**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软件工程2101

学 号： 8203210810

姓 名： 曾凡硕

**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？

函数中返回的值与函数类型一致的原因在于，C++中的函数定义了返回类型，这个类型表示了函数将返回的数据类型。函数执行后，其结果会通过 return 语句返回给调用者，因此返回的值需要与函数的返回类型一致，以符合函数声明和调用的期望。

例如，如果函数声明为 int add(int a, int b);，那么在函数体内，return 语句返回的值必须是 int 类型。

2. 本实验中主函数调用函数时采用的是何种传递方式？

本实验中主函数调用函数时采用的是值传递方式。在值传递中，实参的值被复制给形参，即在函数调用时，被传递给函数的是实参的值而不是实参本身。在函数内部对形参的修改不会影响到实参。

例如，在判断素数的函数调用中，主函数传递整数值给 is\_prime 函数，而在摄氏温度到华氏温度的转换函数中，主函数将摄氏温度的值传递给 celsius\_to\_fah 函数。在这两种情况下，采用的都是值传递方式。

**四、算法分析，程序结果**

1、

#include <iostream>

using namespace std;

*// 函数声明*

void calculateGCDandLCM(int m, int n, int &gcd, int &lcm);

int main() {

*// 输入两个自然数 m 和 n*

    int m, n;

    cout << "输入两个自然数 m 和 n：" << std::endl;

    cin >> m >> n;

*// 计算最大公约数和最小公倍数*

    int gcd, lcm;

    calculateGCDandLCM(m, n, gcd, lcm);

*// 输出结果*

    cout << "最大公约数是：" << gcd << std::endl;

    cout << "最小公倍数是：" << lcm << std::endl;

    return 0;

}

*// 计算最大公约数和最小公倍数的函数*

void calculateGCDandLCM(int m, int n, int &gcd, int &lcm)

{

    int a = m;

    int b = n;

*// 辗转相除法求最大公约数*

    while (b != 0) {

        int temp = b;

        b = a % b;

        a = temp;

    }

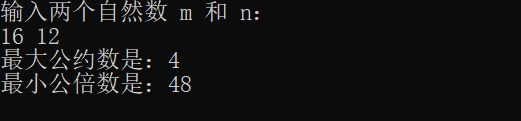
*// 最大公约数*

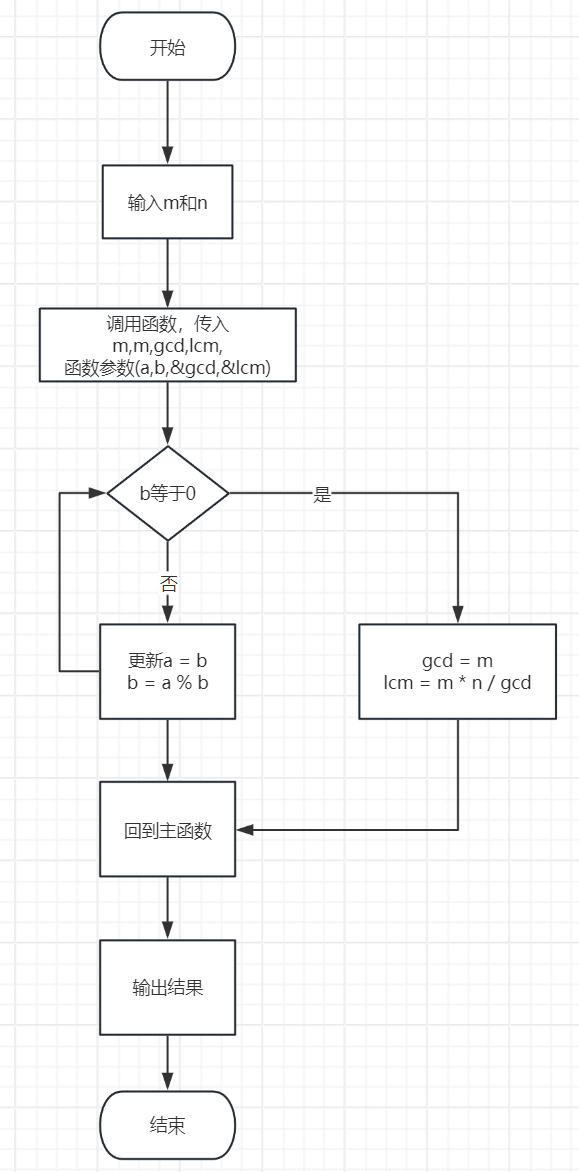
    gcd = a;

*// 最小公倍数 = m \* n / 最大公约数*

    lcm = (m / gcd) \* n;

}





2、

#include<iostream>

#include <iomanip>

using namespace std;

bool is\_prime(int num);

int main()

