**实验5 10-29oj**

1. 实验的目的和要求

学习c++编程的原理和方法

1. 实验内容
2. 实验准备

C++中数组的使用方法，一维，二维的定义，初始化

C++中基本的排序算法，有序数组的查找等问题

(二)实验项目

A

题目描述

输入一个正整数n，然后输入n个整数，将他们从大到小排序后输出

分析：

排序题目直接用快排算法就行了

源程序

using namespace std;

void qsort(int,int);

int a[1000],n;

int main()

{

cin>>n;

for(int i=1;i<=n;i++)cin>>a[i];

qsort(1,n);

for(int i=1;i<=n;i++)cout<<a[i]<<' ';

return 0;

}

void qsort(int l,int r)

{

int mid=a[(l+r)/2],tmp,i=l,j=r;

do{

while(a[i]>mid)i++;

while(a[j]<mid)j--;

if(i<=j)

{

tmp=a[i];

a[i]=a[j];

a[j]=tmp;

i++;

j--;

}

}while(i<=j);

if(i<r)qsort(i,r);

if(j>l)qsort(l,j);

}

测试数据·

Input

15 -3 0 2 1 8 55 9 1 5 44 54 87 65 101 -100

Output

101 87 65 55 54 44 9 8 5 2 1 1 0 -3 -100

B

题目描述

在矩阵中，一个元素在所在行中式最大值，在所在列中式最小值，则被称为鞍点。   
输入两个正整数m,n，然后输入该m行n列矩阵mat中的元素，如果找到mat的鞍点，则输出他的下标，否则输出Not found

分析

这道题首先要理清楚思路应该是两层循环，依据题目的意思首先先确定每一行中的最大的数，再确定好后判断该数在所在列中是否是最小的数。最后设置一个标识符，只要找到鞍点就设置为true否则false

源程序

#include<iostream>

using namespace std;

int main()

{

const int maxn=100;

int a[maxn][maxn],m,n,c,p;

bool f,d;

cin>>m>>n;

d=false;

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

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

cin>>a[i][j];

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

{

c=a[i][1];p=1;

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

if(a[i][j]>c)

{

c=a[i][j];

p=j;

}

f=true;

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

if(a[j][p]<a[i][p])f=false;

if(f)

{

cout<<"mat["<<i<<"]["<<p<<"]="<<c;

d=f;

}

}

if(!d)cout<<"Not Found";

return 0;

}

测试数据

Input

3 4   
3 2 1 4   
5 1 6 8   
6 7 8 9

Output

mat[0][3]=4

Input

3 4

3 2 1 9

4 7 6 8

5 1 8 3

Output

Not Found

C

题目描述

有n盏灯，编号为1-n，第一个人把所有灯打开，第二个人按下所有编号为2的倍数的开关，第三个人按下所有编号为3的倍数的开关，依此类推。   
一共有k个人，问最后哪些灯开着，编写一个程序，输入n和k，输出开着的灯的编号 （k<=n<=1000）.

分析

模拟法，用一个布尔型数组，循环k次，对该数组依此按照题目意思操作，最后只要数组数据为真则输出即可

源程序

#include<iostream>

using namespace std;

int main()

{

const int maxn=10000;

bool f[maxn]={0};

int n,k;

cin>>n>>k;

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

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

if((j%i)==0)f[j]=!f[j];

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

if(f[i])cout<<i<<' ';

return 0;

}

测试数据

Input

15 9

Output

1 4 9 10 11 12 13 14 15

D

题目描述

计算1900年1月1日 ~ 1900+n-1年12月31日中13号落在周一到周日的次数（n<400)   
已知1900年1月1日是星期一

分析

虽然本题使用的是模拟法但是编程起来难度不小，并且极其繁琐，为了简化程序使用两个函数ju函数用于判断闰年，re函数则主要用来执行过程，对于re函数中的几个参数做一定解释flag为闰年标识参数，n是年份，k用于表示星期几的几。随后在执行操作中i用于表示月份，date则是日期，将date对于month做同余操作就可以准确表示日期，month中的数值已经初始化，最后只要是13号就day[k]++就可以实现。

源程序

#include <iostream>

using namespace std;

int day[10];

int month[2][12]={{31,28,31,30,31,30,31,31,30,31,30,31},{31,29,31,30,31,30,31,31,30,31,30,31}};

int ju(int n)

{

if((n%4==0&&n%100!=0)||(n%400==0))

return 1;

return 0;

}

int re(int flag,int n,int k)

{ int date=1;

for(int i=0;i<12;i++)

for(int j=1;j<=month[flag][i];j++)

{date++;

date%=month[flag][i]; k++;k%=7;

if(date==13)

day[k]++;

}

return k;}

int main()

{ int n,k=1;

cin>>n;

for(int i=1900;i<=1900+n-1;i++)

{k=re(ju(i),i,k);}

cout<<day[1]<<" "<<day[2]<<" "<<day[3]<<" "<<day[4]<<" "<<day[5]<<" "<<day[6]<<" "<<day[0]<<endl;

return 0;

}

测试数据

Input

500

Output

857 856 859 854 860 857 857

1. 实验小结

本次实验着重于对于数组的熟悉和使用，主要涉及的算法是模拟法法，第一题主要是了解排序算法，第二三题属于较简单的模拟法。只要理清楚思路即可。最后一题模拟法略有难度，可以使用函数简化，必要的分析是非常重要的。