**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： 8209240214

姓 名： 程世勋

**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？

2. 本实验中主函数调用函数时采用的是何种传递方式？

1因为函数类型决定了返回值的类型，所以函数中返回的值为什么与函数类型一致。

2.本实验1中调用函数采用的是引用传递 实验2、3、5中采用的是值传递。

**四、算法分析，程序结果**

**1.**

#include<iostream>

using namespace std;

int ys(int& m, int& n)//定义最大公因数函数

{

int Y;//定义最大公因数Y

for (int i = 1;i <= m && i <= n;i++)

{

if (m % i == 0 && n % i == 0)

{

Y = i;//将目前最大的i覆盖赋值到临时变量Y

}

}

return Y;//返回最大公因数Y

}

int bs(int& m, int& n)//定义最小公倍数函数

{

int temp;//定义临时变量

for (int i = m; i <= (m \* n); i++)

{

if (i % m == 0 && i % n == 0)//i分别取余a,b,如果结果为0 就是a,b的公倍数

{

temp = i;//将公倍数赋值到临时变量

break;//找到的第一个公倍数就是最小公倍数 跳出循环

}

}

return temp;//返回临时变量

}

int main()

{

int m, n;//定义m,n

cout << "请输入m的值:";//提示输入m的值

cin >> m;//输入m

cout << "请输入n的值:";//提示输入n的值

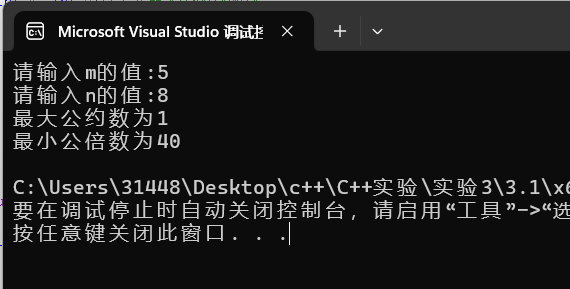
cin >> n;//输入n

cout << "最大公约数为" << ys(m, n) << endl;

cout << "最小公倍数为" << bs(m, n) << endl;

return 0;

}

****

**2.**

#include<iostream>

using namespace std;

bool is\_prime(int num)//定义判断素数的函数

{

int j = 0;//定义j=0

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

{

if (num % i == 0)

{

j++;//每有一个因数,j就+1

}

}

if (j == 2)//如果j=2,即num只有两个因数,就是素数,返回true

{

return true;

}

else

{

return false;

}

}

int main()

{

bool prime;//定义prime用于接收函数返回值

int n = 1;//定义n用来枚举

for (int j = 0; ;)

{

n++;//n+1(从2开始枚举)

prime = is\_prime(n);//接收函数返回值

if (prime == true)//如果是素数

{

cout << n << ' ';//输出该数字

j++;//计数+1

if (j % 10 == 0)//如果已经输出了10个数字

{

cout << endl;//换行

}

}

if (j == 200)//如果输出了200个数字

{

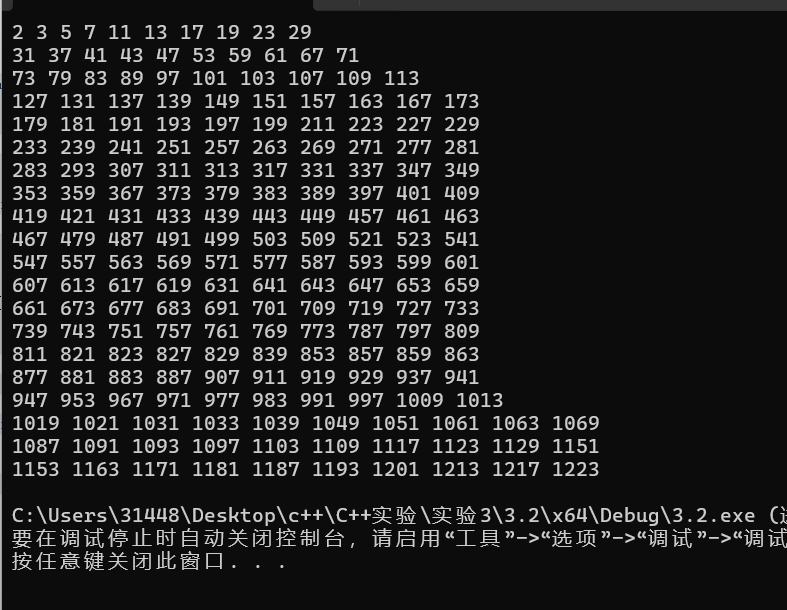
break;

