**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软件2103班

学 号： 8305211004

姓 名： 曾涛涛

**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）

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

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）

5、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）。

**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？

在C++中，与C语言一样，函数的返回值类型与函数声明中指定的类型一致，这是为了确保函数的调用者能够正确地处理函数返回的值。C++继承了C语言的静态类型系统，这意味着在编译时需要知道变量的类型。

函数的返回类型在函数声明时被指定，这样调用者就能够知道函数返回的是什么类型的值。如果函数的实现中返回的值与声明的类型不一致，编译器就会发出错误，因为这可能导致在调用点出现类型不匹配的错误。

例如：

#include <iostream>

int add(int a, int b) {

return a + b;

}

int main() {

int result = add(3, 4);

std::cout << "Result: " << result << std::endl;

// 如果尝试接收错误类型的返回值，编译器会报错

// double incorrectResult = add(1, 2);

// 错误：不能从'int'转换为'double'

return 0;

}

在这个例子中，add 函数返回一个整数，而在 main 函数中，我们正确地将返回的整数值存储在一个整数变量中。如果我们尝试用一个不匹配的类型来接收返回值，编译器会报错。这种强类型检查有助于避免潜在的类型错误。

2. 本实验中主函数调用函数时采用的是何种传递方式？

除了第一题的第二问是 引用传递，其他题目都是值传递。

引用传递，在函数内对变量的修改会直接体现到函数外的变量上。

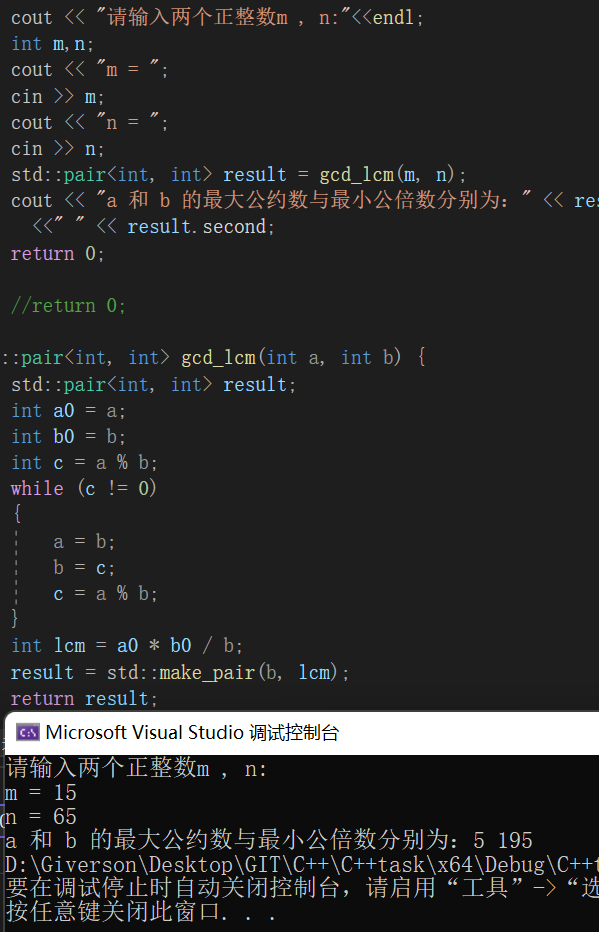
值传递，也就是传给函数的只是变量的副本，在函数内修改变量对函数外的变量没有影响。

