2014 级 C++第七次上机解题报告

14211076 马国瑞

A jhljx 转圈圈

Problem Description

jhljx 最近学了二维数组，决定用二维数组来打印一些图形。他找到了一块正方形的小木板，木板上有一些小方格。他决定把一些数字填入到这些小方格里。但是他喜欢转圈圈，他喜欢让这些数字螺旋的排列在正方形的小方格里。听起来很有趣，希望你们来帮助他吧。

Input

输入多组测试数据。
每组测试数据为一个数字 n。(1<=n<=1000)

Output

输出这个正方形的图案，并且计算出主对角线(左上角到右下角)上元素的值。(输出结果详见样例)

Sample Input

```
3
5
```

Sample Output

```
1 2 3
8 9 4
7 6 5
15
1 2 3 4 5
16 17 18 19 6
15 24 25 20 7
14 23 22 21 8
13 12 11 10 9
73
```

二、解题思路

这道题使用二维数组标记正方形的横纵坐标，一圈一圈从外到内套，从左上角到右上角、右下角、左下角再到终点，然后再进行 n/2 圈，使用嵌套 for 循环实现即可，注意 for 循环起点与终点。注意如果 n 为奇数则必须给整个图形最中间的数赋值 n×n。

还有，大号的数组要设置成全局变量，并且适时清空数组以免出错。

三、参考代码

```
#include<iostream>
#include<cstring>              //清空数组函数用
using namespace std;
int a[1001][1001];
int main()
{
    int n;
    int num, sum;
    while(cin>>n)
    {
        num=1;
        sum=0;
        for(int p=0; p<n/2; p++)          //控制第几圈
        {
            for(int q=p; q<n-p; q++)      //一圈中最上面一行从左到右
            {
                a[p][q]=num;
                num++;
            }
            for(int r=p+1;r<n-p;r++)      //一圈中最右边一列从上到下
            {
                a[r][n-p-1]=num;
                num++;
            }
            for(int s=n-p-2;s>=p;s--)     //一圈中最下边一行从右到左
            {
                a[n-p-1][s]=num;
                num++;
            }
            for(int t=n-p-2;t>p;t--)       //一圈中最左边一列从下到上
            {
                a[t][p]=num;
                num++;
            }
        }
```

```
        if(n%2!=0)
            a[n/2][n/2]=n*n;
        for(int i1=0; i1<n; i1++)                    //输出阵列
        {
            for(int i2=0; i2<n; i2++)
            {
                cout << a[i1][i2];
                if(i2!=n-1)
                    cout<<" ";
            }
            sum+=a[i1][i1];                          //计算最后的数
            cout << endl;
        }
        cout<<sum<<endl;
        memset(a,0,sizeof(a));                       //清空数组的姿势
    }
}
```

B jhljx 下楼梯

Problem Description

jhljx 最近喜欢上下楼梯玩耍，他找到了这样一种楼梯。

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
...
```

这种楼梯好神奇吖。。
他希望你们用会二维数组。jhljx 温馨提醒：不要耍小聪明来水过。

Input

输入多组数据。
每组数据第一行为一个整数 n(1<=n<=100),表示这个楼梯有多少层。第二行为一个整数 k(1<=k<=100)，表示有 k 次访问。
后边 k 行每行为两个正整数 x,y。表示第 x 层，第 y 列的位置。(保证 x,y 在 int 范围内)

Outut

每组数据输出一行，如果 a[x][y]存在，输出这个位置的数，反之输出-1。

Sample Input

```
5
5
1 1
3 2
4 3
5 3
5 7
```

Sample Output

```
1
2
3
6
-1
```

Hint

请用 long long 类型的二维数组实现

二、解题思路

本题的数组依旧要使用全局变量。

发现到，此题使用杨辉三角可以“创造”所谓的”楼梯“，用二项式定理在理论上也可以得到一样的值。A[x][y]不存在的情况为：x 不在[1,n]范围，y 不在[1,x]范围。

三、参考代码

```
#include<iostream>
using namespace std;
const int n_ = 101;
long long a[n_][n_];
void f(long long n)                    //数组的构建
{
    a[0][0]=1;
    a[1][0]=1;
    a[1][1]=1;
    for ( long long b=2; b<n; b++ )
```