}

}

return 0;

}



**3.**

#include<iostream>

#include <iomanip>

#include "mytemperature.h";

using namespace std;

int main()

{

double cel, fah;//定义摄氏度和华氏度

cout << "请输入摄氏温度";

cin >> cel;

printf("%.1f %.1f\n", cel, celsius\_to\_fah(cel));

cout << "请输入华氏温度 ";

cin >> fah;

printf("%.1f %.2f\n", fah, fahrenheit\_to\_cels(fah));

return 0;

}



**5.**

#include<iostream>

using namespace std;

int peaches(int day)

{

if (day == 10)

{

return 1;// 第10天只剩下1个桃子

}

else

{

return (peaches(day + 1) + 1) \* 2;// 根据递推公式，计算第day-1天的桃子数

}

}

int main()

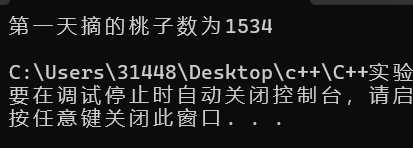
{

int n = 1;

cout << "第一天摘的桃子数为" << peaches(n) << endl;

return 0;

}



**五、遇到的问题与解决方法**

**实验3**

1.在编写代码中，发现输出的结果与效果图不一致，不是两位小数的形式。同时，对如何输出2为小数的方式感到困惑，于是查阅书籍，上网查阅资料，对代码进行了修改。

2.不知道怎么分为3个文件进行编写，但查阅书籍后对格式有所了解后就解决了问题。

**六、体会**

1.写递归函数时，要正确地表达函数，并且对情况进行分类，分为开始和n与n-1的递归，也不能忘了输出。

2.实验3分为3个文件的方式是第一次接触，要多多练习，增加熟练度。开始对如何编写代码有点手忙脚乱，但细心发现问题后就能够解决。

**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**

**一数组**

1.

#include <iostream>

using namespace std;

bool contains(int arr[], int size, int value) //定义一个contains函数

{

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

{

if (arr[i] == value) //如果存在相同数，它返回false

{

return false;

}

}

return true;//如果不存在相同数，它返回true

}

int main() {

int numbers[10]; // 存储输入的数

int uniqueNumbers[20]; // 存储不同的数

int uniqueCount = 0; // 不同数的计数

cout << "请输入10个数：" << endl;

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

{

cin >> numbers[i];

if (contains(uniqueNumbers, uniqueCount, numbers[i])) //如果不存在相同数，元素+1

{

uniqueNumbers[uniqueCount++] = numbers[i];

}

}

cout << "不同的数有：" << endl;

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

{

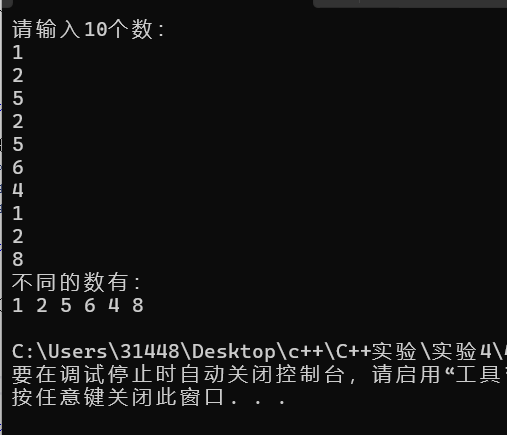
cout << uniqueNumbers[i] << " ";//输出数组中的元素

}

cout << endl;

return 0;

}



2.#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++)//比较轮数:比较一次 可以比出 一个最大值 要将10个数排序 仅需比出9次

