**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软工2303班

学 号： 8210232227

姓 名： 张程熙

**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？

我认为函数的返回值在默认情况下和函数类型保持一致是为了防止未定义行为，增强代码的运行效率，同时增加代码的可读性，还有返回值在转换的过程中也可能产生丢失，如果希望更改返回的值的类型我认为也是可以的。

2. 本实验中主函数调用函数时采用的是何种传递方式？

有按引用调用，有通过返回值调用，有按值调用

**四、算法分析，程序结果**

（1）1.

#include<iostream>

using namespace std;

int gys(int m, int n) {

int temp,temp1;

if (m < n) {

temp = m;

m = n;

n = temp;

}

while (n != 0) {

temp1 = n;

n = m % n;

m = temp1;

}

return m;

}

int main() {

int m, n;

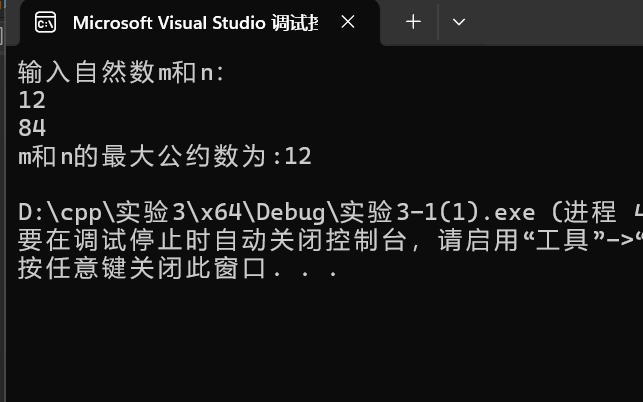
cout << "输入自然数m和n：" << endl;

cin >> m >> n;

cout << "m和n的最大公约数为:" << gys(m, n) << endl;

return 0;

}



2.

#include<iostream>

using namespace std;

int gys(int m, int n) {

int temp, temp1;

if (m < n) {

temp = m;

m = n;

n = temp;

}

while (n != 0) {

temp1 = n;

n = m % n;

m = temp1;

}

return m;

}

int gbs(int m, int n) {

int temp, temp1;

if (m < n) {

temp = m;

m = n;

n = temp;

}

int i = 1;

while (n \* i % m) {

i++;

}

return n \* i;

}

int main() {

int m, n;

cout << "输入自然数m和n：" << endl;

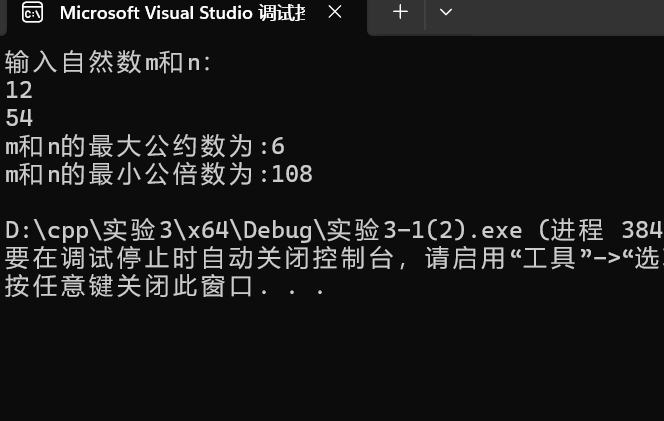
cin >> m >> n;

cout << "m和n的最大公约数为:" << gys(m, n) << endl;

cout << "m和n的最小公倍数为:" << gbs(m, n) << endl;

return 0;

}



（2）#include<iostream>

using namespace std;

bool is\_prime(int num) {

if (num <= 1 ) {

return false;

}

if (num <= 3) {

return true;

}

if (num % 2 == 0 || num % 3 == 0) {

return false;

}

for (int i = 5; i \* i <= num; i += 6) {

if (num % i == 0 || num % (i + 2) == 0) {

return false;

}

}

return true;

}

int main() {

int i = 1;

int j = 2;

while (i <= 200) {

while (is\_prime(j)) {

cout << j << " ";

if (i % 10 == 0) {

cout << endl;

