Intersección entre un segmento y un círculo

```
/*  Verifies the intersection of a segment and a circle
**  The line segment is defined from points p1 to p2
**  The circle is of radius r and centered at point c
**  There are potentially two points of intersection given by
**  p = p1 + mu1 (p2 - p1)
**  p = p1 + mu2 (p2 - p1)
**  mu1 and mu2 are updated via reference
**  Return FALSE if the segment doesn't intersect the circle */

#include <cmath>

const double eps = 1e-10;

class punto {
public:
     double x, y; //coordenadas
     punto(){}
     punto(double x, double y){ this->x = x; this->y = y; }
};

class circulo {
public:
     punto c; //centro
     double r; //radio
     circulo(){}
     circulo(punto c, double r){ this->c = c; this->r = r; }
};

bool cross(punto p1, punto p2, circulo p, double &mu1, double &mu2, punto
&inter1, punto &inter2)
{
    double a, b, c, d;
    punto t;
    t.x = p2.x - p1.x;
    t.y = p2.y - p1.y;
    a = t.x*t.x + t.y*t.y;
    b = 2.0 * (t.x * (p1.x - p.c.x) + t.y * (p1.y - p.c.y));
    c = p.c.x*p.c.x + p.c.y*p.c.y;
    c += p1.x*p1.x + p1.y*p1.y;
    c -= 2.0 * (p.c.x * p1.x + p.c.y * p1.y);
    c -= p.r*p.r;
    d = b * b - 4 * a * c;
    if(fabs(a) < eps || d < -eps) {
        mu1 = mu2 = 0.0;
        return false;
    }
    mu1 = (-b + sqrt(d)) / (2.0 * a);
    mu2 = (-b - sqrt(d)) / (2.0 * a);

    //devuelve los puntos donde se intersectan
    inter1.x = p1.x + mu1*(p2.x - p1.x);
    inter1.y = p1.y + mu1*(p2.y - p1.y);
    inter2.x = p1.x + mu2*(p2.x - p1.x);
    inter2.y = p1.y + mu2*(p2.y - p1.y);
    return true;
}
```