{

for (int j = 0; j < len - 1 - i; j++)//每次比较的回数为len-i-1

{

if (arr[j] > arr[j + 1])//如果前一项大于后一项 调换前后两项

{

int temp = arr[j];

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++)

{

cin >> arr1[i];//输入数组中的各元素

}

int num = (sizeof(arr1) / sizeof(arr1[0]));//计算数组长度

bubbleSort(arr1, num);//调用函数

cout << "排序后的数组为" << endl;//提示排序结果

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

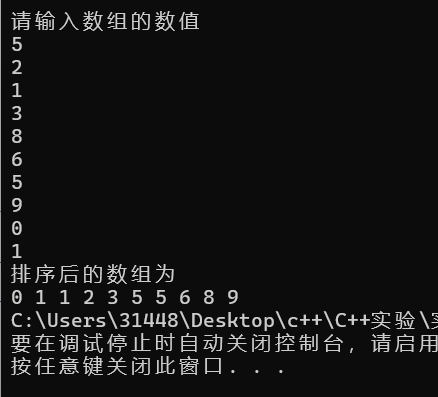
{

cout << arr1[i] << ' ';//输出数组

}

return 0;

}



3.

#include<iostream>

using namespace std;

void change\_state(bool\* arr, int num1) // 定义改变状态函数

{

for (int i = num1 - 1; i < 100; i += num1) // 从第i项开始，步长为num1

{

arr[i] = !arr[i]; // 第i项取反

}

}

int main()

{

bool arr[100];//定义一个布尔类型的数组

for (int i = 0; i < 100; i++)//第一个学生打开所有的存物柜，将数组的所有值赋为true(开)

{

arr[i] = true;

}

for (int i = 2; i <=100; i++)//从学生2到学生100改变状态

{

change\_state(arr, i);//调用改变状态函数

}

cout << "开着的门有:" << endl;

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

{

if (arr[i] == true)//如果开着

{

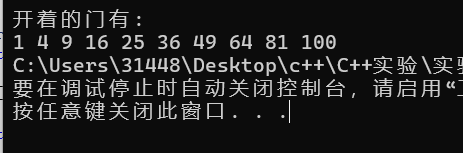
cout << (i + 1) << ' ';//打印编号

}

}

return 0;

}



4.

#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++) // 对list3赋值

{

if (i < size1) // 如果i小于list1的长度，将list1中的数值依次赋给list3

{

list3[i] = list1[i];// i超出了list1的长度，将list2中的数值依次赋给list3

}

else

{

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] << ' '; // 输出合并后的结果

}

cout << endl;

}

int main()

{

int size1, size2; // 定义数组长度

cout << "请输入list1的长度:";

cin >> size1; // 输入size1

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

cout << "请输入list1的数值:"; // 提示输入list1数值

for (int i = 0; i < size1; i++) // 建立循环

{

cin >> list1[i]; // 输入数值

}

cout << "请输入list2的长度:";

cin >> size2; // 输入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:"; // 提示输入list2数值

for (int i = 0; i < size2; i++) // 建立循环

{

cin >> list2[i]; // 输入数值

}

int\* list3 = new int[size1 + size2]; // 使用new 定义数组list3

merge(list1, size1, list2, size2, list3); // 调用函数

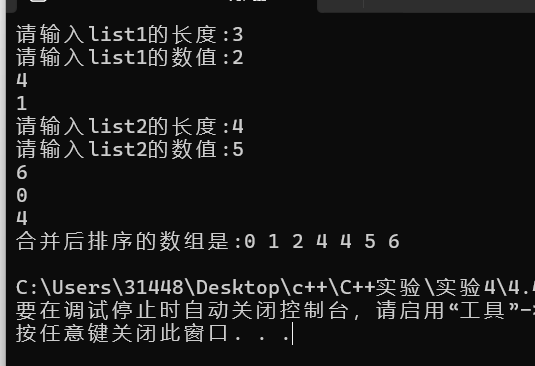
delete[] list1; // 释放list1内存

delete[] list2; // 释放list2内存

delete[] list3; // 释放list3内存

return 0;

}



5.