}

i++;

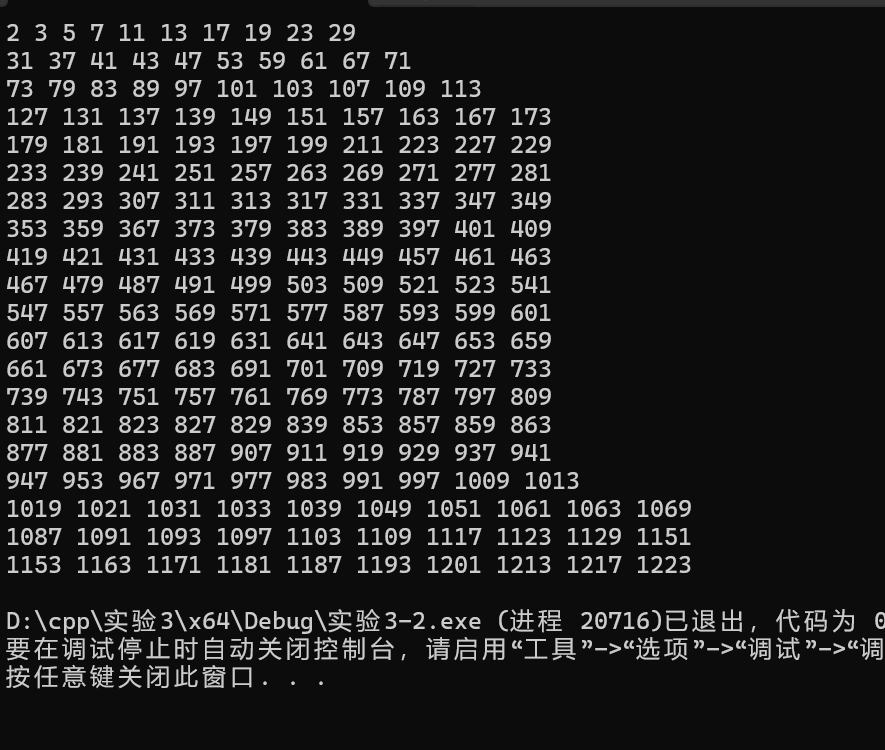
j++;

}

j++;

}

}



（3）头文件

#ifndef MY\_TEMPERATURE\_H

#define MY\_TEMPERATURE\_H

double celsius\_to\_fah(double cel);

double fahrenheit\_to\_cels(double fah);

#endif // MY\_TEMPERATURE\_H

源文件

#include "mytemperature.h"

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

double cel, fah;

std::cout << std::fixed;

std::cout << " Celsius Fahrenheit | Fahrenheit Celsius" << std::endl;

std::cout << std::setprecision(1) << " 40.0 " << celsius\_to\_fah(40.0)

<< " | "

<< std::setprecision(2) << "120.0 " << fahrenheit\_to\_cels(120.0 )<< std::endl;

std::cout << std::setprecision(1) << " 39.0 " << celsius\_to\_fah(39.0)

<< " | "

<< std::setprecision(2) << "110.0 " << fahrenheit\_to\_cels(110.0) << std::endl;

std::cout << std::setprecision(1) << " …… " << "……"

<< " | "

<< std::setprecision(2) << "…… " << "……" << std::endl;

std::cout << std::setprecision(1) << " 31.0 " << celsius\_to\_fah(31.0)

<< " | "

<< std::setprecision(2) << "30.0 " << fahrenheit\_to\_cels(30.0) << std::endl;

return 0;

}

#include "mytemperature.h"

double celsius\_to\_fah(double cel) {

double fah = (cel \* 9.0 / 5.0) + 32;

return fah;

