**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软件工程2402班

学 号： 8209240205

姓 名： 宁帅铭

**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）

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）

**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）。

**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？

**解：**在C++中，函数的返回类型是其声明的一部分，它指定了函数在成功执行后应该返回的数据类型。当函数返回一个值时，这个值必须与函数的返回类型一致（或在某些情况下可以进行隐式类型转换），这是类型安全的一部分。以下是一些关键点来解释为什么函数的返回值必须与函数类型一致：

1）类型安全：

C++是一种强类型语言，要求变量和表达式具有明确的数据类型。

当函数返回一个值时，这个值必须能够明确地匹配函数的返回类型，以避免类型不匹配导致的错误或不确定行为。

2）编译器检查：

在编译时，编译器会检查函数的返回类型与返回语句中表达式的类型是否一致。

如果类型不匹配，编译器会报错，提示开发者修正错误。

3）隐式类型转换：

在某些情况下，如果返回类型与返回表达式的类型不同，但可以进行隐式类型转换（如int到double），编译器会允许这种转换。

但是，如果转换可能会导致数据丢失或精度下降，编译器可能会给出警告。

4）函数调用的一致性：

当一个函数被调用时，调用者期望得到特定类型的结果。

如果函数返回了不同类型的结果，调用者可能无法正确处理这个值，从而导致运行时错误。

5）函数重载和模板：

在C++中，函数可以基于返回类型和其他参数进行重载。

模板函数也依赖于返回类型的明确性来实例化正确的函数版本。

6）清晰的代码意图：

保持函数返回类型与返回值的类型一致有助于其他开发者理解代码的意图和功能。

一致的返回类型使得代码更加可读和可维护。

**示例**

假设有一个函数声明为返回int类型：

cpp

复制代码

int add(int a, int b) {

return a + b;

}

在这个例子中，add函数必须返回一个int类型的值，这与函数的返回类型一致。如果尝试返回其他类型（如double或std::string），编译器将报错。

**隐式类型转换的示例**

cpp

复制代码

double divide(int a, int b) {

return static\_cast<double>(a) / b;

}

在这个例子中，尽管函数返回double类型，但返回表达式中的int类型被隐式转换为double，这是合法的，因为int到double的转换是安全的。

总之，函数的返回值必须与函数的返回类型一致，这是C++类型安全机制的一部分，有助于确保程序的正确性和稳定性。

2. 本实验中主函数调用函数时采用的是何种传递方式？

**解：**第一题采用引用传递，其他题采用值传递

**四、算法分析，程序结果**

**1．（1）**

#include <iostream>

using namespace std;

int outside1(int& a, int& b)//定义外部函数，用来求传入数值的最大公因数

{

int min2;

if (a > b)

{

min2 = b;

}

else if (a < b)

{

min2 = a;

}

else

{

min2 = a;

}//利用if条件判断取得传入数据的最小值

for (; min2 <= a && min2 <= b;)

{

if (a % min2 == 0 && b % min2 == 0)

{

break;

}

else

{

min2--;

}

}//利用for循环求最大公因数

return min2;

}

int main()

{

int min1, m, n;

cout << "请输入两个自然数，本程序可以利用函数求得这两个数的最大公因数，并在主函数中输出" << endl;

cin >> m >> n;//在主函数中进行输入

min1=outside1(m, n);//调用外部函数求出最大公因数并返还给min1

cout << "最大公因数为" << min1 << endl;//在主函数中进行输出

return 0;

}



**1.（2）**

#include <iostream>

using namespace std;

void outside1(int& a, int& b)//定义外部函数，用来求传入数值的最大公因数和最小公倍数，其中用引用参数接受传入的实参

{

int max2, min2;

if (a > b)

{

max2 = a;

min2 = b;

}

else if (a < b)

{

max2 = b;

min2 = a;

}

else

{

max2 = a;

min2 = a;

}//利用if条件判断取得传入数据的最大值和最小值

for (; min2 <= a && min2 <= b;)

{

if (a % min2 == 0 && b % min2 == 0)

{

break;

}

else

{

min2--;

}

}//利用for循环求最大公因数

for (;max2>= a && max2 >= b;)

{

if (max2%a == 0 && max2%b == 0)

{

break;

}

else

{

max2++;

}

}//利用for循环求最小公倍数

cout << "利用引用参数求得两个数的最大公因数为:" << min2 << endl;

cout << "利用引用参数求得两个数的最小公倍数为:" << max2 << endl; //输出

}

int main()

{

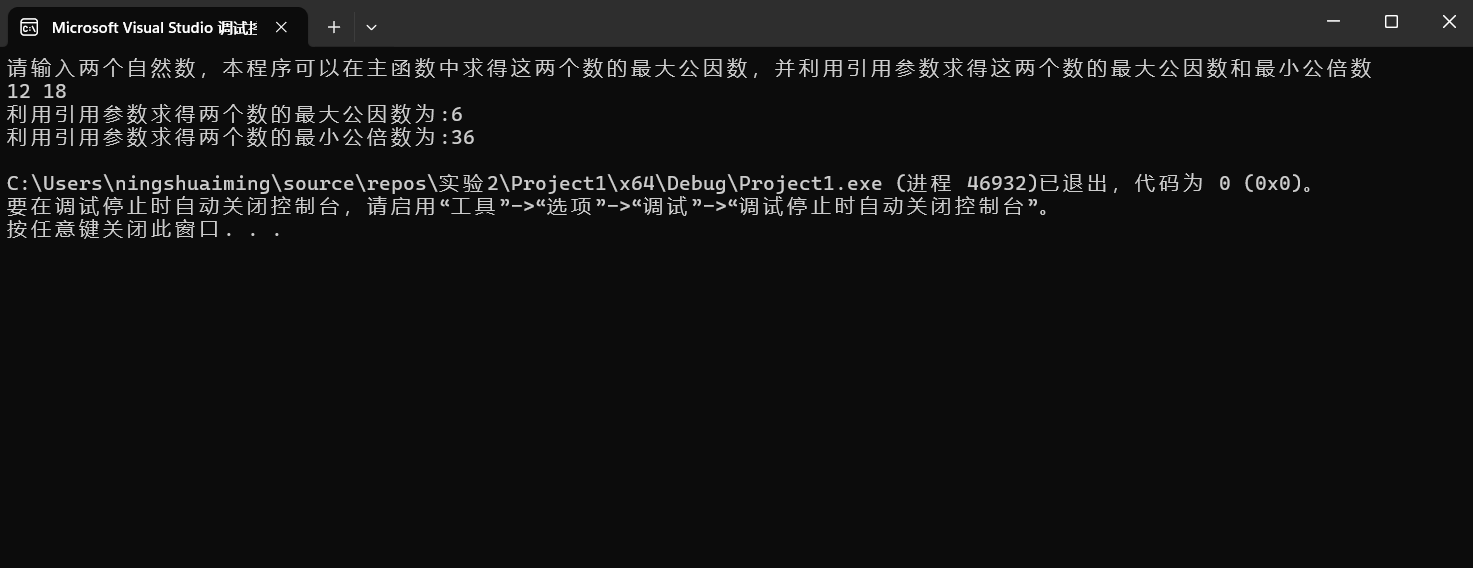
int min1, m, n;

cout << "请输入两个自然数，本程序可以在主函数中求得这两个数的最大公因数，并利用引用参数求得这两个数的最大公因数和最小公倍数" << endl;

cin >> m >> n;

outside1(m, n);//调用外部函数输出最大公因数和最小公倍数

return 0;

}

**2.**

#include<iostream>

#include <cmath>

using namespace std;

bool is\_prime(int num)//定义一个判断一个整数是否为素数的bool型函数

{

for (int in = 2; in <= sqrt(num); )//若传2，3，则in>num的开方，说明他们是素数。一个数若不是素数，它的一个非1非本身因数一定小于等于它的开方，故利用for循环和if条件判断是否存在大于1在小于等于它的开方范围内的整数因数便可

{

if (num % in == 0)

{

return false;

break;

}//若存在相应范围内的整数因数，说明传入的参数不是素数，返回false

else

{

in++;

}

}

return true;//最终未能找到相应范围内的因数，则传入的参数是素数，返回true

}

int main()

{

for (int i = 1,li=0,num=2; i <= 200;li=0 )//定义num初始值为2开始逐个判断是不是素数，list用来记录每一行的输出个数，i用来记录总输出个数

{

for (; li % 10 != 0 || li == 0; )

{

if (is\_prime(num) == 1)

{

cout << num<<'\ ';

i++;

li++;

num++;

}

else

{

num++;