**四、算法分析，程序结果**

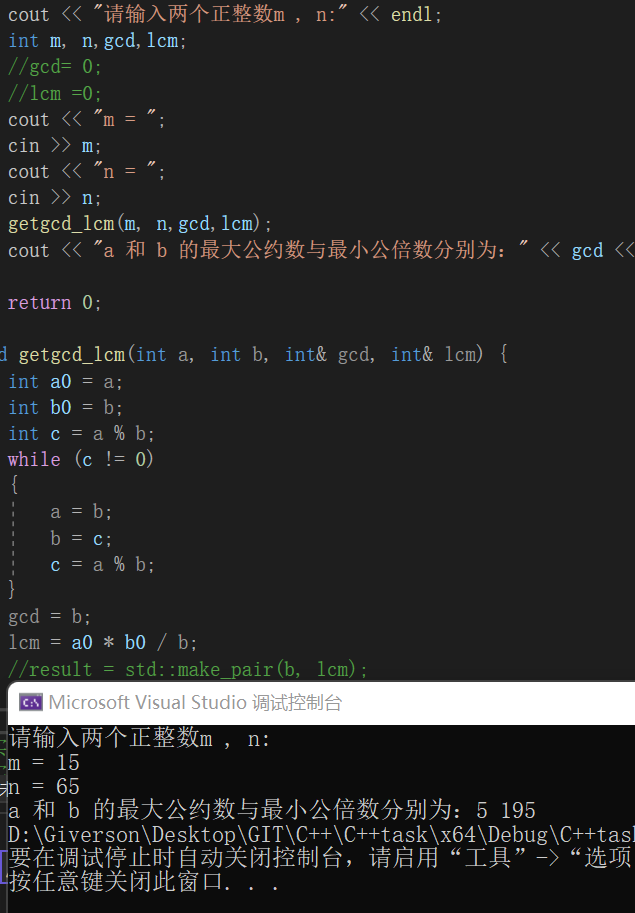
1、输入自然数m和n，

（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）

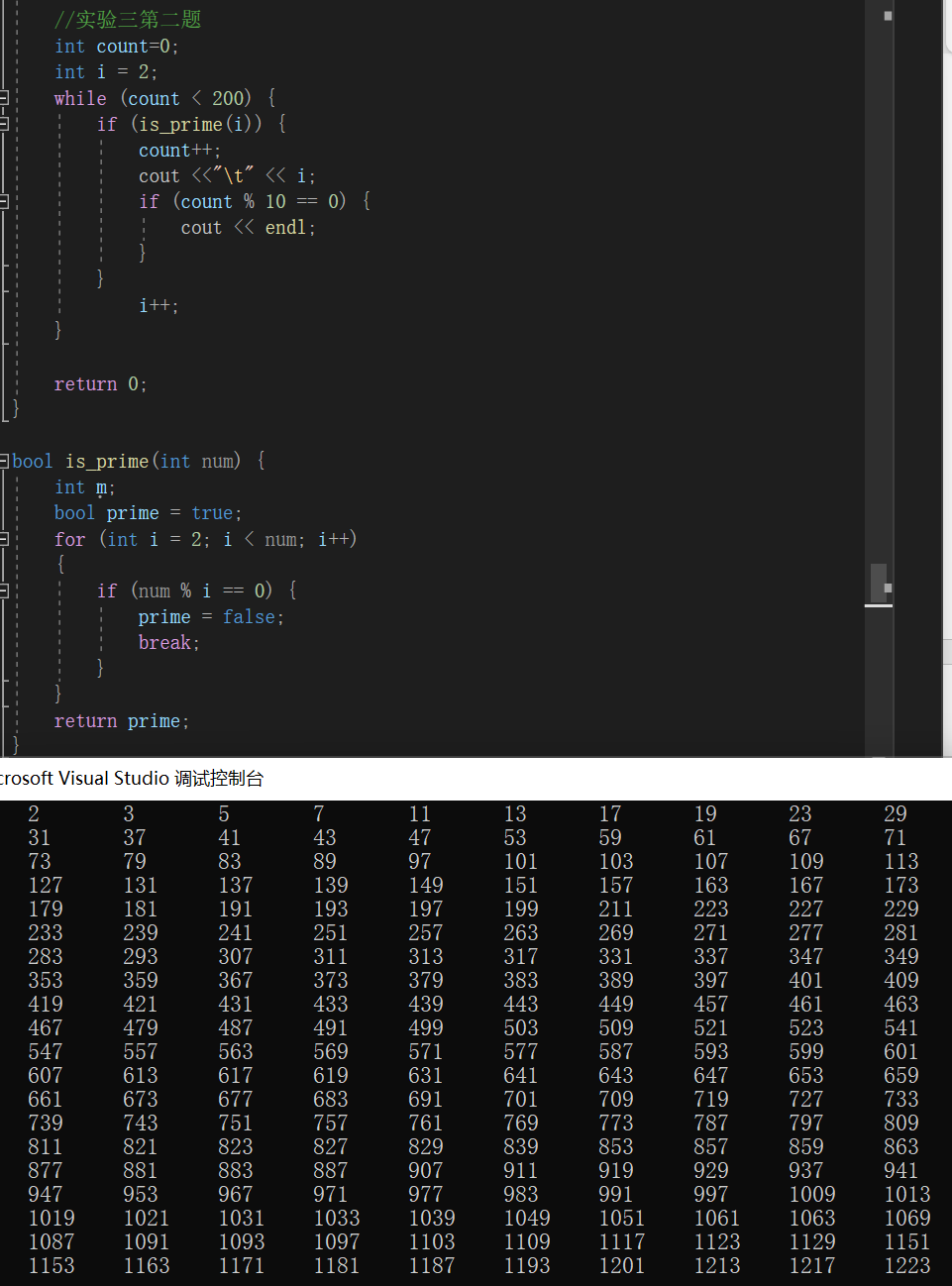


2. 编写程序满足：声明一个函数，判断一个整数是否为素数，使用如下函数头：

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



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

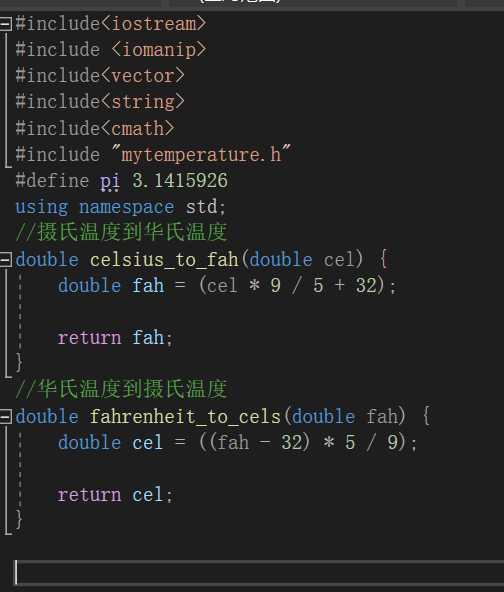
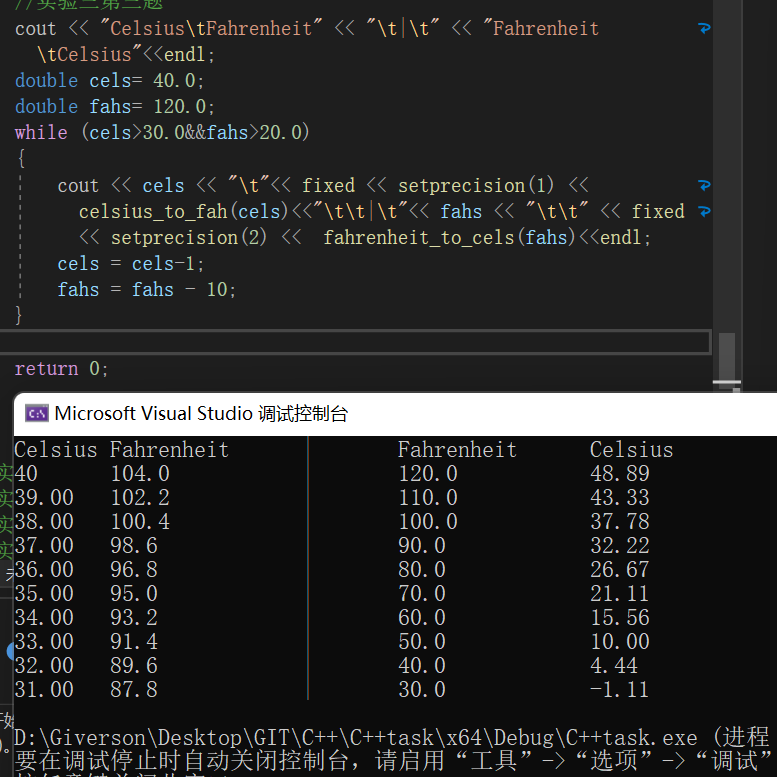
40.0       105.0        |   120.0            48.89

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))

