**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\_\_\_\_\_\_\_\_\_\_\_8209220605\_\_\_\_\_\_\_\_\_

姓 名\_\_\_\_\_\_\_\_\_\_\_\_王一丁\_\_\_\_\_\_\_\_\_\_\_\_\_

**实验报告成绩：**

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

**批阅教师：**

**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 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)

{

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]<<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);

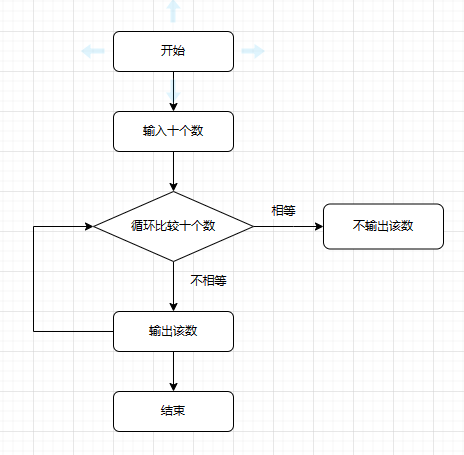
如：调用函数 parseHex(“A5”);返回 165

(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、算法分析、程序与结果**

**（一）数组**

1. **算法：**



**程序：**

#include<iostream>

using namespace std;

int main()

{

const int size = 10;

int list[size];

cout << "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;

}

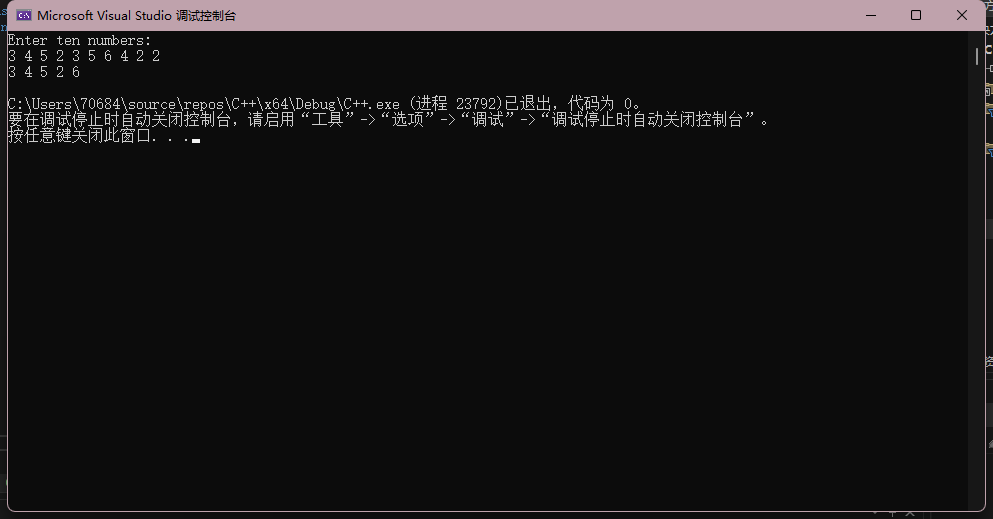
}

cout << endl;

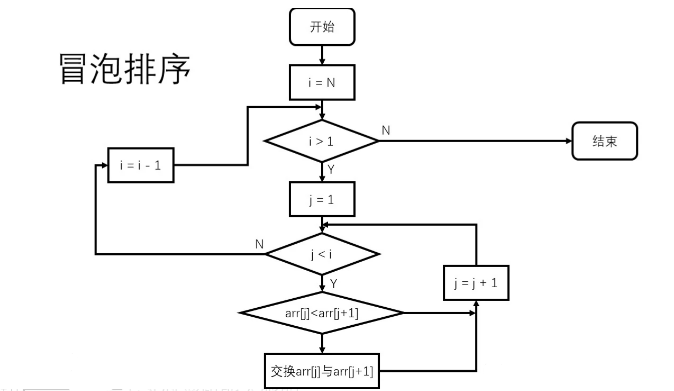
return 0;

