**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软工2303

学 号： 8209230323

姓 名： 夏子轩

**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）

#include<iostream>

using namespace std;

int gcd(int &a,int& b)

{

int t=0;

while (b != 0) {

t = b;

b = a % b;

a = t;

}

return a;

}

int main()

{

int m, n, x, y;

cout << "请输入两个自然数" << endl;

cin >> m >> n;

x = m, y = n;

cout<<"最大公约数为 " << gcd(m, n) << endl;

cout << "最小公倍数为 " << x \* y / m << endl;

return 0;

}

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

#include<iostream>

using namespace std;

bool is\_prime(int num)

{

if (num == 1 || num == 2) { return true; }

else

{

for (int i = 2; i < num; i++)

{

if (num % i == 0)

{

return false;

}

}

return true;

}

}

int main()

{

int i = 2, n = 0;

for (i; i <= 200; i++)

{

if (is\_prime(i))

{

cout << i << "\t";

n++;

if (n % 10 == 0)cout << endl;

}

}

return 0;

}

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）

头文件：

#ifndef MY\_HEADER\_FILE\_H

#define MY\_HEADER\_FILE\_H

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

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

#endif

源文件1：

#include<iostream>

#include<cmath>

using namespace std;

#include "mytriangle.h"

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

{

if (side1 + side2 <= side3 || side1 + side3 <= side2 || side2 + side3 <= side1) {

return false;

}

else {

return true;

}

}

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

{

double s = (side1 + side2 + side3) / 2;

return sqrt(s\*(s - side1)\*(s - side2)\*(s - side3));

}

源文件2：

#include<iostream>

using namespace std;

#include"mytriangle.h"

int main()

{

double side1, side2, side3;

cout << "请输入三角形三边长" << endl;

cin >> side1 >> side2 >> side3;

if (is\_valid(side1, side2, side3))

cout<<"面积为"<<area(side1, side2, side3);

return 0;

}

**3与4选一个完成**

1. 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）。

#include<iostream>

using namespace std;

int f(int n)

{

if (n == 10)return 1;

else {

return (f(n + 1) + 1) \* 2;

}

}

int main()

{

cout << f(1);

return 0;

}

**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：函数的返回值类型用于指定函数执行完毕后将要返回的数据类型。返回值类型应当函数声明或定义中指定的类型一致，以确保函数的调用者能够正确使用函数的返回值。

1. 本实验中主函数调用函数时采用的是何种传递方式？

答：在不需要改变参数值的时候，我们采用了值传递，当我们需要改变参数的值的时候，我们可以使用引用传递。

**四、算法分析，程序结果**

**1.算法：使用了数学方法解决了求最大公约数的办法，然后在主函数内进行了最小公倍数的计算，然后对结果进行输出；**

**结果：可以对任意自然数进行求解。**

**2.算法：利用循环遍历来2到num-1的每一个数，看是否能整除num，如果存在能够整除num的数字，说明num不是素数，返回false，如果未找到，那么num是素数，返回true。**

**结果：通过函数找出前200个素数，并按每行10个输出**

**3.算法：编写两个函数，一个用于判断三角形是否合法，一个用于计算三角形面积；**

**结果：在主函数中判断并输出三角形面积。**

1. **算法：将题干条件转化为数学公式，并用函数递归的模式将其表示出来；**

**结果：主函数中输出了第一天的桃子数。**

1. **遇到的问题与解决方法**

**在实验中可能会遇到头文件引用错误或者递归函数没有合适的终止条件，我们需要在写代码时仔细，确保头文件的路径正确，并且头文件的内容和实现文件一致。也要记得不要让递归函数出错。**

1. **体会**

**在这次实验中，我有许多体会：**

1. **函数的重要性，函数提高了代码的模块性，可读性和可维护性；**
2. **问题分析的能力，在遇到问题时，我需要培养分析问题并且查找解决方法的能力，这对于提高编程能力至关重要；**
3. **多文件编程，我慢慢理解如何将程序分割成多个文件，提高代码的组织性；**
4. **参数传递方式：我更进一步来了解了值传递和引用传递。**

