**计算机程序设计基础（C++)**

**实验报告**

专业班级：软件工程2403

学 号：8209240306

姓 名： 赵岩清

**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？

答：1、避免错误，函数返回与声明不匹配的类型，会导致运行错误。

2、函数类型与实际返回值一致时，代码可读性更好。

3、确保语言的一致性和可预测性

2. 本实验中主函数调用函数时采用的是何种传递方式？

答：求解最大公约数和最小公倍数时，使用了引用参数，直接修改原始变量的值。

判断素数、温度转换、猴子吃桃则使用值传递

**四、算法分析，程序结果**

**1、**

**（1）代码为：**

**#include <iostream>**

**using namespace std;**

**int gcd(int a, int b)**

**{**

**while (b != 0)//利用欧几里得算法（即辗转相处法）可算最大公因数**

**{**

**int temp = a % b;**

**a = b;**

**b = temp;**

**}**

**return a;**

**}**

**int main()**

**{**

**int m, n, gcd\_, lcm\_;**

**cout << "请输入第一个自然数" << endl;**

**cin >> m ;**

**cout << "请输入第二个自然数" << endl;**

**cin >> n;**

**if (m <= 0 || n <= 0) //检验m、n是否为正整数**

**{**

**cout << "请输入正整数！" << endl;**

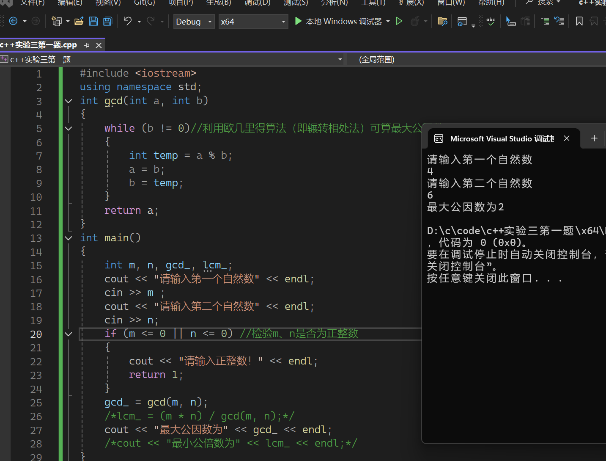
**return 1;**

**}**

**gcd\_ = gcd(m, n);**

**cout << "最大公因数为" << gcd\_ << endl;**

**运算结果：**

****

**（2）代码：**

**#include <iostream>**

**using namespace std;**

**void gcd\_lcm(int a, int b, int& gcd, int& lcm);//函数声明**

**int main() {**

**int m, n, gcd\_result, lcm\_result;**

**cout << "请输入第一个自然数: ";**

**cin >> m;**

**cout << "请输入第二个自然数: ";**

**cin >> n;**

**if (m <= 0 || n <= 0) //检查是否输入正整数**

**{**

**cout << "请输入正整数！" << endl;**

**return 1;**

**}**

**gcd\_lcm(m, n, gcd\_result, lcm\_result);//调用gcm\_lcm函数计算最大公约数和最小公倍数**

**cout << "最大公因数为: " << gcd\_result << endl;**

**cout << "最小公倍数为: " << lcm\_result << endl;**

**return 0;**

**}**

**void gcd\_lcm(int a, int b, int& gcd, int& lcm)**

**{ int \_a = a, \_b = b;//保留原始a、b**

**while (b != 0) //利用欧几里得算法（即辗转相处法）可算最大公因数**

**{**

**int temp = a % b;**

**a = b;**

**b = temp;**

**}**

**gcd = a;**

**lcm = (\_a \* \_b) / gcd; //此时a、b已经改变，因此要用之前的a、b**

**}**

**运行结果：**

**文本

描述已自动生成**

**2、**

**代码：**

**#include <iostream>**

**#include <iomanip>**

**using namespace std;**

**// 定义is\_prime函数，判断num是否为素数**

**bool is\_prime(int num)**

**{**

**if (num <= 1) return false; // 1和负数不是素数**

**if (num == 2) return true; // 2是最小的素数**

**if (num % 2 == 0) return false; // 排除所有偶数**

**for (int i = 3; i \* i <= num; i += 2) // 只需要检查到一个i小于等于num的平方根。