**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软工2302班

学 号： 821023127

姓 名： 王贤书

**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？

这样函数具有可预测性，可以建设函数中的错误，同时也利用代码的复用，能够确保代码质量和可维护性。

2. 本实验中主函数调用函数时采用的是何种传递方式？

主要采用按值传递的方式。函数接收的是实参的一个副本。任何在函数内部对参数所做的修改都不会影响原始变量。

**四、算法分析，程序结果**

第1题：

#include <iostream>

using namespace std;

int fun\_max(int a, int b);

int fun\_min(int a, int b);

int main(){

int a, b;

cout << "请输入两个数:";

cin >> a >> b;

if (a < b) {

int t = a;

a = b; b = t;}

cout << a<<"和"<<b<<"的最大公约数为：";

cout << fun\_max(a, b) << endl;

cout << a << "和" << b << "的最小公倍数为：";

cout << fun\_min(a, b) << endl;

}

int fun\_max(int a, int b){

while (a % b != 0){

int c = a % b;

a = b;

b = c;}

return b;

}

int fun\_min(int a, int b) {

int c = a;

int i = c;

for (; i <= a \* b; i++) {

if (i % a == 0 && i % b == 0) {

break;}}

return i;

}文本

描述已自动生成

第2题：

#include<iostream>

using namespace std;

bool is\_prime(int num) {

if (num == 1)

return false;

if (num == 2)

return true;

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

if (num % i == 0) {

return false;

break;}}

return true;

}

int main() {

int line = 0;

for (int i = 1; i <= 200; i++) {

if (is\_prime(i)) {

cout << i << "\t";

line++;

}if (line == 10) {

cout << endl;

line = 0;

}

}

return 0;

屏幕上有字

描述已自动生成}

第4题：

头文件：

#pragma once

#include<iostream>

#include<cmath>

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

int side12, side23, side13;

side12 = side1 + side2;

side23 = side2 + side3;

side13 = side1 + side3;

if ((side12 > side3) && (side13 > side2) && (side23) > side1)

return true;

return false;

}

double double\_area(double side1, double side2, double side3) {

double area;

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

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

area = sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));

return area;

}

else

std::cout<<"不存在该这三边组成的三角形，故输出默认值";

return 0;

}

cpp文件：

#include <iostream>

#include"mytemperature.h"

using namespace std;

int main(){

double a, b, c;

cout << "请输入三边长:";

cin >> a >> b >> c;

cout << double\_area(a, b, c) << endl;

文本

描述已自动生成}

第5题：

#include<iostream>

using namespace std;

int eat\_peach(int days, int num = 1) {

if (days == 1)

return num;

else {

eat\_peach(days - 1, (num + 1) \* 2);

}

}

int main() {

cout << "第一天摘的桃子数为：";

cout << eat\_peach(10, 1) << endl;

return 0;

文本

描述已自动生成}

**五、遇到的问题与解决方法**

第4 题判断是否为三角形时，我刚开始认为应当将三边排序后再判断，但这样好像违背了题目的初衷，所以我就每种情况都判断了一遍。

**六、体会**

C++代码题要多做才能更好地巩固知识。猴子摘桃那题让我体会到递归函数的简介之美。

**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）。

**三、算法分析，程序结果**

数组：

第1题：

#include <iostream>

#include<vector>

using namespace std;

int main(){

int a[10];

vector<int> b;

cout << "Enter ten numbers:";

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

cin >> a[i];

}

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

bool flag = false;

int sizeb = b.size();

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

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

flag = true;

}if (!flag) {

b.push\_back(a[i]);

}

}

int t = b.size();

cout << "The distinct numbers are:";

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

cout << b[i] << " ";

}cout << endl;

文本

描述已自动生成}

第2题：

#include<iostream>

using namespace std;

int main() {

double list[10];

int listSize = 10;

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

cin >> list[i];

bool changed = true;

do

{

changed = false;

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

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

{

swap(list[j],list[j + 1]);

changed = true;

}

} while (changed);

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

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

return 0;

文本

描述已自动生成}

第3题：

#include<iostream>

using namespace std;

int main() {

bool l[101];

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

l[i] = false;

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

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

if (j % i == 0)

l[j] = (!l[j]);

}

}cout << "开着的柜子号码为：";