```
    {
        a[b][0]=1;
        a[b][b]=1;
        for ( long long c=0; c<b; c++ )
        {
            a[b][c]=a[b-1][c-1]+a[b-1][c];
        }
    }
}
int main()
{
    long long n, k, x, y;
    bool z;
    while ( cin >> n )
    {
        cin >> k;
        f(n);
        while ( k-- )
        {
            z = true;
            cin >> x >> y;
            if ( x<1 || x>n || y<1 || y>x )
                cout << "-1" << endl;
            else
                cout << a[x-1][y-1] << endl;
        }
    }
}
```

C jhljx 学排序

Problem Description

jhljx 学习了数组的冒泡排序,给你 n 个数，请你按照从大到小的顺序排列。

Input

输入多组数据。
每组数据一行为一个正整数 n(1<=n<=1000)。
接着输入 n 个正整数。(保证数字在 int 范围内)

Output

每组数据输出两行。保证每组的 n 个数从大到小排列，并输出冒泡排序中相邻两个数交换的次数。

Sample Input

```
5
6 -1 3 7 10
```

Sample Output

```
10 7 6 3 -1
8
```

Hint

请用冒泡排序实现。禁用 STL。

二、解题思路

冒泡排序，额，就是对 n 个数排序，扫描 n-1 遍，第 i 遍扫描 n-i 次对相邻两个数进行比较，如果不符合要求则交换这两个数，依此类推就可得到。

三、参考代码

```
#include<iostream>
using namespace std;
int main()
{
    int n, x, counter, mid;
    while ( cin >> n )
    {
        counter = 0;
        const int n_ = n;
        int a[n_];
        for ( int i=0; i<n; i++ )
        {
            cin >> x;
            a[i] = x;
        }
        for ( int r=n-1; r>0; r-- )
        {
            for ( int s=0; s<r; s++ )
            {
                if ( a[s]<a[s+1] )
                {
```

```
                    mid = a[s+1];
                    a[s+1]=a[s];
                    a[s]=mid;
                    counter++;
                }
            }
        }
        for ( int i1=0; i1<n; i1++ )
        {
            cout << a[i1];
            if ( i1 != n-1 )
                cout << " ";
        }
        cout << endl << counter << endl;
    }
}
```

D 糖果魔女

Problem Description

KamuiKirito 遇到一只萌萌哒怪物

这只萌萌哒怪物喜欢吃糖果。
现在有 n 块糖果。每块糖果有质量和含糖量。
现在这只怪物只能吃质量和为 k 的糖果。
而这只怪物每次会选择单位质量含糖量最高的糖果吃掉。
而如果这颗糖果她已经吃不下了，她就会停止并且放弃这颗糖果。
那么她会获得多少糖分呢。

Input

输入多组数据。
每组数据第一行为两个整数 n(1<=n<=1000),k（0<=k<=100000)表示有 n 块糖果,怪物最多可以吃质量和为 k 的糖果。

第二行为 n 个整数 ai(a1,a2,……an)，代表第 i 块糖果的质量。
第三行为 n 个正整数 bi(b1,b2,……bn)，代表第 i 块糖果的含糖量。

Outut

每组数据输出一行，为怪物最多能吃到的糖分。

Sample Input

3 4
1 2 3
3 2 1

Sample Output

5

二、解题思路

此题是基于对排序的考查。注意对所有的数**同时进行排序**，否则会出错（我开始就错在这里了）。
注意有怪物能把糖分全部吃完的情况，还有怪物吃不完的时候要把多加的一颗糖的糖分减去。

三、参考代码

```
#include<iostream>
using namespace std;
int main()
{
    long long a[1001];
    long long b[1001];
    long double c[1001];
    long long n, k, s1, s2, mid1, mid2, m, t;
    long double mid3;
    while ( cin >> n >> k )
    {

        s1 = 0;
        s2 = 0;
        for ( long long i1=0; i1<n; i1++ )
        {
            cin >> m;
            a[i1] = m;
            s1=s1+m;
        }
        for ( long long i2=0; i2<n; i2++ )
```