}

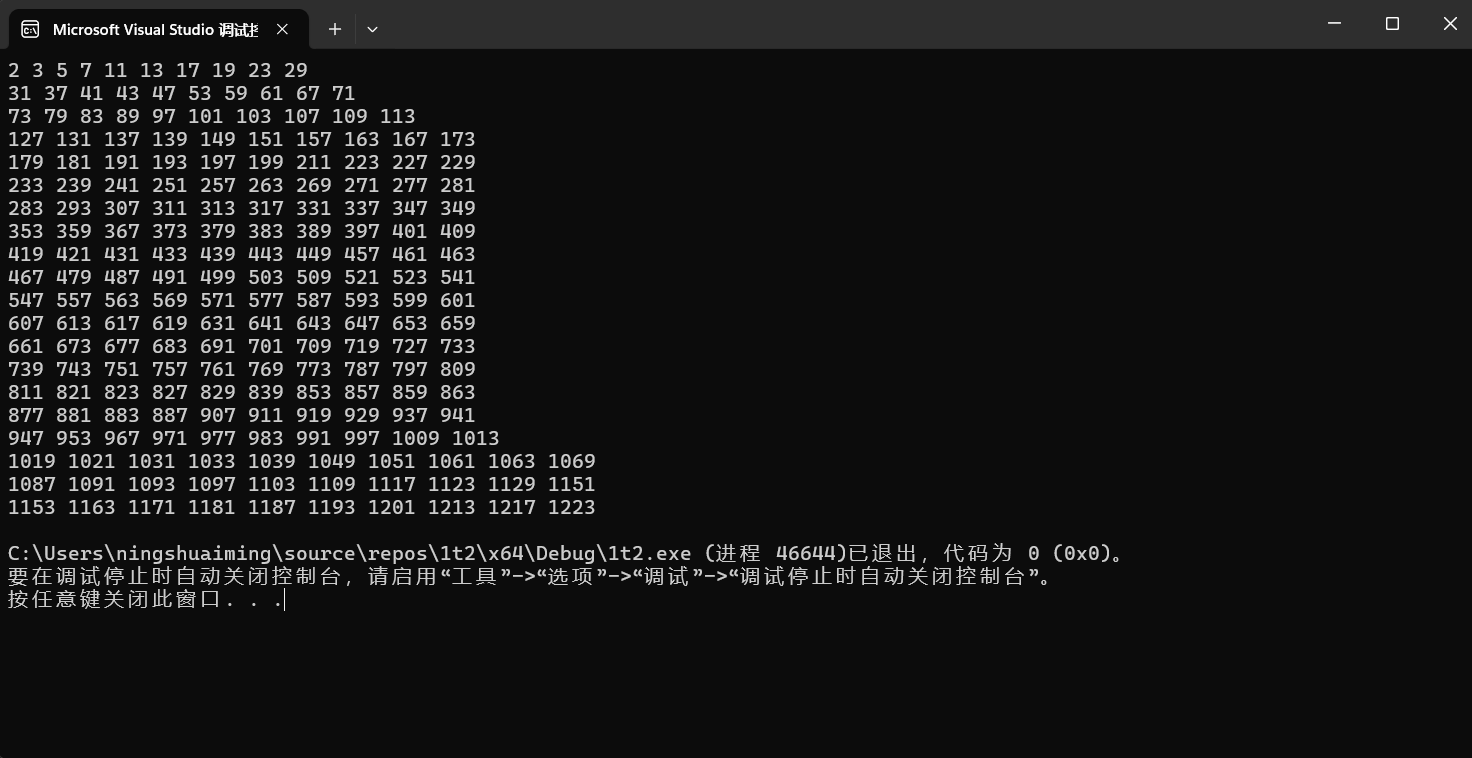
}

cout << endl;

}//利用for循环和if判断，并调用is\_prime（）函数，找出前200个素数，并按每行10个输出

return 0;

}



**3与4选一个完成**

**选择4，代码和结果如下：**

**文件名：mytriangle.h**

#pragma once//防止头文件被重复包含

#include <iostream>

using namespace std;

bool is\_valid(double side1, double side2, double side3);//声明

double area(double side1, double side2, double side3);//声明

**文件名：mytriangle.cpp**

#include<iostream>

#include<cmath>//包含头文件cmath，便于后续开方计算

#include"mytriangle.h"//包含头文件mytriangle.h

using namespace std;

bool is\_valid(double side1, double side2, double side3) {

if (side1 + side2 > side3 && side2 + side3 > side1)

{

return true;

}

else {

return false;

}

}//函数的定义

double area(double side1, double side2, double side3) {

double s = (side1 + side2 + side3) / 2, squ = sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));

return squ;

}//函数的定义

**文件名：1t4main.cpp**

#include <iostream>

using namespace std;

#include "mytriangle.h"//包含头文件mytriangle.h

int main() {

double a, b, c;

cout << "请输入三角形的三条边，本程序能为你计算这个三角形的面积" << endl;

cin >> a >> b >> c;

if (is\_valid(a, b, c) == 1) { //调用is\_valid函数判断是否能构成三角形

cout << "你所输入的三条边能构成三角形，该三角形的面积为" << area(a, b, c) << endl; //调用area函数求三角形面积

}

else {

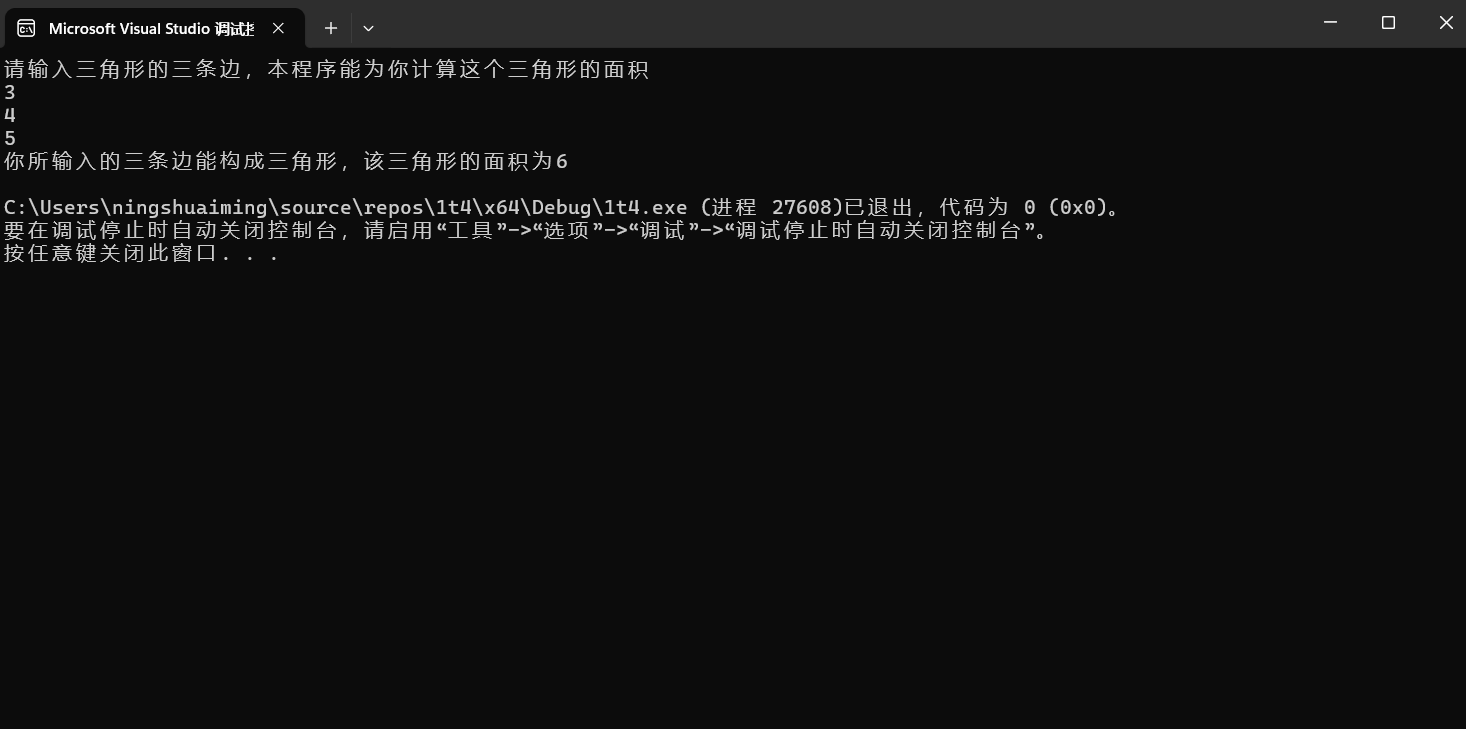
cout << "你所输入的三条边不能构成三角形" << endl;

