cout << "indexOf(“ " << s1 << "”" << "“" << s2 << "”）is ：" << indexOf(s1, s2) << endl;

return 0;

}

**Result：**





**第六题：**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。

**Code：**

#include <iostream>

using namespace std;

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

{

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

{

counts[i] = 0;

}

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

{

if ((s[j] <= 90) && (s[j] >= 65))

{

counts[s[j] - 65]++;

}

else if ((s[j] <= 122) && (s[j] >= 97))

{

counts[s[j] - 97]++;

}

}

}

void show(int counts[])

{

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

if (counts[t] != 0)

{

cout << static\_cast<char>('a' + t) << ":" << counts[t] << "times" << endl;

}

}

int main()

{

char s[100];

int counts[26];

cout << "请输入字符串:" << endl;

cin.getline(s, 100);

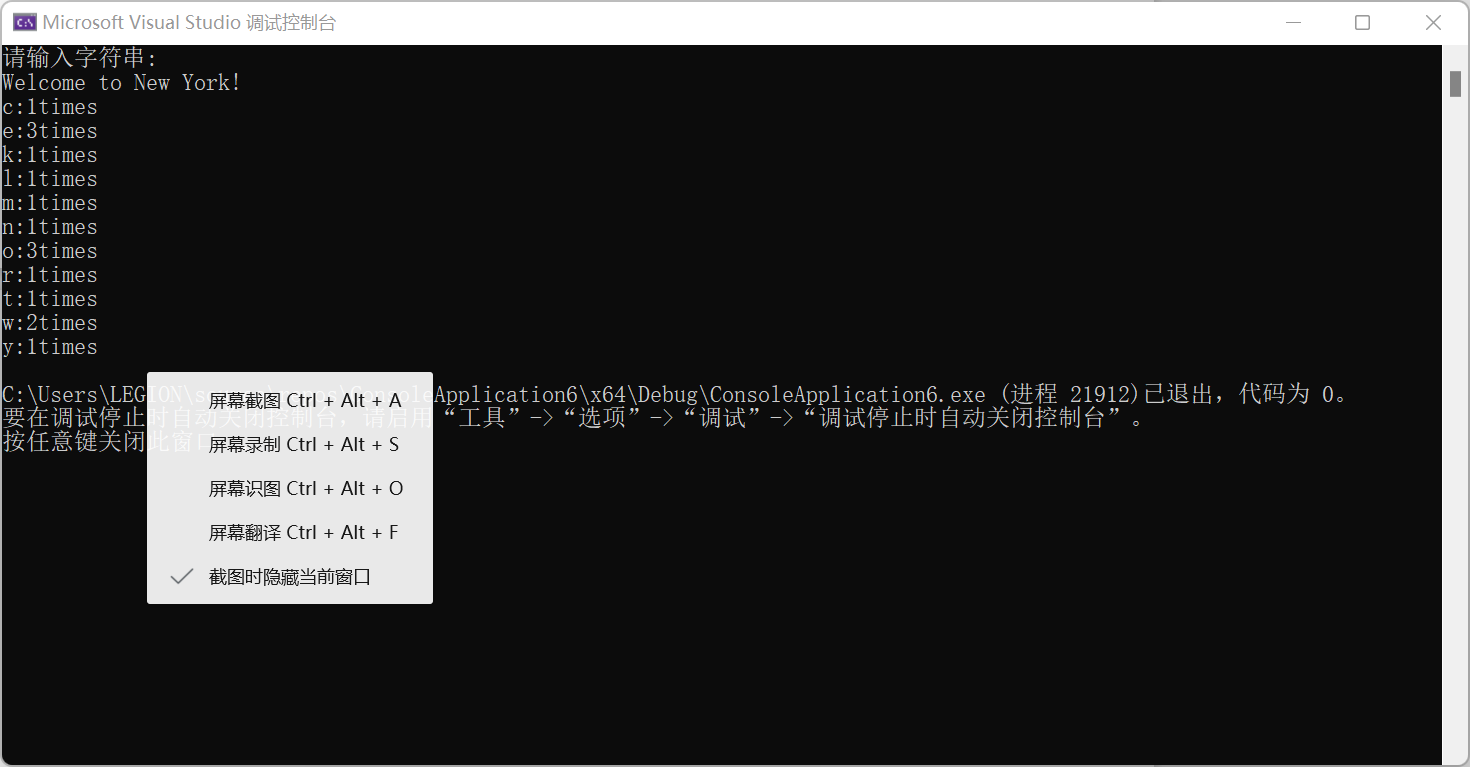
count(s, counts);

show(counts);

return 0;

}

**Result：**



1. **指针**

**1.**

**（1）**

**Code：**

#include <iostream>

using namespace std;

int main()