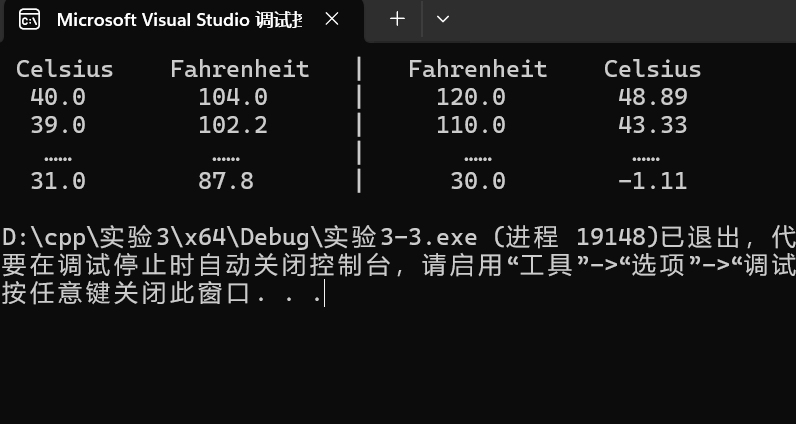
}

double fahrenheit\_to\_cels(double fah) {

double cel = (fah - 32) \* 5.0 / 9.0;

return cel;

}



（5）#include<iostream>

using namespace std;

int peach(int n) {

if (n == 1) {

return 1;

}

else {

return (peach(n - 1) + 1) \* 2;

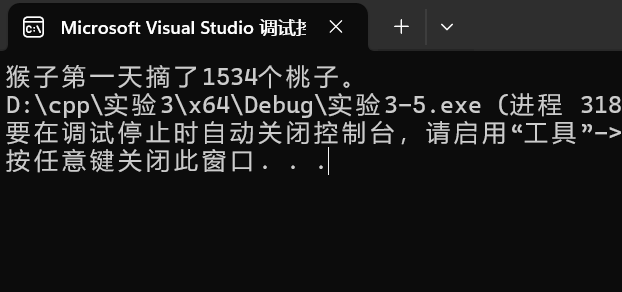
}

}

int main() {

cout << "猴子第一天摘了" << peach(10) << "个桃子。";

}



**五、遇到的问题与解决方法**

实验三的前面两题我没有什么问题，第三题的输出我个人觉得有点难度，首先这个格式如果是靠一个一个输出来做的话，最大的难度在于如何保证输入和输出在同一行，我也参考了一些材料，后面觉得难度有点大，如果做这个的话可能都偏离实验内容了，于是就采用最正常的输出方式，第五题递归我之前做算法的题的时候遇见的比较多，所以也是比较轻松就完成了。

**六、体会**

我做了这部分实验之后，对于函数的创建，还有这门语言的掌握程度提高了，写代码的速度也提升了，也了解了头文件是一个什么东西，以及更加深入的了解了全局变量和局部变量的区别，相信在之后的实验以及之后的编程中会有更大的进步和更多知识了解。

**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;

}

原程序不能保证 p[0] 输出 1 和 p[1] 输出 2，因为函数 f 返回的 list 是一个局部数组，其在函数返回时会被销毁。返回指向这种局部数组的指针会导致未定义行为。

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

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**

**1.**

(1) #include<iostream>

using namespace std;

int pd(int a[],int b[]) {

bool x;

int y = 0;

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

x = true;

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

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

x = false;

}

if (x) {

b[y] = a[i];

y++;

}

}

for (int k = 0; k < y; k++) {

cout << b[k] << " ";

}

return 0;

}

int main() {

int a[10];

int b[10];

cout << "Enter ten number:";

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

cin >> a[i];

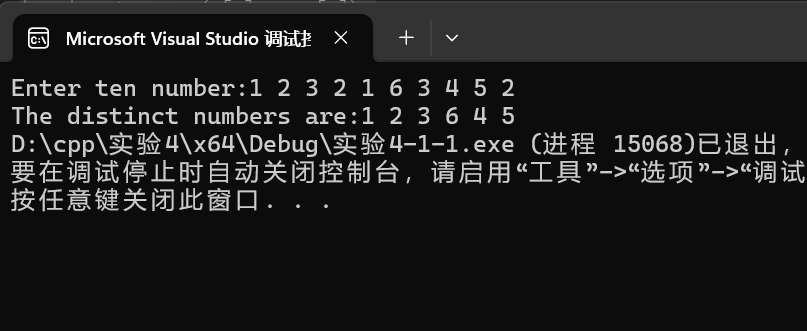
}

cout << "The distinct numbers are:";

pd(a,b);

return 0;

