**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软件工程2402

学 号： 8209240218

姓 名： 郑文俊

**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）

#include <iostream>

using namespace std;

int A(int m1, int n1)//定义最大公因数函数

{

int i = 1;

int temp;//定义临时变量

for (; i <= m1 && i <= n1; i++)//循环

{

if (m1% i== 0&&n1 % i== 0)//求公因数，至最大

temp = i;//将i赋值给temp

}

return temp;//返回临时变量

}

int B(int m2, int n2)//定义最小公倍数函数

{

int temp;//定义临时变量

for (int i=0; i <= (m2 \* n2); i++)//循环

{

if (i % m2 == 0 && i % n2==0)//若i能整除二值，则其为公倍数

temp = i;//将i赋值给temp

}

return temp;//返回临时变量

}

int main()

{

int m ;

int n ;

cout << "输入第一个数" << endl;//输入第一个数

cin >> m;

cout << "输入第二个数" << endl;//输入第二个数

cin >> n;

int A1 = A(m, n);//调用最大公因数函数

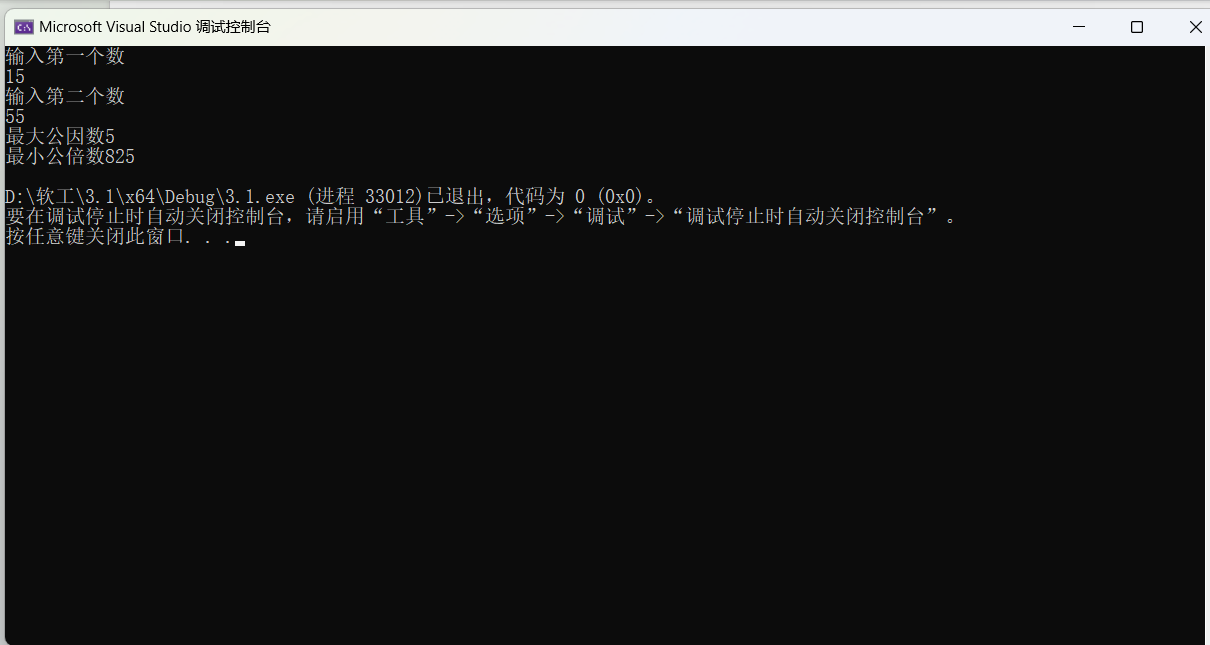
int B1 = B(m, n);//调用最小公倍数函数

cout << "最大公因数" << A1 << endl;

cout << "最小公倍数" << B1 << endl;

return 0;

}

1. 

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

#include <iostream>

using namespace std;

bool is\_prime(int num)//定义素数函数

{

if (num == 1)//当num为一时，不是素数

{

return false;//返回false

}

for (int i = 2; i \* i <= num; i++) //若num是出现过数的倍数，则不是素数

{

if (num % i == 0)//判定是否有其他因数

{

return false;//返回false

}

}

return true; //返回true

}

int main()

{

int a = 0;

for (int i = 2; a<=200; i++) //循环，直到找出前200个素数

{

if (is\_prime(i))//调用素数函数

{

cout << i<<"\t";//输出

a++;

if (a % 10 == 0)//如果一行超过十个，换行

{

cout << endl;