{

 int idx = 2;

 int cnt = 0;

 while(cnt != 200)

 {

  if(is\_prime(idx))

  {

   cout << setw(6) << idx;

   cnt ++;

   if(cnt% 10 == 0)

    cout << endl;

  }

  idx ++;

 }

}

bool is\_prime(int num)

{

 if(num == 1) return false;

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

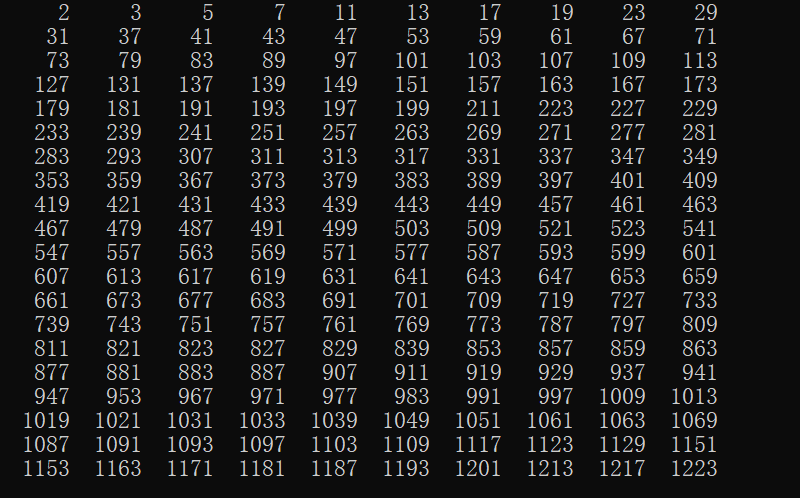
  if(num % i == 0)

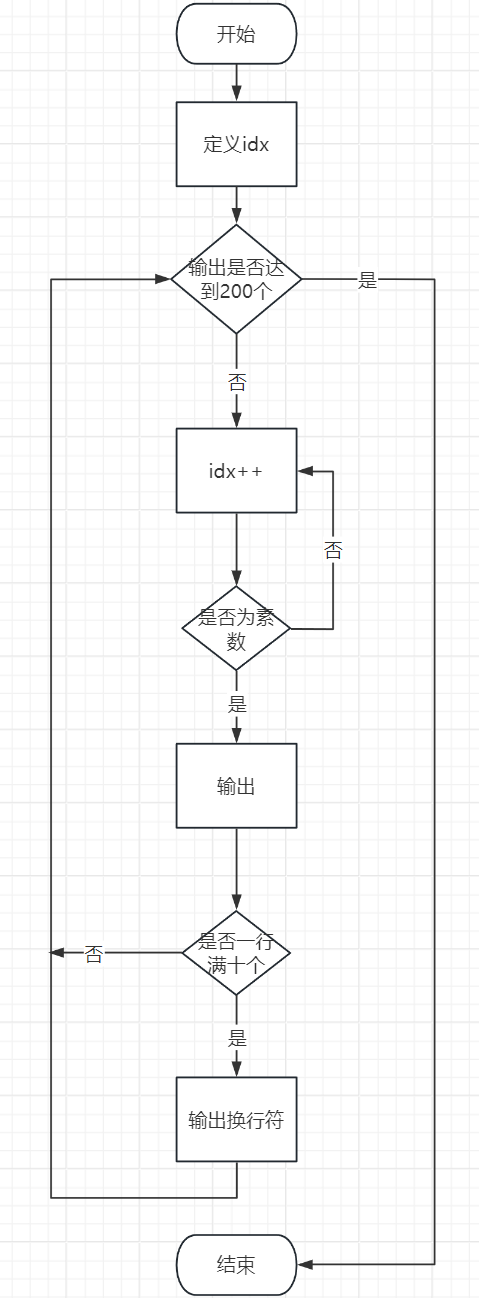
   return false;

 }

 return true;

}





3、

主程序：

#include<iostream>

#include "mytemperature.h"

#include<iomanip>

using namespace std;

int main()

{

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

    for (double celsius = 40.0; celsius >= 31.0; celsius -= 1.0) {

        cout << fixed << setprecision(1)

             << setw(5) << celsius << "       "

             << setw(9) << celsius\_to\_fah(celsius) << "        |   "

             << setw(9) << celsius + 80.0 << "       "

             << setw(9) << fahrenheit\_to\_cels(celsius + 80.0) << endl;

    }

    for (double fahrenheit = 110.0; fahrenheit >= 30.0; fahrenheit -= 1.0) {

        cout << fixed << setprecision(1)

            << setw(5)  << "              "

            << setw(9)  << "               |   "