因为如果num是合数则至少有一个因素小于等于其平方根**

**{**

**if (num % i == 0) return false;**

**}**

**return true;**

**}**

**int main()**

**{**

**int count = 0; // 记录已找到的素数数量**

**int number = 2; // 从2开始检查因为2是最小的素数**

**while (count < 200) // 直到找到200个素数**

**{**

**if (is\_prime(number))**

**{**

**cout.width(5); // 设置输出宽度为5，确保对齐**

**cout << number;**

**if (++count % 10 == 0) // 每输出10个素数换一行**

**cout << endl;**

**else**

**cout << " "; // 输出空格分隔**

**}**

**++number;**

**}**

**return 0;**

**}**

**运算结果：**

**日历

描述已自动生成**

**3、**

**代码：**

**头文件（mytemperature.h）：**

**#ifndef MYTEMPERATURE\_H**

**#define MYTEMPERATURE\_H**

**double celsius\_to\_fah(double cel);//摄氏度到华氏度**

**double fahrenheit\_to\_cels(double fah);//华氏度到摄氏度**

**#endif**

**源文件1（mytemperature.cpp）:**

**#include "mytemperature.h"**

**double celsius\_to\_fah(double cel) //摄氏度到华氏度**

**{**

**return (cel \* 9.0 / 5.0) + 32.0;**

**}**

**double fahrenheit\_to\_cels(double fah) //华氏度到摄氏度**

**{**

**return (fah - 32.0) \* 5.0 / 9.0;**

**}**

**源文件2（mian.cpp）:**

**#include <iostream>**

**#include <iomanip>**

**#include "mytemperature.h"**

**using namespace std;**

**int main()**

**{**

**double f2 = 120.0;**

**cout << fixed << setprecision(2);**

**cout << "Celsius Fahrenheit | Fahrenheit Celsius" << endl;**

**for (double c1 = 40.0;c1 >= 31.0;c1-=1)**

**{**

**double f1 = celsius\_to\_fah(c1);**

**cout << setw(5) << c1 << " "**

**<< setw(7) << f1 << " | ";**

**for (f2;f2>=30;f2-=10)**

**{**

**double c2 = fahrenheit\_to\_cels(f2);**

**cout << setw(9) << f2 << " "**

**<< setw(11) << c2 << " " << endl;**

**f2 -= 10;**

**break;**

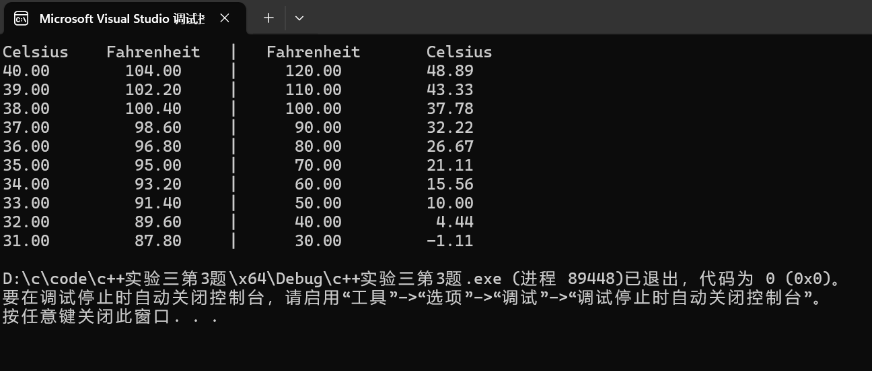
**}**

**}**

**return 0;**

**}**

**运算结果：**

****

**5、**

**代码：**

**#include <iostream>**

**using namespace std;**

**int peaches(int day);//函数声明**

**int main()**

**{**

**int day = 1;**

**int p = peaches(day);**

**cout << "第一天猴子共摘了 " << p << " 个桃子." << endl;**

**return 0;**

**}**

**int peaches(int day) //定义递归函数**

**{**

**if (day == 10)**

**{**

**return 1;**

**}**

**else**

**{**

**return (peaches(day + 1) + 1) \* 2;**

**//函数关系为p2=p1/2-1,即p1=(p2+1)\*2,因为要倒推第一天。**

**}**

**}运算结果：**

**文本

描述已自动生成**