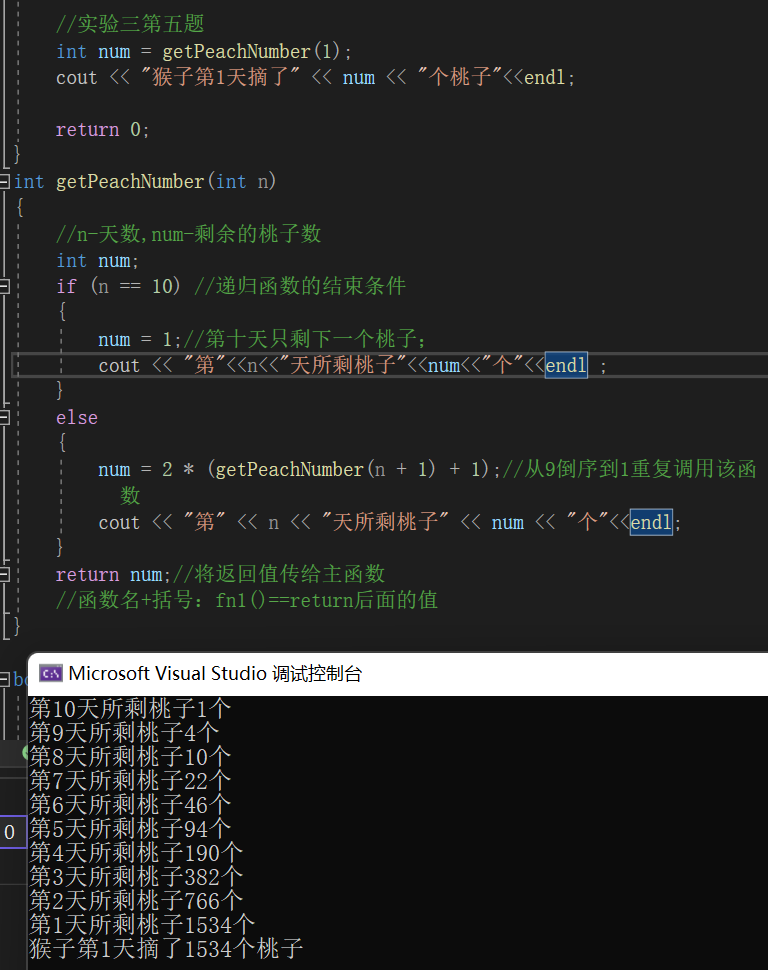
其中s=(side1+side2+side3)/2

写测试程序：读取三角形三边长，如输入合法（判断是否为三角形即可），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）

**3与4选一个完成**

5、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）。



**五、遇到的问题与解决方法**

**1、第一题第一问，因为GCD和LCM基本上是同时求出来的，所以我不想同样的过程写两次，所以就把函数合并在一起了；但是函数的返回值变成了两个，通过查找资料，学会了返回一个、两个、三个（多个）值的方法，这里返回两个值用的是std::pair<int,int>作为函数返回类型，用std::make\_pair<x,y>把两个值存到返回值里面去；**

**2、第一题第二问，要求使用引用参数的方法，有点不太清楚如何去写函数方法了，一开始的时候传入参数是&m和&n，所以没有能够把结果保存到gcd和lcm中，后面把传入参数改为了m,n,&gcd,&lcm就可以了；**

**3、第二题，素数的求法用的是最暴力的求法，简化版的有点不记得了，甚至简单版的都不太记得了；查阅资料解决**

**4、第三题，主要是输出格式不太对劲，调试几次后，就可以了；**

**5、第五题，递归有点忘了，不知道如何去嗲用自身，也不知道如何利用结束条件，搞的有点迷糊了，查阅资料后解决；**

**六、体会**

**编程题目就是这样，不练习就会生疏；以致遗忘。所以像什么算法，数据结构的相关知识都需要在空余时间利用大量的练习，形成一种肌肉记忆；这样进行编程的时候才方便；**

**通过本次实验，我对函数的概念和使用有了更深刻的理解，具体包括函数的定义、声明、编写、调用、参数传递、变量作用域以及多文件编程方法。**

1. **函数的定义与声明： 了解了函数的定义和声明的方法，明白如何正确地声明和定义函数，使得程序结构更加清晰。**
2. **函数的编写要求： 学会按照任务要求编写函数，确保函数的功能正确实现，参数传递正确，返回值符合预期。**
3. **函数的调用方法： 通过实验，熟悉了如何在主函数中调用不同的函数，实现模块化的程序设计，提高了代码的可读性和可维护性。**
4. **函数参数的传递方法： 了解了函数参数的传递方式，包括值传递和引用传递。在实际应用中，根据需要选择合适的传递方式，提高程序的效率。**
5. **变量的作用域： 学会控制变量的作用域，避免命名冲突和数据混乱。了解局部变量和全局变量的特性，合理使用变量，确保程序的正确性。**
6. **多文件编程方法： 学会将程序分成多个文件进行编写，了解头文件和源文件的概念，提高了代码的组织结构和可维护性。**
7. **任务实践： 通过实际的任务，如最大公约数计算、素数判断、温度转换和递归计算等，深入理解了函数在解决实际问题中的应用。**

**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

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

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

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

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

**（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]<<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

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）。

【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(代码+cmd运行)**

**（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）。

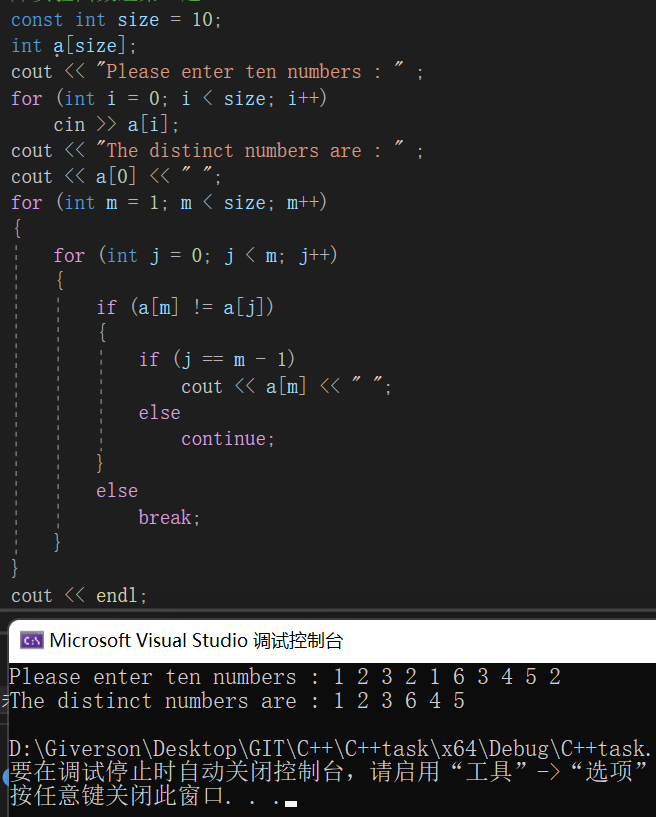
提示：读入的数如果是一个新的值，则将其存入一个数组。否则，将其丢弃。输入完毕后，数组中保存的就是不同的数。

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



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];