}

**结果：**



1. **算法：**



**程序：**

#include<stdio.h>

int main(void) {

int i, j, t;

int a[10];

printf("Please enter ten numbers:\n");

//从键盘接收需要排序的数并存入数组

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

scanf\_s("%d", &a[i]);

//起泡法排序

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

for (i = 0; i < 9 - j; i++) {

if (a[i] > a[i + 1]) {

t = a[i];

a[i] = a[i + 1];

a[i + 1] = t;

}

}

}

//输出排好序的数组

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

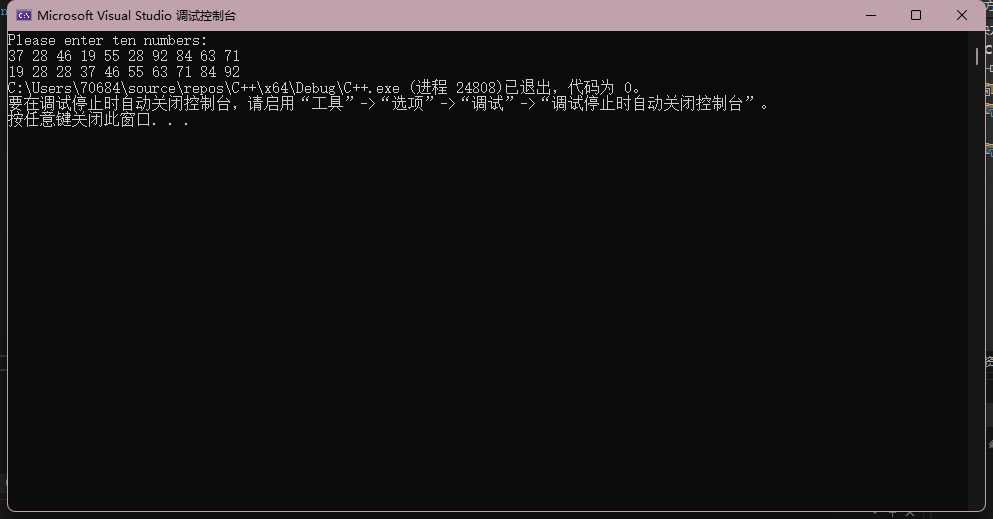
printf("%d ", a[i]);

}

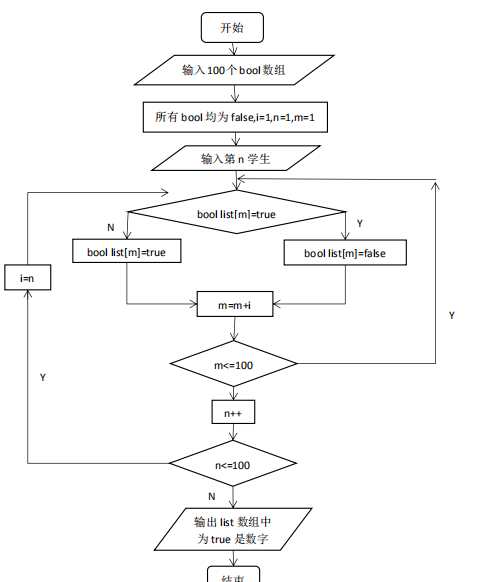
return 0;

}

**结果：**



1. **算法：**



**程序：**

#include<iostream>

using namespace std;

int main()

{

const int size = 100;

bool list[size];

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

{

list[i] = false;

}

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

{

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

{

if (list[k])

list[k] = false;

else

list[k] = true;

}

}

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

{

if (list[m])

cout << m + 1 << " ";