**五、遇到的问题与解决方法**

**1、在第一题中，做第一问时没考虑到引用参数，以至于在做第二题的时候需要重新写代码，利用引用参数的方式进行传递。**

**2、第二题中，在#include \<iomanip>头文件前提下的输出格式的具体格式并不是很明确，**

**如利用setw(n)调控输出宽度**

**3、第三题中，如何把4列两组数据放在同一行输出的同时又使其两两关联，最后是通过for循环的嵌套，再加上2中提到的输出格式操作，利用空格填充来达成整齐地目的。**

**4、第五题中，对递归函数的运用不够熟练。且最初会把通过for循环进行正向的推导和利用递归逆向推导搞混。**

**六、体会**

**广泛了解达成某一目的的不同方式，并具体深入了解二者的差异和适用情况。例如在函数调用传递时可以用引用参数和值传递，再例如`cout.width();`和`setw(n)`都可以设置输出格式中的字段宽度，再例如for循环和递归都可以达成某些相同的目的。**

**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**

**数组第一题：**

**代码：**

#include<iostream>

using namespace std;

int main()

{

const int size = 10;

int list[size];

cout << "请输入10个数 : " << endl;

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

cin >> list[i];

cout << list[0] << " ";

for (int m = 1; m < size; m++)

{

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

{

if (list[m] != list[j])

{

if (j == m - 1)

cout << list[m] << " ";

else

continue;

}

else

break;

}

}

cout << endl;

return 0;

}**输出结果：**

**文本

描述已自动生成**

**数组第二题：**

**代码：**

#include <iostream>

using namespace std;

void bubbleSort(double list[], int listSize) {

bool changed = true;

do {

changed = false;

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

{

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

{

double temp = list[j];

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

list[j + 1] = temp;

changed = true;

}

}

} while (changed);

}

int main()

{

const int SIZE = 10;

double numbers[SIZE];

cout << "请输入10个双精度数字：" << endl;

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

{

cin >> numbers[i];

}

bubbleSort(numbers, SIZE);

cout << "排序后的数字为：" << endl;

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

{

cout << numbers[i] << " ";

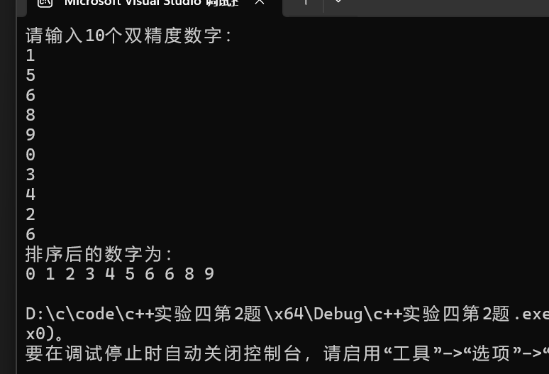
}

cout << endl;

return 0;

}

**运算结果：**

****

**数组第三题：**

**代码：** #include <iostream>

using namespace std;

int main()

{

const int nLOCKERS = 100;

bool lockers[nLOCKERS] = { false };

for (int student = 1; student <= nLOCKERS; ++student)

{

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

{

lockers[locker] = !lockers[locker];

}

}

cout << "开着的存物柜编号为：" << endl;

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

{

if (lockers[i])

{

cout << (i + 1) << " ";

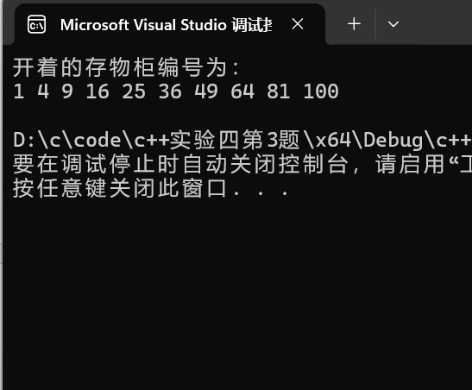
}

}

cout << endl;

return 0;