{

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

pi = &i;

pj = &j;

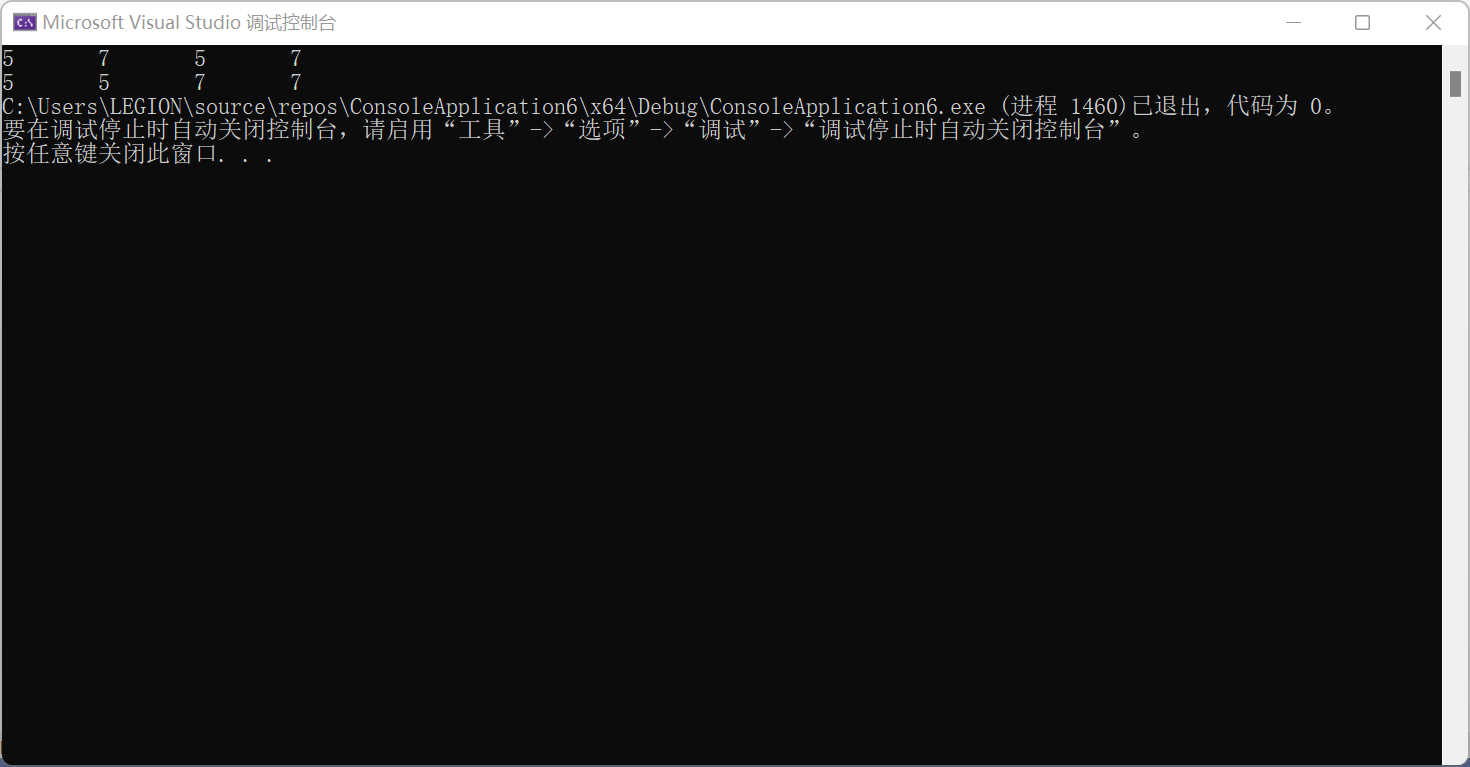
i = 5; j = 7;

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

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

}

**Result：**



**（2）**

**Code：**

#include <iostream>

using namespace std;

int main()

{

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

int\* p, i;