            << setw(9) << fahrenheit << "       "

            << setw(9) << fahrenheit\_to\_cels(fahrenheit) << endl;

    }

    system("pause");

    return 0;

}

Mytemperature.h

#ifndef MYTEMPERATURE\_H

#define MYTEMPERATURE\_H

double celsius\_to\_fah(double cel);

double fahrenheit\_to\_cels(double fah);

#endif *// MYTEMPERATURE\_H*

Mytemperature.cpp

#include "mytemperature.h"

double celsius\_to\_fah(double cel) {

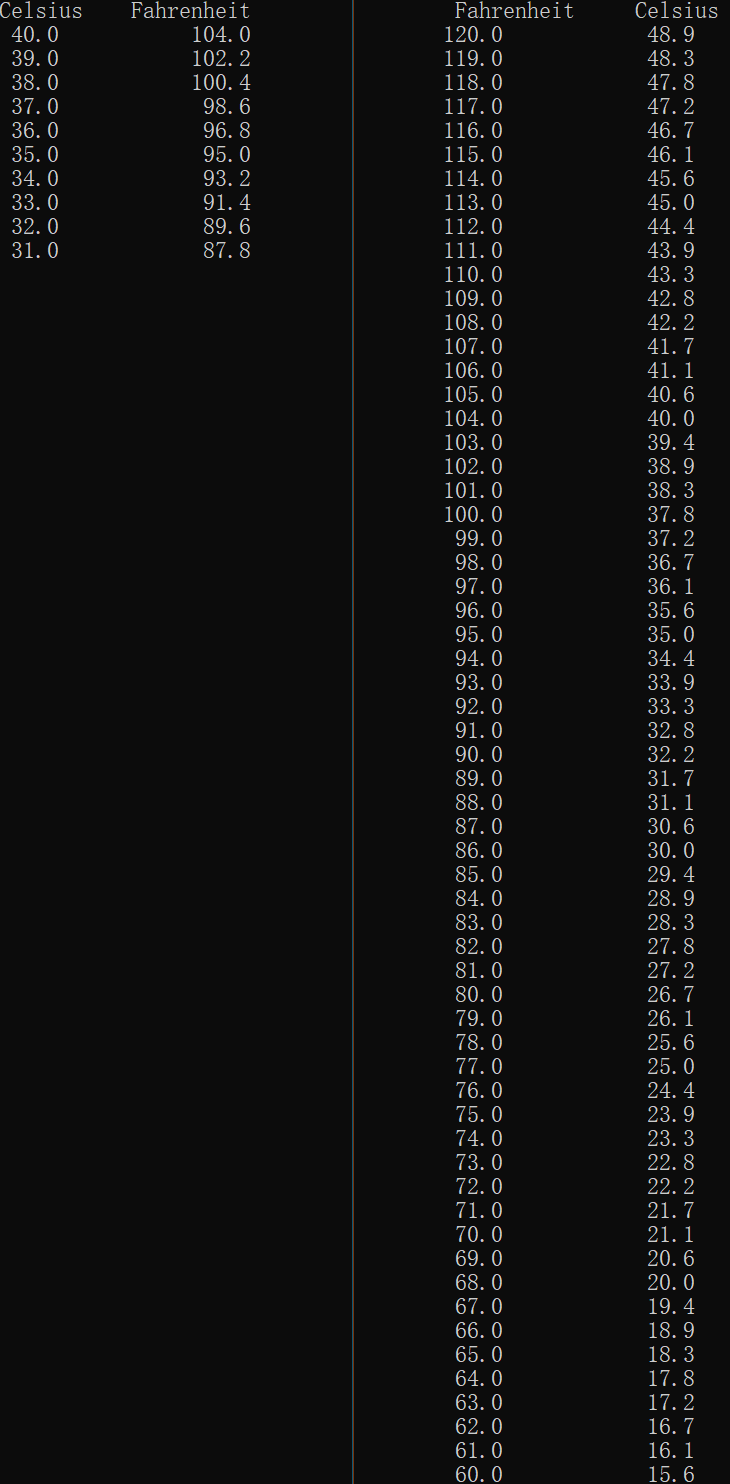
    return cel \* 9.0 / 5.0 + 32.0;