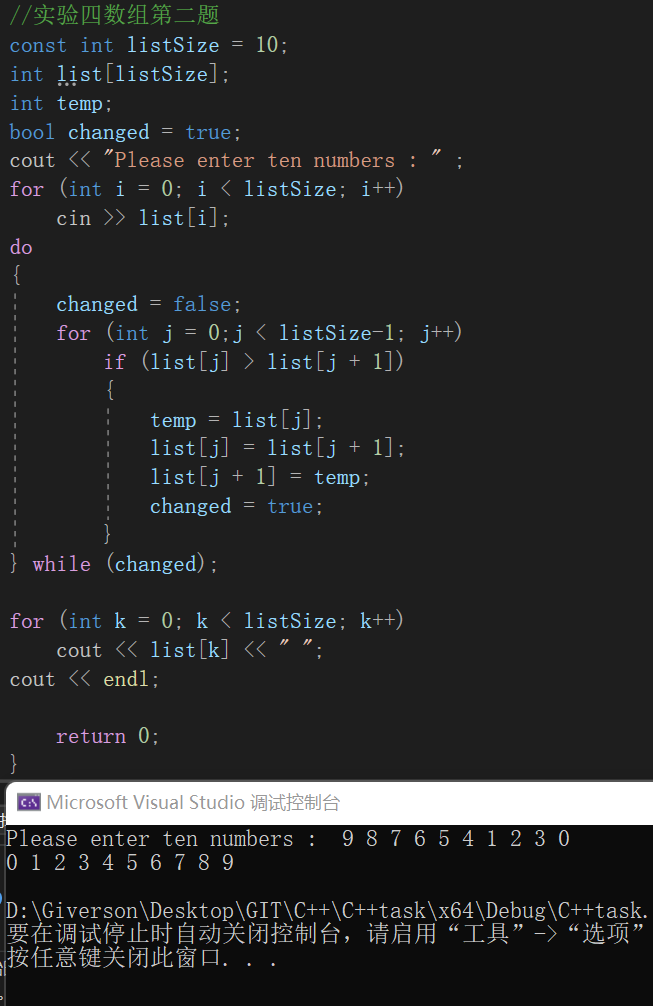
changed = true;

}

} 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。

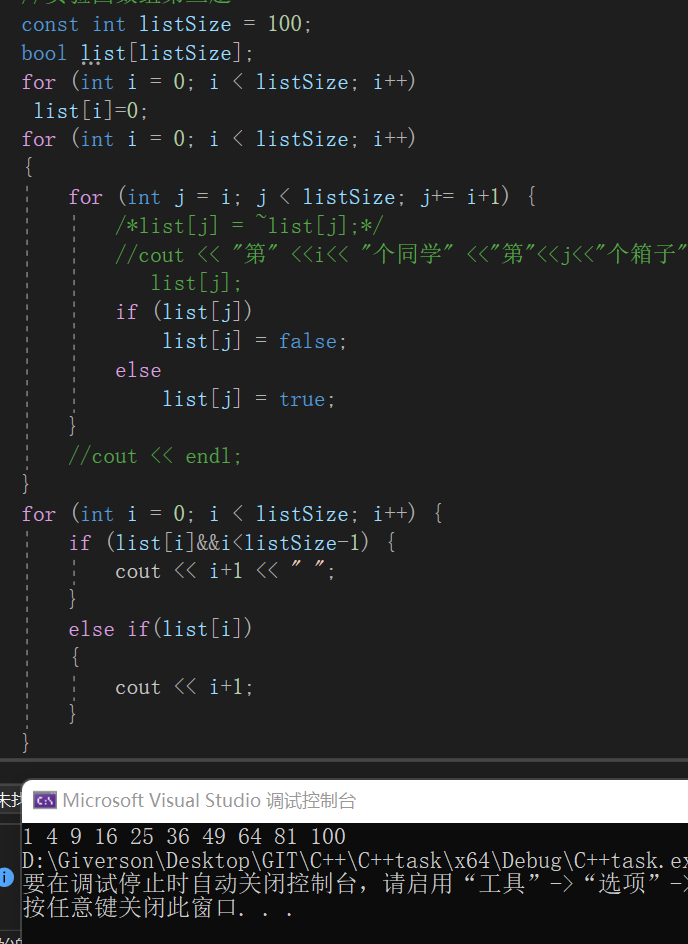


3、游戏：存物柜问题：

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。**

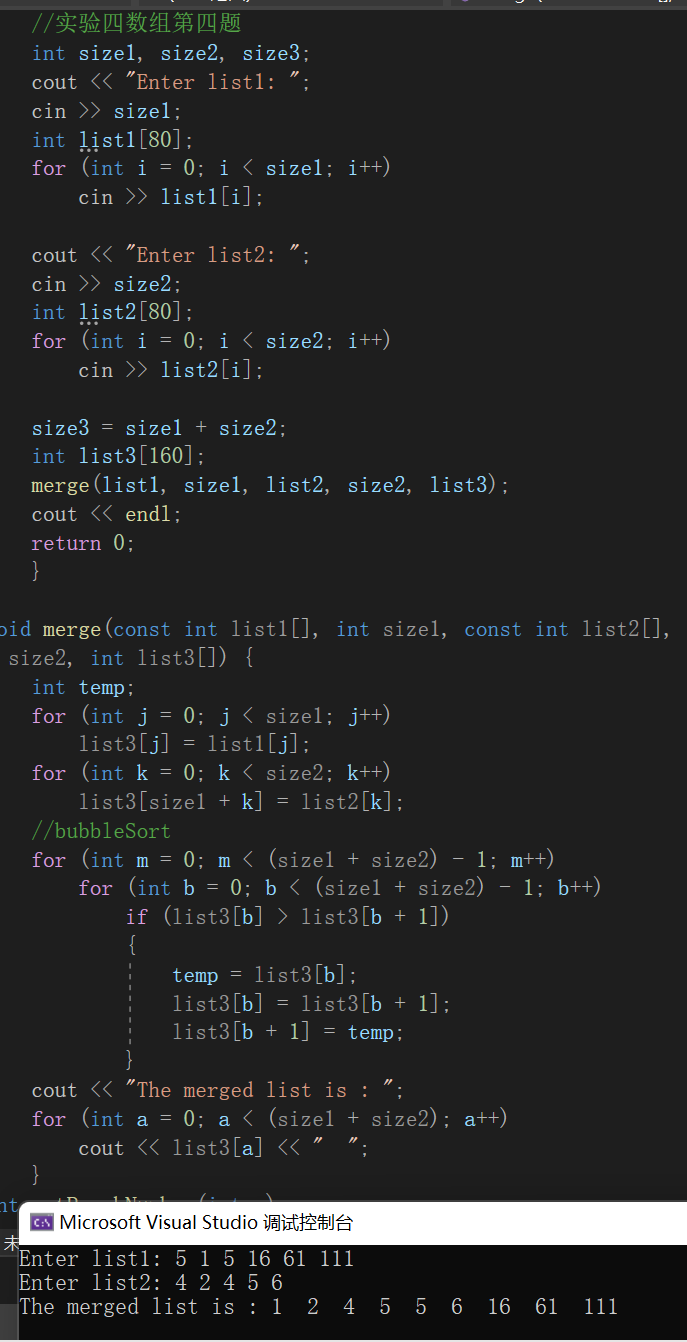
Enter

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



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!