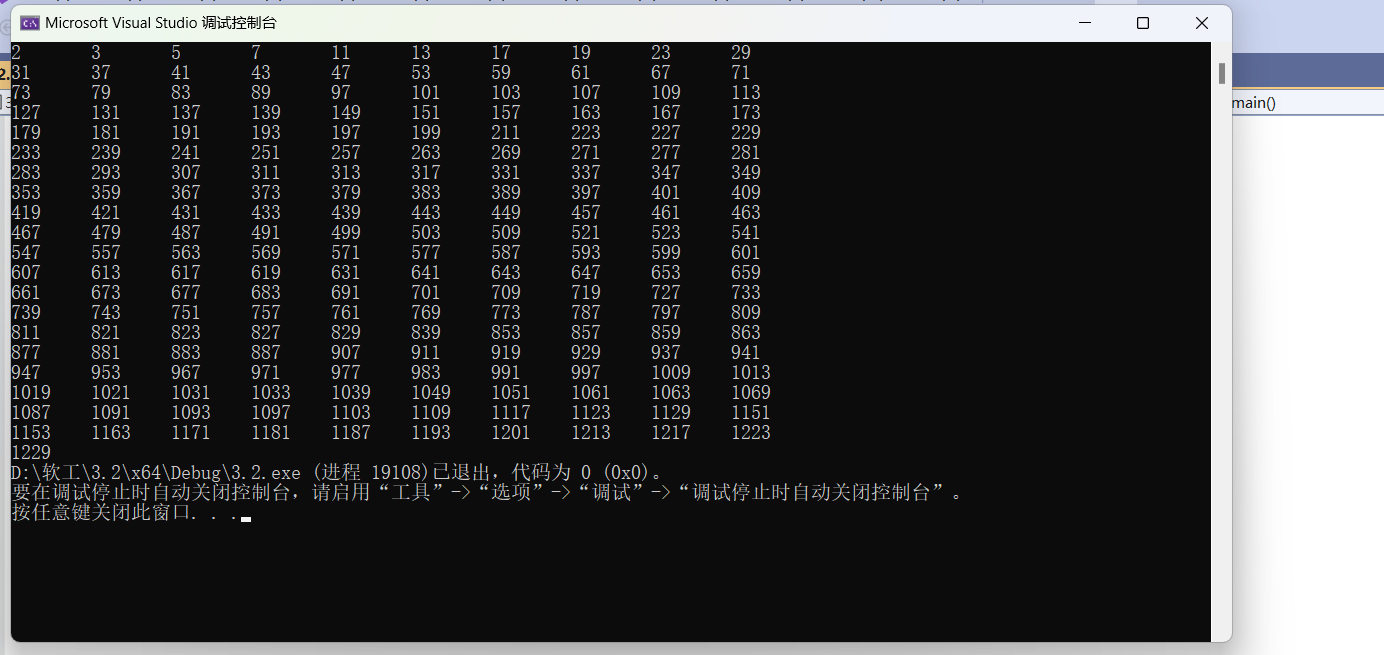
}

}

}

return 0;

}



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）

mytriangle.h

#pragma once

#include<iostream>

using namespace std;

bool is\_valid(double side1, double side2, double side3);//定义判断三边是否构成三角形函数

double area(double side1, double side2, double side3);//定义三角形面积函数

mytriangle.cpp

#include"mytriangle.h"

bool is\_valid(double side1, double side2, double side3)//判断三边是否构成三角形

{

if (side1 + side2 > side3 && side3 + side2 > side1 && side1 + side3 > side2)//如果两边之和大于第三边，则三角形成立

{

return true;//返回true

}

else

{

return false;//返回false

}

}

double area(double side1, double side2, double side3)//计算三角形面积

{

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

double square = sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));

return square;//返回square

}

3.4

#include"mytriangle.h"

int main()

{

int side1, side2, side3;//定义三边长

cout << "三边边长为" << endl;//提醒输入三边长

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

if (is\_valid(side1, side2, side3))//判断三边是否构成三角形，调用判断三边是否构成三角形函数

{

double s = area(side1, side2, side3); //调用三角形面积函数

cout << "三角形面积为：" << s << endl;//输出三角形面积

}

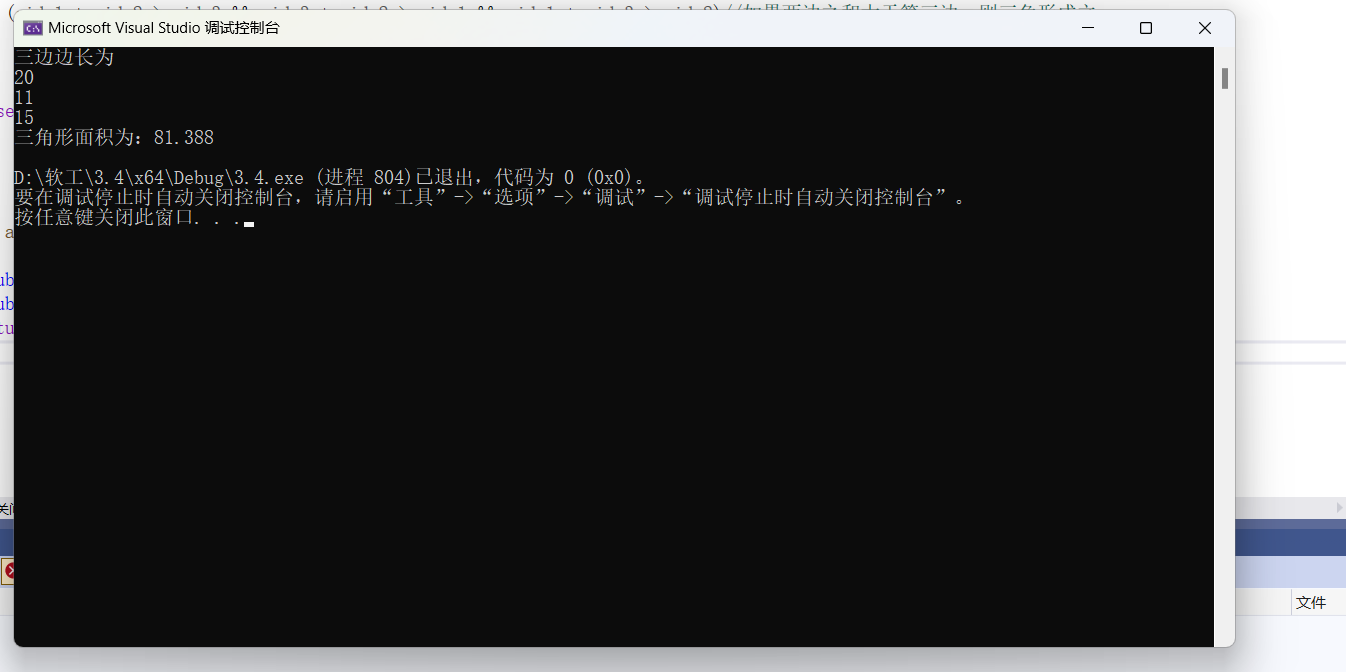
else

{

cout << "无法构成三角形" << endl;//如果判断失败，输出无法构成三角形

}

return 0;

v

}

**3与4选一个完成**

1. 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）。

#include<iostream>

using namespace std;

int peach\_number(int num1, int num2)//定桃子数的函数

{

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

{

return num1;//返回num1

}

}

int main()