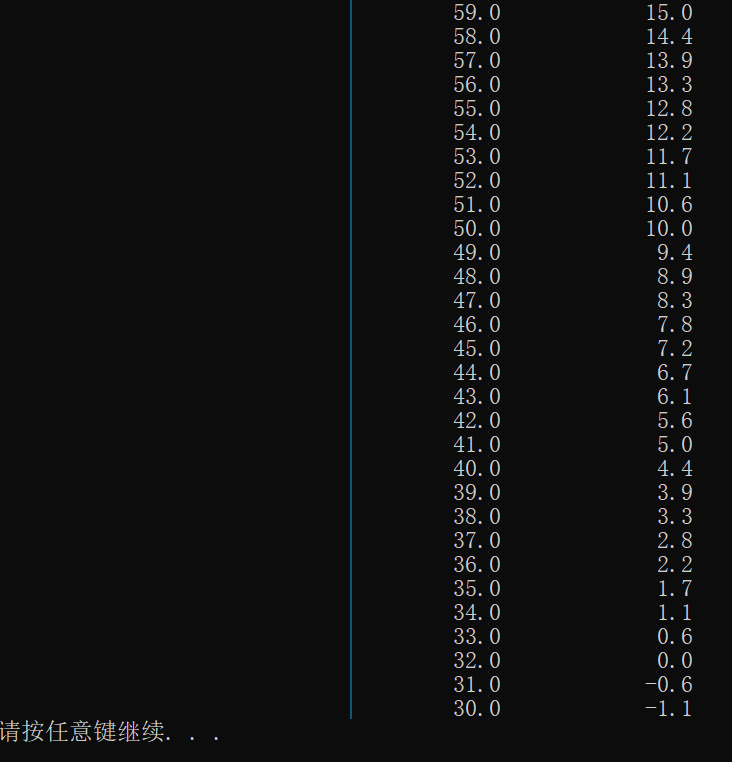
}

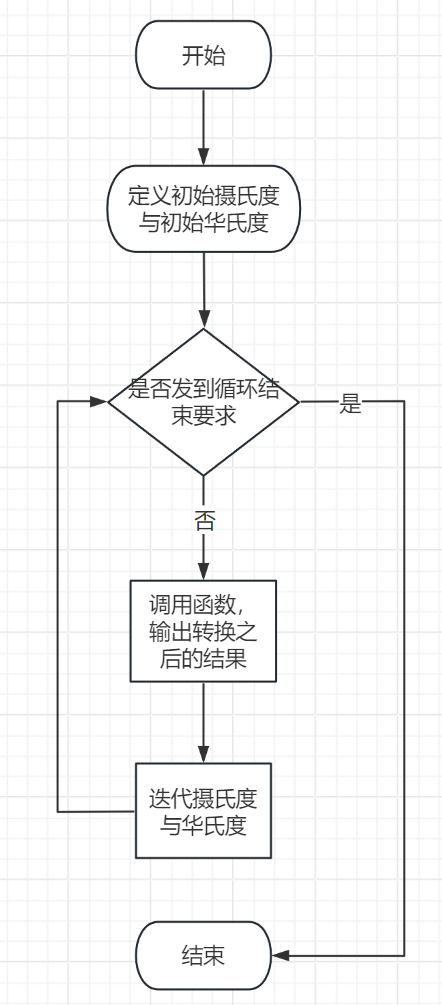
double fahrenheit\_to\_cels(double fah) {

    return (fah - 32.0) \* 5.0 / 9.0;

}







5、

#include<iostream>

using namespace std;

int peachNum(int day);

int main()

{

 cout << "猴子第一天摘了 " << peachNum(1) << " 个桃子" << endl;

 return 0;

}

int peachNum(int day)