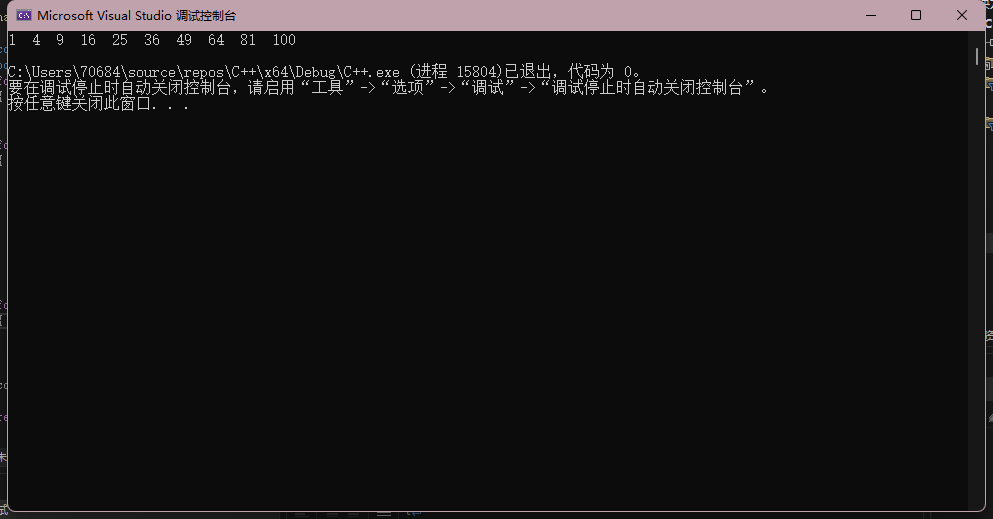
}

cout << endl;

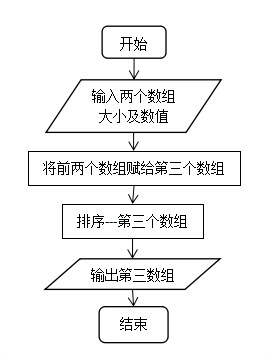
return 0;

}

**结果：**



1. **算法：**



**程序：**

#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 << "第一个数组的大小：";

cin >> n;

cout << "第二个数组的大小：";

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++)

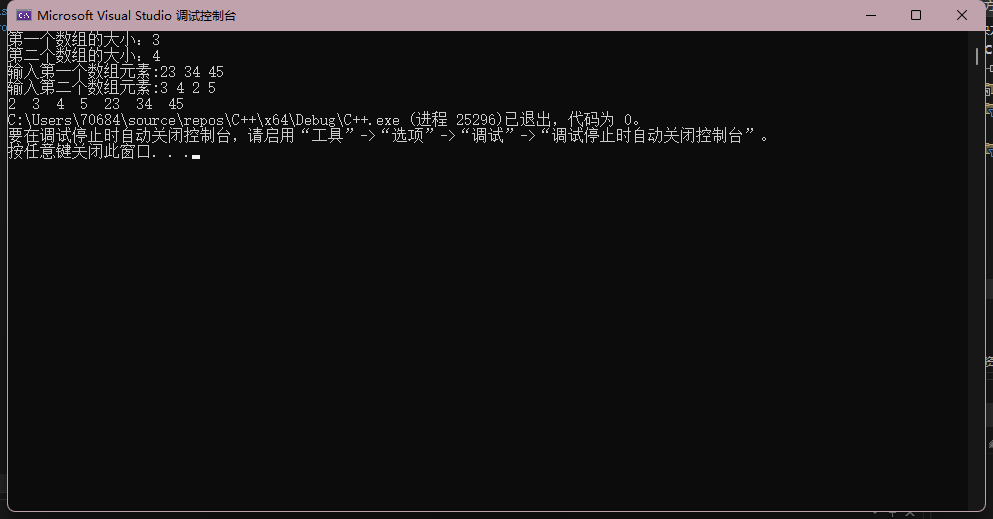
cin >> q[j];

merge(p, n, q, m);