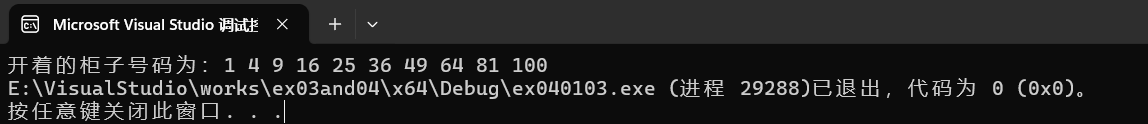
for (int i = 1; i <= 100; i++) {

if (l[i])

cout << i<<" ";

}

return 0;

}

第4题：

#include<iostream>

using namespace std;

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

int l = 0, r = 0, t = 0;

while ((l < size1) && (r < size2)) {

if (list1[l] < list2[r]) {

list3[t] = list1[l];

l++;

}else{

list3[t] = list2[r];

r++;}

t++;}

if (l == size1) {

for (; r < size2; r++) {

list3[t] = list2[r];

t++;}}

else{

for (; l < size1; l++) {

list3[t] = list1[l];

t++;}}}

int main() {

int size1, size2;

cout << "Enter list1:";

cin >> size1;

int\* list1 = new int[size1];

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

cin >> list1[i];

cout << "Enter list2:";

cin >> size2;

int\* list2 = new int[size2];

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

cin >> list2[i];

int size3 = size1 + size2;

int\* list3 = new int[size3];

cout << "The merged list is:";

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

for (int i = 0; i < size3; i++)

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

delete[]list3;

delete[]list2;

delete[]list1;

return 0;

文本

描述已自动生成}

第5题：

#include<iostream>

using namespace std;

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

int i = 0, k = -1;

while (s2[i] != '\0') {

if (s2[i] == s1[0]) {

int ii = 0;

while (s1[ii] != '\0') {

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

break;

ii++;

}

if (s1[ii] == '\0') {

k = i;

break;}}

i++;}

return k;

}

int main() {

char s1[1024];

char s2[1024];

cout << "Enter the first string:";

cin.getline(s1, 1024);

cout << "Enter the second string:";

cin.getline(s2, 1024);

int k = indexOf(s1, s2);

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

}文本

描述已自动生成文本

描述已自动生成

第6题：

#include<iostream>

using namespace std;

int counts[26];

char s[1024];

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

int i = 0;

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

counts[i] = 0;

while (s[i] != '\0') {

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

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

counts[j]++;

i++;}}

int main() {

cout << "Enter a string:";

cin.getline(s, 1024);

count(s, counts);

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

if (counts[i] != 0) {

char k = 'a' + i;

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

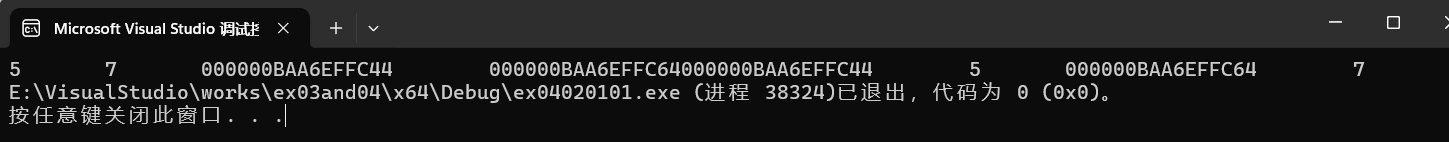
}

}

return 0;

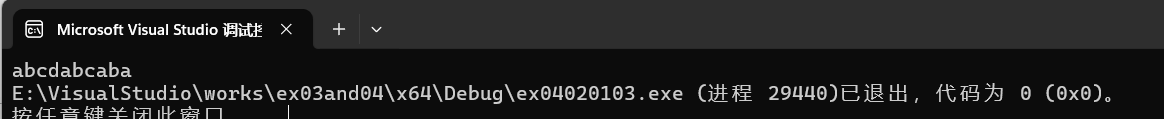
}

指针：

1.(1):

文本

描述已自动生成1.(2):

1.(3):

1.(4):

修改之后为：

#include<iostream>

using namespace std;

int\* f()

{

int\* list = new int[4];

\*list = 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;

文本

描述已自动生成}

2.(1):

#include<iostream>

using namespace std;

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

int i = 0, k = -1;

while (s2[i] != '\0') {

if (s2[i] == s1[0]) {

int ii = 0;

while (s1[ii] != '\0') {

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

break;

}ii++;