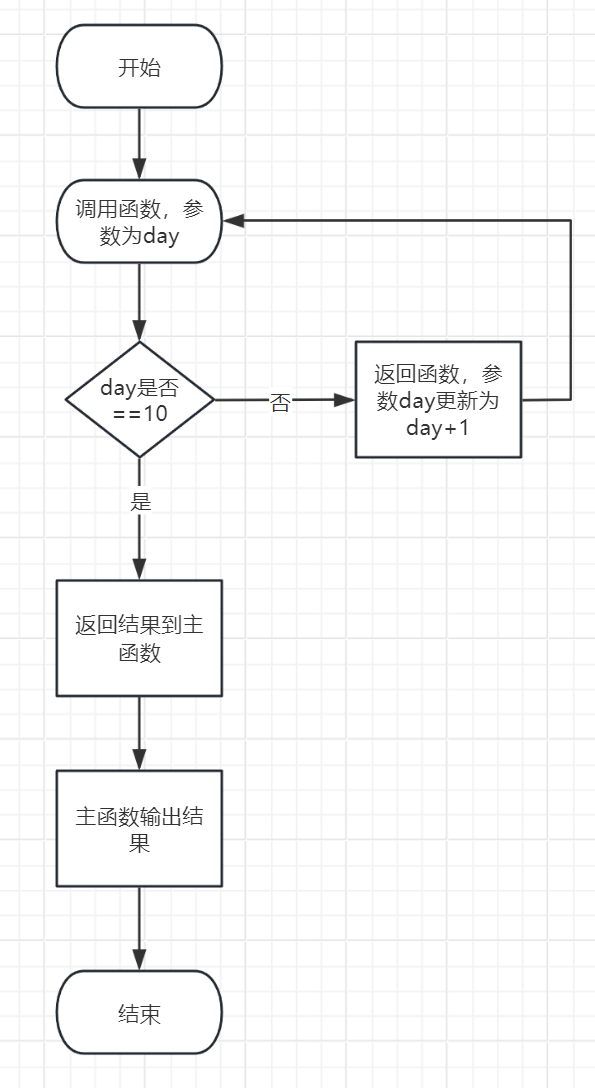
{

 if(day == 10) return 1;

 else return (peachNum(day + 1) + 1) \* 2;

}





**五、遇到的问题与解决方法**

1、忘记摄氏度转华氏度以及华氏度转摄氏度的公式，网上搜索后解决了问题。

2、设置每个输出占用4个字符的时候使用setw(4)函数需要引用头文件<iomanip>

**六、体会**

函数的定义与声明： 了解了函数的定义和声明方法，以及如何正确使用函数头进行声明，使得函数可以在其他文件中被引用。

函数的编写要求： 实践了函数的编写要求，包括函数的参数传递和返回值的设定。学会了如何在函数中使用引用参数，以便修改传入参数的值。

多文件编程方法： 通过实现多个文件的程序，深刻理解了多文件编程的重要性。头文件的声明和源文件的定义分离，提高了代码的可维护性和可读性。

素数判断和输出： 编写了一个判断素数的函数，并成功找出并输出了前200个素数。这锻炼了我对函数的调用和使用的能力。

温度转换程序： 学到了如何将函数的声明和定义分离到头文件和源文件中，同时编写了一个测试程序，通过调用函数实现摄氏温度到华氏温度的转换。

递归实现： 学会了使用递归实现问题。递归的思想在解决特定类型的问题时具有独特的优势。

**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\_\_\_ 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;

}

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）**

1、

#include<iostream>

using namespace std;

const int N = 15;

int a[N];

void save(int x, int cnt);

int idx = 0;

int main()

{

 for(int i = 0; i < 10; i ++)

 {

  int x;

  cin >> x;

  save(x, i);

 }

 for(int i = 0; i < a[10]; i ++)

 {

  cout << a[i] <<" ";

 }

 return 0;

}

void save(int x, int cnt)

{

 bool flag = true;

 if(cnt > 0)

 {

  for(int i = 0; i <= cnt - 1; i ++)

  {

   if(a[i] == x)

   {

    flag = false;

    break;

   }

  }

  if(flag)

  {

   a[idx ++] = x;

   a[10] ++;

  }

 }

 else

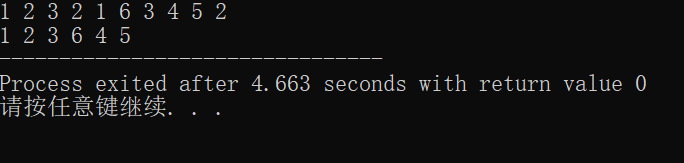
 {

  a[idx ++] = x;

  a[10] ++;

 }

}



2、

#include <iostream>

using namespace std;

*// 交换数组中两个元素的值*

void swap(double &a, double &b);

*// 起泡排序函数*

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

int main() {

    const int listSize = 10;

    double myList[listSize];

*// 输入数组元素*

    cout << "请输入10个双精度数字：" << endl;

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

        cin >> myList[i];

    }

*// 调用排序函数*

    bubbleSort(myList, listSize);

*// 输出排列后的数字*

    cout << "排列后的数字为：" << endl;

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

        cout << myList[i] << " ";

    }

    cout << endl;

    return 0;

}

void swap(double &a, double &b) {

    double temp = a;

    a = b;

    b = temp;

}

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]) {

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

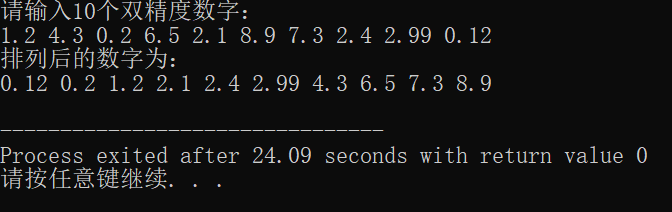
                changed = true;

            }

        }

    } while (changed);

}



3、

#include <iostream>

using namespace std;

int main() {

    const int numLockers = 100;

    bool lockers[numLockers] = {false};

    for (int student = 1; student <= numLockers; student ++) {

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

            lockers[locker] = !lockers[locker];

