计算几何

误差修正

```
const double eps=1e-8;
int cmp(double x)
{
    if (fabs(x)<eps) return 0;
    if (x>0) return 1;
    return -1;
}
```

多边形类

点类

- det 计算两个向量的叉积
- dot 计算两个向量的点积
- dist 计算两个点的距离
- rotate_point(p,A) 计算向量$\vec{op}$绕原点逆时针旋转弧度A

```
const double pi=acos(-1.0);
inline double sqr(double x){return x*x;}
struct point
{
    double x,y;
    point(){}
    point(double a,double b):x(a),y(b){}
    void input(){scanf("%lf%lf"),&x,&y;}
    friend point operator + (const point &a,const point &b)
    {
        return point(a.x+b.x,a.y+b.y);
    }
    friend point operator - (const point &a,const point &b)
    {
        return point(a.x-b.x,a.y-b.y);
    }
    friend bool operator == (const point &a,const point &b)
    {
        return cmp(a.x-b.x)==0&&cmp(a.y-b.y)==0;
    }
    friend point operator * (const point &a,const double &b)
    {
        return point(a.x*b,a.y*b);
    }
    friend point operator * (const double &a,const point &b)
    {
        return point(a*b.x,a*b.y);
    }
    friend point operator / (const point &a,const double &b)
    {
        return point(a.x/b,a.y/b);
    }
    double norm()
    {
        return sqrt(sqr(x)+sqr(y));
    }
};
double det(const point &a,const point &b)
{
    return a.x*b.y-a.y*b.x;
}
double dot(const point &a,const point &b)
{
    return a.x*b.x+a.y*b.y;
}
double dist(const point &a,const point &b)
{
    return (a-b).norm();
}
point rotate_point(const point &p,double A)
{
    double tx=p.x,ty=p.y;
    return point(tx*cos(A)-ty*sin(A),tx*sin(A)+ty*cos(A));
}
```