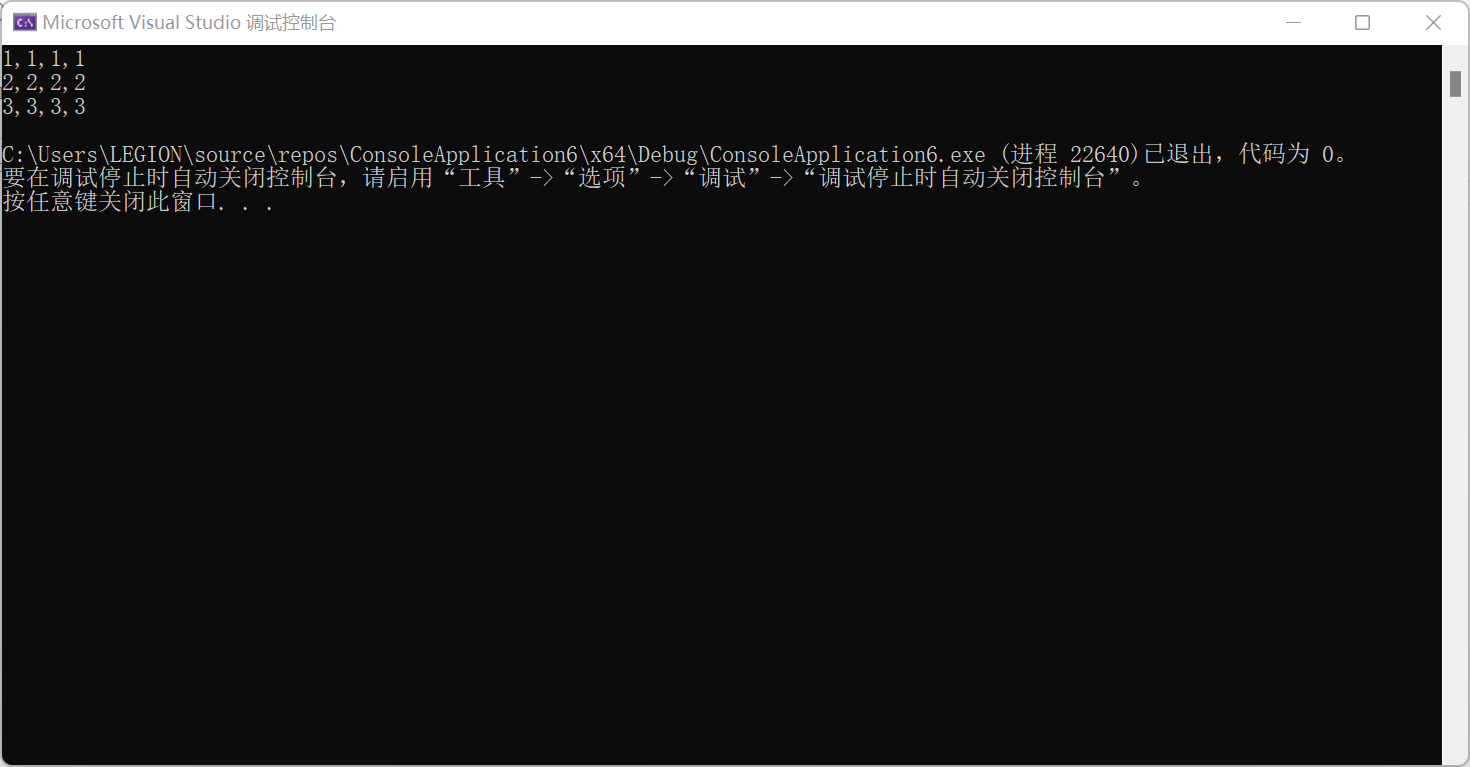
p = a;

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

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

}

**Result:**



**(3)**

**Code：**

#include <iostream>

using namespace std;

void f(char\* st, int i)

{

st[i] = '\0';

cout << st; // printf(“%s\n”,st);

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

}

void main()

