E3

Partie I – classe Point

Nous souhaitons écrire une classe Point permettant de représenter les points du plan.

La classe doit être pourvue des méthodes suivantes :

- un constructeur prenant les coordonnées en argument,
- des fonctions permettant de récupérer ou modifier les coordonnées :

```
float getX () const   //retourne l'abscisse
float getY () const   //retourne l'ordonnée
void setX (float nouveauX) //modifie l'abscisse
void setY (float nouveauY) //modifie l'ordonnée
```

- une fonction d'affichage
- une fonction testant l'égalité avec un autre point
- une fonction retournant la distance à un autre point

Partie II – classe Segment

Nous souhaitons écrire une classe Segment, un segment étant constitué de deux objets Point, l'origine et l'extrémité, en partie privée.

Ecrire un constructeur recevant deux objets Point en paramètre.

Ecrire une fonction d'affichage.

Ecrire une fonction faisant subir une translation au segment. Cette fonction recevra en paramètre un objet Point indiquant la nouvelle position de l'origine.

Partie III – classe Polygone

Nous souhaitons maintenant développer une classe Polygone. Cette classe a deux données membres :

- un entier indiquant le nombre de points constituant le polygone (appelés points extrêmes)
- un tableau de points (points extrêmes du polygone)

Ecrire un constructeur recevant en entrée un entier et un tableau de point. A l'appel la syntaxe suivante pourra être utilisée :

```
Point tableau[3] = { Point(0,0),Point(0,4),Point(2,2) };
Polygone p(tableau, 3);
```

Ecrire le destructeur et le constructeur par recopie.

Ajouter une fonction d'affichage.

Ajouter une fonction calculant le périmètre du polygone.

Partie IV – suite classe Polygone

Nous souhaitons maintenant doter la classe Polygone d'une fonction qui retourne l'enveloppe convexe d'un polygone. L'enveloppe convexe d'un polygone P est un polygone convexe minimal contenant P.

Cette enveloppe se calcule à l'aide du pseudo-code suivant (marche de Jarvis):

```
R ← ensemble vide de points
p0 ← point extrême le plus à gauche du polygone P
     (et le plus haut en cas d'égalité)
p ← p0
répéter
     ajouter p à R
     q ←point extrême quelconque de P autre que p
     pour tout point extrême u de P
          si  (xq-xp)(yu-yp) – (xu-xp)(yq-yp) < 0
          q ← u
     p ← q
tant que p != p0
retourner le polygone constitué à partir des points de R
```

(R pourra être défini comme un tableau de points de dimension égale au nombre de points de P ; il faudra alors ajouter un constructeur par défaut à la classe Point pour pouvoir allouer R)

Référence :

Jarvis, Ray A. On the identification of the convex hull of a finite set of points in the plane. Information processing letters, 1973, vol. 2, p. 18-21.
https://www.sciencedirect.com/science/article/pii/0020019073900203?via%3Dihub