{

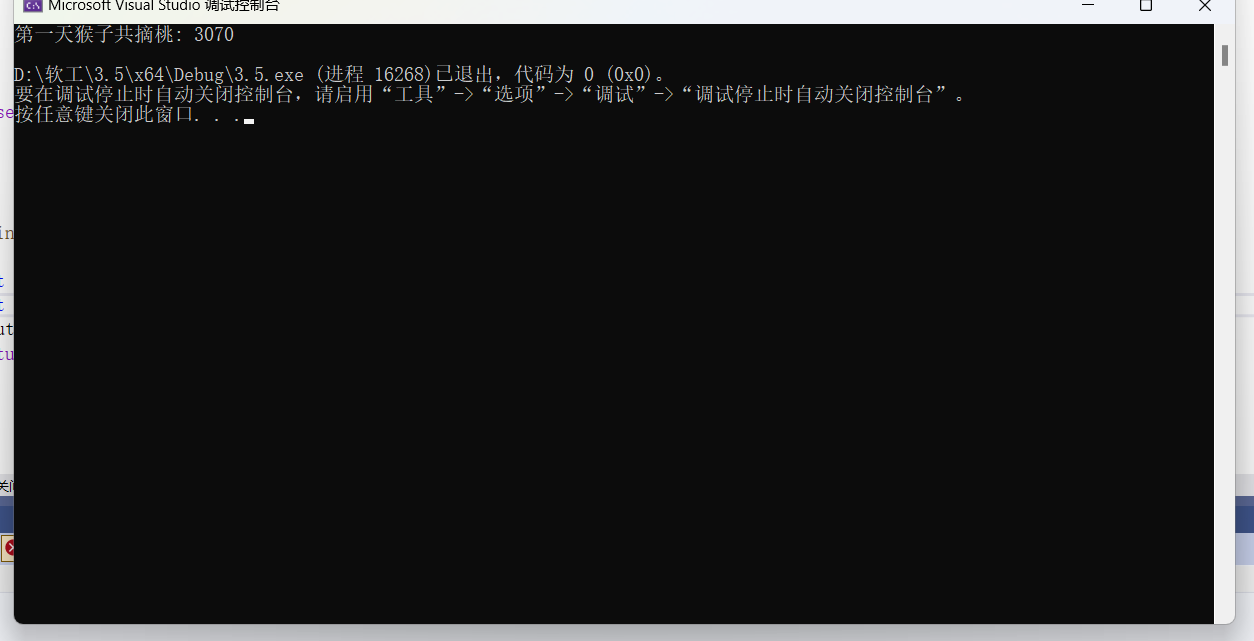
int n = 1;//第十天只剩一个桃子

int num = peach\_number(n, 10);

cout << "第一天猴子共摘桃: " << num << endl;//输出总摘桃数

return 0;

}

1. 

**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？

2. 本实验中主函数调用函数时采用的是何种传递方式？

**四、算法分析，程序结果**

**五、遇到的问题与解决方法**

**六、体会**

**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

#include<iostream>

using namespace std;

int main()

{

int arr[10];//定义数组

int temparr[10];

int num = 0;

int temp = 0;

cout << "请输入10个数" << endl;

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

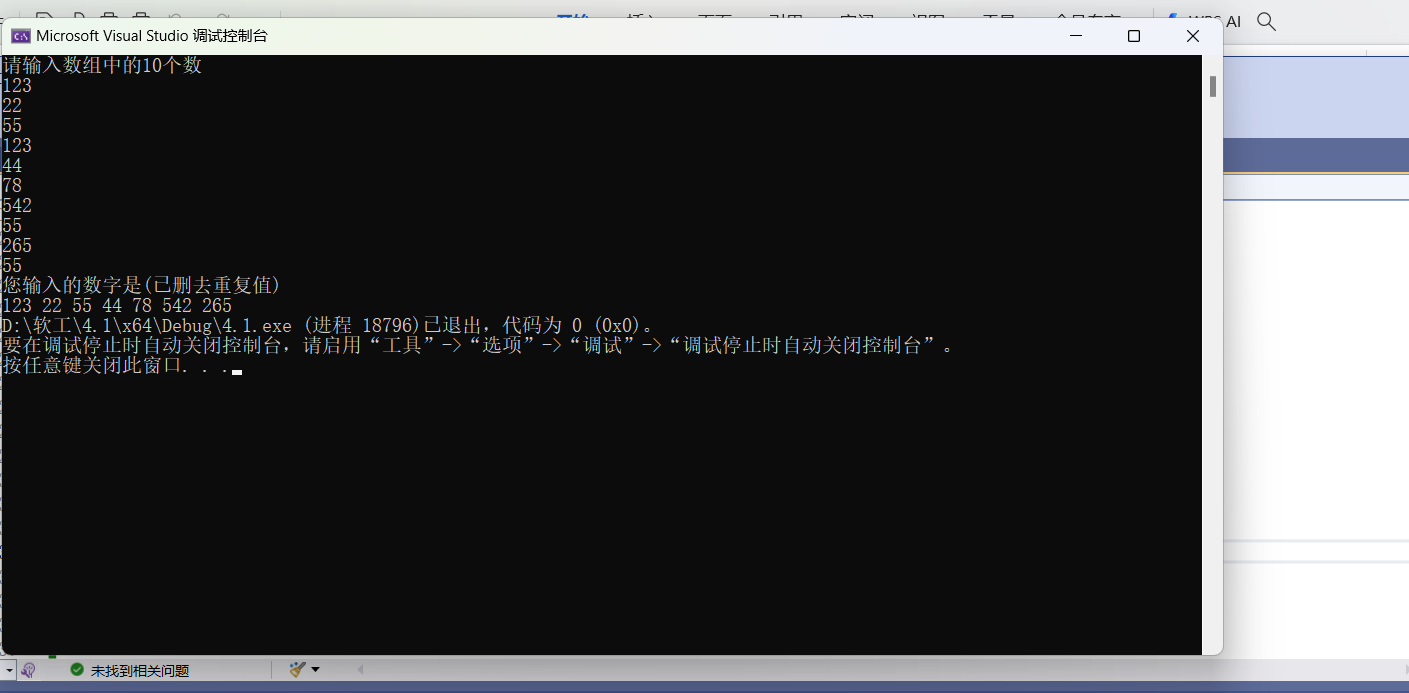
{

cout << arr[i] <<" ";//打印数据

}

return 0;

}



2、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。

#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++)//确定排序

{

for (int j = 0; j < len - 1 - i; j++)//选出比较中的最大值

{

if (arr[j] > arr[j + 1])//当前一项大于后一项时，调换顺序

{

int temp = arr[j];//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;