#include <iostream>

#include <cstring> // 用于strlen函数

using namespace std;

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

int len1 = strlen(s1);

int len2 = strlen(s2);

if (len1 > len2)

{

return -1; // 如果s1比s2长，不可能是子串

}

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

bool found = true;

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

{

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

{

found = false;

break;

}

}

if (found)

{

return i; // 找到匹配的子串，返回下标

}

}

return -1; // 没有找到匹配的子串，返回-1

}

int main() {

const int MAX\_LEN = 200; // 假设字符串最大长度为200

char s1[MAX\_LEN], s2[MAX\_LEN];

cout << "Enter the first string: ";

cin.getline(s1, MAX\_LEN);

cout << "Enter the second string: ";

cin.getline(s2, MAX\_LEN);

int result = indexOf(s1, s2);

if (result != -1) {

cout << "indexOf(\"" << s1 << "\", \"" << s2 << "\") is " << result << endl;

}

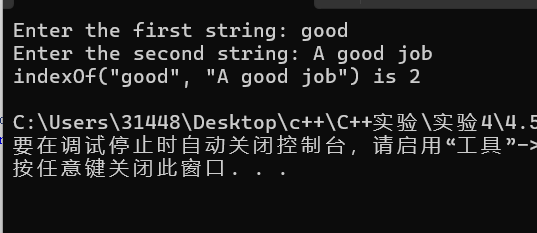
else {

cout << "indexOf(\"" << s1 << "\", \"" << s2 << "\") is –1" << endl;

}

return 0;

}



6.#include <iostream>

using namespace std;

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

while (\*s) // 遍历字符串直到遇到空字符

{

char ch = \*s;

if (ch >= 'A' && ch <= 'Z') // 如果字符是大写字母

{

ch += 'a' - 'A'; // 转换为小写字母

}

if (ch >= 'a' && ch <= 'z') // 如果字符是小写字母

{

counts[ch - 'a']++; // 对应字母的计数增加

}

s++; // 移动到下一个字符

}

}

int main() {

const int SIZE = 26; // 字母的数量

int counts[SIZE] = { 0 }; // 初始化计数数组

char input[1000]; // 假设输入的字符串不超过999个字符

cout << "Enter a string: ";

cin.getline(input, 1000); // 读入字符串

count(input, counts); // 调用count函数

// 显示每个字母出现的次数

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

if (counts[i] > 0) // 如果次数非零

{

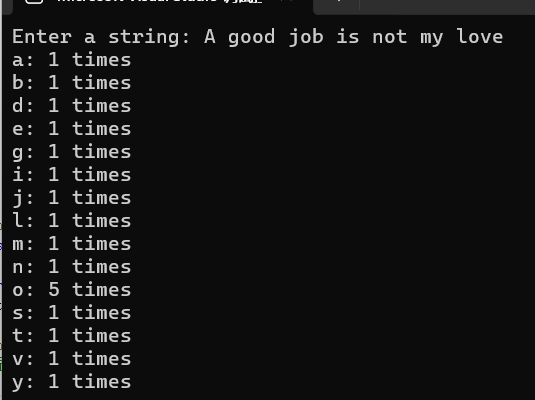
cout << char('a' + i) << ": " << counts[i] << " times" << endl;

}

}

return 0;

}



指针

1.1

#include<iostream>

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;

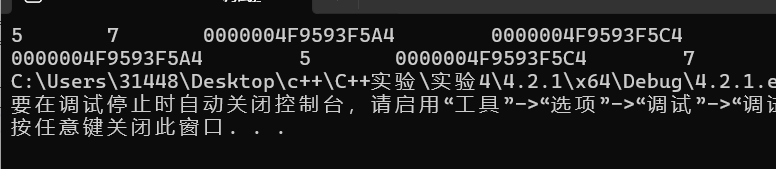
pj = &j;

i = 5; j = 7;

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

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

}



1.2

#include<iostream>

using namespace std;

int main() //C语言程序，要了解

{

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

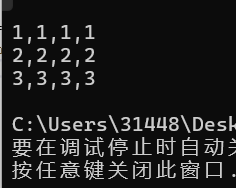
int\* p, i;