}

return 0;

}





**5.**

#include <iostream>

using namespace std;

int cal(int fin,int day)//定义fin记录桃子数，day记录天数

{

if (day != 0)

{

fin = (fin + 1) \* 2;//求上一天桃子数

day = day--;//天数减一

cal(fin, day);//利用递归重复计算

}

else

{

return fin;//返回第一天桃子数

}

}

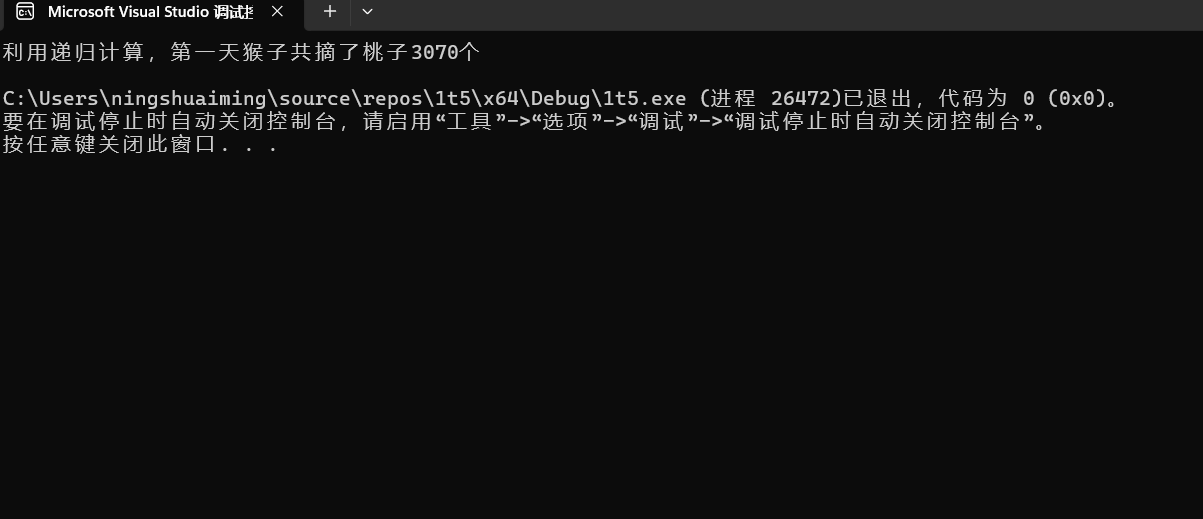
int main()

{

cout << "利用递归计算，第一天猴子共摘了桃子" << cal(1, 10) << "个" << endl;

return 0;

}



**五、遇到的问题与解决方法**

**1.不熟悉如何在程序中判断一个数是不是素数**

**解决方法：经过查询，了解到可以通过如下代码实现素数判断**

**for (int in = 2; in <= sqrt(num); )**

**{**

**if (num % in == 0)**

**{**

**return false;**

**break;**

**}//若存在相应范围内的整数因数，说明传入的参数不是素数，返回false**

**else**

**{**

**in++;**

**}**

**}**

**return true；**

**若主函数中传给num，2，3，则in>num的开方，说明他们是素数。一个数若不是素数，它的一个非1非本身因数一定小于等于它的开方，故利用for循环和if条件判断是否存在大于1在小于等于它的开方范围内的整数因数便可，如果不存在则是素数，如果存在则不是素数。**

**2.不熟悉如何实现题目中要求的“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”**

**解决方法：经过与同学们讨论，要满足题目的这个要求，只需要首先在****头文件mytriangle.h中声明所需要的函数；然后再CPP文件mytriangle.cpp先包含头文件mytriangle.h，再对头文件中声明的函数进行定义；最后只需要在main( )函数所在的CPP文件中包含头文件mytriangle.h，便可以调用该头文件中的函数，实现相关功能。**

**六、体会**

**在完成C++实验后，我对于编程有了更深一层的理解和感悟。这次实验不仅是对所学知识的实践应用，更是对基础知识重要性的深刻认识。以下是我实验后的几点体会：**

**1.重视基础知识**

**在实验过程中，我深刻体会到书本和课程中的基础知识对于编程的重要性。例如，递归作为一种重要的编程思想，在实验中的多个问题上都得到了广泛应用。通过递归，我能够更简洁、更高效地解决一些看似复杂的问题。同时，文件的包含也是编程中不可或缺的一部分，它帮助我们组织和管理代码，提高代码的可读性和可维护性。**

**2.积极巩固基础知识**

**在实验过程中，我发现自己对于一些基础知识的掌握还不够牢固，有些知识点甚至已经淡忘。面对这种情况，我积极采取了多种方式来巩固基础知识。首先，我会翻阅课本和课程资料，重新学习并理解这些知识点。其次，我会利用网络资源进行查询，找到相关的教程和案例，进一步加深对这些知识点的理解。最后，我还会与同学们进行讨论和交流，通过分享和探讨来巩固自己的知识。**

**3.实践出真知**

**除了重视基础知识外，我还深刻体会到实践的重要性。只有通过实践，才能真正理解并掌握所学知识。在实验过程中，我不断尝试将所学知识应用于实际问题中，通过不断试错和修正，逐渐掌握了编程的技巧和方法。同时，实践也让我更加深入地理解了基础知识的重要性和实用性。**

**总结与反思**

**通过这次实验，我不仅巩固了基础知识，提高了编程能力，还学会了如何积极查找和解决问题。我意识到，编程不仅是一门技术，更是一种思维方式和解决问题的方法。在未来的学习和工作中，我将继续重视基础知识的学习和实践应用，不断提高自己的编程能力和解决问题的能力。**

**总之，这次C++实验让我收获颇丰，我深刻体会到了基础知识的重要性以及实践的价值。**

**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<iostream>

using namespace std; #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\_\_abcdabcaba\_\_\_\_\_\_

(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**

**(一)1.**

#include <iostream>

using namespace std;

int main() {

double a[10];

double te;

int i, num;

cout << "请你输入十个数,本程序将输出其中不同的数（即如果一个数出现多次，只打印一次）。" << endl;

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

{

cin >> te;

for (int n = 0; n < i; n++)//利用for循环，判断新输入的数与之前的数是否存在重复

{

if (a[n] == te)

{

goto jump;

}//若存在重复，跳转到jump位置，不将该数加入数组，而是继续读取数字对该位置进行填充

}

a[i]=te;//若不存在重复，将该数加入数组

continue;

jump:i--;

}

cout << "输出结果为："<<' ';

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

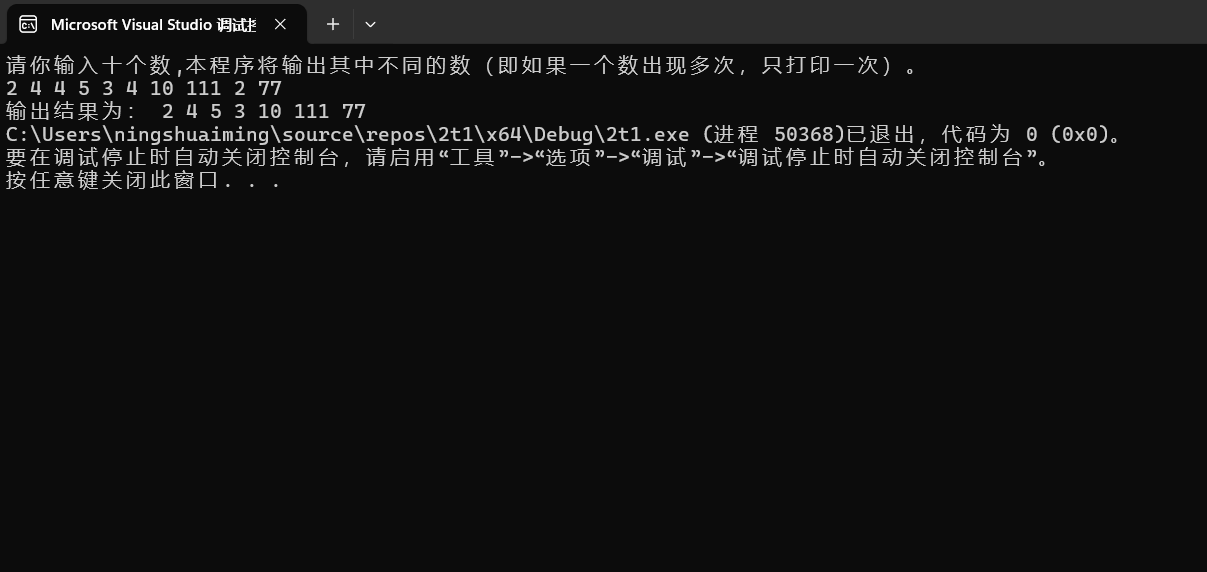
{

cout << a[outp] << ' ';//逐个输出

}

return 0;