}



(2) #include<iostream>

using namespace std;

void bubble\_Sort(double list[], int n) {

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

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

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

int temp = list[j];

list[j] = list[j + 1];

list[j + 1] = temp;

}

}

}

}

int main() {

int n = 10;

double list[10];

cout << "输入十个数字：" << endl;

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

cin >> list[i];

}

bubble\_Sort(list, n);

cout << "排序后的结果为:" << endl;

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

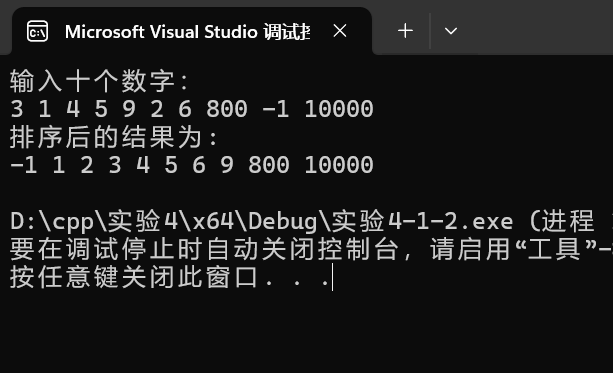
cout << list[i] << " ";

}

cout << endl;

return 0;

}



(3) #include<iostream>

using namespace std;

bool OC[100];

int n = 100;

int main() {

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

if (j % i == 0)

OC[j] = !OC[j];

}

}

cout << "开着的柜子的号码有：";

for (int i = 1; i <= n; i++) {

if (OC[i]) {

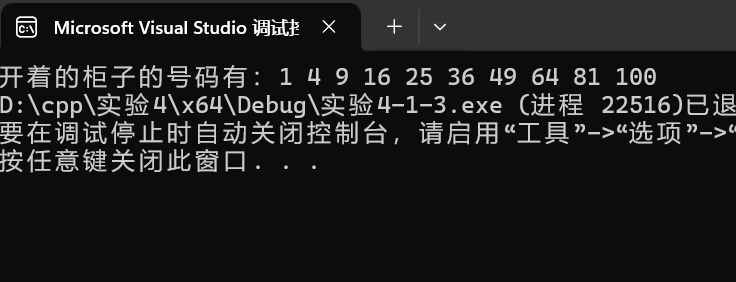
cout << i << " ";

}

}

return 0;

}



(4) #include<iostream>

using namespace std;

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

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

list3[i] = list1[i];

}

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

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

}

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

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

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

int temp = list3[h];

list3[h] = list3[h + 1];

list3[h + 1] = temp;

}

}

}

}

int main() {

int m, n;

cout << "Enter list1:";

cin >> m;

int\* A = new int[m];

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

cin >> A[i];

}

cout << "Enter list2:";

cin >> n;

int\* B = new int[n];

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

cin >> B[j];

}

int\* C = new int[m + n];

merge(A, m, B, n, C);

cout << "The merged list is ";

for (int k = 0; k < m + n; k++) {

cout<<C[k]<<" ";

}

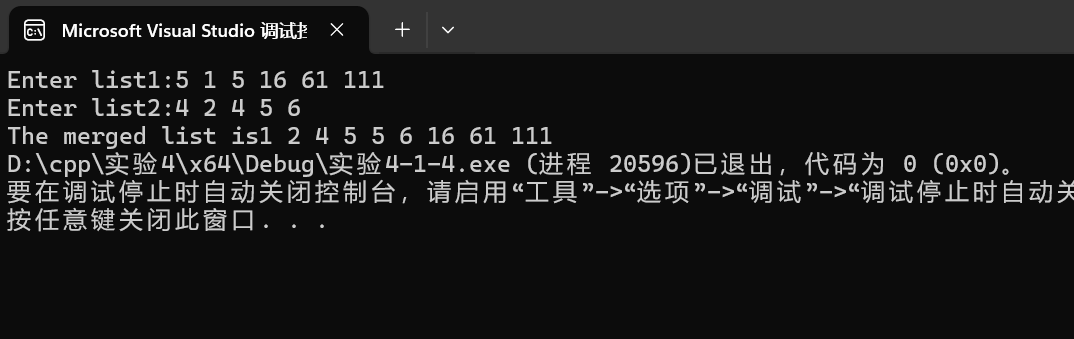
delete[]A;

delete[]B;

delete[]C;

return 0;