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

}



1.3

#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);

}

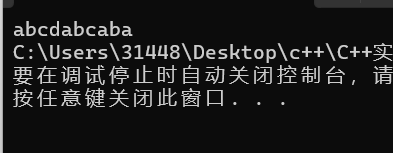
void main()

{

char st[] = "abcd";

f(st, 4);

}



1.4

#include<iostream>

using namespace std;

int\* f()

{

return new int[4] {1, 2, 3, 4};// 使用new动态分配内存

}

int main()

{

int\* p = f();

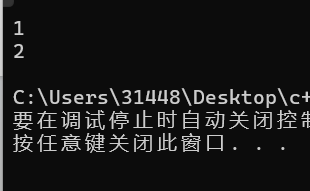
cout << p[0] << endl;

cout << p[1] << endl;

delete[] p;// 使用delete[]释放内存

return 0;

}



指针2.1

#include <iostream>

using namespace std;

int myStrlen(const char\* str) // 计算字符串长度的函数

{

int length = 0;

while (str[length] != '\0')

{

length++;

}

return length;

}

int indexof(const char\* s1, const char\* s2) // 函数实现

{

int len1 = myStrlen(s1);

int len2 = myStrlen(s2);

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

{

return -1; // 如果s1为空，s2为空，或者s1比s2长，则直接返回-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; // 如果字符不匹配，跳出内层循环

}

}

if (match)

{

return i; // 如果找到匹配的子串，返回第一次匹配的下标

}

}

return -1; // 如果没有找到匹配的子串，返回-1

}

int main()

{

char s1[100], s2[100]; // 假设字符串长度不超过99

cout << "请输入字符串s1: ";

cin.getline(s1, 100); // 使用getline读取字符串

cout << "请输入字符串s2: ";

cin.getline(s2, 100);

int result = indexof(s1, s2);

if (result != -1)

{

cout << "字符串s1是字符串s2的子串，第一次匹配的下标是: " << result << endl;

}

else

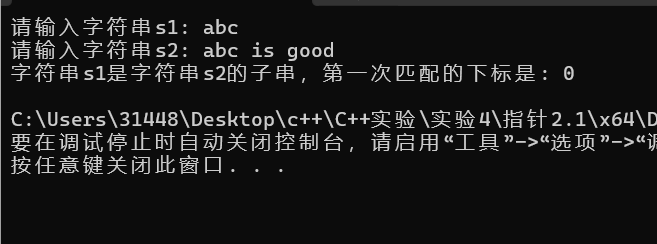
{

cout << "字符串s1不是字符串s2的子串。" << endl;

}

return 0;

}



2.2

#include <iostream>

#include <string>

#include <cstring> // 用于处理字符串函数

using namespace std;

int parseHex(const char\* const hexString) // 函数实现

{

if (hexString == nullptr || \*hexString == '\0')

{

cout << "Input is empty or null!" << endl;

return -1; // 返回-1表示输入无效

}

int decimal = 0;

int base = 1; // 16的0次方

size\_t length = strlen(hexString); // 获取字符串长度

for (int i = length - 1; i >= 0; --i)

{

char c = hexString[i];

if (c >= '0' && c <= '9')

{

decimal += (c - '0') \* base;

}

else if (c >= 'A' && c <= 'F')

{

decimal += (c - 'A' + 10) \* base;

}

else if (c >= 'a' && c <= 'f')

{

decimal += (c - 'a' + 10) \* base;

}

else

{

cout << "Invalid hexadecimal character: " << c << endl;

return -1; // 返回-1表示输入无效

}

base \*= 16; // 移动到下一个16进制位

}

return decimal;

}

int main() {

string hexStr; // 用于存储用户输入的16进制字符串

cout << "Enter a hexadecimal number: ";

cin >> hexStr; // 从用户输入中读取16进制字符串

int decimal = parseHex(hexStr.c\_str()); // 调用parseHex函数

if (decimal != -1)

{

cout << "The decimal value of " << hexStr << " is " << decimal << endl;