}**运算结果：**

****

**数组第四题：**

**代码：**#include <iostream>

using namespace std;

// 合并两个已排序数组

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

{

int i = 0, j = 0, k = 0;

while (i < size1 && j < size2)

{

if (list1[i] < list2[j])

{

list3[k++] = list1[i++];

}

else

{

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

}

}

while (i < size1)// 如果list1还有剩余元素

{

list3[k++] = list1[i++];

}

while (j < size2)// 如果list2还有剩余元素

{

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

}

}

int main()

{

const int MAX\_SIZE = 80;

int list1[MAX\_SIZE], list2[MAX\_SIZE], list3[MAX\_SIZE \* 2];

int size1, size2;

cout << "请输入第一个数组的元素数（不超过80）：";

cin >> size1;

cout << "请输入第一个数组的元素（已排序）：" << endl;

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

{

cin >> list1[i];

}

cout << "请输入第二个数组的元素数（不超过80）：";

cin >> size2;

cout << "请输入第二个数组的元素（已排序）：" << endl;

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

{

cin >> list2[i];

}

merge(list1, size1, list2, size2, list3);

cout << "合并后的数组为：" << endl;

for (int i = 0; i < size1 + size2; ++i)

{

cout << list3[i] << " ";

}

cout << endl;

return 0;

}

**运算结果：**

**文本

描述已自动生成**

**数组第五题：**

**代码：**

#include <iostream>

#include <cstring> // 用于strlen

using namespace std;

int indexOf(const char s1[], const char s2[]) // 检验s1是否是s2的子串

{

int len1 = strlen(s1);

int len2 = strlen(s2);

for (int i = 0; i <= len2 - len1; ++i)// 遍历s2中的每个可能的起始位置

{

bool match = true;

for (int j = 0; j < len1; ++j)// 比较s1和s2的子串

{

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

{

match = false;

break;

}

}

if (match)

{

return i;

}

}

return -1;

}

int main()

{

const int MAX\_LENGTH = 100;

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

cout << "请输入第一个字符串s1: ";

cin.getline(s1, MAX\_LENGTH);

cout << "请输入第二个字符串s2: ";

cin.getline(s2, MAX\_LENGTH);

int index = indexOf(s1, s2);

if (index != -1)

{

cout << index << endl;

}

else

{

cout << index << endl;

}

return 0;

}

**运算结果：**

**手机屏幕的截图

描述已自动生成**

**数组第六题：**

**代码：**#include <iostream>

using namespace std;

const int num1 = 26;

const int num2 = 99;

void countLetters(const char list[], int counts[])

{

for (int i = 0; i < num1; i++)

counts[i] = 0;

for (int j = 0; j < num2; j++)

{

counts[list[j] - 'a']++;

counts[list[j] - 'A']++;

}

}

void displayCounts(const int counts[])

{

for (int i = 0; i < num1; i++)

{

if (counts[i] != 0)

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

}

}

int main()

{

char list[num2];

int counts[num1];

cout << "输入一串字符: ";

cin.getline(list, num2);

countLetters(list, counts);

displayCounts(counts);

return 0;

}

运算结果：

文本

描述已自动生成

指针1、

（4）：#include <iostream>

using namespace std;

int\* f()

{

int\* list = new int[4] {1, 2, 3, 4}; // 使用new动态分配内存

return list;

}

int main()

{

int\* p = f();

cout << p[0] << endl;

cout << p[1] << endl;

delete[] p; // 释放动态分配的内存

return 0;

}

运算结果：



指针2（1）

#include <iostream>

#include <cstring> // 用于strlen

using namespace std;

int indexOf(const char s1[], const char s2[]) // 检验s1是否是s2的子串

{

int len1 = strlen(s1);

int len2 = strlen(s2);

for (int i = 0; i <= len2 - len1; ++i)// 遍历s2中的每个可能的起始位置

{

bool match = true;

for (int j = 0; j < len1; ++j)// 比较s1和s2的子串

{

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

{

match = false;

break;