}if (s1[ii] == '\0') {

k = i;

break;

}

}i++;

}return k;

}

int main() {

char s1[1024];

char s2[1024];

cout << "Enter the first string:";

cin.getline(s1, 1024);

cout << "Enter the second string:";

cin.getline(s2, 1024);

int k = indexOf(s1, s2);

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

文本

描述已自动生成文本

描述已自动生成}

2.(2):

#include<iostream>

#include<cmath>

using namespace std;

char hexString[20];

int parseHex(const char\* const hexString) {

int len = 0,sum=0,hexNum[20];

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

len++;

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

switch (hexString[i]) {

case '1':

hexNum[i] = 1; break;

case '2':

hexNum[i] = 2; break;

case '3':

hexNum[i] = 3; break;

case '4':

hexNum[i] = 4; break;

case '5':

hexNum[i] = 5; break;

case '6':

hexNum[i] = 1; break;

case '7':

hexNum[i] = 1; break;

case '8':

hexNum[i] = 1; break;

case '9':

hexNum[i] = 1; break;

case '0':

hexNum[i] = 1; break;

case 'a':

hexNum[i] = 10; break;

case 'A':

hexNum[i] = 10; break;

case 'b' :

hexNum[i] = 11; break;

case 'B':

hexNum[i] = 11; break;

case 'c':

hexNum[i] = 12; break;

case 'C':

hexNum[i] = 12; break;

case 'd':

hexNum[i] = 13; break;

case 'D':

hexNum[i] = 13; break;

case 'e':

hexNum[i] = 14; break;

case 'E':

hexNum[i] = 14; break;

case 'f':

hexNum[i] = 15; break;

case 'F':

hexNum[i] = 15; break;

default:

break;

}}

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

int t = hexNum[i];

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

t \*= 16;

}sum += t;

}return sum;

}

int main() {

cin >> hexString;

cout <<hexString<<"的十进制数为：" << parseHex(hexString) << endl;

文本

描述已自动生成}

2.(3):

#include<iostream>

using namespace std;

int main() {

int n;

cout << "请输入数组元素个数：";

cin >> n;

int\* p = new int[n];

cout << "请输入数组元素：";

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

cin >> p[i];

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

cout <<"数组中 " << p[i] << "的地址为：" << p + i << endl;

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

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

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

int temp = p[j];

p[j] = p[j + 1];

p[j + 1] = temp;

}

}

cout << "排序后的数组为：";

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

cout << p[i] << " ";

cout << endl;

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

cout << "数组中 " << p[i] << "的地址为：" << p + i << endl;

delete[]p;

return 0;

文本

描述已自动生成}

**四、遇到的问题与解决方法**

数组部分：

第1题我本来想用第2 个数组的修改来解决是否重复的问题，但实现不太容易，所以我转变思路，用可变数组vector来解决问题。

合并两个排列好的数组这题中，我联想到了归并排序的合并部分。具体的思路是将两个数组一对一比大小，较小的元素纳入新数组，并将下标加一。最后一定有一个数组有剩余，便将其全部加入新数组。

指针部分：

指针部分我刚开始忘了指向数组的指针加中括号和下标是表示值，而直接将指针变量加上数字指的是数组中元素的地址。翻书后又将这一部分知识巩固了一遍。

十六进制转换为十进制那题我本来想用if语句加上范围的限制来解决问题。但转头一想没有switch…case…语句来的直接，刚好我对于switch语句使用教室便用这个语句解题。虽然switch语句代码量看起来更大，实践上都是重复的内容，实现非常方便。

最后一题，我误将新分配内存这一个步骤写到主函数之外，导致程序虽然可以正常运行，但总是出现内存泄漏。仔细排查后才知道自己没有认真看题。

**五、体会**

对于数组的修改这部分的内容反应较慢，思考的时间较长。指针部分我相对于其他部分没有那么熟练。本来我对指针这部分内容没有完全掌握好，看到涉及很多指针的代码心里有点怕怕的。通过这次实验，我巩固了这些知识，使用时也更加熟练。另外，程序题要多多实践，这样便于打开思路。实验中遇到的问题迫使自己去思考和寻找解决方案，提高了解决问题的能力。

在实验中，我遇到了大大小小各种各样的问题。刚开始我还很怕程序报错，现在我理解了报错是好事，它让我清晰地找到自己的错误。学会了如何分解复杂问题，逐步解决。