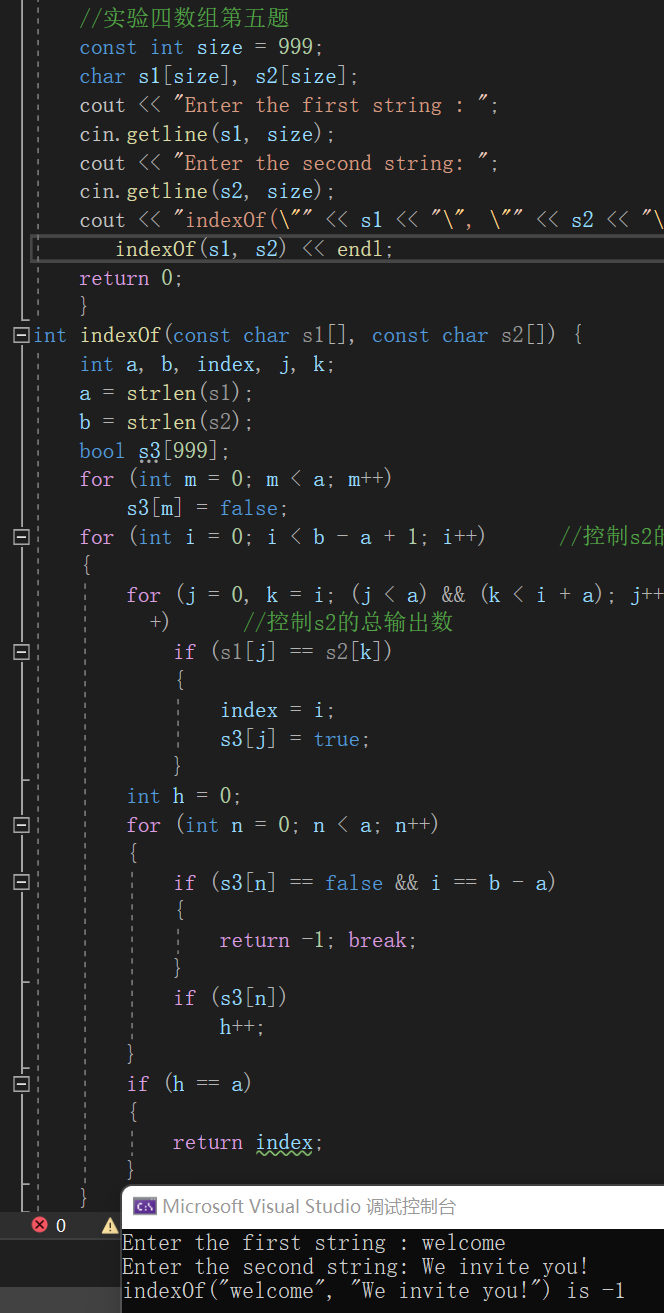
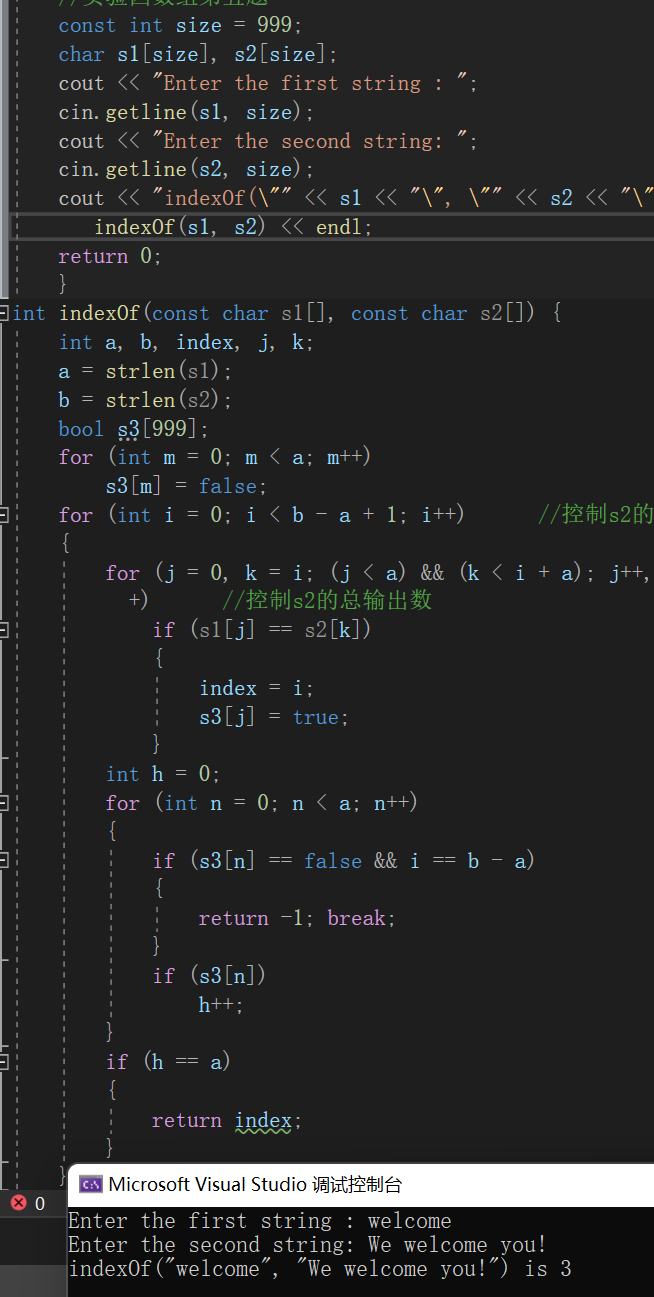
indexOf(“welcome”, “We welcome you!”) is 3

Enter

Enter the first string: welcome

Enter the second string: We invite you!