        }

    }

    cout << "开着的储物柜号码为：";

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

        if (lockers[i]) {

            cout << i + 1 << " ";

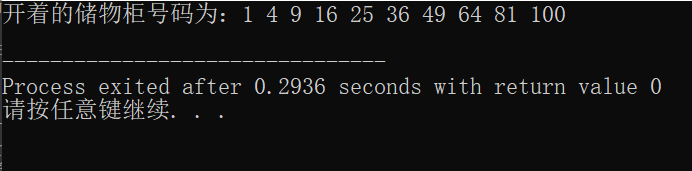
        }

    }

    cout << endl;

    return 0;

}



4、

#include<iostream>

using namespace std;

const int N = 170;

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

int main()

{

 int list1[N], list2[N], list3[N];

 int size1, size2, size3;

 cout << "Enter list1:";

 cin >>size1;

 for(int i = 0; i < size1; i ++)

  cin >> list1[i];

 cout << "Enter list2:";

 cin >>size2;

 for(int i = 0; i < size2; i ++)

  cin >> list2[i];

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

 cout << "The merged list is ";

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

  cout << list3[i] <<" ";

 return 0;

}

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) {

        list3[k++] = list1[i++];

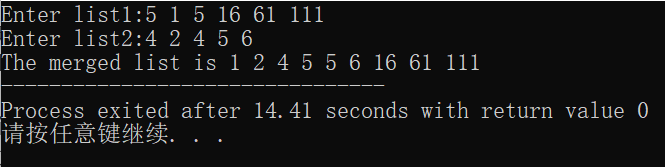
    }

    while (j < size2) {

        list3[k++] = list2[j++];

    }

}



5、

#include<iostream>

#include <cstring>

using namespace std;

const int N = 100;

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

int main()

{

 char s1[N], s2[N];

    cout << "Enter the first string: ";

    cin.getline(s1, N);

    cout << "Enter the second string: ";

    cin.getline(s2, N);

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

}

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

{

 int len1 = strlen(s1);

    int len2 = strlen(s2);

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

        int j;

        for (j = 0; j < len1; ++j) {

            if (s2[i + j] != s1[j]) {

                break;

            }

        }

        if (j == len1) {

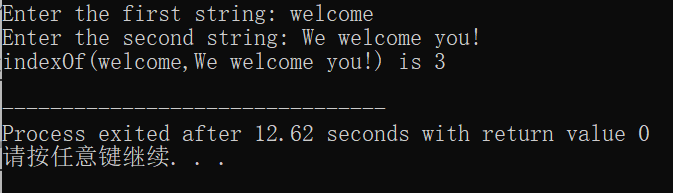
            return i;

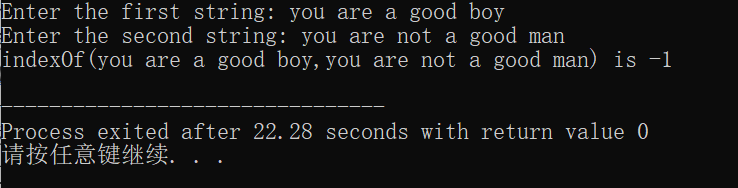
        }

    }

    return -1;

}





6、

#include <iostream>

#include <string>

using namespace std;

void count(string str, int counts[]){

 for (int i = 0; i < 26; i++) counts[i] = 0;

 for (int i = 0; i < str.length(); i++)

 {

  if (str[i] >= 'A' && str[i] <= 'Z')

   counts[str[i] - 65]++;

  if (str[i] >= 'a' && str[i] <= 'z')

   counts[str[i] - 97]++;

 }

}

void display(const int counts[]){

 for (int i = 0; i < 26; i++)

 {

  if (counts[i] != 0)

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

 }

}

int main()

{

 string str;

 int counts[26];

 cout << "Enter a string : ";

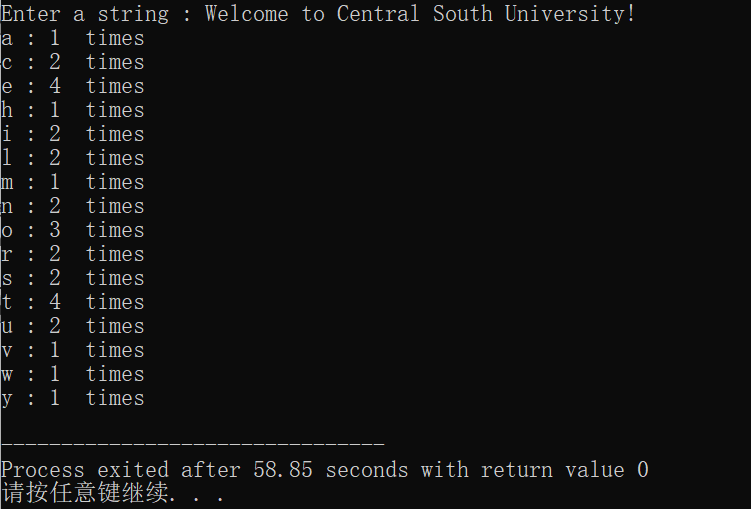
 getline(cin, str);

 count(str, counts);

 display(counts);

 return 0;