}



(5) #include <iostream>

#include <cstring>

using namespace std;

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

int len1 = strlen(s1);

int len2 = strlen(s2);

if (len1 == 0 || len2 == 0 || len2 < len1) {

return -1;

}

for (int i = 0; i < len2 - len1; i++) {

bool match = true;

for (int j = 0; j < len1; j++) {

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

match = false;

break;

}

}

if (match) {

return i;

}

}

return -1;

}

int main() {

const int MAX = 100;

char s1[MAX], s2[MAX];

cout << "Enter the first string: ";

cin.getline(s1, MAX);

cout << "Enter the second string: ";

cin.getline(s2, MAX);

int index = indexOf(s1, s2);

if (index != -1) {

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

}

else {

cout << "indexOf(“" << s1 << "”, “" << s2 << "”) is -1" << endl;

}

return 0;

}





(6) #include<iostream>

#include<cstring>

using namespace std;

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

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

counts[i] = 0;

}

int len = strlen(s);

for (int j = 0; j < len; j++) {

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

if (s[j] == 'a' + k || s[j] == 'A' + k) {

counts[k]++;

}

}

}

}

int main() {

int counts[26];

char s[100];

cout << "Enter a string：";

cin.getline (s,100);

count(s, counts);

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

char x = 'a' + i;

if (counts[i] != 0) {

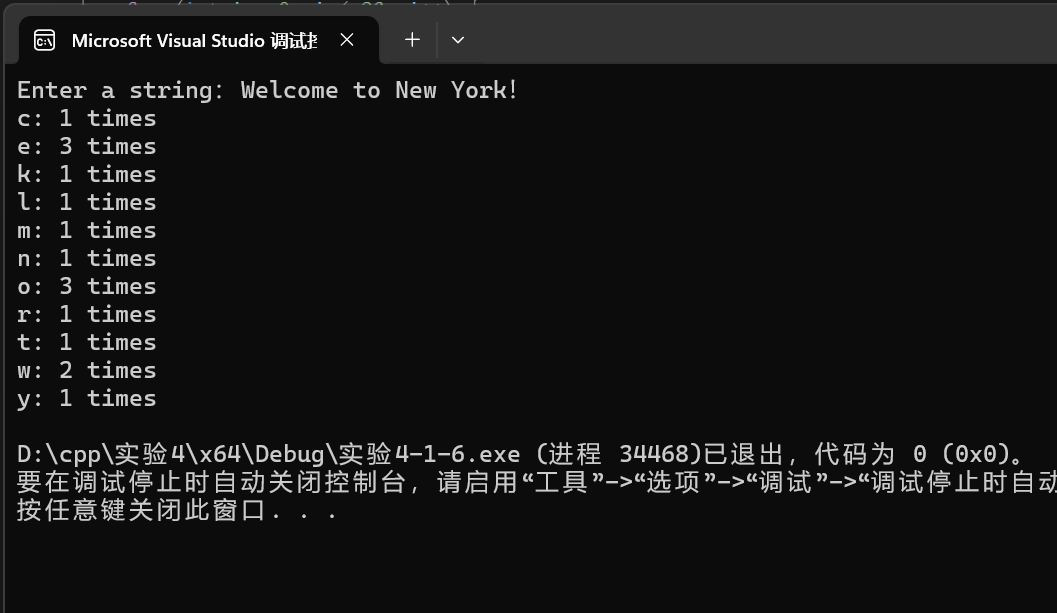
cout << x << ": " << counts[i] << " times" << endl;

