**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软工2404

学 号： 26

姓 名： 郑俊晖

**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>

#include<iomanip>

using namespace std;

int func(int m, int n) {

int i = 1, ans = 1;

while (i <= m && i <= n) {

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

ans = i;

i++;

}

return ans;

}

int main() {

int m, n;

cin >> m >> n;

cout << "最大公约数为" << func(m, n) << endl;

cout << "最小公倍数为" << m \* n / func(m, n) << 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>

#include<iomanip>

using namespace std;

bool is\_prime(int num) {

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

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

}

return true;

}

void is\_prime() {

int count = 2;

int row = 2;

cout << "2 3 ";

for (int i = 5; count < 200; i += 2) {

bool isPrime = true;

for (int j = 2; j <= sqrt(i); j++) {

if (i % j == 0) {

isPrime = false;

break;

}

}

if (isPrime) {

cout << i << " ";

count++;

row++;

if (row == 10) {

cout << endl;

row = 0;

}

}

}

cout << endl;

}

int main() {

int a = 0;

cin >> a;

if (is\_prime(a) == 1) cout << "true" << endl;

else cout << "false" << endl;

is\_prime();

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）

**3与4选一个完成**

**T4.**#pragma once

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

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

#include<iostream>

#include"mytriangle.h"

using namespace std;

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

if (side1 + side2 > side3 && side1 + side3 > side2 && side2 + side3 > side1 && abs(side1 - side2) < side3 && abs(side1 - side3) < side2 && abs(side2 - side3) < side1) {

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

return true;

}

else return false;

}

else return false;

};

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

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

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

};

#include<iostream>

#include<iomanip>

#include"mytriangle.h"

using namespace std;

int main() {

double a, b, c;

cin >> a >> b >> c;

if (is\_valid(a, b, c)) {

cout << "Size is: " << area(a, b, c);

}

else cout << "Not A Triangle";

return 0;

}

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>

#include<iomanip>

using namespace std;

int eat(int num,int times) {

if (times == 9) {

num = (num + 1) \* 2;

return num;

}

num = (eat(num, times + 1) + 1) \* 2;

return num;

}

int main() {

cout<<"第一天摘了"<<eat(1,1)<<"个桃子";

return 0;

}

输出结果：1534

**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？

函数的类型本身就是为了识别返回值的类型。

1. 本实验中主函数调用函数时采用的是何种传递方式？

值传递。

**四、算法分析，程序结果**

**T2.2 3 5 7 11 13 17 19 23 29**

**31 37 41 43 47 53 59 61 67 71**

**73 79 83 89 97 101 103 107 109 113**

**127 131 137 139 149 151 157 163 167 173**

**179 181 191 193 197 199 211 223 227 229**

**233 239 241 251 257 263 269 271 277 281**

**283 293 307 311 313 317 331 337 347 349**

**353 359 367 373 379 383 389 397 401 409**

**419 421 431 433 439 443 449 457 461 463**

**467 479 487 491 499 503 509 521 523 541**

**547 557 563 569 571 577 587 593 599 601**

**607 613 617 619 631 641 643 647 653 659**

**661 673 677 683 691 701 709 719 727 733**

**739 743 751 757 761 769 773 787 797 809**

**811 821 823 827 829 839 853 857 859 863**

**877 881 883 887 907 911 919 929 937 941**

**947 953 967 971 977 983 991 997 1009 1013**

**1019 1021 1031 1033 1039 1049 1051 1061 1063 1069**

**1087 1091 1093 1097 1103 1109 1117 1123 1129 1151**

**1153 1163 1171 1181 1187 1193 1201 1213 1217 1223**

**T5.第一天摘了1534个桃子**

**五、遇到的问题与解决方法**

**头函数的使用与函数的调用 寻找参考资料**

1. **体会**

**明确了函数的使用方法，注意到了头文件的使用与代码的模块化处理**

**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

#include<iostream>

#include<iomanip>

using namespace std;

int main() {

int num[10], a, index = 1;

cin >> a;

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

num[i] = a;

}

for (int i = 1; i <= 9; i++) {

int cur;

bool again = false;

cin >> cur;

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

if (num[j] == cur) {

again = true;

break;

}

}

if(again==false){

num[index] = cur;

index++;

}

}

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

cout << num[i] << " ";

}

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>

#include<iomanip>

using namespace std;

void sort(double num[]) {

bool changed = true;

do

{

changed = false;

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

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

{

int cur = num[j];

num[j] = num[j + 1];

num[j + 1] = cur;

changed = true;

}

} while (changed);

}

int main() {

double num[10];

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

cin >> num[i];

}

sort(num);

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

cout << num[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>

#include<iomanip>

using namespace std;

int main() {

bool chest[101];

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

chest[i] = false;

}

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

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

chest[j] = !chest[j];

