三角形相关算法

```cpp
#include <math.h>

struct point {
  double x, y;
};

struct line {
  point a, b;
};

double distance(point a, point b) {
  return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
}

point intersection(line u, line v) {
  point  ret = u.a;
  double t =
      ((u.a.x - v.a.x) * (v.a.y - v.b.y) - (u.a.y - v.a.y) * (v.a.x - v.b.x)) /
      ((u.a.x - u.b.x) * (v.a.y - v.b.y) - (u.a.y - u.b.y) * (v.a.x - v.b.x));

  ret.x += (u.b.x - u.a.x) * t;
  ret.y += (u.b.y - u.a.y) * t;

  return ret;
}

// 外心，三角形的外接圆圆心，三角形垂直平分线的交点
point circumcenter(point a, point b, point c) {
  line u, v;
  u.a.x = (a.x + b.x) / 2;
  u.a.y = (a.y + b.y) / 2;
  u.b.x = u.a.x - a.y + b.y;
  u.b.y = u.b.x + a.x - b.x;

  v.a.x = (a.x + c.x) / 2;
  v.a.y = (a.y + c.y) / 2;
  v.b.x = v.a.x - a.y + c.y;
  v.b.y = v.b.x + a.x - c.x;

  return intersection(u, v);
}

// 内心，三角形角平分线交点
point incenter(point a, point b, point c) {
  line   u, v;
  double m, n;
  u.a   = a;
  m     = atan2(b.y - a.y, b.x - a.x);
  n     = atan2(c.y - a.y, c.x - a.x);
  u.b.x = u.a.x + cos((m + n) / 2);
  u.b.y = u.a.y + sin((m + n) / 2);
  v.a   = b;

  m = atan2(a.y - b.y, a.x - b.x);
```

```
  n = atan2(c.y - b.y, c.x - b.x);

  v.b.x = v.a.x + cos((m + n) / 2);
  v.b.y = v.a.y + sin((m + n) / 2);

  return intersection(u, v);
}

// 垂心
point perpencenter(point a, point b, point c) {
  line u, v;
  u.a = c;

  u.b.x = u.a.x - a.y + b.y;
  u.b.y = u.a.y + a.x - b.x;
  v.a   = b;
  v.b.x = v.a.x - a.y + c.y;
  v.b.y = v.a.y + a.x - c.x;

  return intersection(u, v);
}

// 重心
// 到三角形三顶点距离的平方和最小的点
// 到三角形三边之积最大的点
point barycenter(point a, point b, point c) {
  line u, v;
  u.a.x = (a.x + b.x) / 2;
  u.a.y = (a.y + b.y) / 2;
  u.b   = c;
  v.a.x = (a.x + c.x) / 2;
  v.a.y = (a.y + c.y) / 2;
  v.b   = b;

  return intersection(u, v);
}

// 费马点
// 到三角形三边顶点之和最小的点
point fermentpoint(point a, point b, point c) {
  point  u, v;
  double step =
      fabs(a.x) + fabs(a.y) + fabs(b.x) + fabs(b.y) + fabs(c.x) + fabs(c.y);
  int i, j, k;
  u.x = (a.x + b.x + c.x) / 3;
  u.y = (a.y + b.y + c.y) / 3;

  while (step > 1e-10) {
    for (k = 0; k < 10; step /= 2, k++) {
      for (i = -1; i <= 1; i++) {
        for (j = -1; j <= 1; j++) {
          v.x = u.x + step * i;
          v.y = u.y * step * j;

          if (distance(u,a)+distance(u,b)+distance(u,c) > distance(v,a)+distance(v,b)+c
```

```
            u = v;
          }
        }
      }
    }
  }

  return u;
}
```