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);//调用冒泡排序函数

cout << "排序后的数组为" << endl;//提示排序结果

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

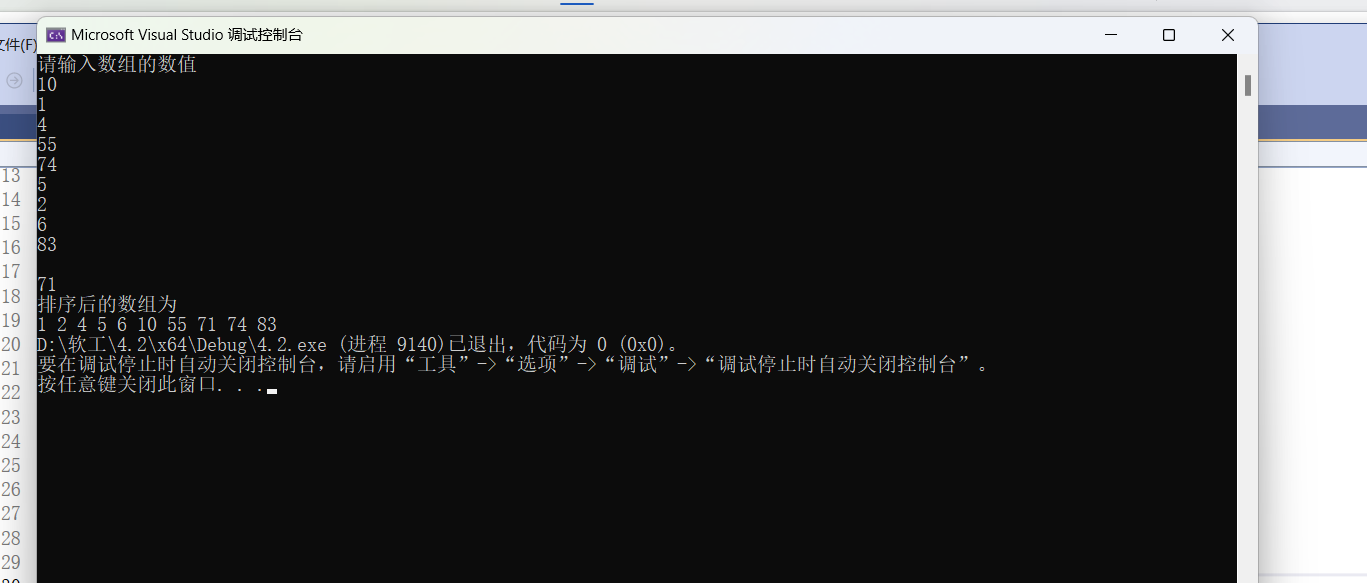
{

cout << arr1[i] << " ";//输出数组

}

return 0;

}



3、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。

#include <iostream>

using namespace std;

bool lockers[100];// 表示存物柜状态的数组

int main()

{

for (int i = 0; i < 100; i++) // 所有存物柜开始为关闭状态

{

lockers[i] = false;

}

for (int i = 0; i < 100; i++)// 如果第一个学生打开所有存物柜

{

lockers[i] = true;

}

for (int student = 2; student <= 100; student++)// 其余学生进行操作

{

for (int locker = student - 1; locker < 100; locker += student)

{

lockers[locker] = !lockers[locker]; // 改变存物柜状态

}

}

for (int i = 0; i < 100; i++)// 输出打开的的号码

{

if (lockers[i])

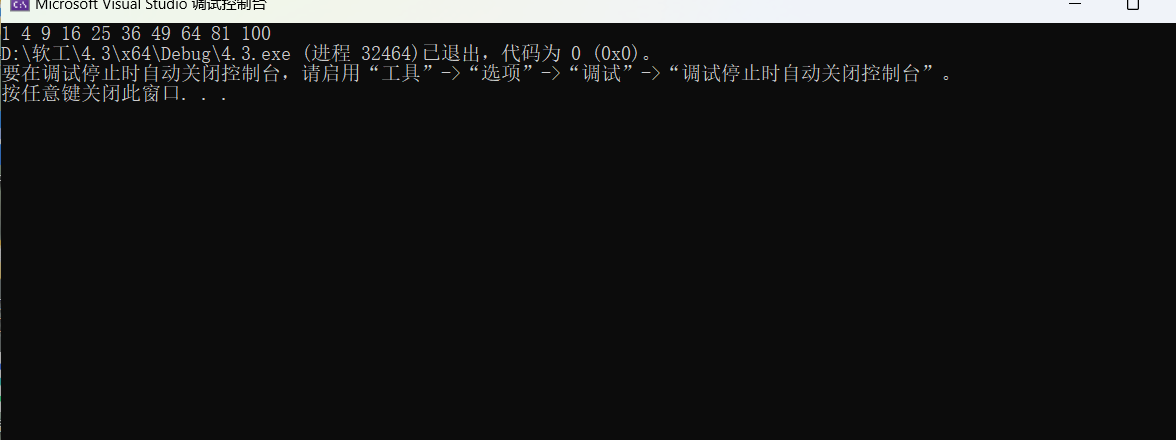
{

cout << i + 1 << " ";

}

}

return 0;

}

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

#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

{

if (i < size1)//如果i<list1 将list1中的数值赋给list3

list3[i] = list1[i];

else//i超出了list1 将list2中的数值赋给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:";

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

{

cout << list3[i] << ' ';

}

}

int main()

{

int size1, size2;

cout << "请输入list1的长度:";

cin >> size1;

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

cout << "请输入list1的数值:";

for (int i = 0; i < size1; i++)//循环

{

cin >> list1[i];

}

cout << "请输入list2的长度:";

cin >> 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:";

for (int i = 0; i < size2; i++)//循环

{

cin >> list2[i];

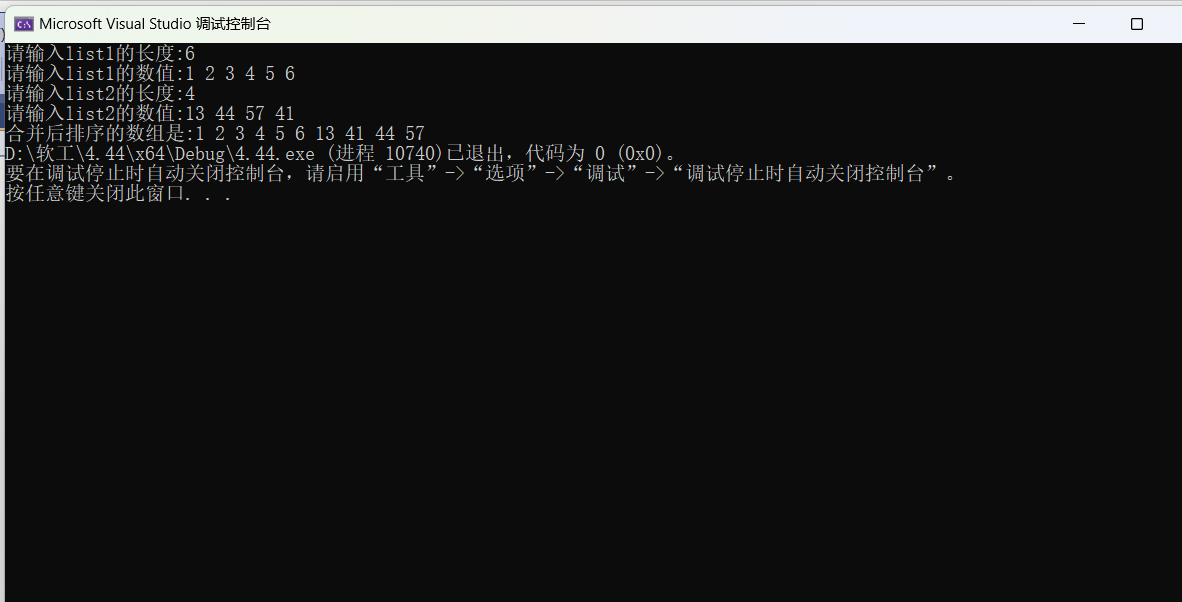
}

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、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