}



**(一)2.**

#include <iostream>

using namespace std;

void ada(double a[] ,int size ) {

bool changed = true;

do

{

changed = false;//初始值设为false

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

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

{

int help;

help = a[j];

a[j] = a[j + 1];

a[j + 1] = help;

changed = true;

}//一旦发生一次调换，调换的两个数之前的数就有了需要调换的嫌疑，需要把changed值设为true，在此执行检索调换循环

} while (changed==true);//函数执行条件changed值为true

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

{

cout << a[i] << ' ';//将起泡排序后的数组输出

}

}

int main() {

double fin[10];

cout << "请输入10个双精度数字作为一个数组，本程序将调用函数把这个数组进行起泡排序并输出" << endl;

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

{

cin >> fin[i];

} //输入数组

cout << "进行起泡排序后的数组为:"<<' ';

ada(fin, 10); //调用ada（）函数，对数组进行起泡排序并输出

return 0;

****}

**(一)3.**

#include <iostream>

using namespace std;

int main() {

bool a[100];//定义bool数组代表柜子的开关状态

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

a[i] = false;

}//实现开学第一天所有柜子都关着

for (int S = 1; S <= 100; S++) {

for (int L = S; L <= 100; L = L + S) {

a[L - 1]=!a[L - 1];

}

} //根据学生学号对柜子状态进行改变

cout << "所有开着的柜子的号码为：";

for (int n = 0; n < 100; n++) {

if (a[n] == true) {

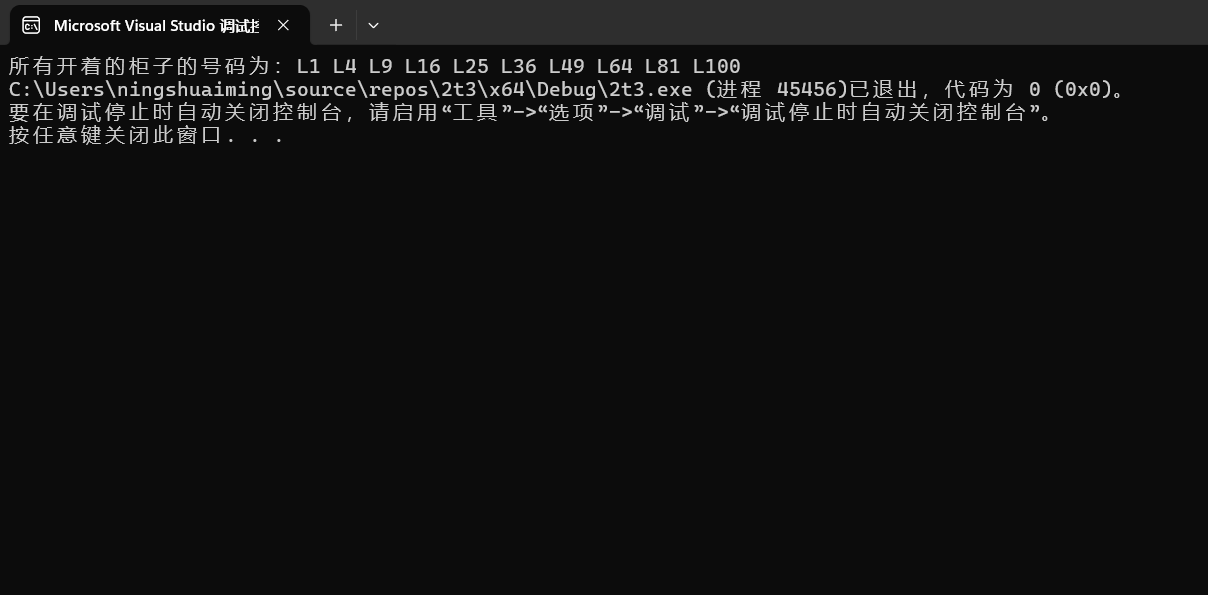
cout << "L" << n + 1 << ' ';//用' '表示空格进行分隔

}

} //把所有状态为true的柜子代号进行输出

return 0;

}



**(一)4.**

#include <iostream>

using namespace std;

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

bool changed = false;

do {

changed = true;

for (int j = 0; j < size1 + size2 - 1; j++) {

if (list3[j] > list3[j + 1]) {

int mid;

mid = list3[j];

list3[j] = list3[j + 1];

list3[j + 1] = mid;

changed = false;

}

}

} while (changed == false);//进行冒泡排序

cout << "新的排列好的数组为：";

for (int n = 0; n < size1 + size2; n++) {

cout << list3[n] << ' ';

}//输出排序好的数组

}

int main() {

int size1, size2, size3;

cout << "请输入一个不大于80的数作为第一个数组的长度" << endl;

cin >> size1; //输入数组长度

cout << "请再输入一个不大于80的数作为第二个数组的长度" << endl;

cin >> size2; //输入数组长度

size3 = size1 + size2; //求得合并后的数组长度

int\* list1 = new int[size1];

int\* list2 = new int[size2];

int\* list3 = new int[size3];

cout << "请输入第一个数组" << endl;

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

cin >> list1[i];

}//输入第一个数组

cout << "请输入第二个数组" << endl;

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

cin >> list2[i];

}//输入第二个数组

for (int m = 0; m < size1; m++) {

list3[m] = list1[m];

}

for (int n = 0; n < size2; n++) {

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

}//利用两个for循环把两个数组合并为一个数组

merge( list1, size1, list2, size2, list3); //调用函数把合并后的数组进行冒泡排序并输出

delete []list1, list2, list3;//释放动态内存

return 0;

}



**（一）5.**

#include<iostream>

#include<vector> //调用<vector>头文件，便于在程序中创建vector型数组，实现动态地对数组长度进行改变

using namespace std;

void indexOf(const char s1[] , const char s2[]) {

const char\* re = strstr(s2, s1); //利用strstr()函数判断s1是否为s2的子串，若是将返回s1在s2中首次出现处s2元素的地址，若不是则返回NULL

if (re != nullptr) {

cout << "s1是s2的子串，s1在s2中的下标为：" << re - s2 << endl;

}

else {

cout << "s1不是s2的子串，返回值：" << -1 << endl;

}

}

int main() {

vector<char>s1;

vector<char>s2; //创建两个vector型字符数组

char mid1, mid2;

cout << "本程序可以判断一个字符串s1是否为另一个字符串s2的字串" << endl;

cout << "请输入第一个字符串s1" << endl;

while (mid1 = cin.get()) { //读入输入的字符

if (mid1 != '\n') {

s1.push\_back(mid1); //将所有非回车字符添加进数组中

}

else {

s1.push\_back('\0'); //当读到回车，代表输入结束。不向数组中加入回车，而是加入'\0',表示字符串的结束

break;

}

}

cout << "请输入第二个字符串s2" << endl;

while (mid2 = cin.get()) { //读入输入的字符

if (mid2 != '\n') {

s2.push\_back(mid2); //将所有非回车字符添加进数组中

}

else {

s2.push\_back('\0'); //当读到回车，代表输入结束。不向数组中加入回车，而是加入'\0',表示字符串的结束

break;

}

}

int size1 = s1.size();

int size2 = s2.size();

char\* S1 = new char[size1];

char\* S2 = new char[size2];

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

S1[i] = s1[i];

} //将vector型数组转换为普通型数组

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

S2[i] = s2[i];