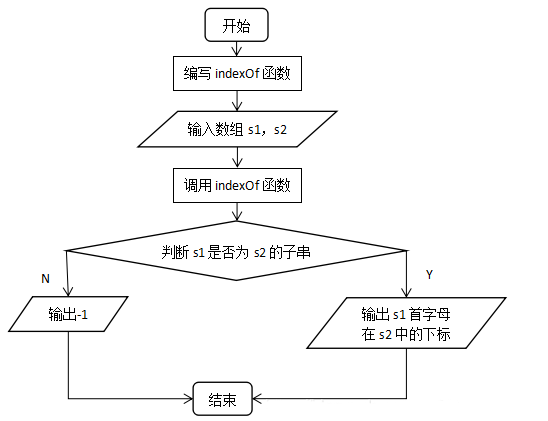
return 0;

}

**结果：**



1. **算法：**



**程序：**

#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;

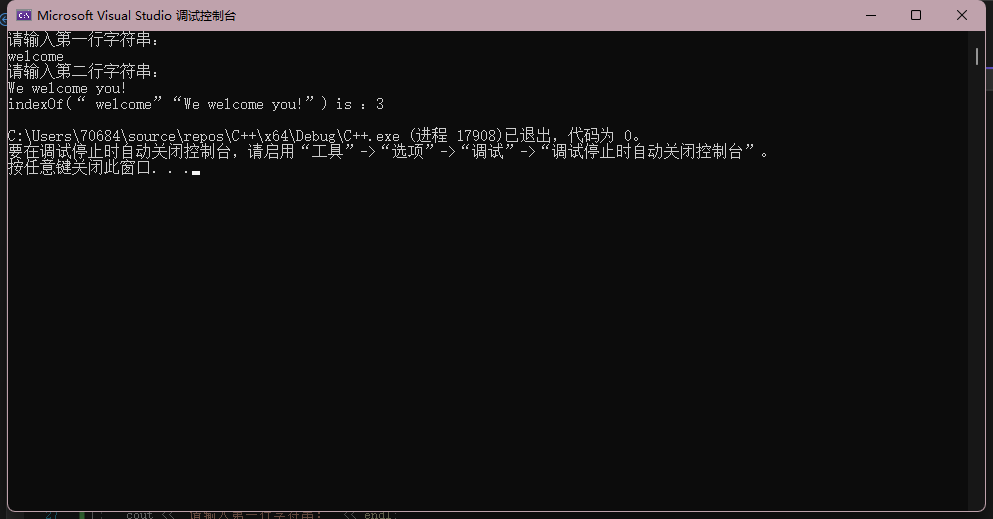
cin.getline(s2, 100);

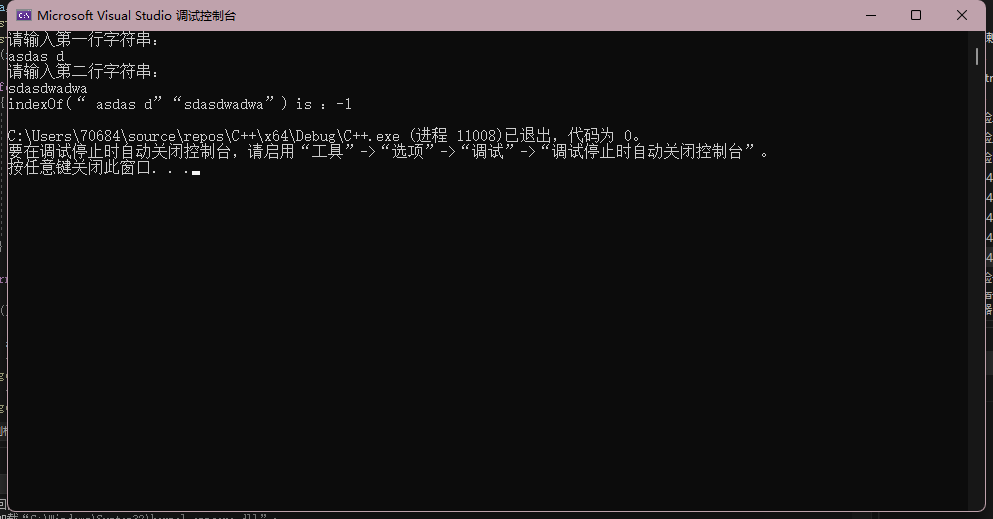
cout << "indexOf(“ " << s1 << "”" << "“" << s2 << "”）is ：" << indexOf(s1, s2) << endl;