#include<iostream>

using namespace std;

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

{

int len\_s1 = strlen(s1);//求s1长

int len\_s2 = strlen(s2);//求s2长

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 m = 1; m < len\_s1; m++)//循环

{

i++;//i+1

if (s1[m] == s2[i])//若子串下一项和母串下一项相等

{

equal++;//计数+1

}

else

{

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;//下次循环

}

}

}

return -1;//返回-1

}

int main()

{

char s1[100], s2[100];

cout << "Enter the first string::";

cin.getline(s1, 100);

cout << "Enter the second string:";

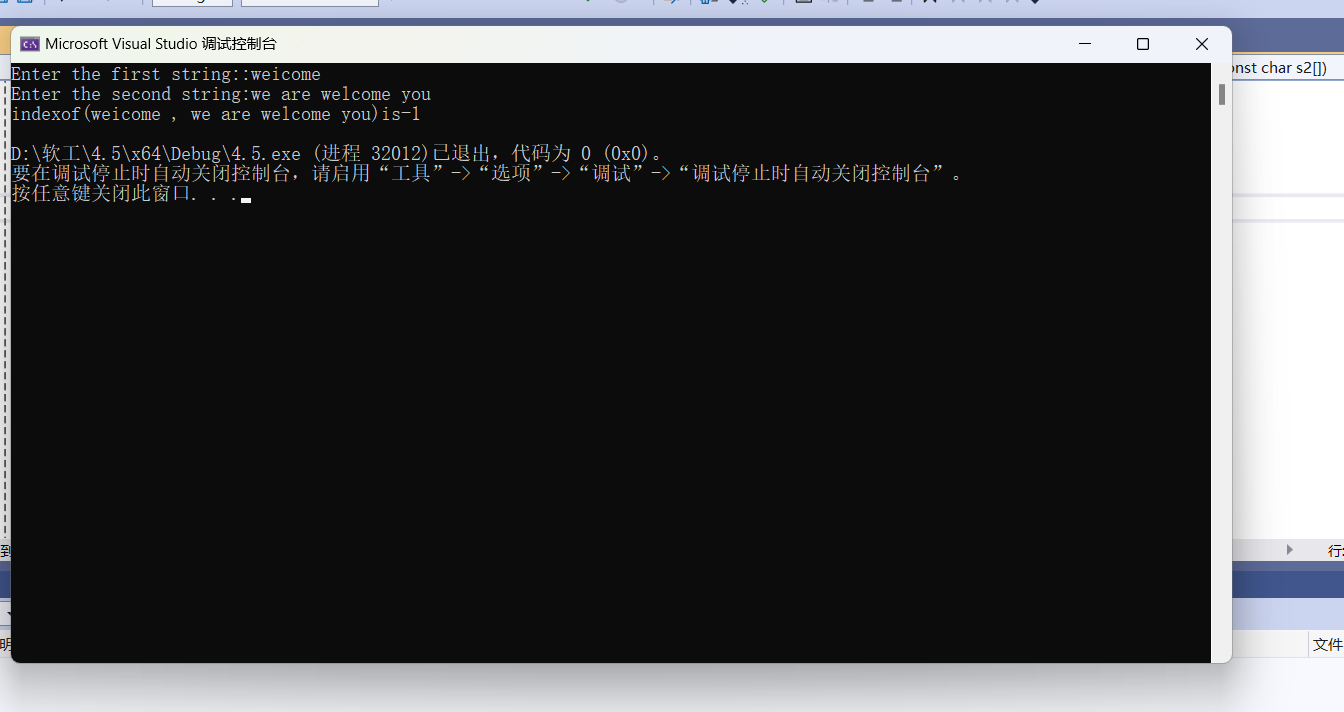
cin.getline(s2, 100);

int indexof = indexOf(s1, s2);//调用函数

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

return 0;

}



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

#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++)//循环

{

int m = int(s[i]) - 65;//m对应的ASCII

if (m >= 32 && m < 58)//m为小写字母

{

counts[m - 32]++;

}

if (m >= 0 && m < 26)//m为大写字母

{

counts[m]++;

}

}

}

int main()

{

char s[100];

cout << "输入:";

cin.getline(s, 100);