indexOf(“welcome”, “We invite you!”) is –1



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

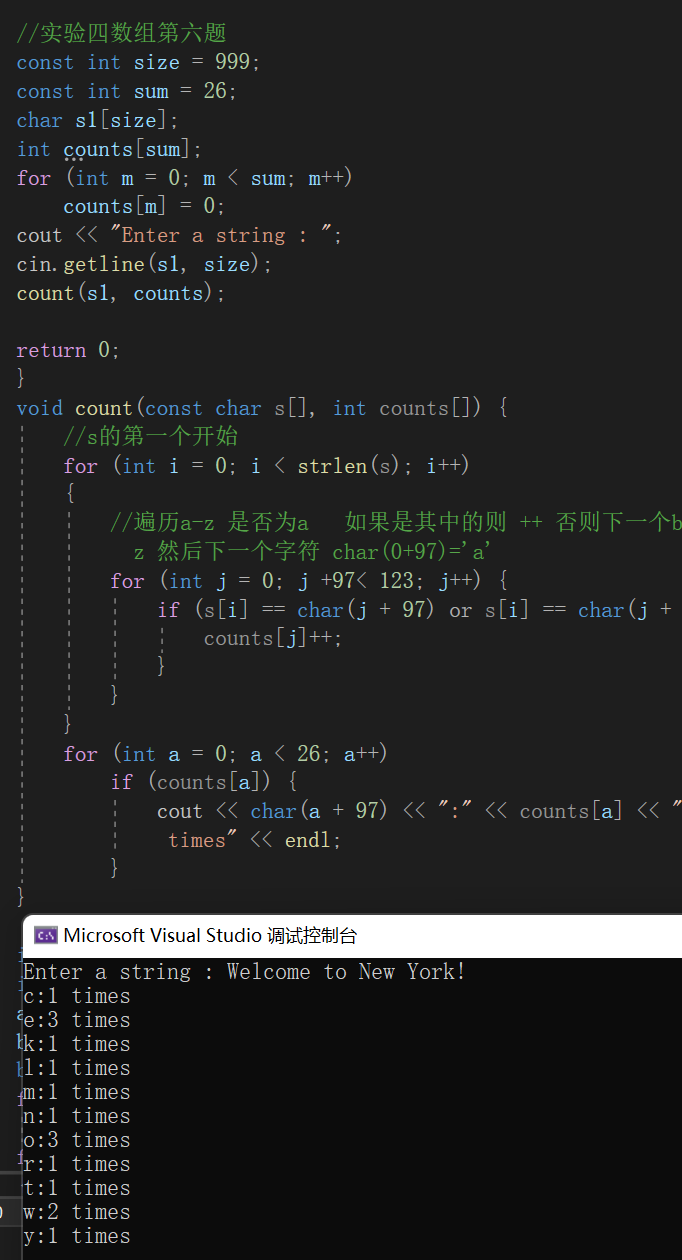
o: 3 times

r: 1 times

t: 1 times

w: 2 times

y: 1 times



**（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;

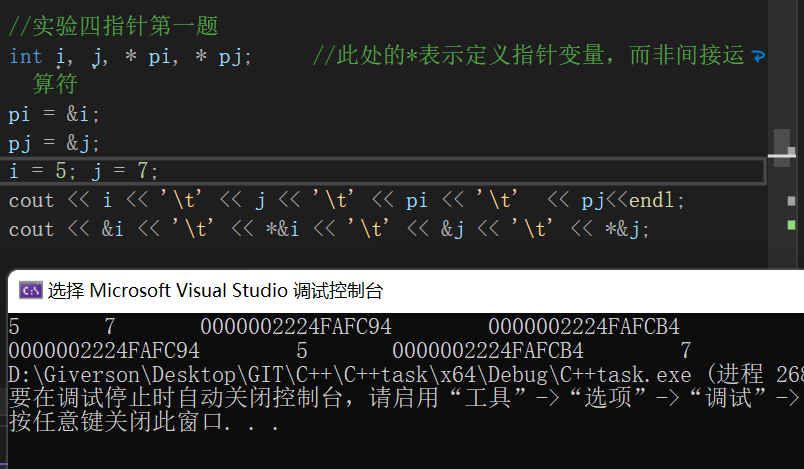
i=5;j=7;

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

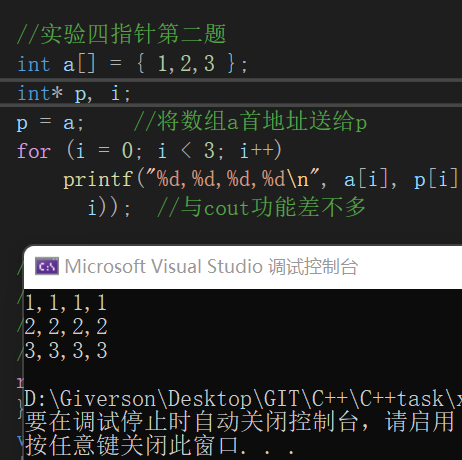
3,3,3,3

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()

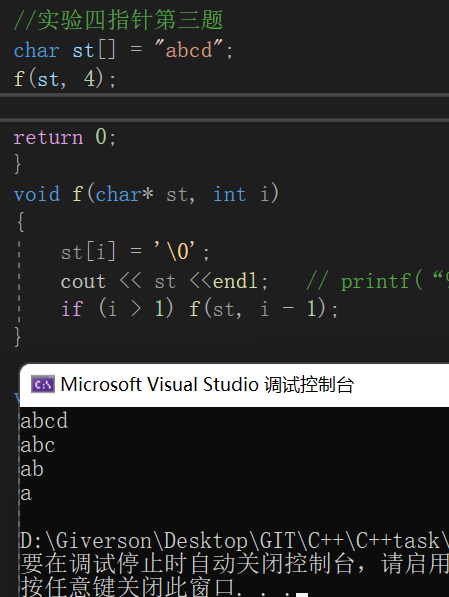
{

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()

{

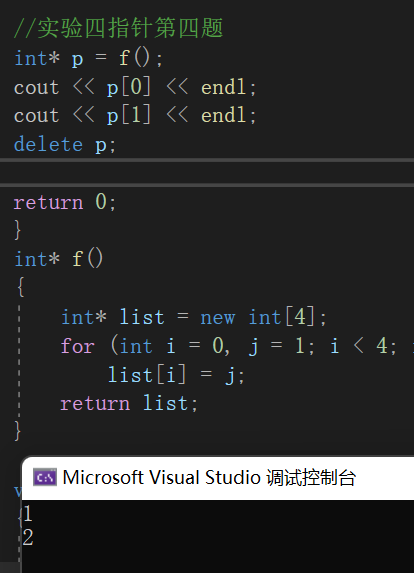
int \*p=f();

cout<<p[0]<<endl;

cout<<p[1]<<endl;