}

else

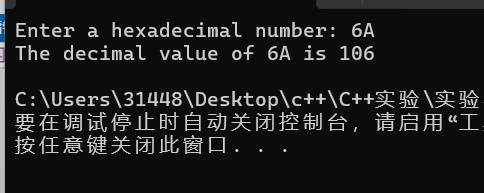
{

cout << "Invalid hexadecimal string: " << hexStr << endl;

}

return 0;

}



2.3

#include <iostream>

using namespace std;

void sortArray(int\* arr, int size) // 排序函数实现（简单选择排序）

{

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

{

int minIndex = i;

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

{

if (arr[j] < arr[minIndex])

{

minIndex = j;

}

}

if (minIndex != i) // 交换元素

{

int temp = arr[i];

arr[i] = arr[minIndex];

arr[minIndex] = temp;

}

}

}

int main()

{

int size;

cout << "请输入数组的元素个数: ";

cin >> size; // 用户输入数组的大小

int\* arr = new int[size];// 使用new动态分配数组

cout << "请输入 " << size << " 个数组元素:" << endl;

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

{

cin >> arr[i]; // 用户输入数组元素

}

cout << "原始数组内容：" << endl;// 打印数组原始内容

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

{

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

}

cout << endl;

sortArray(arr, size);

cout << "排序后的数组内容：" << endl;

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

{

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

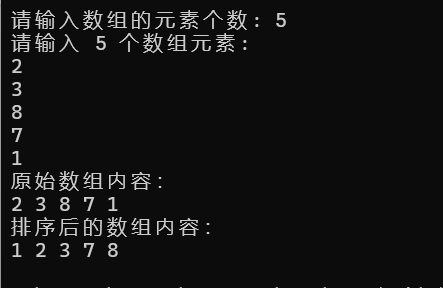
}

cout << endl;

delete[] arr;// 释放数组内存

return 0;

}



**四、遇到的问题与解决方法**

**问题：检测子串功能不完全**

解决方法：

在最开始编写的一版程序中，我可以完成子串的检查，但是只能检测连续的字符串。

考虑到是因为我使用cin输入字符数组的值，中间不能带有空格，便将cin替换为了cin.getline（），在替换后，程序就可以正常带有空格了。

问题：对存物柜问题理解有误。

解决方法：

报告上写的是每隔n个存物柜改变其状态，但通过网上查询资料得知是每隔n-1个存物柜改变其状态，故改变了源代码。

**五、体会**

**1. 深入掌握数组相关知识是十分必要的**

数组能够与函数、循环相结合，实现复杂的功能。但要实现更高级的功能，就需要更深入地理解和掌握数组的相关知识。

例如，在完成本次实验后，我意识到使用常量定义数组大小的方法在某些情况下存在局限性，这时就需要引入动态数组的概念。我已经掌握了`new`操作符的使用，通过进一步学习动态数组的原理和定义方法，我能够更快地掌握其定义和使用。使用动态数组后，程序的计算过程和代码编写变得更加便捷。不过，使用动态数组时，记得在不再需要时通过`delete`释放内存。

数组的概念体系相当复杂，进行计算时需要精确地编写表达式；要更灵活地运用数组，就需要通过实践和练习来加强数组知识的掌握。

**2. 指针的灵活运用能够实现多样化的功能**

与普通变量定义不同，指针存储的是变量的内存地址，这一特性使得指针的应用场景非常广泛。

在函数调用时，通过指针传递可以进行地址传递，使得函数能够修改实参的值，这是值传递无法实现的。虽然引用可以在一定程度上替代指针，但在某些情况下指针仍有其不可替代的作用。

此外，通过将函数中的变量定义为`const`，可以防止指针的值或指向的地址被修改，如果函数不需要修改指针的值或地址，这种方法可以避免无意中的错误。

在本次实验中，我主要学习了指针的相关内容，比如`\*p`、`&p`、`\*&p`之间的区别等，对这些概念有了更清晰的认识后，将来在编写指针相关的代码时就可以避免错误，提高效率。

总的来说，指针的使用可以增强程序的功能，为函数提供更多的操作性，提升代码的可读性。在未来，我将继续巩固指针知识，以便在我的程序中实现更多的可能性。