Ponto 2D / Vetor

Representação de pontos e vetores como pontos.

Estrutura

```
const double EPS = 1e-9;
const double PI = acos(-1);

struct Point{
    double x, y;

    Point(double _x, double _y) : x(_x), y(_y) {} ;
    Point(){ x=y=0.0; };

    // Operadores
    bool operator <( Point other ) const {
        if(fabs( x-other.x )>EPS) return x<other.x;
        else return y < other.y;
    }
    bool operator ==(Point other) const {
        return fabs(x-other.x) < EPS && fabs(y-other.y)<EPS;
    }
    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); }

    // Vetor normalizado
    Point normalized(){ return Point(x, y)*(1.0/norm()); }

    // 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
double inner(Point p1, Point p2){
    return p1.x*p2.x+p1.y*p2.y;
}

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

// Teste counter-clockwise
bool ccw(Point p, Point q, Point r){
    return cross(q-p, r-p)>0;
}

// Teste de colinearidade de r com a reta p-q
bool collinear(Point p, Point q, Point r){
    return fabs(cross(q-p, r-p))<EPS;
}

// Rotação do ponto em relação a origem
Point rotate(Point p, double rad){
    return Point(p.x*cos(rad)-p.y*sin(rad),
                 p.x*sin(rad)+p.y*cos(rad));
}

// Angulo formado pelas retas que conteem os pontos a e b e se crusam no
ponto o
double angle(Point a, Point o, Point b){
    return acos(inner(a-o, b-o)/(dist(o, a)*dist(o,b)));
}
```