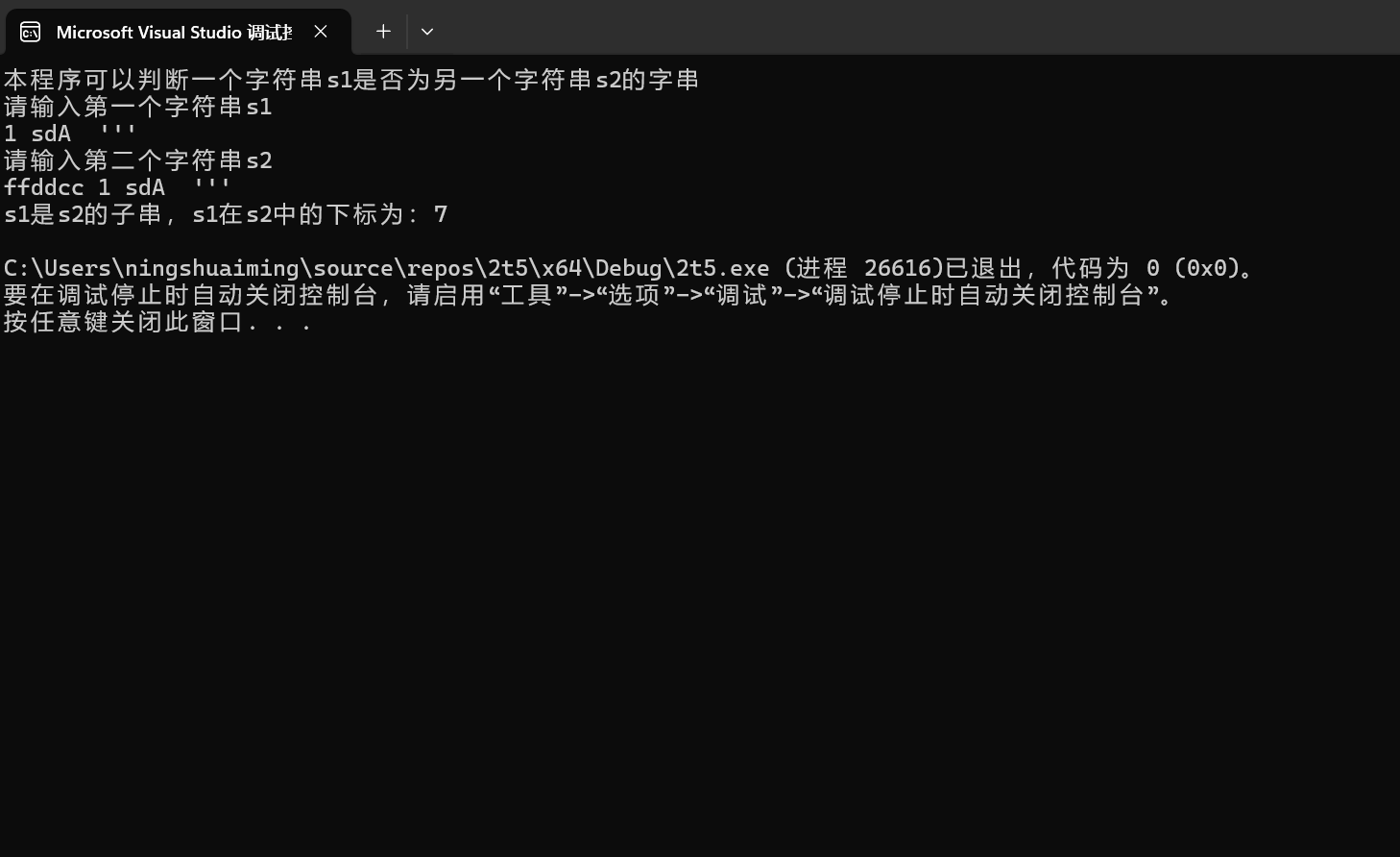
} //将vector型数组转换为普通型数组

indexOf(S1, S2); //调用函数进行字串判断

delete []S1, S2; //释放动态内存

return 0;

}

****

**（二）6.**

#include <iostream>

#include <vector> //引入头文件<vector>,便于后续创建vector型数组，实现数组的动态调整

using namespace std;

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

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

counts[i] = 0;

for (int n = 0; s[n] != '\0'; n++) {

if (s[n] == 65 + i || s[n] == 97 + i) {

counts[i]++;

}

}

} //利用A到Z以及a到z的ascii码对各个字母出现的次数进行统计

for (int c = 0; c < 26; c++) {

if (counts[c] != 0) {

cout << char(97 + c) << ":" << counts[c] << ' ' << "times" << endl;

}

} //将统计结果中字母出现次数非零的部分进行输出

}

int main() {

int counts[26];

vector <char> start;

char box;

cout << "请输入一个字符串，本程序可以数出字符串中每个字母出现的次数，其中字母不分大小写，例如字母A和字母a都被看作a" << endl;

while (box=cin.get()) {

if (box != '\n') {

start.push\_back(box);

}

else {

start.push\_back('\0');

break;

}

} //将输入的字符构建为一个以'\0'结束的字符串

int size = start.size(); //利用size（）函数求得字符串的长度

char\* put = new char[size];

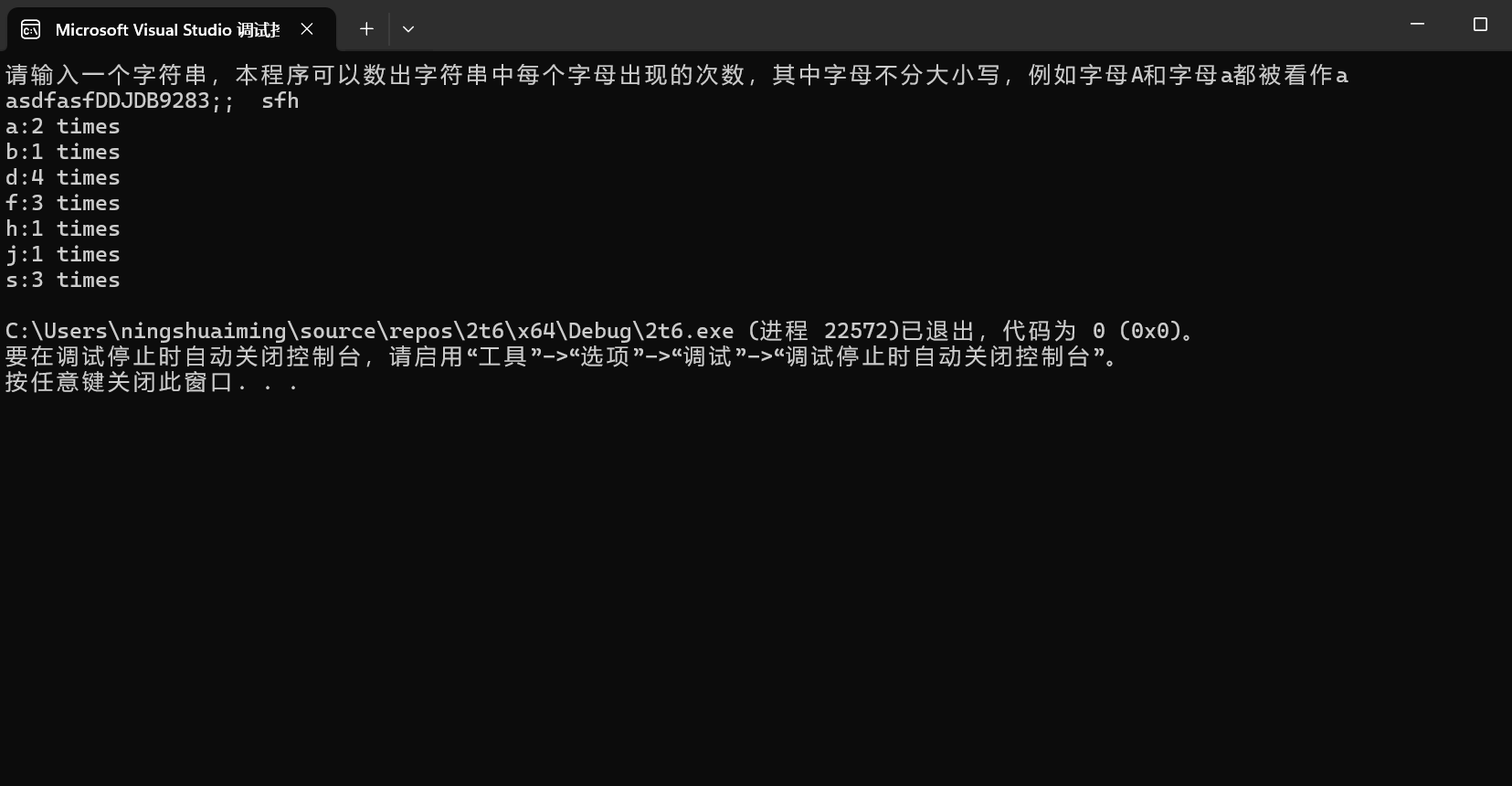
for (int i = 0; i < size; i++) {

put[i] = start[i];

} //将vector型数组转换为普通型数组

count(put, counts); //调用函数数出字符串中每个字母出现的次数

return 0;

****}

**(二) 1. （1）.**

using namespace std;