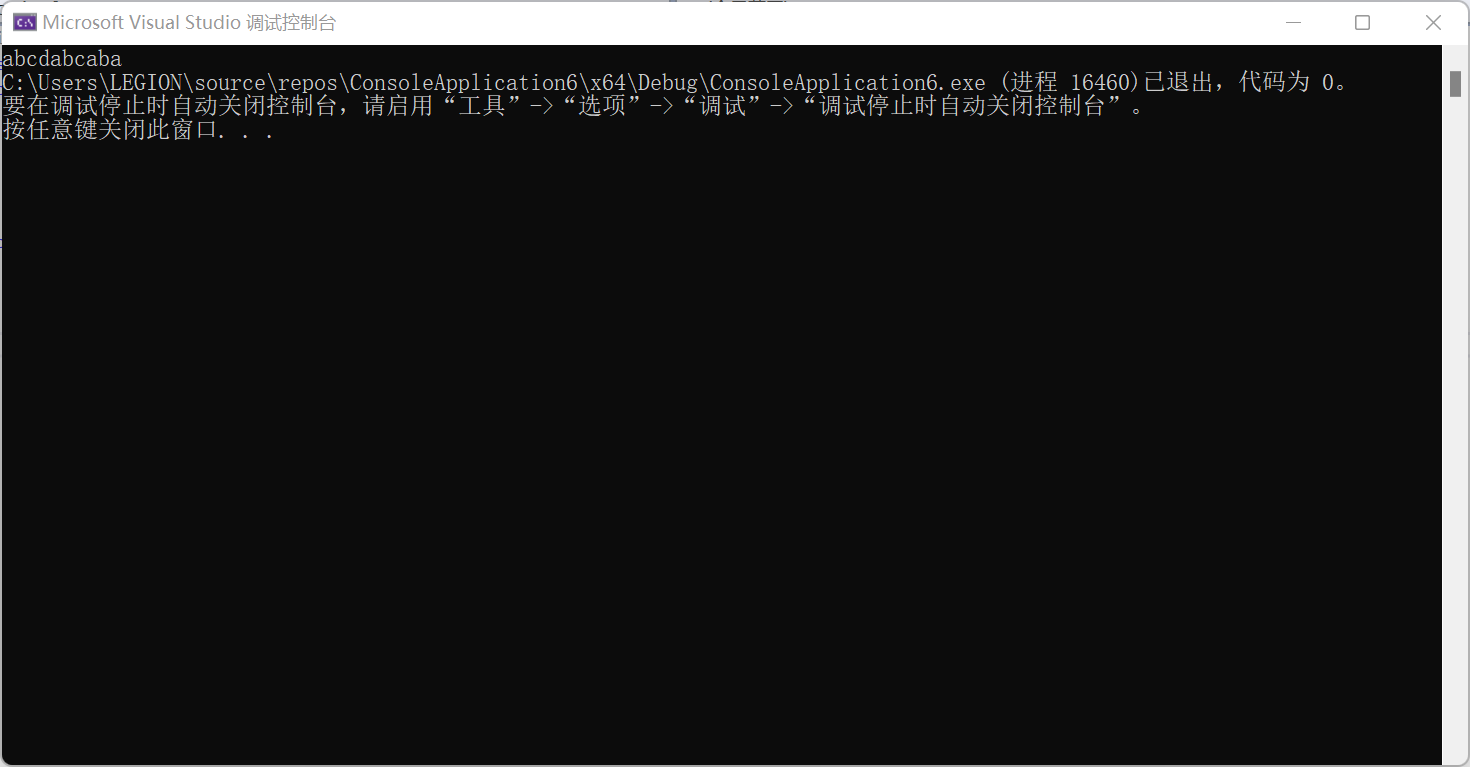
{

char st[] = "abcd";

f(st, 4);

}

**Result:**



**(4)**

**Code：**

#include<iostream>

using namespace std;

int\* f()

{

int\* list = new int[4];

list[0] = 1;

list[1] = 2;

list[2] = 3;

list[3] = 4;

return list;

}

void main()

{

int\* p = f();

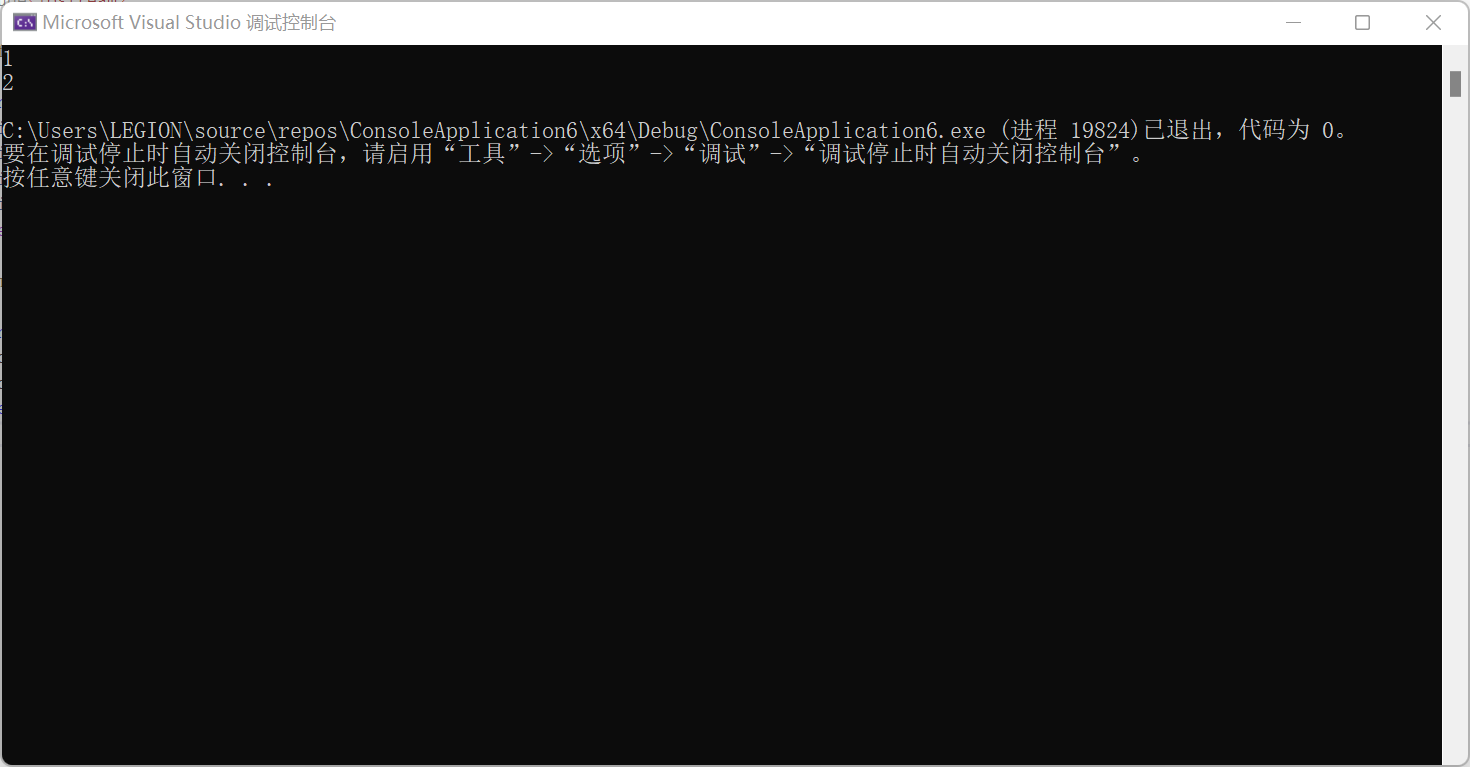
cout << p[0] << endl;