int counts[26];//定义整数数组

for (int i = 0; i < 26; i++)//循环

{

counts[i] = 0;//给每个值都赋为0

}

count(s, counts);//调用计数函数

for (int i = 0; i < 26; i++)//建立循环

{

if (counts[i] != 0)//打印

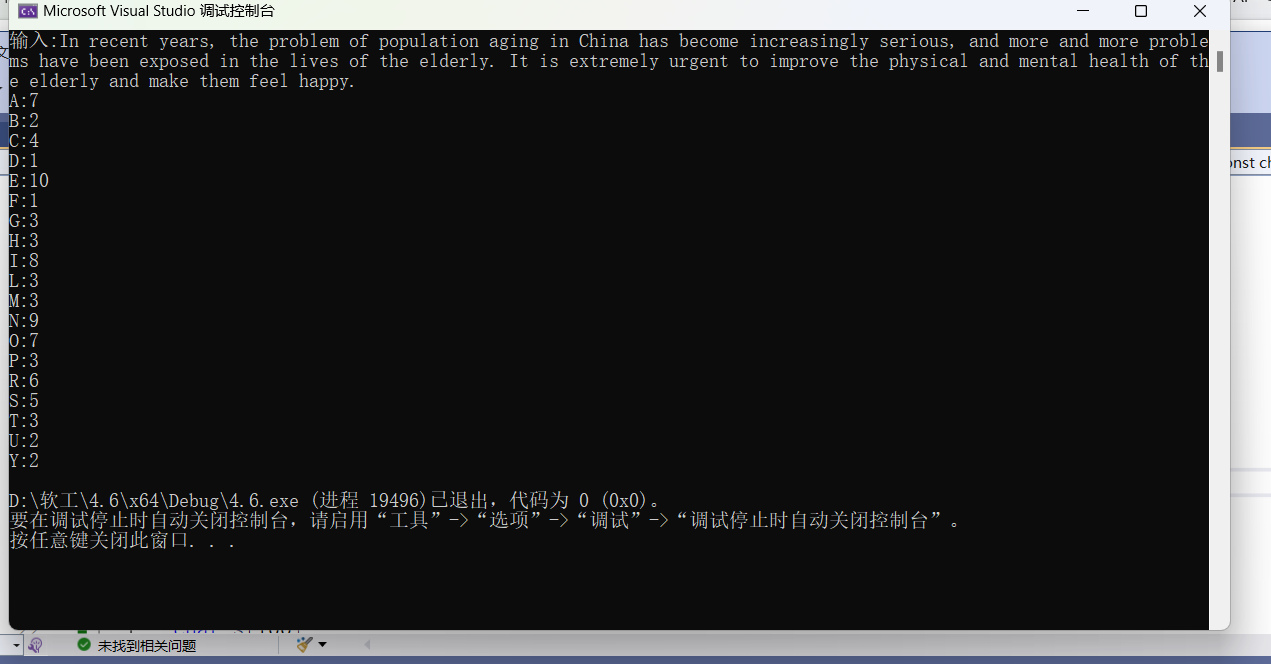
{

cout << char(i + 65) << ":" << counts[i] << endl;//打印字母和各自数量

}

}

return 0;

}

**（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：

#include<iostream>

using namespace std;

void main()

{

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

pi = &i;

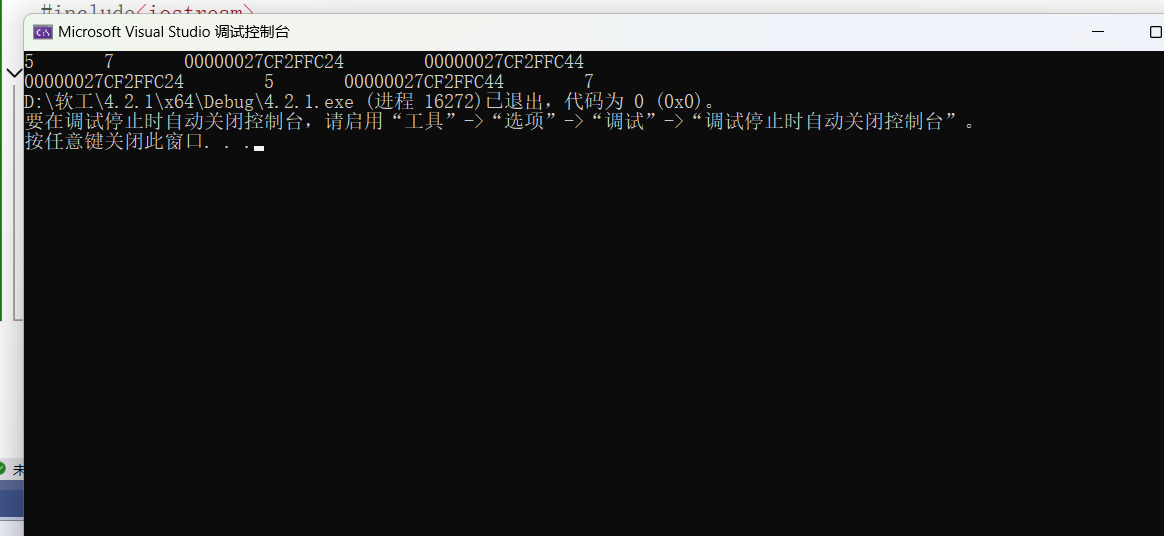
pj = &j;

i = 5; j = 7;

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

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

}



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]。

#include<iostream>

using namespace std;

int main() //C语言程序，要了解

{

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

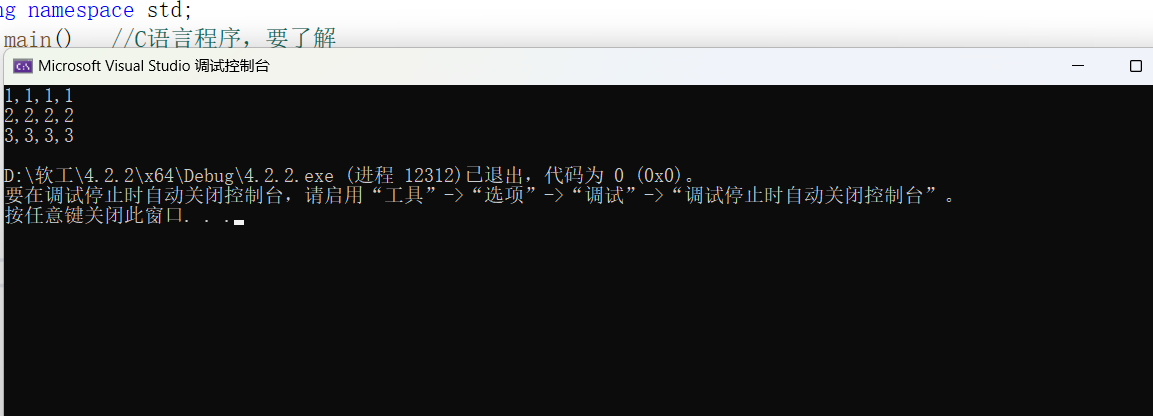
int\* p, i;

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)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\_\_\_\_\_\_\_\_

#include<iostream>

using namespace std;

void f(char\* st, int i)

{

st[i] = '\0';

cout << st <<endl;