}

}

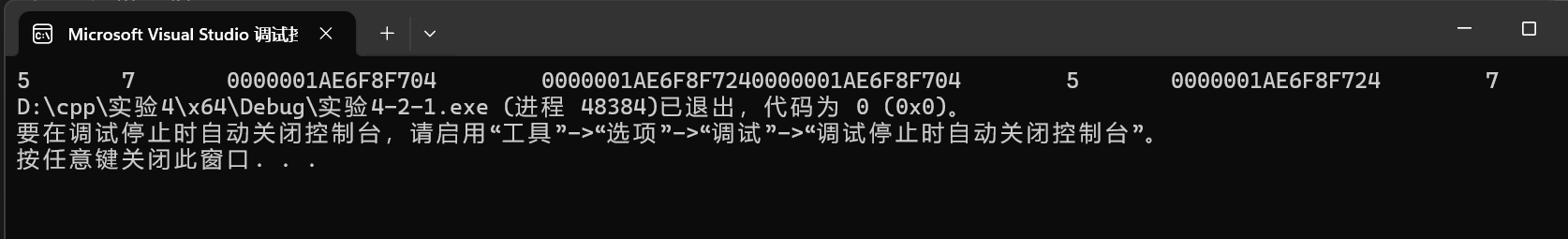
return 0;

}



**2.**

(1)



(2)



(3) #include<iostream>

using namespace std;

void f(char\* st, int i)

{

st[i] = '\0';

cout << st;

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

}

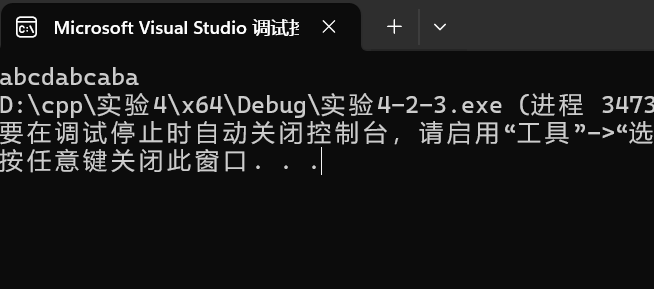
void main()

{

char st[] = "abcd";

f(st, 4);

}



(4) #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()

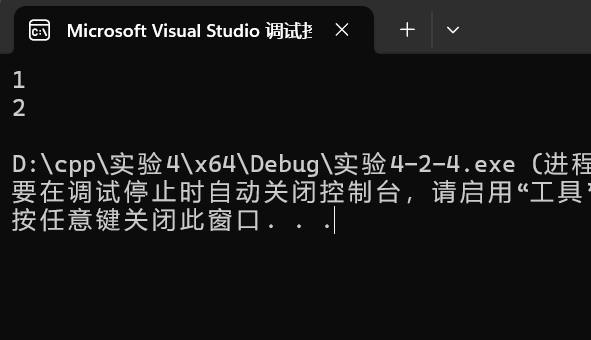
{

int\* p = f();

cout << p[0] << endl;

cout << p[1] << endl;

}



(5) #include<iostream>

#include <cstring>

using namespace std;

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

int len1 = strlen(s1);

int len2 = strlen(s2);

if (len1 == 0 || len2 == 0 || len1 > len2) {

return -1;

}

for (int i = 0; i <= len2 - len1; i++) {

bool match = true;

for (int j = 0; j < len1; j++) {

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

match = false;

break;

}

}

if (match) {

return i;

}

}

return -1;

}

int main() {

const int MAX = 100;

char s1[MAX], s2[MAX];

cout << "Enter the first string: ";

cin.getline(s1, MAX);

cout << "Enter the second string: ";

cin.getline(s2, MAX);

int index = indexof(s1, s2);