**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**

下面是一个运行样例：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。

Enter

Enter ten numbers: 1 2 3 2 1 6 3 4 5 2

The distinct numbers are: 1 2 3 6 4 5

#include<iostream>

using namespace std;

int main()

{

const int size = 10;

int list[size];

cout << "Please enter ten numbers : " << 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;

}

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。

#include<iostream>

using namespace std;

int main()

{

double list[10] = {};

cout << "Input ten numbers: " << endl;

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

{

cin >> list[i];

}

bool changed = true;

do

{

changed = false;

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

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

{

double t;

t = list[j];

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

list[j + 1] = t;

changed = true;

}

} while (changed);

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

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

return 0;

}

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。

#include<iostream>

using namespace std;

int main()

{

int student[100]{};

bool cabnit[100]{};

for (int i = 0; i <= 99; i++)

student[i] = i + 1;

for (int i = 0; i <= 99; i++)

cabnit[i] = true;

for (int i = 2; i <= 100; i++)

{

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

if (j == i || (j - i) % i == 0)

{

if (cabnit[j - 1] ) {

cabnit[j - 1] = false;

continue;

}

else {

cabnit[j - 1] = true;

continue;

}

}

}

for (int i = 1; i <= 100; i++)

if (cabnit[i - 1])

cout << i << " ";

return 0;

}

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

#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 i = 0; i < size2; i++)

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

bool changed = true;

do

{

changed = false;

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

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

{

int t;

t = list3[j];

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

list3[j + 1] = t;

changed = true;

}

} while (changed);

}

int main()

{

const int Max = 80;

int m, n;

cout << "Enter list1: ";

cin >> m;

int\* list1 = new int[m];

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

cin >> list1[i];

cout << endl;

cout << "Enter list2: ";

cin >> n;

int\* list2 = new int[n];

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

cin >> list2[i];

cout << endl;

cout << "The merged list is ";

int\* list3 = new int[m + n];

merge(list1, m, list2, n, list3);

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

{

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

}

delete[]list1;

delete[]list2;

delete[]list3;

return 0;

}

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

Enter the first string: welcome

Enter the second string: We invite you!

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

#include<iostream>

#include<cstring>

using namespace std;

int indexOf(char s1[], char s2[]);

int main()

{

const int size = 999;

char s1[size];

char s2[size];

cout << "Enter the first string :";

cin.getline(s1, size);

cout << "Enter the second string:";

cin.getline(s2, size);

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

return 0;

}

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

{

int a, b, index, j, k;

a = strlen(s1);

b = strlen(s2);

bool\* s3 = new bool[a];

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

s3[m] = false;

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

{

for (j = 0, k = i; (j < a) && (k < i + a); j++, k++)

if (s1[j] == s2[k])

{

index = i;

s3[j] = true;

}

int h = 0;

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

{

if (s3[n] == false && i == b - a)

{

return -1; break;

}

if (s3[n])

h++;

}

if (h == a)

{

return index;

}

}

}

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

#include<iostream>

using namespace std;

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

{

int n = strlen(s);

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

{

for (int j = 0; j < 26; j++)

{

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

counts[j]++;

}

}

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

{

if (counts[i] != 0)

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

}

}

int main()

{

const int max=99;

char\* s = new char[max];

cin.getline(s, max);

int counts[26];

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

counts[i] = 0;

count(s, counts);

delete[]s;

return 0;

}

**（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：5 7 000000C515D4FC14 000000C515D4FC34

000000C515D4FC14 5 000000C515D4FC34 7

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 = 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;

}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);

#include<iostream>

#include<cstring>

using namespace std;

int indexOf(char\* s1, char\* s2);

int main()

{

const int size = 999;

char s1[size];

char s2[size];

cout << "Enter the first string :";

cin.getline(s1, size);

cout << "Enter the second string:";

cin.getline(s2, size);

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

return 0;

}

int indexOf(char\* s1, char\* s2)

{

int a, b, index, j, k;

a = strlen(s1);

b = strlen(s2);

bool\* s3 = new bool[a];