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

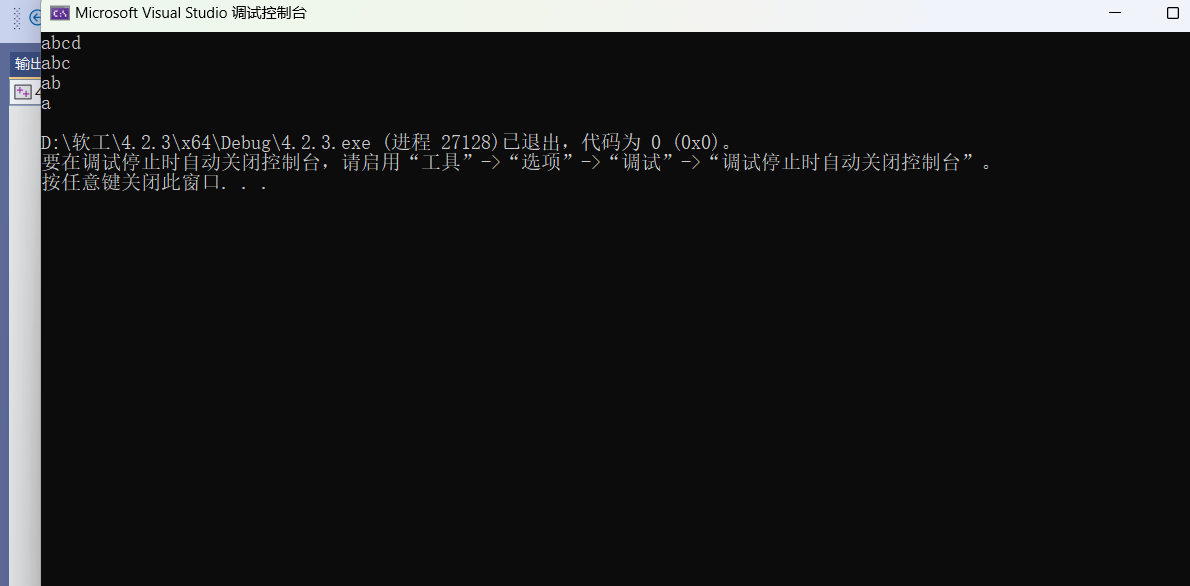
}

int main() {

char st[] = "abcd";

f(st, 4);

return 0;

}

(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;

}

#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()

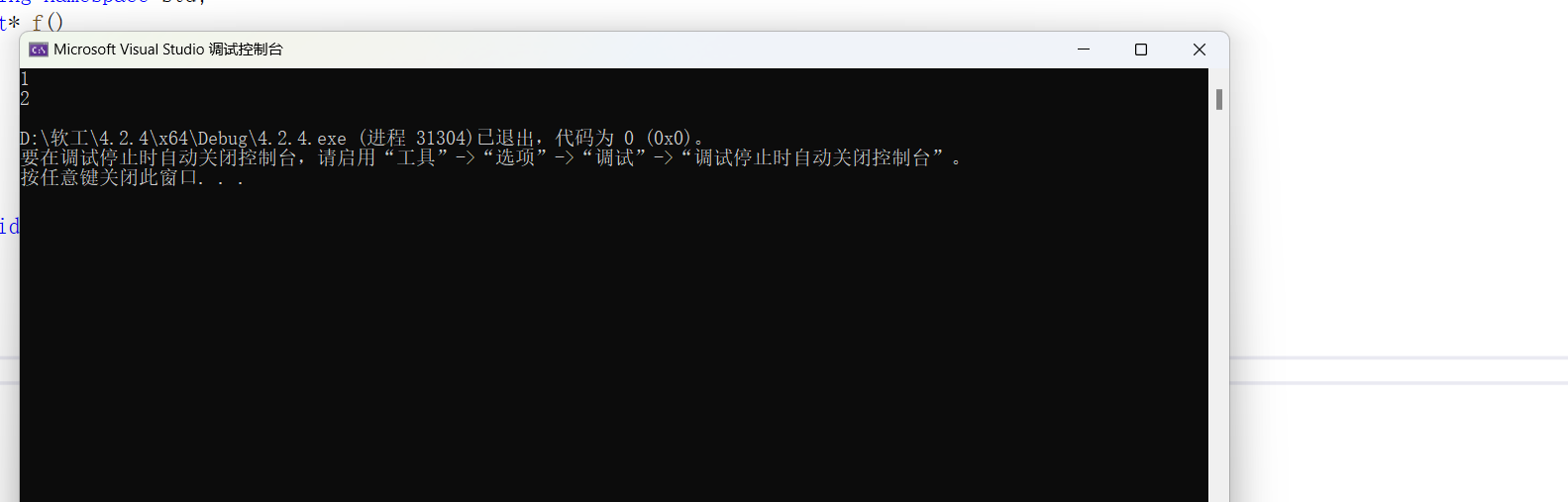
{

int\* p = f();

cout << p[0] << endl;

cout << p[1] << endl;

}



2、程序设计

(1)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);

#include<iostream>

using namespace std;

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

{

int len\_s1 = strlen(s1);//s1长度

int len\_s2 = strlen(s2);//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))

{

equal++;//计数+1

for (int j = 1; j < len\_s1; j++)//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//不相等

{

break;//跳出循环

}

}

if (equal == len\_s1)//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];

cout << "输入字符串s1:";

cin.getline(s1, 100);

cout << "输入字符串s2:";

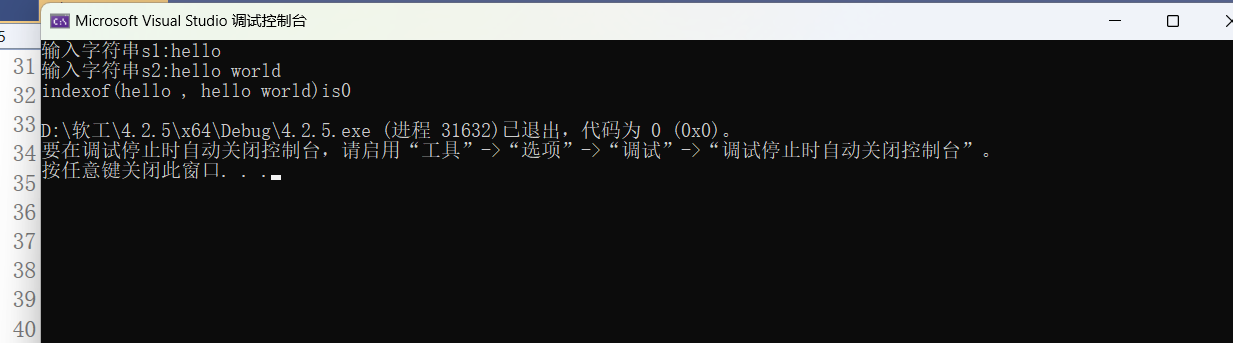
cin.getline(s2, 100);

int index = indexof(s1, s2);//调用函数

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

return 0;