cout << p[1] << endl;

delete[]p;

}

**Result:**



**2.**

**（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。

**Code：**

#include <iostream>

using namespace std;

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

{

int a, b, m = 0, location, i, j;

a = strlen(s1);

b = strlen(s2);

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

{

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

{

if (s1[j] == s2[j + i] && (j == a - 1))

{

return i;

}

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

{

break;

}

}

}

return -1;

}

int main()

{

char s1[100], s2[100];

cout << "请输入第一行字符串：" << endl;

cin.getline(s1, 100);

cout << "请输入第二行字符串：" << endl;

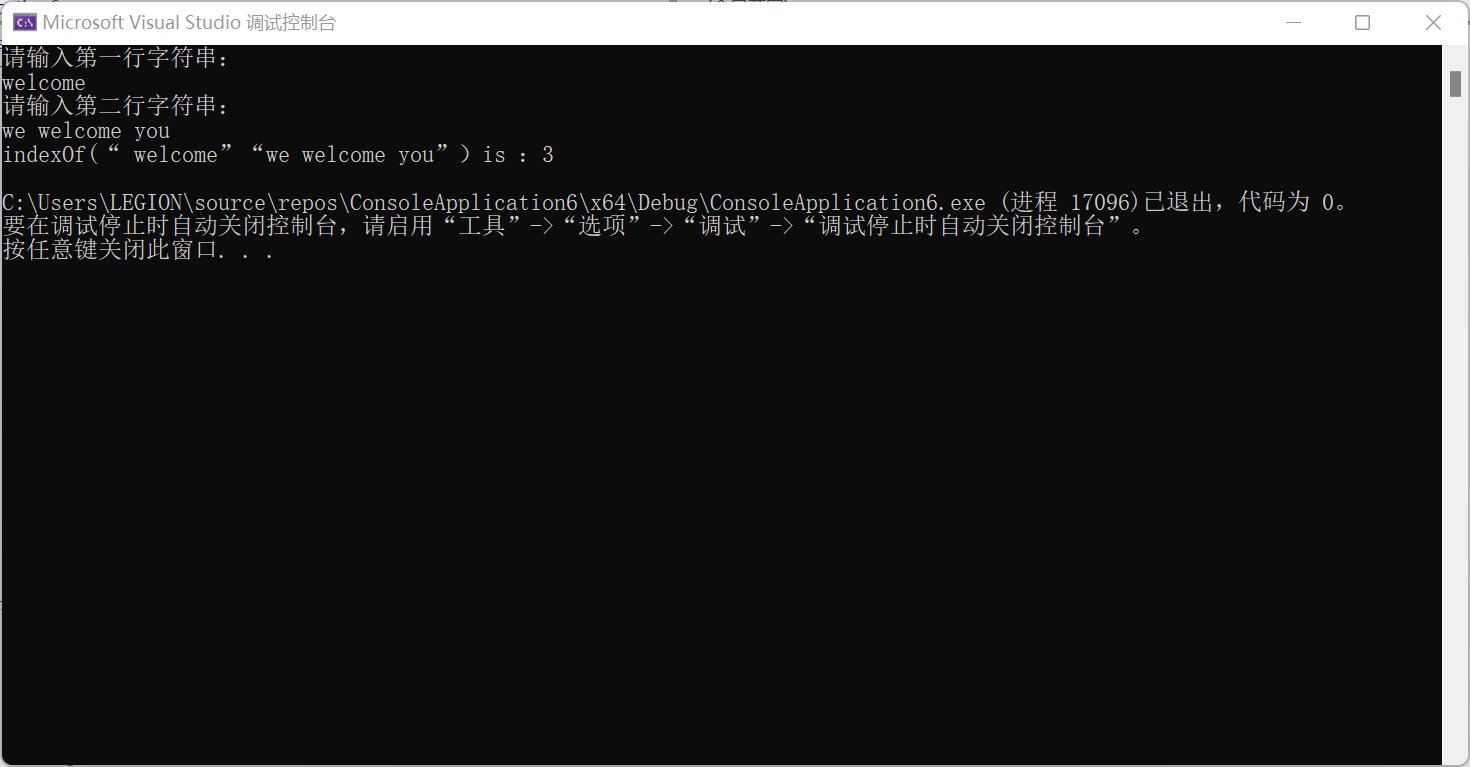
cin.getline(s2, 100);