}

}

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

if (chest[i] == true) cout << "L" << i << " ";

}

return 0;

}

输出结果：L1 L4 L9 L16 L25 L36 L49 L64 L81 L100

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>

#include<iomanip>

using namespace std;

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

int index1 = 0, index2 = 0, index = 0;

while (index1 <= size1-1 && index2 <= size2-1) {

if (list1[index1] <= list2[index2]) {

list3[index] = list1[index1];

index1++; index++;

}

else {

list3[index] = list2[index2];

index2++; index++;

}

}

for (int i = index; i <= size1 + size2; i++) {

if (index1 == size1) {

list3[index] = list2[index2];

index2++; index++;

}

else {

list3[index] = list1[index1];

index1++; index++;

}

}

}

int main() {

int size1, size2;

cin >> size1;

int\* list1 = new int[size1];

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

cin >> list1[i];

}

cin >> size2;

int\* list2 = new int[size2];

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

cin >> list2[i];

}

int\* list3 = new int[size1 + size2];

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

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

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

}

delete list1, list2, 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 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

#include <stdio.h>

#include <string.h>

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

int main() {

char s1[100], s2[100];

printf("请输入字符串s2: ");

fgets(s2, sizeof(s2), stdin);

s2[strcspn(s2, "\n")] = '\0';

printf("请输入字符串s1: ");

fgets(s1, sizeof(s1), stdin);

s1[strcspn(s1, "\n")] = '\0';

int index = indexOf(s1, s2);

if (index != -1) {

printf("字符串 \"%s\" 在字符串 \"%s\" 中的起始下标是: %d\n", s1, s2, index);

}

else {

printf("字符串 \"%s\" 不是字符串 \"%s\" 的子串\n", s1, s2);

}

return 0;

}

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

const char\* ptr = strstr(s2, s1);

if (ptr != NULL) {

return ptr - s2;

}

else {

return -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

#include<iostream>

#include<string>

using namespace std;

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

for (int i = 0; i < strlen(s); i++) {

if (s[i] >= 65 && s[i] <= 90) {

counts[s[i] - 65]++;

}

else {

counts[s[i] - 97]++;

}

}

}

int main() {

char s[100];

int counts[26];

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

counts[i] = 0;

}

cout << "Enter a string:";

fgets(s, sizeof(s), stdin);

count(s, counts);

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

if (counts[i] != 0) {

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

}

}

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 00000031A87AF524 00000031A87AF54400000031A87AF524 5 00000031A87AF544 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[]={1,2,3,4};

return list;

}

void main()

{

int \*p=f();

cout<<p[0]<<endl;

cout<<p[1]<<endl;

}

#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;

}

int 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 <stdio.h>

#include <string.h>

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

int main() {

char s1[100], s2[100];

printf("请输入字符串s2: ");

fgets(s2, sizeof(s2), stdin);

s2[strcspn(s2, "\n")] = '\0';

printf("请输入字符串s1: ");

fgets(s1, sizeof(s1), stdin);

s1[strcspn(s1, "\n")] = '\0';

int index = indexOf(s1, s2);

if (index != -1) {

printf("字符串 \"%s\" 在字符串 \"%s\" 中的起始下标是: %d\n", s1, s2, index);

}

else {

printf("字符串 \"%s\" 不是字符串 \"%s\" 的子串\n", s1, s2);

}

return 0;

}

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

const char\* ptr = strstr(s2, s1);

if (ptr != NULL) {

return ptr - s2;

}

else {

return -1;

}

}

(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 <stdio.h>

#include <string.h>

using namespace std;

int parseHex(const char\* const hexString) {

int ans = 0, digit = 0;

for (int i = strlen(hexString) - 2; i >= 0; i--) {

int cur = 0;

if (hexString[i] >= 65 && hexString[i] <= 90) {

cur = hexString[i] - 55;

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

cur = cur \* 16;

}

}

else {

cur = hexString[i] - 48;

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

cur = cur \* 16;

}

}

ans = ans + cur;

digit++;

}

return ans;

}

int main() {

char num[100];

fgets(num, sizeof(num), stdin);

cout << parseHex(num);

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>

#include <stdio.h>

#include <string.h>

using namespace std;

void sorting(int\* num,int n) {

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

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

if (num[j - 1] > num[j]) {

int cur = num[j - 1];

num[j - 1] = num[j];

num[j] = cur;

}

}

}

}

int main() {

int n = 0;

cin >> n;

int\* num = new int[n];

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

cin >> num[i];

}

sorting(num,n);

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

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

}

delete[] num;

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**

**如上**

**四、遇到的问题与解决方法**

**字符串的处理：查表得出字符串转字符数组的函数，并学会了子串问题的一般解决方法**

**五、体会**

加深了对数组与指针的理解，能运用指针，引用，动态内存分配等方法优化程序