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;

}



**(二)1. （2）.**

#include <iostream>

int main() //C语言程序，要了解

{

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

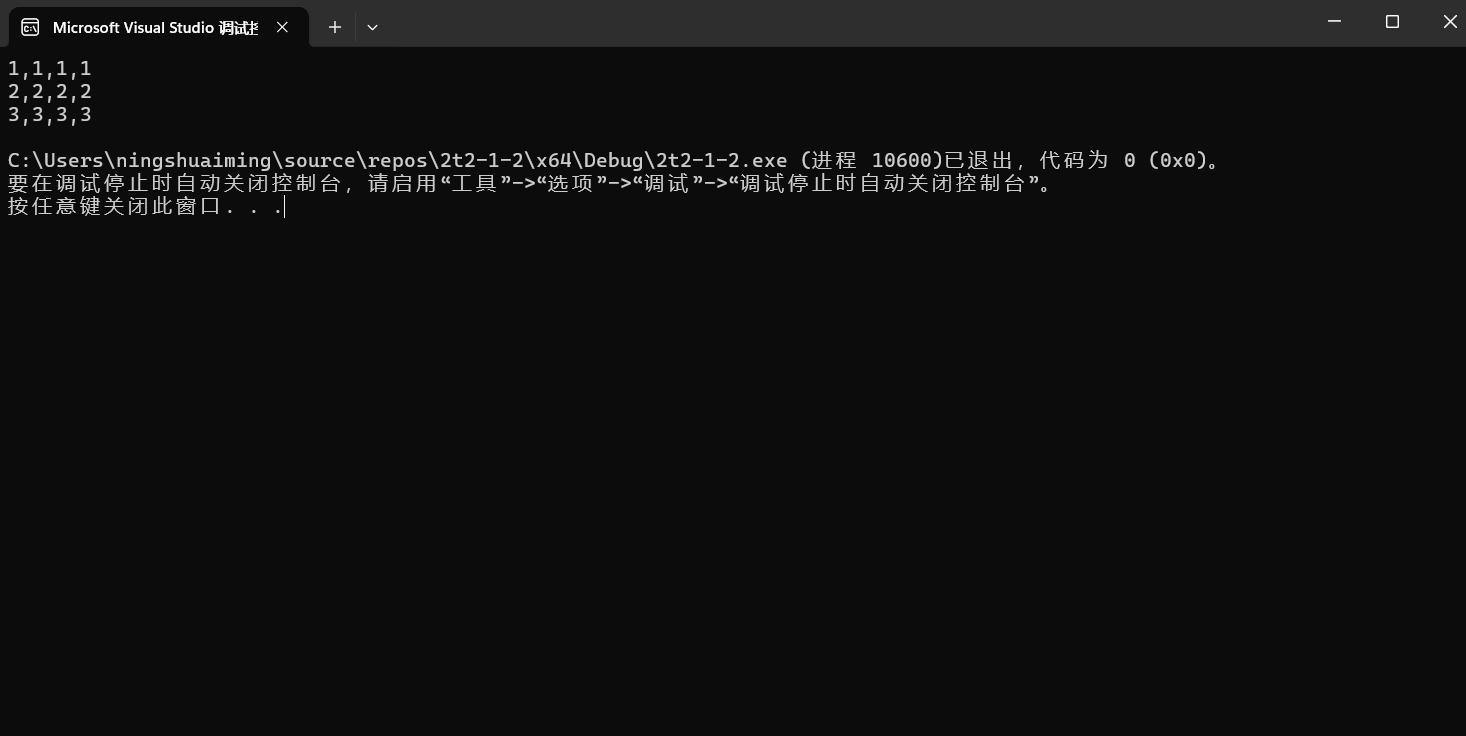
int\* p, i;

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

}

****

**(二)1. （3）.**

#include<iostream>

#include <stdio.h>

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);

}

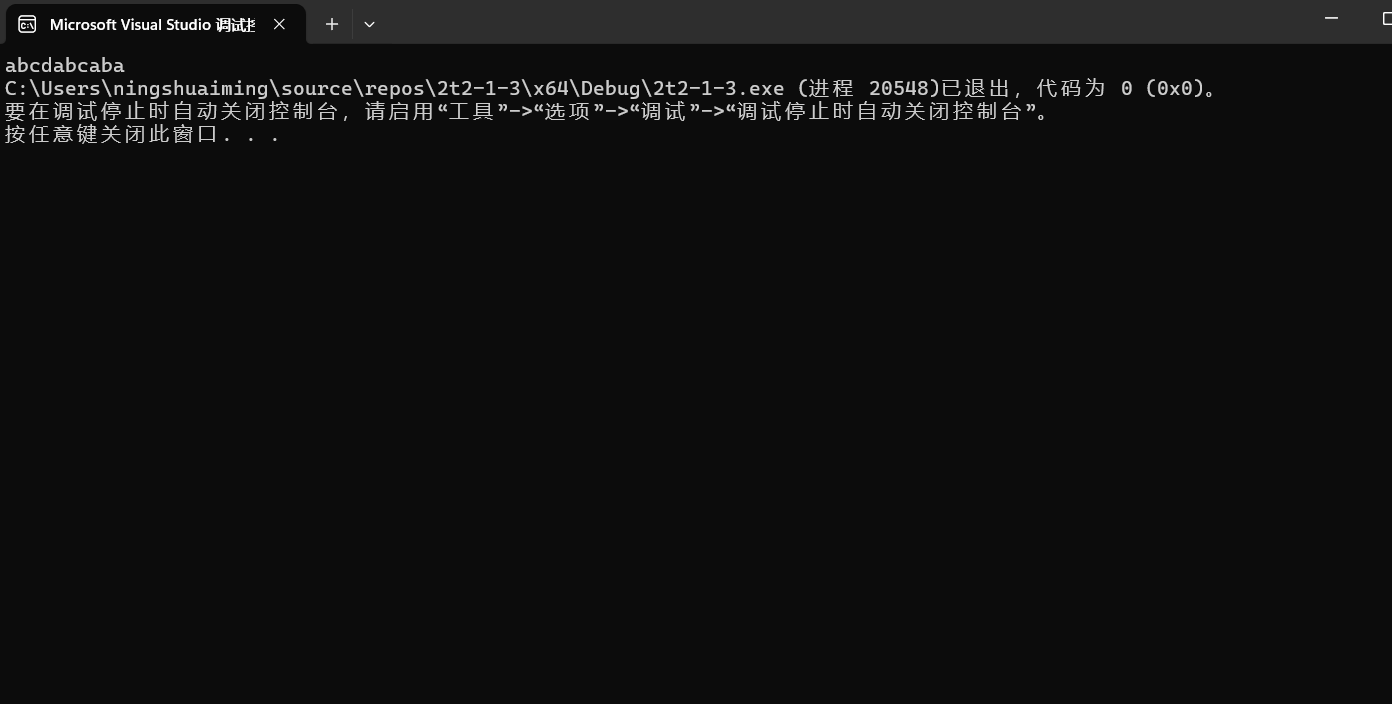
void main()

{

char st[] = "abcd";

f(st, 4);

}



**(二)1.（4）。**

解：原程序能保证p[0]输出1，但不能保证p[1]输出2；

理由如下： 原程序的f函数中list数组未进行动态内存分配，故其返回值返回给main函数中的p后，f执行完毕，list数组所占的内存会被释放。尽管在程序的帮助下这部分内存被最后保留一次，使得p[0]的访问值仍为正确的1，但这之后，这部分内存被彻底释放，再次进行访问p[1]便无法再输出得到正确值2了。

修改方案：利用new为list数组分配动态内存，使这部分内存能一直保存到最后delete释放，使得在进行p[0]、p[1]访问使能够输出正确的值。

修改后代码如下：

#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();

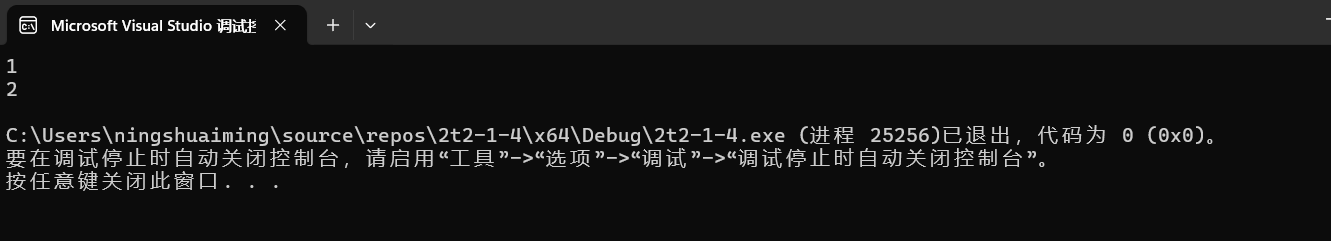
cout << p[0] << endl;

cout << p[1] << endl;

delete[] p;