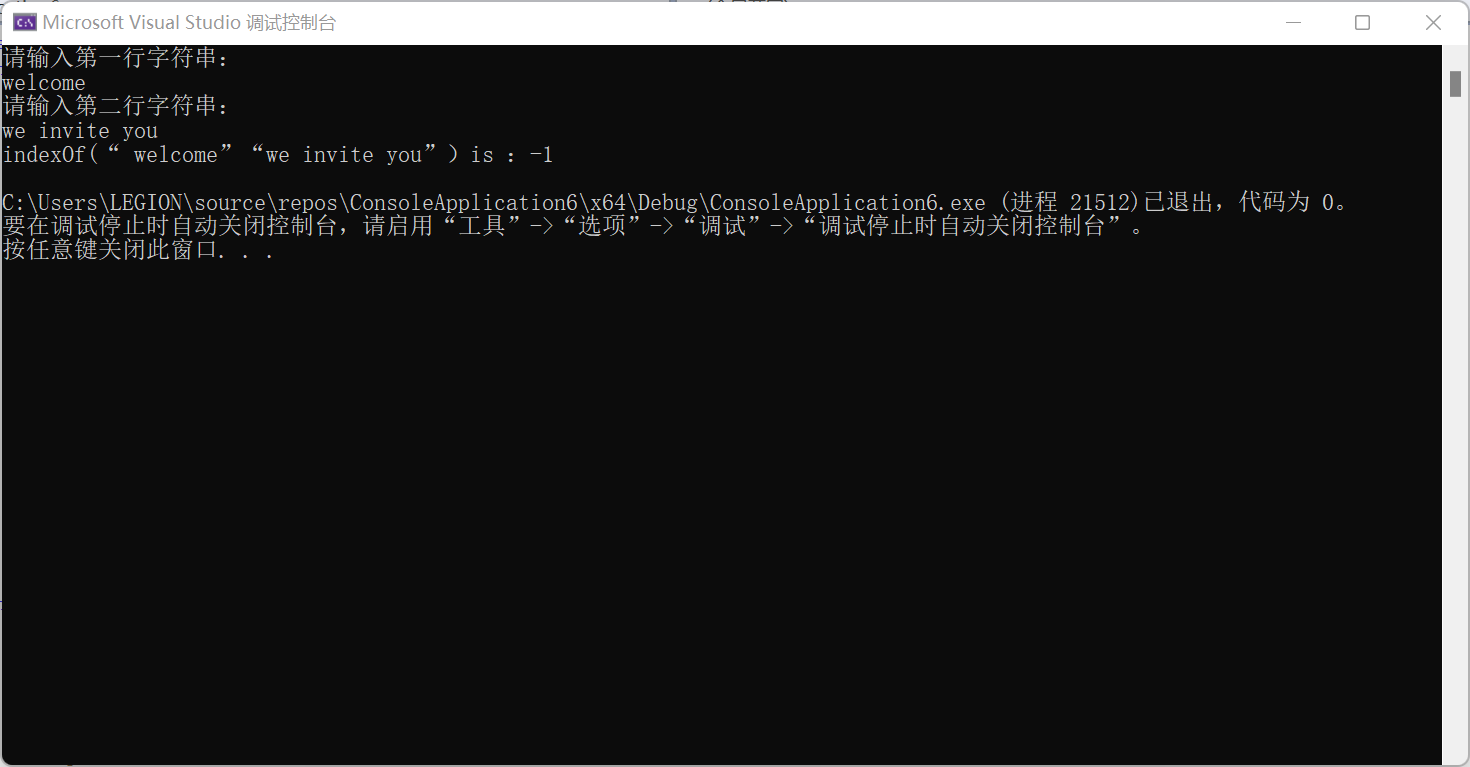
cout << "indexOf(“ " << s1 << "”" << "“" << s2 << "”）is ：" << indexOf(s1, s2) << endl;

return 0;