return 0;

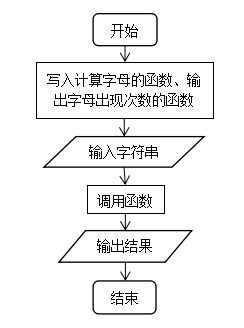
}

**结果：**





1. **算法：**



**程序：**

#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);

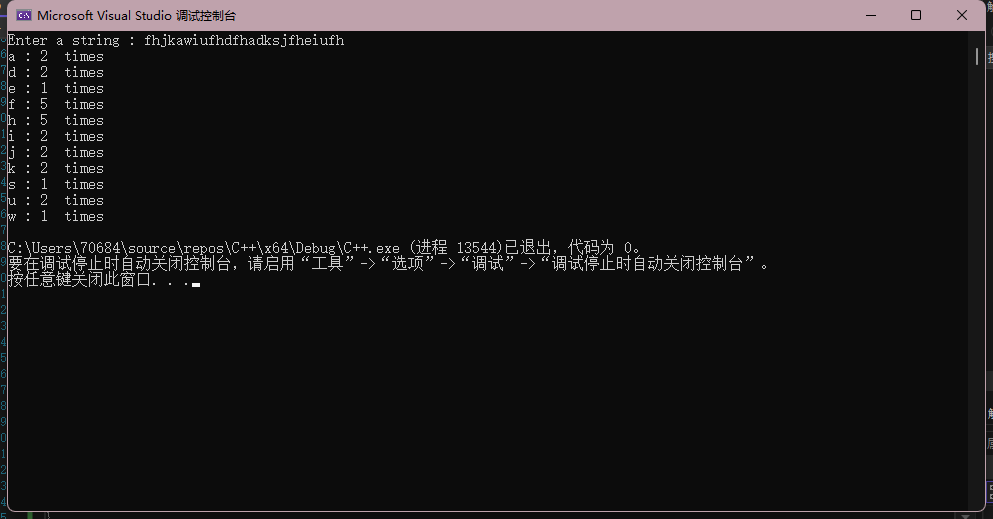
countLetters(list, counts);

displayCounts(counts);

return 0;

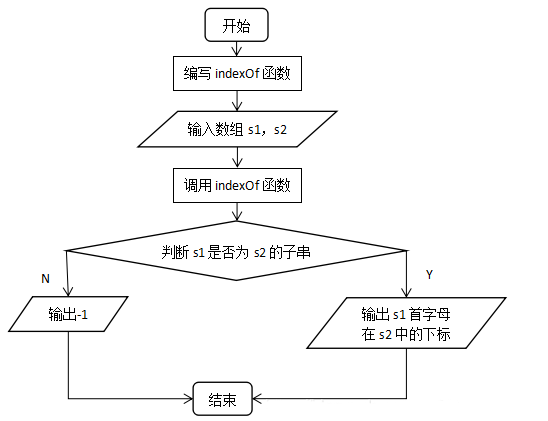
}

**结果：**



**（指针）**

1. **算法：**



**程序：**

#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;

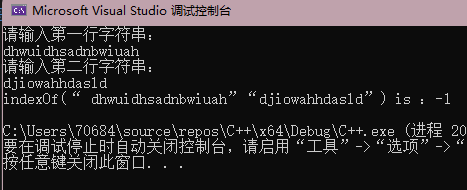
cin.getline(s2, 100);