if (index != -1) {

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

}

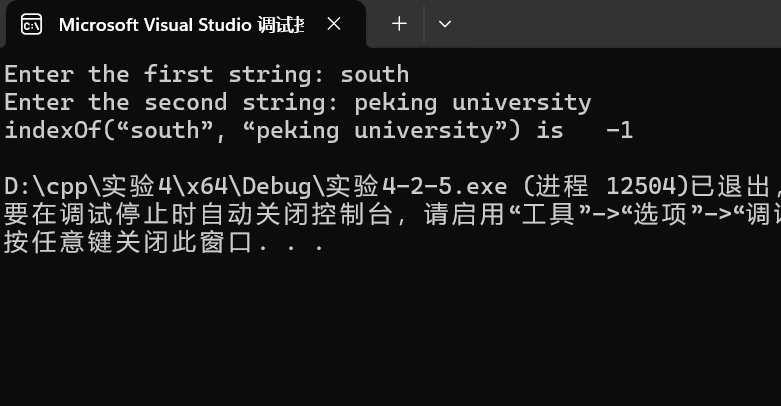
else {

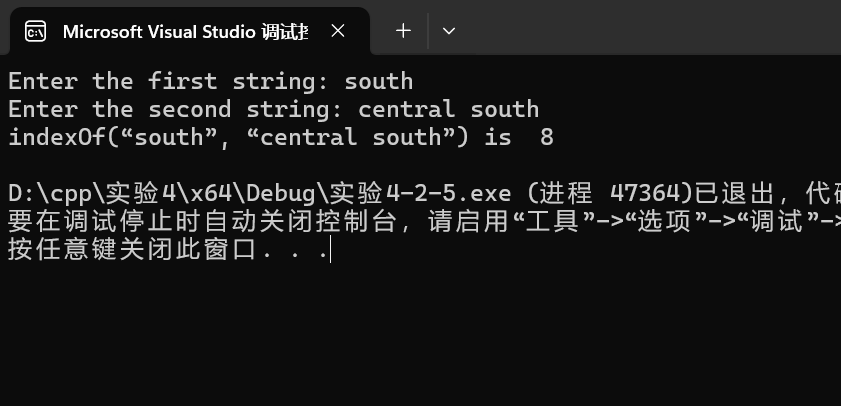
cout << "indexOf(“" << s1 << "”, “" << s2 << "”) is -1" << endl;

}

return 0;

}





(6) #include<iostream>

#include <cstring>

#include<cmath>

using namespace std;

int parseHex(const char\* const hexString) {

int len = strlen(hexString);

int\* copy = new int[len];

int sum = 0;

for (int i = len - 1; i >= 0; i--) {

if (hexString[i] >= '0' && hexString[i] <= '9') {

copy[len - i - 1] = hexString[i] - '0';

}

else if (hexString[i] >= 'a' && hexString[i] <= 'f') {

copy[len - i - 1] = hexString[i] - 'a' + 10;

}

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

copy[len - i - 1] = hexString[i] - 'A' + 10;

}

else {

cout << "输入包含非法字符！" << endl;

}

}

for (int k = 0; k < len; k++) {

sum += copy[k] \* pow(16, k);

}

delete[]copy;

return sum;

}

int main() {

char s[100];

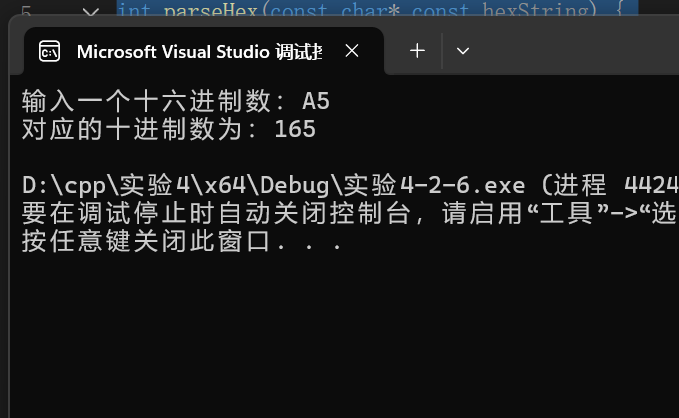
cout << "输入一个十六进制数：";

cin >> s;

int n = parseHex(s);

cout << "对应的十进制数为：" << n << endl;

}



(7) #include<iostream>

using namespace std;

void bubble\_Sort(int\* list, int n) {

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

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

if (\*(list + j) > \*(list + j + 1)) {

int temp = \*(list + j);

\*(list + j) = \*(list + j + 1);

\*(list + j + 1) = temp;

}

}

}

}

int main() {

int n;

cout << "请输入数组的元素个数: ";

cin >> n;

int\* list = new int[n];

cout << "请输入数组的元素: ";

