12.9-函数的应用实例(二)

代码示例1：

已知在坐标系中三个点的坐标,用线段将这些点两两相连；

输入：

六个整数，分别表示三个点的坐标

输出：

(1)若连接后无法构成三角形，输出“Wrong！”

(2)若能构成三角形，判断其是否为直角三角形，如果是则输出“Yes！”，如果不是则输出“No！”

#include<iostream>

#include<math.h>

using namespace std;

struct point

{

int x; //横坐标

int y; //纵坐标

};

void input(point &a)

{

//输入一个点的横纵坐标

cin>>a.x>>a.y;

}

double slope(point a,point b)

{

//求两点构成直线的斜率

return (double)(a.y-b.y)/(a.x-b.x);

}

bool is\_triangle(point a,point b,point c)

{

//判断能否构成三角形

//cout<<slope(a,b)<<" "<<slope(a,c)<<" "<<slope(b,c)<<endl;

if( slope(a,b)==slope(b,c) )

{

//任意两条线段共线，则三点共线，无法构成三角形

return false;

}

return true;

}

double length(point a,point b)

{

//求两点间距离

return sqrt( (double)( (a.x-b.x)\*(a.x-b.x)+(a.y-b.y)\*(a.y-b.y) ) );

}

void swap(double a,double b)

{

double tmp;

tmp=a;

a=b;

b=tmp;

return;

}

bool is\_right\_triangle(point a,point b,point c)

{

//判断能否构成直角三角形

double len[3];

int i,j;

len[0]=length(a,b); //AB的长

len[1]=length(a,c);

len[2]=length(b,c);

for(i=0;i<2;i++)

{

//冒泡排序，从小到大

for(j=0;j<2-i;j++)

{

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

{

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

}

}

}

if(len[0]\*len[0]+len[1]\*len[1]==len[2]\*len[2])

return true;

else

return false;

}

point vector(point a,point b)

{

//求直线AB对应的向量坐标

point l;

l.x=b.x-a.x;

l.y=b.y-a.y;

return l;

}

double vector\_product(point l1,point l2)

{

//求两个向量的数量积

return l1.x\*l2.x+l1.y\*l2.y;

}

bool is\_right\_triangle(point a,point b,point c,int) //无名参数用于重载

{

point l1=vector(a,b);

point l2=vector(a,c);

point l3=vector(b,c);

if( vector\_product(l1,l2)==0 || vector\_product(l1,l3)==0 || vector\_product(l2,l3)==0 )

{

//若两平面向量的数量积为0，则构成直角

return true;

}

return false;

}

int main()

{

point a,b,c;

input(a);

input(b);

input(c);

if(is\_triangle(a,b,c))

{

//cout<<"OK!"<<endl;

if(is\_right\_triangle(a,b,c,0))

{

cout<<"Yes!"<<endl;

}

else

{

cout<<"No!"<<endl;

}

}

else

{

cout<<"Wrong!"<<endl;

}

return 0;

}

代码示例2：

一个 n 行 n 列的螺旋矩阵可由如下方法生成：

从矩阵的左上角（第 1 行第 1 列）出发，初始时向右移动；如果前方是未曾经过的格子，则继续前进，否则右转；重复上述操作直至经过矩阵中所有格子。根据经过顺序，在格子中依次填入1, 2, 3, ... , n^2，便构成了一个螺旋矩阵。

例如下图是n=4时的螺旋矩阵。

1 2 3 4

12 13 14 5

11 16 15 6

10 9 8 7

输入：

第一行输入一个整数n(n<=10)，表示矩阵的大小；

第二行输入两个整数x,y；

输出：

一个整数，表示在n\*n螺旋矩阵中，第x行第y列的数

//非递归暴力算法

#include<iostream>

using namespace std;

int main()

{

int a[11][11];

int n, x, y;

cin >> n >> x >> y;

int tmp = 1;;

int i,j;

for (i = 1; i <= n / 2 + 1; i++)

{

//构造从外层数，第i层的螺旋矩阵

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

a[i][j]=tmp++; //构造上面的一行

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

a[j][n-i+1]=tmp++; //构造右边的一列

for(j = n-i; j >= i; j--)

a[n-i+1][j]=tmp++; //构造下面的一行

for(j = n-i; j > i; j--)

a[j][i]=tmp++; //构造左边的一列

}

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

{

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

{

cout<<a[i][j]<<" ";

}

cout<<endl;

}

cout << a[x][y] << endl;

return 0;

}

//递归算法

int work(int n, int i, int j) {

if (i == 1)

return j;

if (j == n)

return n + i - 1;

if (i == n)

return 3 \* n - 2 - j + 1;

if (j == 1)

return 4 \* n - 4 - i + 2;

// 注意，递归的时候，n 要减 2 而不是减 1

return work(n - 2, i - 1, j - 1) + 4 \* (n - 1);

}