}

运行结果：



**（二）2. （1）.**

#include<iostream>

#include<vector> //调用<vector>头文件，便于在程序中创建vector型数组，实现动态地对数组长度进行改变

using namespace std;

void indexOf(const char\* s1, const char \*s2) { //定义一个函数用来进行子串判断

const char\* re = strstr(s2, s1); //调用strstr（）函数，判断s1是否为s2的子串

if (re != nullptr) {

cout << "s1是s2的子串，s1在s2中的下标为：" << re - s2 << endl;

}

else {

cout << "s1不是s2的子串，返回值：" << -1 << endl;

}

}

int main() {

vector<char>s1;

vector<char>s2; //定义两个vector型的变量

char mid1, mid2;

cout << "本程序可以判断一个字符串s1是否为另一个字符串s2的字串" << endl;

cout << "请输入第一个字符串s1" << endl;

while (mid1 = cin.get()) {

if (mid1 != '\n') {

s1.push\_back(mid1);

}

else {

s1.push\_back('\0');

break;

}

} //输入字符串

cout << "请输入第二个字符串s2" << endl;

while (mid2 = cin.get()) {

if (mid2 != '\n') {

s2.push\_back(mid2);

}

else {

s2.push\_back('\0');

break;

}

} //输入字符串

int size1 = s1.size();

int size2 = s2.size();

char\* S1 = new char[size1];

char\* S2 = new char[size2];

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

S1[i] = s1[i];

}

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

S2[i] = s2[i];

} //将vector型的 数组转换为普通类型数组

indexOf(S1, S2); //调用函数进行子串判断

delete[]S1, S2; //释放动态内存

return 0;

}

**（二）2. （2）.**

#include <iostream>

#include <stdlib.h> //引入<stdlib.h>头文件，便于后续使用strtol（）函数将字符串形式表示的一个16进制数转换为10进制数

#include <vector>

using namespace std;

int parseHex(const char\* const hexString) {

char\* endfin;

long calc = strtol(hexString, &endfin,， 16); //使用strtol（）函数将字符串形式表示的一个16进制数转换为10进制数

if (\*endfin != '\0') { //如果\*endfin != '\0'，说明输入的字符串中有不属于16进制数的字符，无法进行转换，输入非法

cout << "your input string is illegal" << endl;

return -1;

}

cout << "转换结果为" << calc << endl;

return 0;

}

int main() {

cout << "请输入一个正确的字符串，本程序可以将字符串形式表示的一个16进制数转换为10进制数并输出" << endl;

vector <char> inp; //创建vector型数组

char mid;

while (mid = cin.get()) {

if (mid != '\n') {

inp.push\_back(mid);

}

else {

inp.push\_back('\0');

break;

}

} //输入 数组

int size = inp.size(); //利用size（）函数获取数组长度

char\* finbef = new char[size];

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

finbef[i] = inp[i];

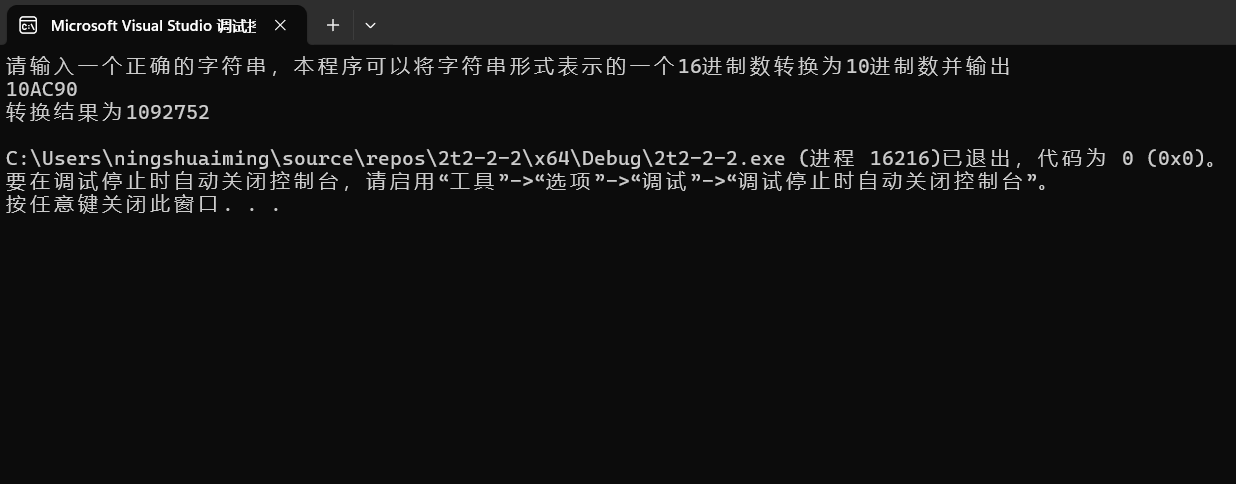
} //将vector型数组转化为普通型数组

const char\* const fin = finbef; //将数组中的元素转化为const型，便于调用函数

parseHex(fin); //调用函数

return 0;

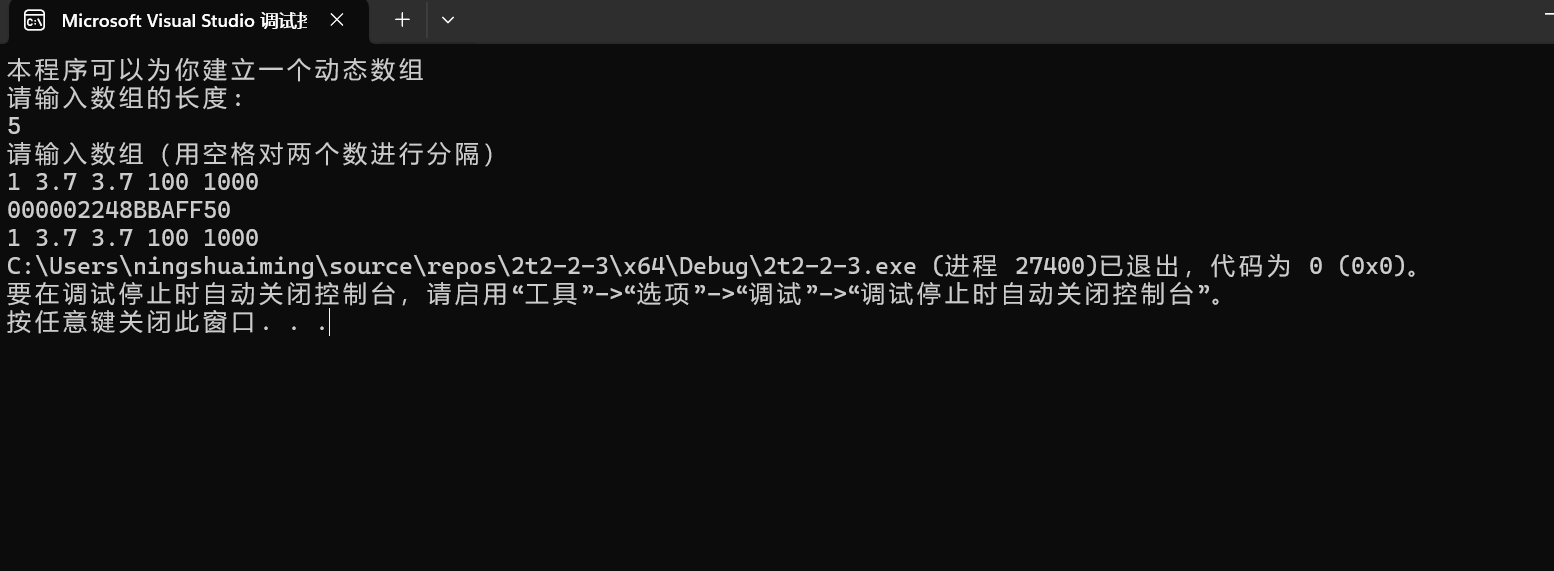
}

****

**（二）2. （3）.**

按要求建立动态数组后，对其动态调试观察指针及指针指向的内容的情况如下：





**最后本题的完整代码及结果如下 ：**

#include <iostream>

using namespace std;

double\* ord( double list[],int Size) { //定义函数，将传入的数组进行由大到小排序

bool help = false;

do {

help = true; //初始值设为false

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

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

double swi = list[i];

list[i] = list[i + 1];

list[i + 1] = swi;

help = false;