}

**Result:**





**（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);

**Code：**

#include<iostream>

using namespace std;

int parseHex(const char\* const hexString)

{

int a, sum = 0;

a = strlen(hexString);

int\* list = new int[a];

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

{

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

list[i] = (static\_cast<int>(hexString[i]) - 55) \* (pow(16, a - 1 - i));

else

list[i] = (hexString[i] - 48) \* pow(16, a - i - 1);

sum = sum + list[i];

}

return sum;

}

int main()

{

int size = 100;

char s[100];

cout << "请输入一个十六进制数字： \n";

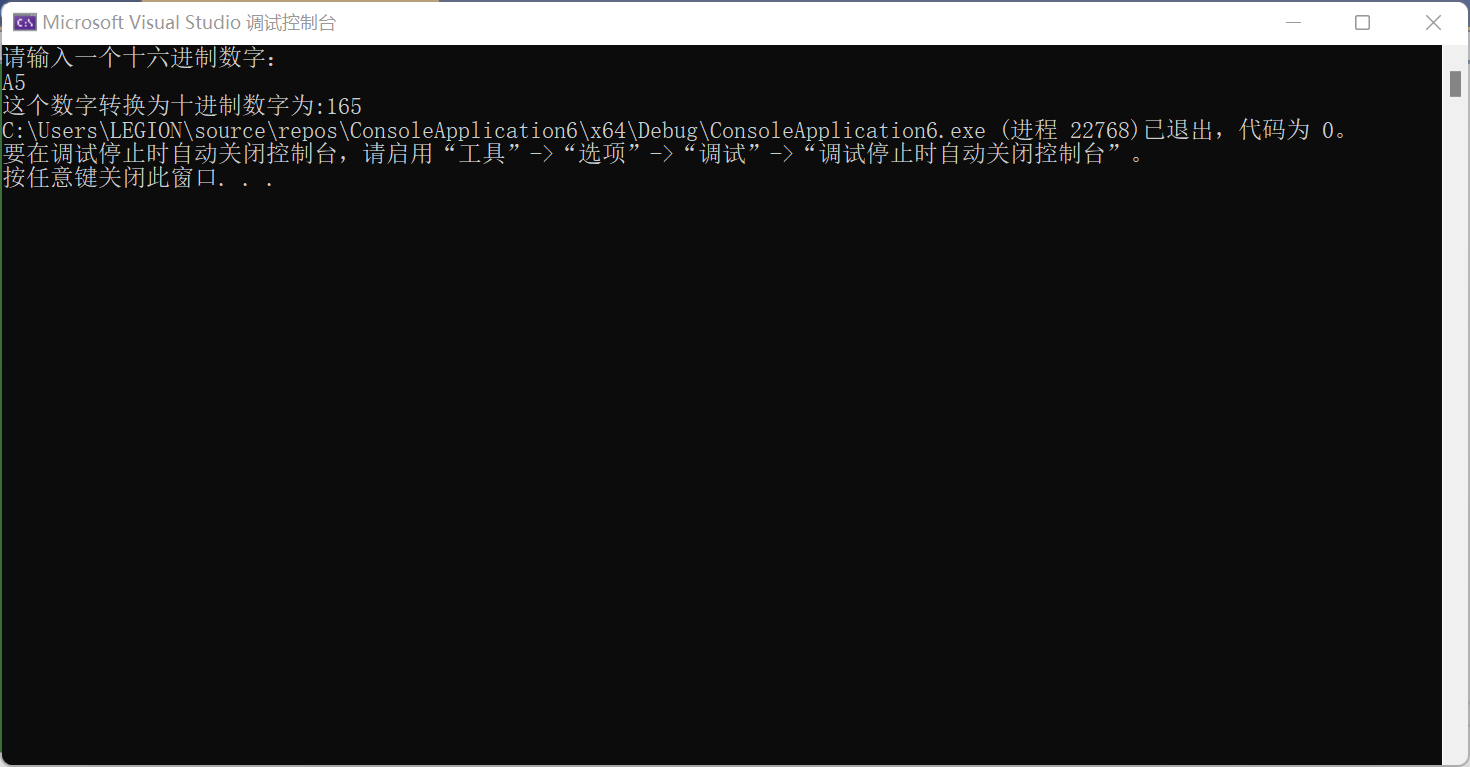
cin.getline(s, size);

cout << "这个数字转换为十进制数字为:" << parseHex(s);

return 0;

}

**Result:**



**（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）。

**Code：**

#include <iostream>

using namespace std;

void bubblesort(double a[], int size)