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

cin >> \*(list + i);

}

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

cout << "指针地址: " << (list + i) << " 对应指向的内容: " << \*(list + i) << endl;

}

bubble\_Sort(list, n);

cout << "排序后的数组: ";

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

cout << \*(list + i) << " ";

}

cout << endl;

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

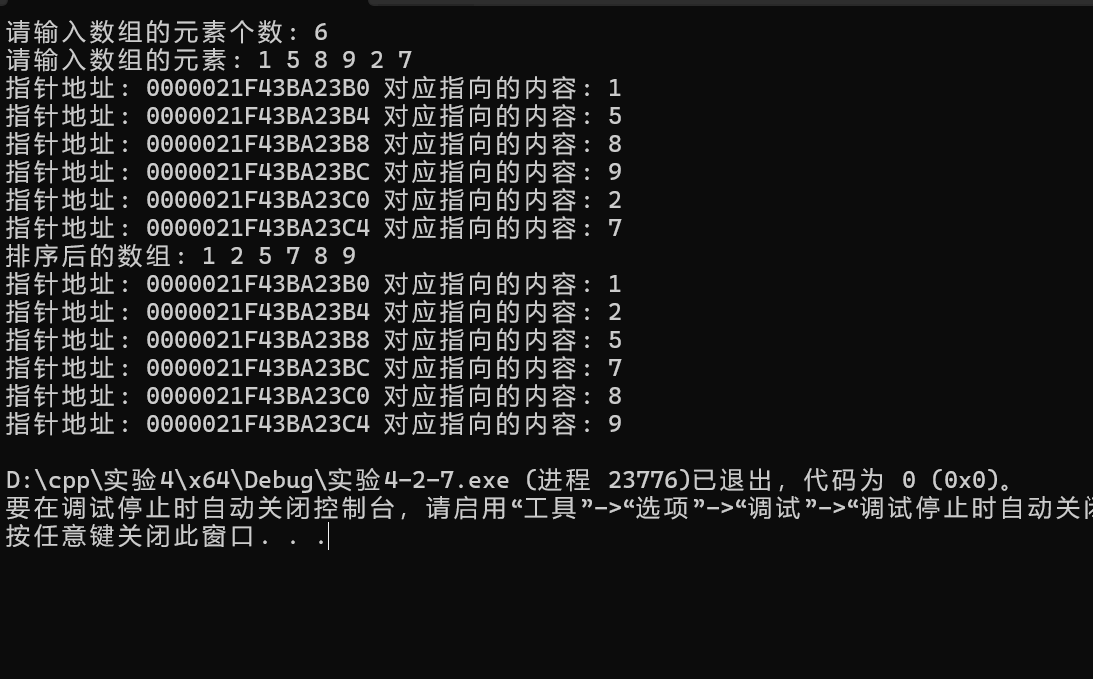
cout << "指针地址: " << (list + i) << " 对应指向的内容: " << \*(list + i) << endl;

}

delete[] list;

return 0;

}



**四、遇到的问题与解决方法**

第一题我在设计循环的时候保存是从后往前保存，很快发现是设计的问题及时修改了。第四题一开始我没打算用动态数组，发现不好做，最后参考了一些网上的资料，在使用动态数组的情况下又优化了结构，第五题我正常运行后产生错误答案，检查之后我发现是因为循环的终止点找错了，导致某些特定位置不能发现是子串，指针部分的第六题我本来打算使用和上面统计字符串一样的方式，但发现字符串不能直接转化成整型，于是采用if来进行判断。

**五、体会**

这个实验首先让我更深刻的了解了指针的作用，通过指向地址来拿值，另一个就是创建动态数组，尤其是创建完需要记得删除，还有一般是在函数里进行创建，如果在函数外创建可能会有溢出，这里面的实验很多我都做过，再一次的实验让我对于它们的逻辑有了深刻地认识，有的时候不了解的程序在重新设计一遍可能有了新的感悟，比如一些细节的地方是可以优化的，比如循环找到了对应的值就可以直接break，没有必要继续循环降低运行效率。