}

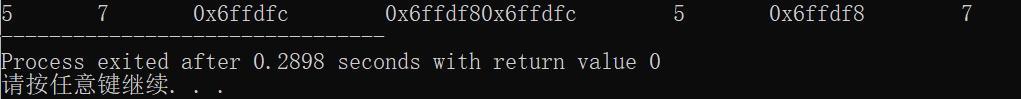


**(二)**

1、

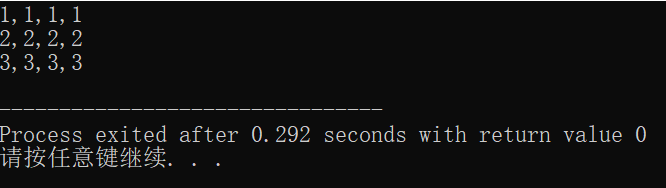
(1)

结果如下：



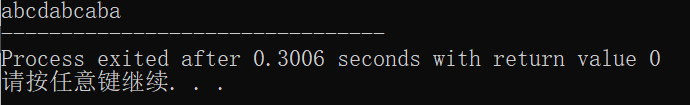
(2)

结果如下：



(3)

结果如下：



(4)

修改代码如下：

#include<iostream>

using namespace std;

int \*f()

{

 int\* list = new int[4]{1, 2, 3, 4};

 return list;

}

int main()

{

 int \*p=f();

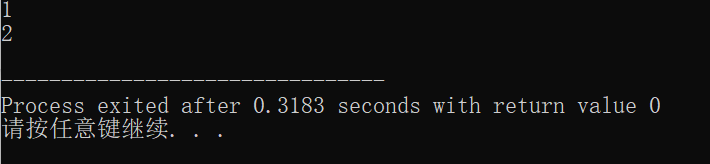
 cout<<p[0]<<endl;

 cout<<p[1]<<endl;

 delete []p;

}

结果如下：



2、

(1)

#include<iostream>

#include <cstring>

using namespace std;

const int N = 100;

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

int main()

{

 char s1[N], s2[N];

    cout << "Enter the first string: ";

    cin.getline(s1, N);

    cout << "Enter the second string: ";

    cin.getline(s2, N);

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

}

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

{

 int len1 = strlen(s1);

    int len2 = strlen(s2);

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

        int j;

        for (j = 0; j < len1; ++j) {

            if (s2[i + j] != s1[j]) {

                break;

            }

        }

        if (j == len1) {

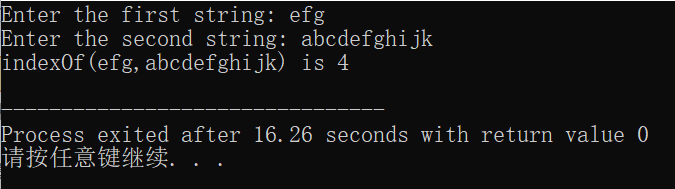
            return i;

        }

    }

    return -1;

}



(2)

#include <iostream>

#include <cstring>

#include <cctype>

using namespace std;

int parseHex(const char \*const hexString) {

    int length = strlen(hexString);

    int result = 0;

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

        char ch = toupper(hexString[i]);

        if (isdigit(ch)) {

            result = result \* 16 + (ch - '0');

        } else if (ch >= 'A' && ch <= 'F') {

            result = result \* 16 + (ch - 'A' + 10);

        } else {

            cerr << "无效的十六进制字符: " << ch << endl;

            return -1;

        }

    }

    return result;

}

int main() {

    const int maxSize = 100;

    char hexString[maxSize];

    cout << "输入16进制数: ";

    cin >> hexString;

    int result = parseHex(hexString);

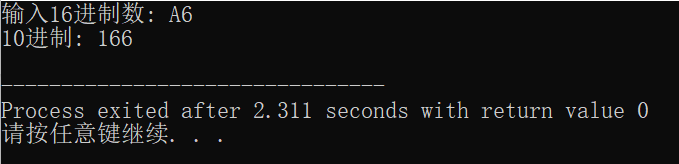
    if (result != -1) {

        cout << "10进制: " << result << endl;

    }

    return 0;