}



(2)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

#include<iostream>

#include<cmath>

using namespace std;

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

{

int s\_len = strlen(hexString);

int num = 0;//定义0

for (int i = 0; i < s\_len; i++)//循环

{

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

{

num = num + (((int(hexString[i])) - 48) \* pow(16, s\_len - i - 1));//转换成int型并乘16的x-1次

}

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

{

num = num + (((int(hexString[i])) - 55) \* pow(16, s\_len - i - 1));//转换为int型并乘16的x-1次

}

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

{

num = num + (((int(hexString[i])) - 87) \* pow(16, s\_len - i - 1));//转换为int型并乘16的x-1次

}

else

{

cout << "数值非16进制" << endl;

return 0;

}

}

return num;

}

int main()

{

char s[50];

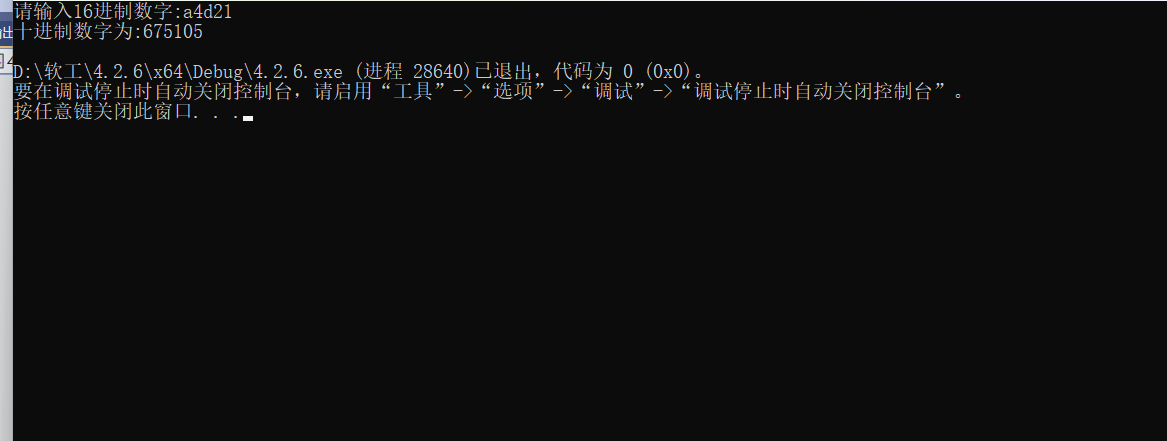
cout << "请输入16进制数字:";

cin.getline(s, 50);

int dec = parseHex(s);//调用函数

cout << "十进制数字为:" << dec << endl;

}



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）。

#include <iostream>

using namespace std;

void swap(int\* m, int\* n) //定义交换元素的函数

{

int temp = \*m;

\*m = \*n;

\*n = temp;

}

void bubbleSort(int\* arr, int size) // 冒泡排序函数

{

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

{

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

{

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

{

swap(&arr[j], &arr[j + 1]);

}

}

}

}

int main()

{

int size; // 数组大小

cout << "请输入数组元素个数: ";

cin >> size;

int\* arr = new int[size];// 使用 new 创建动态数组

cout << "请依次输入数组元素: ";

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

{

cin >> arr[i];

}

cout << "排序前的数组为: ";// 打印排序前的数组

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

{

cout << arr[i] << " ";

}

cout <<endl;

bubbleSort(arr, size); // 调用冒泡排序函数

cout << "排序后的数组为: ";// 打印

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

{

cout << arr[i] << " ";

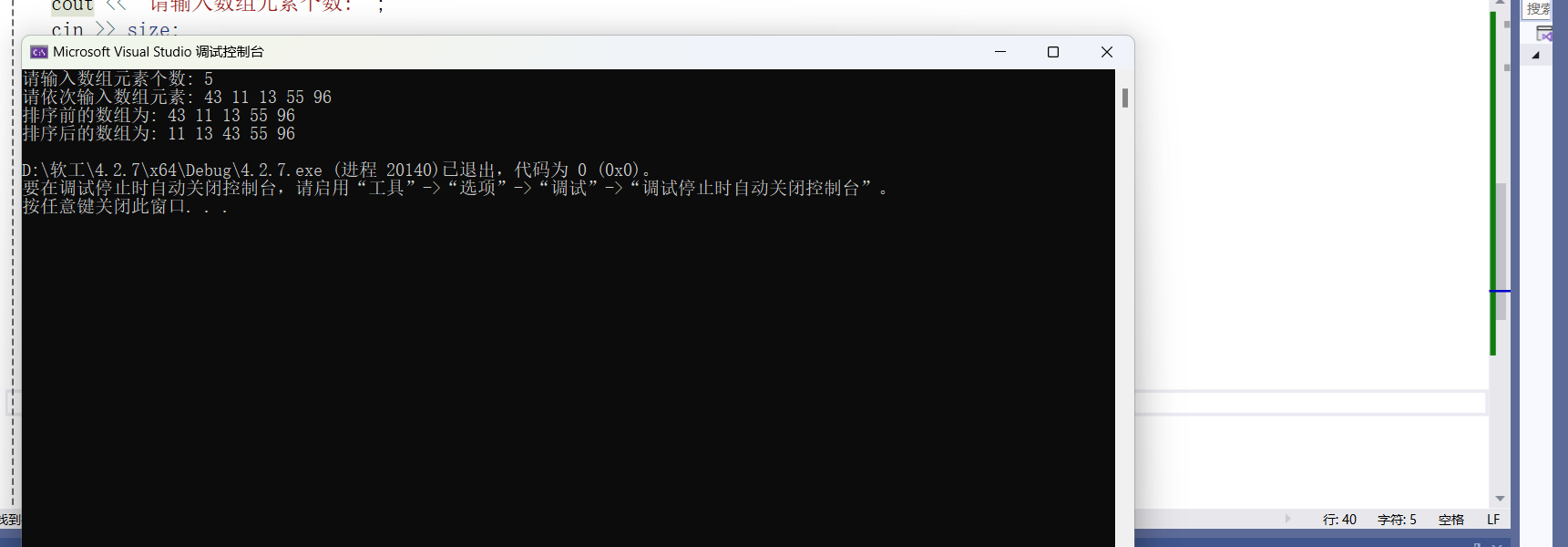
}

cout <<endl;

delete[] arr;// 释放内存

return 0;

}

1. 

【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**

**四、遇到的问题与解决方法**

**五、体会**