}

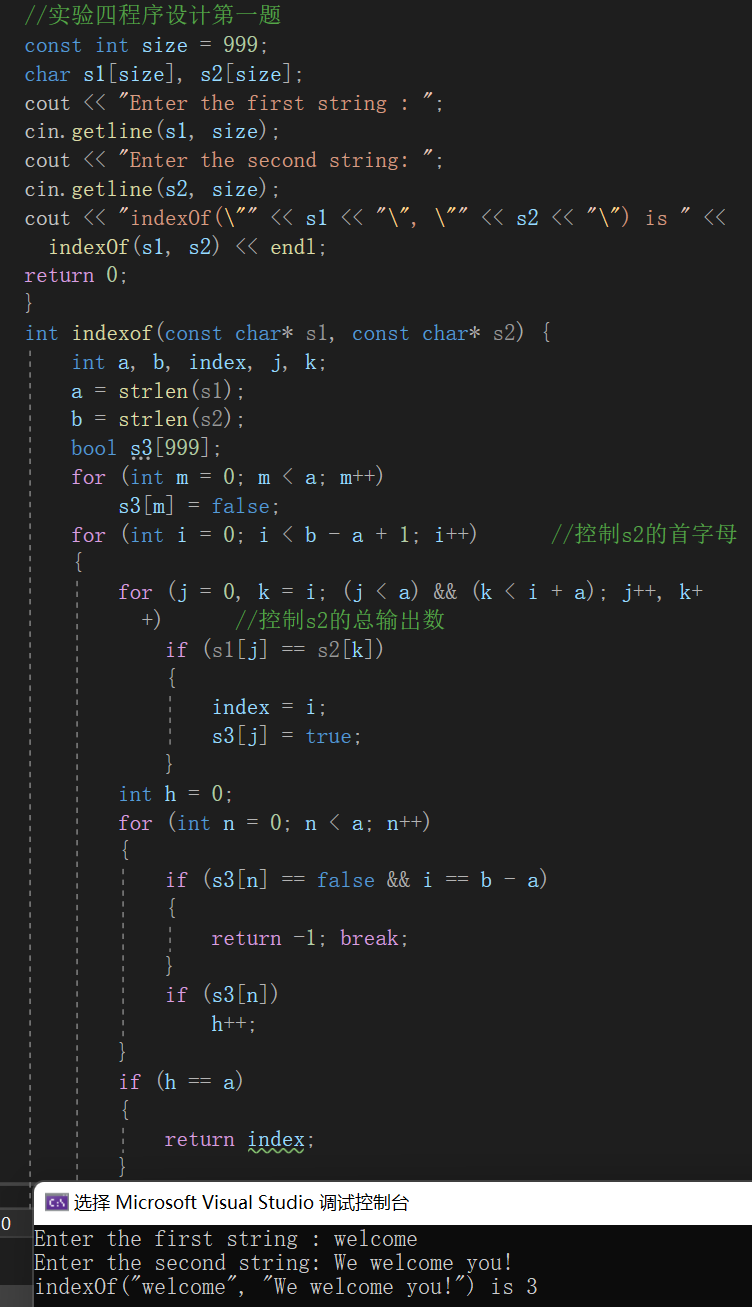
结果：不能保证p[0]输出1，p[1]输出2，修改后的代码为



2、程序设计

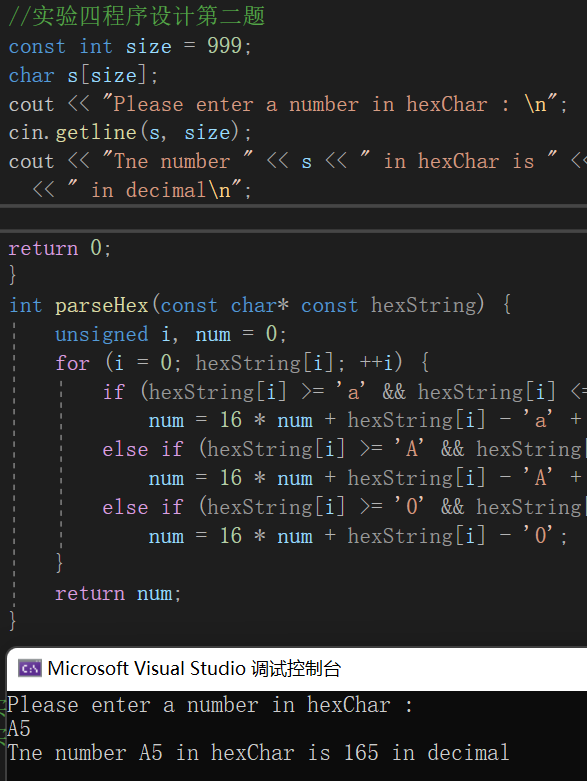
(1)编写函数检查字符串s1是否为字符串s2的子串，若是，返回第一次匹配的下标，否则返回-1。在主程序中输入字符串s1与s2，调用函数实现。

函数原型：int indexof(const char \*s1,const char \*s2);