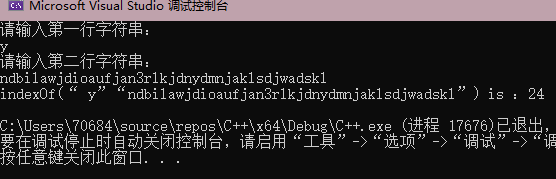
cout << "indexOf(“ " << s1 << "”" << "“" << s2 << "”）is ：" << indexOf(s1, s2) << endl;

return 0;

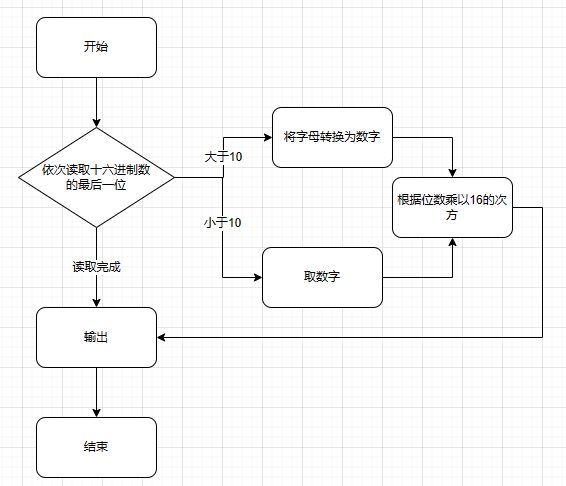
}

**结果：**





1. **算法：**



**程序：**

#include <stdio.h>

//将十六进制的字符串转换成整数

int htoi(char\* str)

{

int n = 0;

if (str == NULL) return -1;

if (\*str == '0' && (\*(str + 1) == 'x' || \*(str + 1) == 'X'))

{

str += 2;

}

while (1)

{

if (\*str >= '0' && \*str <= '9')

{

n = 16 \* n + (\*str - '0');

}

else if (\*str >= 'A' && \*str <= 'F')//十六进制还要判断字符是不是在A-F或者a-f之间

{

n = 16 \* n + (\*str - 'A' + 10);

}

else if (\*str >= 'a' && \*str <= 'f')

{

n = 16 \* n + (\*str - 'a' + 10);

}

else

{

break;

}

str++;

}

return n;

}

int main(void)

{

char str1[] = "0XAAAAB";

int data = 0;

data = htoi(str1);

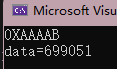
printf("%s\r\n", str1);

printf("data=%d\r\n", data);

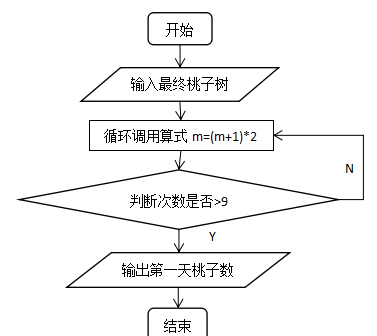
return 0;

}

**结果：**



1. **算法：**



**程序：**

#include<iostream>

using namespace std;

int main()

{

int m = 1;

for (int i = 1; i < 10; i++)

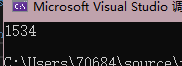
m = (m + 1) \* 2;

cout << m << endl;

return 0;

}

**结果：**



**四、遇到的问题与解决方法**

**这一节实验主要是数组的使用，需要用到输入的值作为数组的大小。我经常把一个变量作为数组的下标，后来查资料才发现正确的方法是使用动态数组。还有三目运算符的问题，三元运算符？：的优先级太低了，比/，+，<这些运算符的优先级还要低，导致我犯了好几个错误。**

**五、体会**

**上课要跟着老师敲代码，课后把代码完善，从上到下读一遍，才能把学到的东西化为己用，如果讲的东西我都掌握了，就不会犯这么多错了。**