Ponto 2D / Vetor

Representação de pontos e vetores como pontos.

Estrutura

```
const double PI = acos(-1);

struct point{

    long long  x, y;

    point(long long _x, long long _y) : x(_x), y(_y) {} ;

    point(){ x=y=0.0; };

    // Operadores
    bool operator <( point other ) const {
        return x==other.x ? y < other.y : x<other.x;
    }
    bool operator ==(point other) const {
        return x==other.x and y==other.y;
    }
    point operator +(point other) const {
        return point( x+other.x, y+other.y);
    }
    point operator -(point other) const {
        return point( x-other.x, y-other.y );
    }
    point operator *(double k) const {
        return point(x*k, y*k);
    }

    // Norma do vetor em relação a origem
    double norm() { return hypot(x, y); }

    // Angulo do vetor com a origem
    double angle() { return atan2(y, x); }

    // Angulo polar em relação a origem
    double polarAngle(){
        double a = atan2(y, x);
        return a < 0 ? a + 2*PI : a;
    }
};
```

Funções

```
// Distância entre 2 pontos
double dist(point p1, point p2){
    return hypot(p1.x-p2.x, p1.y-p2.y);
}

// Produto interno
long long inner(point p1, point p2){
    return p1.x*p2.x+p1.y*p2.y;
}

// Produto vetorial entre 2 pontos (apenas a componente z)
long long cross(point p1, point p2){
    return p1.x*p2.y-p1.y*p2.x;
}

// Teste se os pontos estão no sentido antihorário (DISCRIMINANTE)
bool ccw(point p, point q, point r){
    return cross(q-p, r-p)>0;
}

// Teste de colinearidade entre os pontos
bool collinear(point p, point q, point r){
    return cross(q-p, r-p)==0;
}

// Angulo formado entre vetores a e b com o de origem
double angle(point a, point o, point b){
    return acos(inner(a-o, b-o)/(dist(o, a)*dist(o,b)));
}

// Ponto q está dentro no segmento p r
bool between(point p, point q, point r){
    return collinear(p, q, r) && inner(p-q, r-q)<=0;
}

// Teste de paralelidade
bool parallel(point a, point b){
    return cross(a, b)==0;
}
```