} //一旦发生一次调换，调换的两个数之前的数就有了需要调换的嫌疑，需要把changed值设为true，在此执行检索调换循环

}

} while (help == false);//函数执行条件：changed值为true

double\* ret = new double[Size]; //创建一个动态数组用来储存排序好的元素，便于在主函数中返回该数组首地址后，仍能访问该数组中的元素

for (int i = 0; i < Size; i++) {

ret[i] = list[i];

}

return ret;

}

int main() {

int size;

cout << "本程序可以为你建立一个动态数组，由小到大排序后用指针方式输出数组元素" << endl;

cout << "请输入数组的长度:"<<endl;

cin >> size;

cout << "请输入数组（用空格对两个数进行分隔）" << endl;

double\* inp = new double[size]; //创建动态数组

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

cin >> inp[i];

} //输入

double\* outp = ord(inp, size); //调用函数

cout << "排序后结果为：";

for (int n = 0; n < size; n++) {

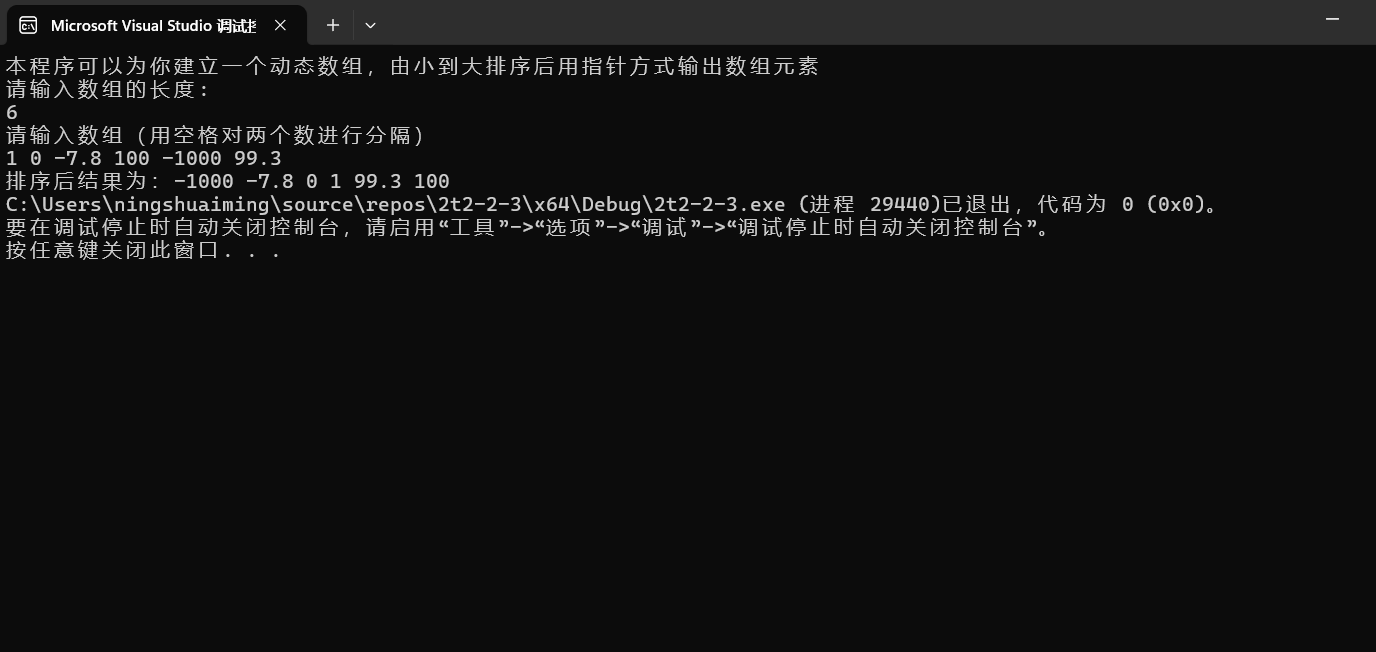
cout << \*outp<<'\ ';

outp++;

} //输出

delete[]inp, outp; //释放动态内存

return 0;

****}

**四、遇到的问题与解决方法**

**1.不熟悉printf（）函数的用法。**

**解决方法 ：通过查询，了解到了printf（）的具体用法 。**

**printf()函数是 C 语言中的标准输出函数，在 C++ 中也可以使用。它定义在<stdio.h>头文件中（在 C++ 中也可以使用<cstdio>头文件，这是 C 标准库头文件的 C++ 兼容版本）。其主要功能是将格式化后的文本输出到标准输出设备（通常是控制台）。**

**基本语法：int printf(const char\* format,...);。其中，format是一个包含格式控制字符串的指针，...表示可变参数列表，用于传递要按照格式输出的具体数据。**

**2.不清楚如何在不对数组长度进行确定的情况下不断地在数组中添加元素。**

**解决方法：通过查询和同学间的讨论了解到，可以包含头文件<vector>,**

**利用语法“数组类型+<vector>+数组名”可以构建一个vector类型的动态数组。当 有新元素需要添加时，只需要利用“数组名.push\_back(新元素)”，即可将新元素添加到数组中，还可以用“数组名.size()”函数返回 此时该数组的元素个数即长度。但是注意：vector类型的数组不能直接作为实参传递给非vector类型数组的形参。需要将vector型数组的值重新一一赋值给另一个普通数组，相当于将vector型转化为了普通型，之后才能进行传参。**

**3.不清楚如何用简洁的方式进行子串判断，同时实现“是子串，返回第一次匹配的下标，否则返回-1。**

**解决方式：通过查询 ，发现可以利用c++中的strstr()函数进行子串判断。具体用法如下：函数原型为：char \*result=strstr(const char \*haystack, const char \*needle);。其中haystack是主字符串，needle是要查找的子字符串。如果找到了子字符串，将返回s1在s2中首次出现处s2元素的地址给result，可以通过(result - haystack)计算出子字符串在主字符串中的起始位置并输出；如果没找到，strstr()函数返回NULL。**

**4.不清楚如何将以字符串形式表示的一个16进制数转换为10进制数**

**解决方式：通过与同学们讨论发现可以引入<stdlib.h>头文件，并在函数中使用strtol（）函数将字符串形式表示的一个16进制数转换为10进制数。具体用法如下：**

**char \*endptr**

**long int a=strtol(const char \*str, &endptr, int base);**

**str：指向要转换为长整数的字符串。**

**endptr：一个字符型指针，若字符串全部转换成功，会把最后’\0’的 地址返回给它，若存在无法转换的字符将把该字符的地址返还给它。**

**base：基数，必须介于2和36之间（包含2和36），或者是特殊值0。基数代表采用的进制方式，例如base值为2就代表采用二进制。**

**该函数的会返还转换结果给a。**

**五、体会**

**1. 灵活衔接利用控制结构实现目标功能**

在这次实验中，我深刻体会到了对if、for、while等控制结构的灵活应用是实现复杂功能的关键。例如，在检索数组中的重复元素时，我巧妙地运用了for循环遍历数组，同时利用if语句判断当前元素是否已出现过（通过辅助的数据结构如vector或set进行记录）。这种结合使用的方式不仅提高了代码的可读性，也显著提升了程序的执行效率。

另外，在实现起泡排序时，我通过for循环控制排序的轮数，并在每一轮中使用嵌套的for循环和if语句进行相邻元素的比较和交换。这种控制结构的灵活运用，使得排序算法得以高效且准确地实现。

**2. 积极查询与学习新知识**

在实验过程中，我遇到了许多用以往代码难以直接实现或实现起来较为复杂的任务。面对这些挑战，我积极利用网络资源进行查询，找到了许多实用的解决方案和新的C++语法知识。

例如，在进行子串判断时，我最初尝试使用简单的逐个元素判断，但发现效率较低。通过网上查询，我学习了strstr（）这种高效的函数，并成功地将其应用于子串判断程序中，大大提高了程序的性能。

同时，我也意识到，在编程过程中，不断学习新的知识和语法是非常重要的。这不仅可以帮助我们解决当前的问题，还能拓宽我们的视野，提高我们的编程能力和解决问题的能力。

总的来说，这次C++实验让我更加深入地理解了控制结构的重要性，并教会了我如何积极查询和学习新知识来应对编程中的挑战。我相信，这些宝贵的经验和体会将对我未来的编程学习和工作产生深远的影响。