}

}

if (match)

{

return i;

}

}

return -1;

}

int main()

{

const int MAX\_LENGTH = 100;

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

cout << "请输入第一个字符串s1: ";

cin.getline(s1, MAX\_LENGTH);

cout << "请输入第二个字符串s2: ";

cin.getline(s2, MAX\_LENGTH);

int index = indexOf(s1, s2);

if (index != -1)

{

cout << index << endl;

}

else

{

cout << index << endl;

}

return 0;

}

运算结果：

**手机屏幕的截图

描述已自动生成手机屏幕的截图

描述已自动生成**

指针2、（2）

代码：

#include <iostream>

#include <cctype> // 用于tolower函数

using namespace std;

int parseHex(const char\* const hexString);

int main()

{

const int MAX\_LENGTH = 100;

char hexString[MAX\_LENGTH]{};

cout << "请输入一个16进制字符串: ";

cin.getline(hexString, MAX\_LENGTH);

int decimalValue = parseHex(hexString);

cout << "16进制字符串 " << hexString << " 对应的10进制数是: " << decimalValue << endl;

return 0;

}

int parseHex(const char\* const hexString)

{

int result = 0;

int length = 0;

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

{

length++;

}

for (int i = 0; i < length; ++i)// 从右向左遍历字符串

{

char c = tolower(hexString[length - 1 - i]);// 将字符转换为小写

int value = 0;

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

{

value = c - '0';

}

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

{

value = c - 'a' + 10;

}

else

{

cerr << "无效的16进制字符: " << c << endl;

return -1; //表示无效

}

result += value \* static\_cast<int>(pow(16, i));

}

return result;

}

运算结果：

文本

描述已自动生成

指针2（3）：

代码：#include <iostream>

using namespace std;

void sortArray(int\* arr, int size);

void printArray(int\* arr, int size);

int main()

{

int n;

cout << "请输入数组的元素个数: ";

cin >> n;

int\* array = new (nothrow) int[n];// 动态分配内存

if (array == nullptr)

{

cerr << "内存分配失败！" << endl;

return 1;

}

cout << "请输入 " << n << " 个整数:" << endl;

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

{

cin >> array[i];

}

sortArray(array, n);

cout << "排序后的数组为: ";

printArray(array, n);

delete[] array;

return 0;

}

void sortArray(int\* arr, int size)

{

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

{

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

{

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

{

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

void printArray(int\* arr, int size) // 使用指针方式输出数组元素

{

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

{

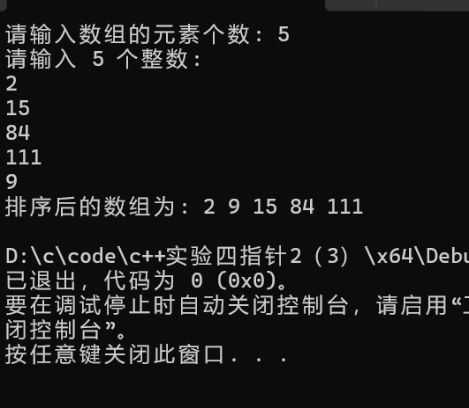
cout << \*(arr + i) << " ";

}

cout << endl;

}

运算结果：



**四、遇到的问题与解决方法**

**1、对基本的for和if在连续嵌套时逻辑不够清楚，运用不够熟练。**

**2、对起泡排序的函数运用不够熟练。**

**3、不了解#include <cstring>头文件下strlen函数的具体用法。**

**4、对递归的认识不够明确**

**5、对十六进制和十进制的转化方法不够明确**

**6、对动态数组理解不到位且不明确**

**五、体会**

不能够熟练运用已掌握的知识解决问题，但没能掌握好的知识又运用不出来。导致面对很多问题都只有一部分思路，又有一部分是难以解决的。在学习一些知识的同时，多了解其在不同情况下的应用，多加练习。