{

int i, j; double t;

for (i = 0; i < size - 1; i++)

{

for (j = i + 1; j < size; j++)

{

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

{

t = a[j]; a[j] = a[i]; a[i] = t;

}

}

}

}

int main()

{

int n;

cout << "数组元素个数：";

cin >> n;

double\* p = new double[n];

cout << "输入数组元素：";

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

cin >> p[i];

bubblesort(p, n);

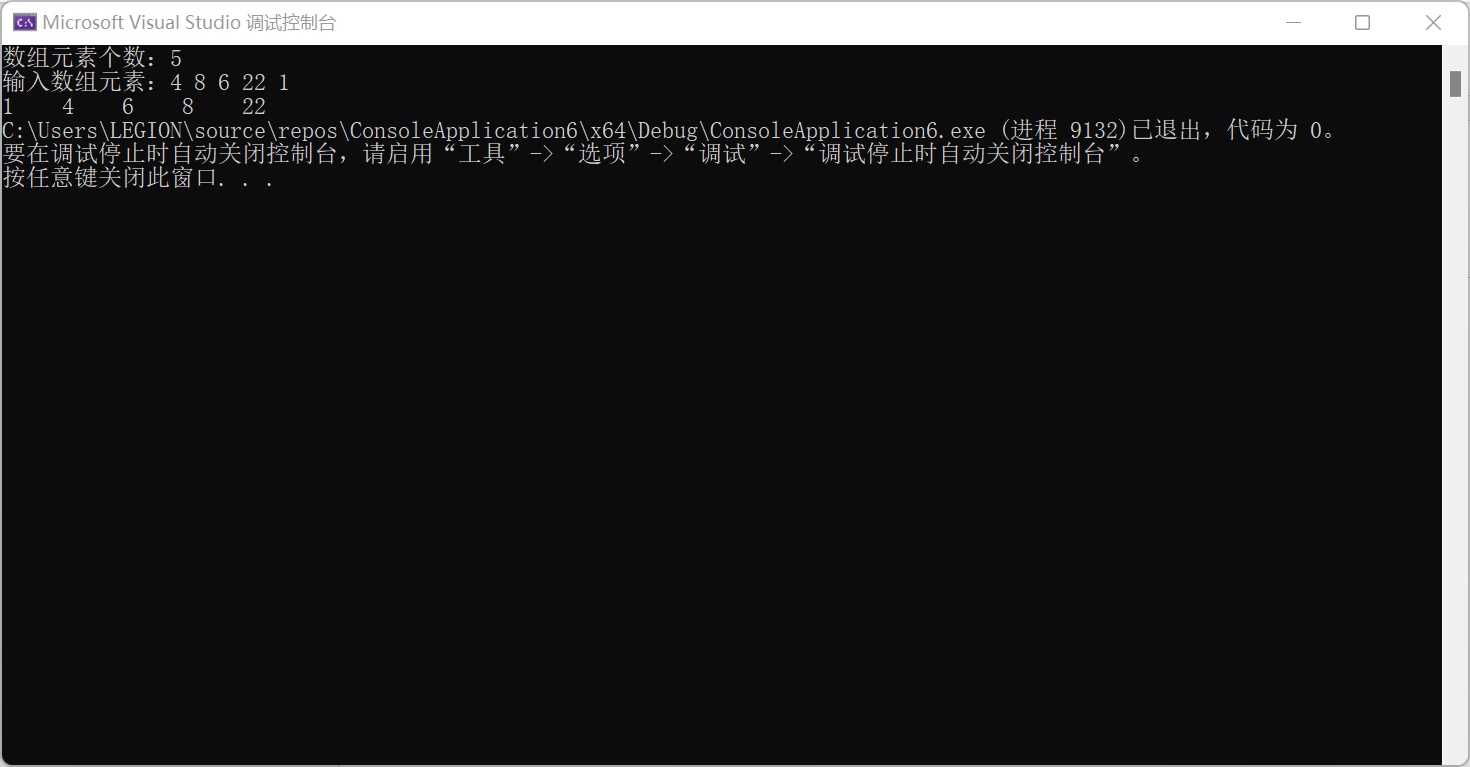
for (int i = 0; i < n; i++)

cout << p[i] << " ";

delete[]p;

}

**Result:**



1. **遇到的问题和解决方法**

数组第五道实验题我自己敲的代码bug太多太多太多了，多到debug好久后直接把代码放弃了，然后上CSDN看了看以及看了看同学的代码才敲出来。第六题一开始忘记限制条件使！也算进字母里了。指针第二题我对十六进制一点都不知道，这道题基本一点都想不出，基本上是看着CSDN里面给的代码敲了一遍。

1. **体会**

好难。