struct point

{

double x, y;

};

struct triangle

{

point a, b, c;

};

struct rect

{

point TL, TR, BL, BR;

};

struct segment

{

point a, b;

};

struct vect

{

double x, y;

};

struct polygon

{

vector <point> points;

int s = points.size();

};

double get\_Length(vect V)

{

return sqrt( pow(V.x, 2) + pow(V.y, 2) );

}

double Dot(vect V1, vect V2)

{

return V1.x \* V2.x + V1.y \* V2.y;

}

double Cross(vect V1, vect V2)

{

return V1.x \* V2.y - V1.y \* V2.x;

}

double get\_Angle(vect V)

{

return atan2(V.x, V.y);

}

vect make\_vect( segment sg)

{

vect v;

v.x = sg.b.x - sg.a.x;

v.y = sg.b.y - sg.a.y;

return v;

}

double get\_S\_PL(point P, segment S)

{

return abs(( (S.b.x-S.a.x)\*(S.b.y-L.a.y) - (L.b.y - L.a.y)\*(L.b.x - L.a.x) )/get\_Length(make\_vect(S)));

}

vect norm(vect V)

{

double len = get\_Length(V);

V.x /= len;

V.y /= len;

return V;

}

double get\_S\_Pol(polygon Pol) // точки в порядке обхода против часовой стрелки

{

double answer = 0.0;

int s = Pol.points.SZ;

for (int i = 0; i < s; i++)

{

double ax = Pol.points[i].x;

double ay = Pol.points[i].y;

double bx = Pol.points[(i+1)%s].x;

double by = Pol.points[(i+1)%s].y;

answer += (ax\*by - ay\*bx);

}

return answer\*0.5;

}

point Reflect(vect P, segment S)

{

point pr;

vect N;

N.x = S.a.y - S.b.y;

N.y = S.b.x - S.a.x;

vect n = norm(N);

double dot2 = 2\*Dot(n, P);

pr.x = P.x - dot2\*n.x;

pr.y = P.y - dot2\*n.y;

return pr;

}