```
{
	cin >> t;
	b[i2] = t;
	s2=s2+t;
}
for ( long long i3=0; i3<n; i3++ )
{
	c[i3]=static_cast<long double>(b[i3])/static_cast<long double>(a[i3]);
}
if ( s1 <= k )                          //怪物能全吃完糖果
	cout << s2 << endl;
else
{
	int counter = 0;
	for ( long long r=n-1; r>0; r-- )
	{
		for ( long long s=0; s<r; s++ )
		{
			if ( c[s]<c[s+1] )              //同时排序
			{
				mid1 = a[s+1];
				a[s+1]=a[s];
				a[s]=mid1;
				mid2 = b[s+1];
				b[s+1]=b[s];
				b[s]=mid2;
				mid3 = c[s+1];
				c[s+1]=c[s];
				c[s]=mid3;
			}
		}
	}
	s1 = 0;
	s2 = 0;
	long long i=0;
	while (s1<=k)
	{
		s1+=a[i];
		s2+=b[i];
		i++;
	}
	if ( s1 > k )          //怪物吃不完的时候要把多加的一颗糖的糖分减去
		s2 -= b[i-1];
	cout << s2 << endl;
```

```
        }
    }
}
```

E 全能者的悖论

Problem Description

全能者能创造出一块他搬不动的石头吗？

现在有若干块石头，每块石头拥有质量和价值。请输出单位质量价值第二大的石头的序号。

Input

输入多组数据。
每组数据第一行为一个整数 n(2<=n<=1000),表示有 n 块石头。
第二行为 n 个整数 ai(a1,a2,……an)，代表第 i 个石头的价值。
第三行为 n 个正整数 bi(b1,b2,……bn)，代表第 i 个石头的质量。

Outut

每组数据输出一行，为单位质量价值第二大的石头的序号。

Sample Input

```
3
1 2 3
3 2 1
```

Sample Output

```
2
```

Hint

保证第一大和第二大不存在并列现象。

二、解题思路

本题只需将数组中所有元素扫两次找第一大数和第二大数即可，然后找第二大的单位质量价值对应的序号即可。

注意：由于价值为整数，但是可能存在 0 价值（依据常理价值不存在负数），所以根据以下参考代码给出的做法，初始的最大价值和第二大价值的数只需不大于 0 即可。

三、参考代码

```
#include<iostream>
using namespace std;
int main()
{
	int n, counter;
	long long a[1000];
	long long b[1000];
	long double c[1000];
	long double maxs, maxs_2;
	while ( cin >> n )
	{
		maxs = 0;
		maxs_2 = 0;
		for ( int i=0; i<n; i++ )
			cin >> a[i];
		for ( int r=0; r<n; r++ )
			cin >> b[r];
		for ( int s=0; s<n; s++ )
			c[s] = static_cast<long double>(a[s])/static_cast<long double>(b[s]);
		for ( int m=0; m<n; m++ )
		{
			if ( c[m] > maxs )
				maxs = c[m];
		}
		for ( int l=0; l<n; l++ )
		{
			if ( c[l] >= maxs_2 && c[l] != maxs )		//注意等号，否则第二大价值为 0 时得不到正解
			{
				counter = l + 1;
				maxs_2 = c[l];
			}
		}
		cout << counter << endl;
	}
}
```

F 这货不是二分

Problem Description

给你一升序数列，以及一组查询，查询某一特定元素是否存在于数列之中，如果存在，则输出该元素首次出现的位置，否则输出"error"。
输出 m 行，每行输出内容见题目描述及样例。

Input

多组测试数据。
每组数据第一行为两个整数 n,m(1<=n,m<=1000000)，表示数列中有 n 个元素以及 m 次查询。
第二行包含 n 个正整数（1<=ai<=2000000)，用空格分隔，表示有序数列。
接下来 m 行，每行一个整数，表示每次查询的元素。

Outut

每组数据输出一行。见样例。

Sample Input

5 3
1 2 3 4 5
3
5
7

Sample Output

3
5
error

二、解题思路

由于有最多 1000000 个元素和 1000000 个查询，且时限 1s，因此不能进行多次计算，并且要使用 C 语言样式输入输出。

因此，可考虑当输入元素 x 时某个数组的第 x 号元素赋值为数列的第几项的值，然后在查询的时候只需要判断语句即可。由于 x 取值[1,2000000]，这里依旧要将数组设置为全局变量，并在每组测试数据 over 之后清空数组。

三、参考代码

```
#include<stdio.h>
#include<string.h>
int A[2000001];
int main()
{
    int n, m, x, y;
```

```
    while ( scanf("%d %d",&n,&m)!=EOF )
    {
        for ( int i=1; i<=n; i++ )
        {
            scanf("%d",&x);
            if ( A[x]==0 )
                A[x]=i;
        }
        while ( m-- )
        {
            scanf("%d",&y);
            if (A[y]==0)
                printf("error\n");
            else
                printf("%d\n",A[y]);
        }
        memset(A, 0, sizeof(A));
    }
}
```