直直的多边形面积

题目描述

Zhoues 最近在研究**直直的多边形**，直直的多边形是所有边均平行于两条坐标轴之一的简单多边形。

现给出一个直直的多边形，Zhoues 想知道这个直直的多边形的面积。

输入格式

第一行一个正整数 t（1≤t≤5），表示数据组数。

对于每组数据，第一行一个正整数 n（4≤n≤4×10^5），表示直直的多边形的顶点数。

接下来 n 行，每行两个整数 xi,yi（−5×104≤xi,yi≤5×10^4），表示直直的多边形一个顶点的坐标。保证顶点坐标按顺时针方向给出，且所有边均平行于两条坐标轴之一。

输出格式

对于每组数据，输出一行一个非负整数，表示直直的多边形的面积。

题目分析

- 两个点的叉积的二分之一的绝对值与这两个点与原点构成的三角形面积相等，根据叉积的性质，当这两个点做叉积的顺序与坐标的右手定则同向时叉积为正，反之为负
- 用一个结构体记录点的坐标，用一个vector记录顶点的集合

```
struct Point {
    long long x, y;
    double len() {
        return sqrt(x * x + y * y);
    }
    Point(long long _=.0,long long __=.0):x(_),y(__) {}
};
vector <Point> poly;
```

输入时需要先清空vector

```
poly.clear();
for (int j=0;j<n;j++) {
    long long x,y;
    scanf("%lld%lld",&x,&y);
    poly.push_back(Point(x,y));
}
```

- 由于题目中的顶点坐标按顺时针方向给出，则按顶点给出顺序对相邻的两个点做叉积并除以二，得到的三角形面积（带正负）相加则得到整个闭合多边形的面积

```
long long area(const vector<Point> &poly) {
    long long rt=0;
    int n=(int)poly.size();
    for (int i=0; i<n; i++)
        rt+=cross(poly[i],poly[(i+1)%n]);
    return abs(rt/2);
}
```

- 做叉积时并不要求相邻的两个点之间的线段平行于坐标轴，因此该解法稍做修改也可以用于求不是直直的多边形的面积

代码

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<iostream>
#include<cmath>
#include<vector>
using namespace std;
const double eps=1e-10;
int n,p;
struct Point {
    long long x, y;
    double len() {
        return sqrt(x * x + y * y);
    }
    Point(long long _=.0,long long __=.0):x(_),y(__) {}
};
Point operator +(const Point &r,const Point &s) {
    return Point(r.x+s.x,r.y+s.y);
}
Point operator -(const Point &r,const Point &s) {
    return Point(r.x-s.x,r.y-s.y);
}
Point operator *(const Point &r,double s) {
    return Point(r.x*s,r.y*s);
}
Point operator /(const Point &r,double s) {
    return Point(r.x/s,r.y/s);
}
//点
vector <Point> poly;
int sign(double a) {
    return (a>eps)?1:(a<-eps)?-1:0;    //判断正负
}
long long dot(const Point &r, const Point &s) {
    return r.x*s.x+r.y*s.y;    //点积
}
long long cross(const Point &r, const Point &s) {
    return r.x*s.y-r.y*s.x;    //叉积
}
long long area(const vector<Point> &poly) {
    long long rt=0;
```

```
    int n=(int)poly.size();
    for (int i=0; i<n; i++)
        rt+=cross(poly[i],poly[(i+1)%n]);
    return abs(rt/2);
}//多边形的面积
int main() {
    int t;
    scanf("%d",&t);
    for (int i=0; i<t; i++) {
        scanf("%d",&n);
        poly.clear();
        for (int j=0;j<n;j++) {
            long long x,y;
            scanf("%lld%lld",&x,&y);
            poly.push_back(Point(x,y));
        }
        printf("%lld\n",area(poly));
    }
    return 0;
}
```