}



(3)

#include<iostream>

using namespace std;

void sort(int \*array, int n) {

 for (int i = 0; i < n-1; i++)

 {

  for (int j = 0; j < n - i - 1; j++)

  {

   if (array[j] > array[j + 1])

   {

    int temp = array[j];

    array[j] = array[j + 1];

    array[j + 1] = temp;

   }

  }

 }

}

int main(){

 int n;

 cout << "请输入数组长度：";

 cin >> n;

 int\* array = new int[n];

 cout << "请输入各元素：";

 for (int i = 0; i < n; i++)

  cin >> array[i];

 int index;

 cout << "请输入所要查询元素的下标（输入-1结束查询）：";

 cin >> index;

 while(index != -1)

 {

  cout << "array[" << index << "]为" << array[index] << endl;

  cout << "指针为：" << &array[index] << endl;

  cout << "请输入所要查询元素的下标（输入-1结束查询）：";

  cin >> index;

 }

 sort(array, n);

 cout << "排序后的数组为：" << endl;

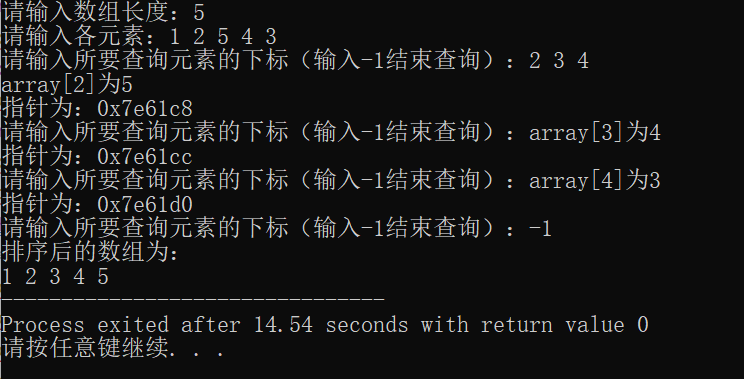
 for (int i = 0; i < n; i++)

  cout << array[i] << " ";

 delete[] array;

 return 0;

}



**四、遇到的问题与解决方法**

1.起初程序会输出很大的负数，这是因为b[i]没有初始化，以为没有初始化是null，而事实上里面存的是一个随机的数字，改正方法是重复的数字对应的b[i]赋值为=-1。

2.因为起泡排序已编写多次，已比较熟练，没有遇到什么问题。

3、运行时总是会报以上错误，最后发现是数组越界，原因只是cabinet[j-1] = !cabinet[j-1];写成了cabinet[j] = !cabinet[j];

4．因为学习了算法课，这个函数就是归并排序里的一个函数，因此编写思路很明确，主要不要忘记其中一个数组排列完毕后另一个数组有剩余的情况。

5.当母串匹配到末尾时，有可能会发生越界，解决方案是只需匹配到母串还剩余子串长度时。

6.利用ASCII码实现统计，避免了代码的大量重复书写。

7.功能和实现方法和405类似，只需要做简单的修改。

8.起初输出是一个随机的数字，花费了很长时间，最后意识到只有第一个字符输入成功了，不能直接cin，而用了cin.getline。

9.没有问题。

**五、体会**

在第一个程序中，指针的概念得到了初步的体验。pi 和 pj 分别存储了 i 和 j 的地址，而 &i 和 &j 分别表示 i 和 j 的地址。这里学到了指针变量的定义和使用。

在第二个程序中，数组名 a 本身就是数组首地址，而 p 是指向数组首地址的指针。通过 \*(p+i) 或者 p[i] 可以访问数组元素。这里强调了数组和指针之间的关系。

在第二个程序中，通过 p=a; 将数组首地址赋给指针，强调了数组名即指向数组首地址的指针。

第一个程序中的 pi 和 pj 是指针，但它们被用来存储整数变量 i 和 j 的地址。这并不是通常的使用方式，而且在实际应用中可能会导致问题，因为 i 和 j 是栈上的局部变量，它们的地址在函数结束后可能会变得无效。

动态内存分配与释放：

在第四个程序中，如果没有使用动态内存分配，而是直接返回局部数组的地址，就会出现指针指向已释放内存的情况。通过使用 new 分配内存，并在 main 函数中使用 delete 释放内存，避免了这个问题。

在第三个程序中，函数 f 递归输出字符串的前缀，强调了递归函数的定义和调用。递归是一种重要的编程技巧，这里体验了一个简单的递归过程。

在实验的第二部分，进行了一系列的程序设计。包括字符串的比较、16进制到10进制的转换、动态数组的排序和释放等。这些任务帮助理解了指针、数组、字符串处理和动态内存的使用。