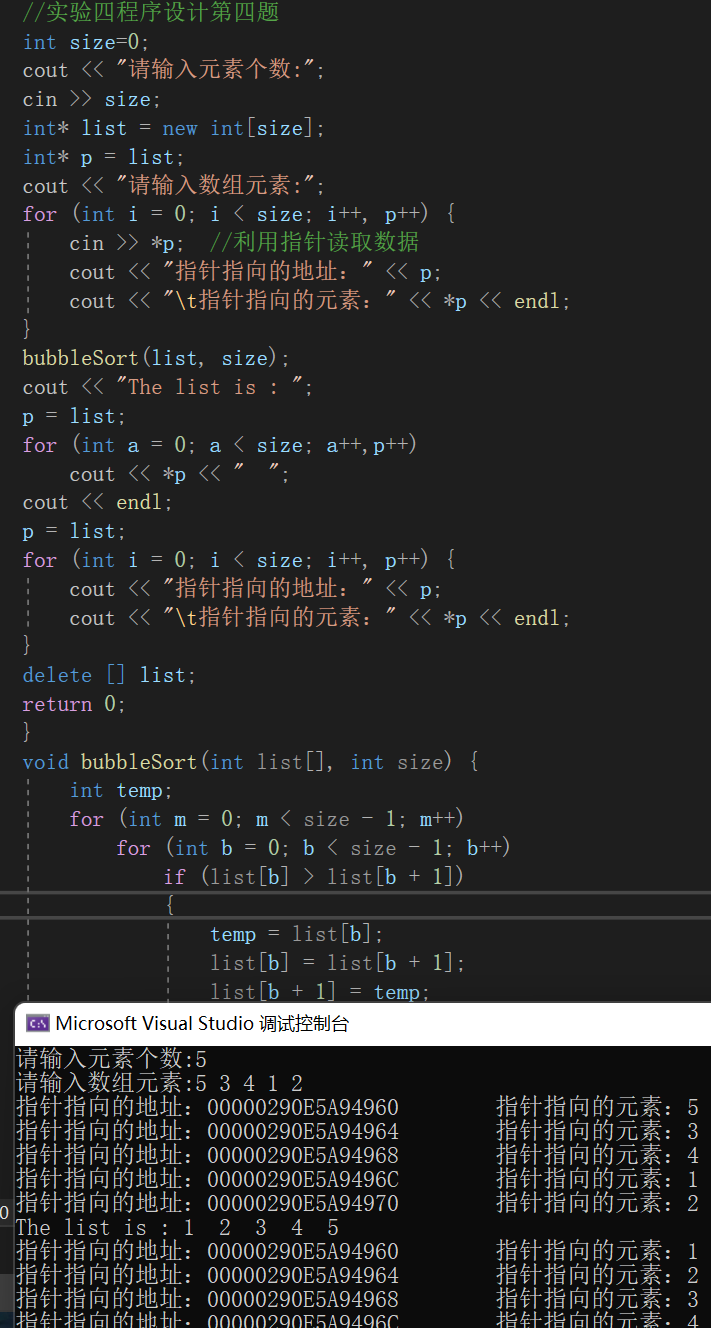
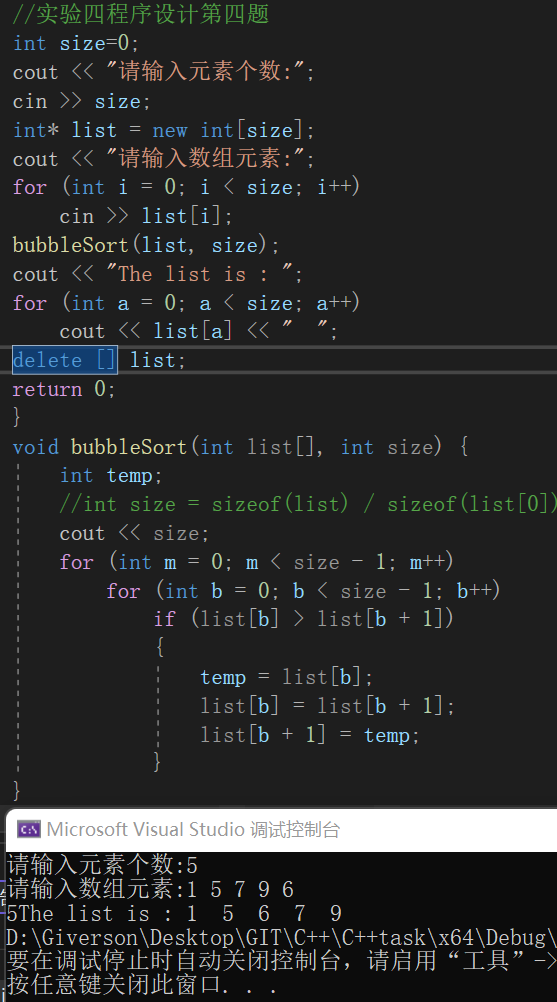


(2)编写一个函数将以**字符串形式**表示的一个16进制数转换为10进制数，并在主函数中测试。函数原型 int parseHex(const char \*const hexString);

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）。



**四、遇到的问题与解决方法**

1、数组第一题，一开始没有意识到需要先把数组一起性全部输入，考虑的是一个一个存起来，如果重复就不存，所以纠结了一下。后面发现要先都存起来的话，只需要遍历选择输出之前的所有数，只要都不想等就可以输出，否则选择下一个数就可以了；

2、数组第二题，起泡排序，有一点忘了，大概就是利用进行对比，然后temp进行交换；

3、数组第三题，存储柜一开始，用的的非~对元素进行转换，后面发现用不了这个东西，只能写if else代码了；

4、数组第四题，合并函数比较简单，先乱序都存起来，然后冒泡排序即可；

5、数组第五题，检验子串主要是，需要两重循环对其进行遍历；

6、数组第六题，这个主要知道ascii码就比较好写了；

7、指针第一题，直接运行即可，稍微调整了一下输出格式；

8、指针第二题，直接运行即可；

9、指针第三题，char st[]是const string型要用””把abcd引用起来；

10、指针第四题，动态数组的创建有点忘了，查阅一下资料之后完成；

11、程序设计第一题，这个之前做过；

12、程序设计第二题，主义分清有那些比较的情况就可以了，还有就是从数组的第一个数开始的话，其实是属于最内圈，然后逐层乘以radix；

13、程序设计第三题，这个之前做过类似的，主要是动态调试观察指针及指针指向的内容一开始不知道是什么意思，后面查阅资料得知；另外，计算数组大小的时候，这个数组是作为函数参数传进来的，此时已经变成了指针。

**五、体会**

**对于指针，数组，函数传参，这三者之间的组合使用，存在一定的难度，也是常常产生程序bug的地方，需要更加深入的学习；**

**数组与指针**

1. **数组的处理：** 通过实验中对数组的不同处理方法，我加深了对数组定义和处理的理解。特别是在涉及到排序、搜索等算法时，我学会了如何更有效地利用数组。
2. **指针的使用：** 实验中涉及到了指针的声明、赋值、运算等操作。理解了指针与数组之间的关系，以及指针在数组和函数中的应用。这让我更灵活地处理内存和数据。

**搜索与排序算法**

1. **搜索算法：** 通过实现字符串子串检验函数和检查数组中元素的函数，我理解了搜索算法的设计和实现。这对于处理大量数据时的查找操作非常重要。
2. **排序算法：** 实现了冒泡排序算法对数组进行排序。这加深了我对排序算法的理解，并培养了对算法复杂度的考虑。

**指针与动态内存分配**

1. **指针与数组关系：** 通过指针与数组的关系，我能更加灵活地访问和处理数组中的元素。这对于大规模数据的操作和优化至关重要。
2. **动态内存分配：** 学习了使用 **new** 进行动态内存分配，这让我能够灵活地管理内存，避免了静态数组大小不足的问题。同时，也意识到了释放内存的重要性。

**递归函数**

1. **递归函数：** 虽然递归函数在本次实验中未详细讲解，但通过问题的描述，我对递归的概念有了初步的了解。递归是一种解决问题的有趣方式，我期待在以后的学习中深入了解和运用递归。