Université de Pau et des Pays de l'Adour **Année 2017-2018**
Département de Mathématiques

L2-Info–Calcul scientifique
TD n^o 5

Exercice 1

1. Ecrire une classe ‘‘`Point`’’ permettant de manipuler des points de l'espace.
   Cette classe contiendra:
   Les attributs privés:
   `double _x,_y,_z`; représentant les coordonnées du point,
   Les constructeurs (au moins ceux-là):
   `Point();` constructeur sans paramètre
   `Point(double x, double y, double z);` constructeur prenant x, y z en paramètre servant à initialiser _x,_y,_z
   `Point(double x, double y);` ne prenant que x,y en paramètre, initialisant _x et _y aux valeurs de x et y et initialisant _z à 0 (pour les points du plan)
   les fonctions membres publiques :
   `double getX() const;`
   `double getY() const;`
   `double getZ() const;` donnant accés aux coordonnées en lecture
   `double& getX();`
   `double& getY();`
   `double& getZ();` donnant accés aux coordonnées en écriture
   `double distance(const Point& P) const`; renvoyant la distance de this à P, passé en paramètre
   `void translate(const Point& P)` ajoutant à this les coordonnées de P, passé en paramètre
   Les deux opérateurs :
   ‘‘<<’’ et ‘‘>>’’ pour permettre aux flux de sortie et d'entrée d'écrire et de lire des objets de type “`Point`”.

2. Ecrire une classe ‘‘`Triangle`’’ permettant de manipuler des triangles du plan. On supposera que pour tous les sommets des triangles de cette classe, _z=0.
   Cette classe contiendra:
   Les attributs privés :
   `std::vector<Point> _nodes;` représentant le vecteur des sommets et qui devra être dimensionné à la taille 3 dans les constructeurs,
   `double _area;` représentant l'aire du triangle
   `double _perimeter;` représentant le périmètre du triangle
   les fonctions membres privées:
   `void _computeArea();` calculant l'aire du triangle et initialisant l'attribut `_area` (voir formule en fin de sujet).
   `void _computePerimeter();` calculant le périmètre du triangle et initialisant l'attribut `_perimeter`
   Les constructeurs (au moins ceux-là):
   `Triangle();` constructeur sans paramètre
   `Triangle(const Point& P1, const Point& P2, const Point& P3);` prenant les sommets en paramètre

`Triangle(const std::vector<Point>& vpts);` prenant un veteur de sommets en paramètre
les fonctions membres publiques:
`void initGeometry();` mettant à jour la valeur de l'aire et du périmètre
`const Point& getNode(int i) const`; donnant accés au sommet i en lecture
`Point& getNode(int i)`; donnant accés au sommet i en écriture
`double getArea() const;` donnant accés à l'aire du triangle en lecture
`double getPerimeter() const;` donnant accés au périmètre du triangle en lecture"
Les deux opérateurs :
"<<" et ">>" pour permettre aux flux de sortie et d'entrée d'écrire et de lire des objets de type "`Triangle`".

3. Écrire un programme principal testant vos deux classes.

Exercice 2

1. Récupérer le fichier "Domain.vtk" sur le site du cours. Ce fichier contient la description d'un domaine recouvert par un ensemble de triangles que l'on appelle "maillage".
2. Depuis une fenêtre terminal taper la commande "paraview".
3. Sous paraview, lire le fichier "Domain.vtk" pour visualiser le doamine et le maillage.
   Combien y a-t-il de triangles et de sommets dans le maillage recouvrant le domaine?
4. Ecrire un programme qui lit ce fichier Domain.vtk et qui, à l'aide de vos classes de l'exercice 1, calcule l'aire du domaine maillé.

Formule du calcul de l'aire d'un triangle T du plan, de sommets $P_0(x_0, y_0)$, $P_1(x_1, y_1)$, $P_2(x_2, y_2)$:

$$Aire(T) = \frac{1}{2}|(x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0)|.$$