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

s3[m] = false;

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

{

for (j = 0, k = i; (j < a) && (k < i + a); j++, k++)

if (s1[j] == s2[k])

{

index = i;

s3[j] = true;

}

int h = 0;

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

{

if (s3[n] == false && i == b - a)

{

return -1; break;

}

if (s3[n])

h++;

}

if (h == a)

{

return index;

}

}

}

(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

#include<iostream>

#include<string>

#include<cmath>

using namespace std;

int parseHex(const char\* const hexString)

{

int result = 0;

int n = strlen(hexString);

int\* list = new int[n];

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

list[i] = 0;

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

{

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

list[i] = hexString[i] - 'A' + 10;

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

list[i] = hexString[i] - '0';

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

list[i] = hexString[i] - 'a' + 10;

}

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

result +=list[i]\* pow(16, n -1 - i);

delete[]list;

return result;

}

int main()

{

string a;

cout << "请输入一个十六进制的数字";

cin >> a;

const char\* hexString = a.c\_str();

cout<<"它的十进制为： "<<parseHex(hexString);

return 0;

}

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）。

#include<iostream>

using namespace std;

void sort(int\*\*pa,int n) {

bool changed = true;

do

{

changed = false;

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

if ((\*pa)[j] > (\*pa)[j+1])

{

int t = (\*pa)[j ];

(\*pa)[j ] = (\*pa)[j+1];

(\*pa)[j+1] = t;

changed = true;

}

} while (changed);

}

int main()

{

int n;

cout << "请输入元素个数";

cin >> n;

cout << "请输入元素";

int\* list = new int[n];

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

cin >> list[i];

int\*\* pa = &list;

cout <<"第一个元素为："<< \* \*pa << endl;

cout << "你想查询第几个元素：";

int m;

cin >> m;

if (m <= n)

cout << (\*pa)[m - 1] << endl;

else

cout << "您定义的数组里没有这么多元素" << endl;

sort(pa,n);

cout << "元素从小到大排序为：";

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

cout << (\*pa)[i]<< " ";

delete[]list;

return 0;

}

【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）.使用嵌套循环，外层循环用于遍历所有元素，内层循环用于检查当前元素是否在前面的遍历中已经出现过，如果没出现过，就进行输出；**

**（2）.起泡排序，利用布尔型变量‘changed’判断是否需要下一趟循环；**

**（3）.利用布尔型变量‘cabinet’表示存物柜的开关状态，通过嵌套循环模拟学生打开和关上存物柜的过程，最终输所有开着的柜子的号码；**

**（4）.将两个排列好的数组合并成一个新数组，然后使用起泡算法对合并后的数组进行排序；**

**（5）.检验子串，利用两边循环遍历字符串，通过比较字符是否相等来找到子串在主串中的位置；**

**（6）.实用数组‘counts’统计每个字母出现的次数**

**2.程序设计：**

**（1）.使用两个嵌套循环来遍历输入字符串的字母，函数初始化一个布尔型数组来跟踪在‘s1’中匹配的字符，如果找到匹配，函数就返回第一次索引，否则返回-1；**

**（2）.将十六进制字符串转化为其十进制等值，用‘list’存储每个十六进制等职并累加结果；**

**（3）.创建一个动态数组，用户输入其元素，使用冒泡排序对数组进行升序，并用指针输出元素，最后释放动态分配的内存。**

**四、遇到的问题与解决方法**

**在这次实验中，由于不够仔细，出现过忘记释放动态分配的内存等问题，但是经过我认真仔细的检查，最终发现了问题所在，最后解决了问题。**

1. **体会**

通过这次实验，我学到了如何处理数组，指针和字符串以及如何设计和实现一些基本的算法。在使用指针对数组进行操作时，我对指针的使用和数组的动态分配有了更深刻的认识。我还在慢慢养成确保释放动态内存的习惯，防止资源泄露。我认为，在面对不同问题和挑战时，持续学习和改进的态度时必要的。反思自己的代码水平，寻找潜在的改进点，并尝试用不同的方法，有助于不断提升编程水平。