Рекомендуемый интерфейс для решения задач вычислительной геометрии 2.0

```
const
        EPS = 1e-9;

type
        TAngle = extended;

        TPoint = record
                x, y: extended;
        end;

        TPolarPoint = record
                {phi in [0..2 * PI)}
                r, phi: extended;
        end;

        TVector = record
                x, y: extended;
        end;

        TSegment = record
                a, b: TPoint;
        end;

        TRay = record
                a, b: TPoint;
        end;

        TLine = record
                a, b: TPoint;
        end;

        TTriangle = record
                a, b, c: TPoint;
        end;

        TCircle = record
                c: TPoint;
                r: extended;
        end;

        TPolygon = record
                size: longint;
                p: array [1..MAXN] of TPoint;
        end;

function CreatePoint(x, y: longint): TPoint;
function CreateVector(a, b: TPoint): TVector;
function CreateSegment(a, b: TPoint): TSegment;
function CreateRay(a, b: TPoint): TRay;
```

```
function CreateLine(a, b: TPoint): TLine;
function CreateTriangle(a, b, c: TPoint): TTriangle;
function CreateCircle(c: TPoint; r: longint): TCircle;

procedure PrintPoint(p: TPoint);
procedure PrintVector(v: TVector);

function VectorLength(v: TVector): extended;
function Distance(a, b: TPoint): extended;
function Distance(s: TSegment s; a: TPoint): extended;
function Distance(ray: TRay; a: TPoint): extended;
function Distance(line: TLine; a: TPoint): extended;
function Distance(a, b: TSegment): extended;

function Sum(a, b: TVector): TVector;
function Mult(a: TVector; b: extended): TVector;
function Normalize(a: TVector): TVector;

function DotProduct(a, b: TVector): longint;
function CrossProduct(a, b: TVector): longint;

function PolarToCartesian(a: TPolarPoint): TPoint;
function CartesianToPolar(a: TPoint): TPolarPoint;

function IsIntersected(a, b: TSegment): boolean;
function IsIntersected(a, b: TRay): boolean;
function IsIntersected(a, b: TLine): boolean;

function IsBelonged(s: TSegment s; a: TPoint): boolean;
function IsBelonged(ray: TRay; a: TPoint): boolean;
function IsBelonged(line: TLine; a: TPoint): boolean;

function IsInside(t: TTriangle; a: TPoint): boolean;
function IsInside(c: TCircle; a: TPoint): boolean;
function IsInside(p: TPolygon; a: TPoint): boolean;
function IsConvex(p: TPolygon): boolean;

function IsOrthogonal(a, b: TVector): boolean;

function TriangleArea(a: TTriangle): extended;
function CircleArea(a: TCircle): extended;
function PolygonArea(a: TPolygon): extended;

function GetAngle(a, b: TVector): extended;

function Less(a, b: extended): boolean;
function Equal(a, b: extended): boolean;
function Greater(a, b: extended): boolean;
```