Examen C – Session 2
ING1 – Programmation C
21 février 2013

Modalités

- Durée : 3 heures
- Toutes vos affaires (sacs, vestes, trousse, etc.) doivent être placées à l'avant de la salle.
- Aucun document papier n'est autorisé.
- Aucune sortie n'est autorisée avant une durée incompressible de 30 minutes.
- Aucun déplacement n'est autorisé.
- **Aucune question au professeur n'est autorisée**. Si le sujet ne vous semble pas clair, à vous d'expliquer dans des commentaires pourquoi est ce que ça ne vous semble pas clair, et quelles modifications vous effectuez pour réaliser l'exercice.
- Aucun échange, de quelle que nature que ce soit, n'est possible.
- **Les différentes fonctions/procédures demandées seront toutes à tester dans un main.**
- Vous traiterez chaque partie dans un module différent.
- La présence d'un `Makefile` et des commentaires doxygens sont un plus, et seront appréciés.
- La lisibilité du code et les commentaires seront pris en compte dans la notation, de même que la séparation des différentes procédures/fonctions dans des fichiers d'en-tête (.c .h).
- **Une erreur de compilation entrainera une pénalité de -50% sur la note finale. Si une partie de votre code ne compile pas, commentez-le. Ceci implique évidemment que vous compiliez au fur et à mesure!**
- Chaque warning de compilation entrainera une **pénalité de 0.5 point**.
- Tout ce qui sera dans le dossier **rendu** correspondra à votre rendu, et sera récupéré à la fin de l'examen. Je vous conseille vivement de travailler directement dans ce dossier.

Problème : développement d'un logiciel de géométrie sur un plan

Point

1. Définir une structure `point` qui contient les coordonnées du point ainsi que son nom ;
2. Définir la fonction `saisirPoint` qui permet de saisir un point ;
3. Définir la procédure `afficherPoint`qui permet d'afficher un point ;
4. Définir la méthode `translaterPoint(int int_deltaX, int int_deltaY)` qui permet de translater le point sur le plan;
5. Définir la fonction `distancePoint` qui permet de mesurer la distance entre deux points.

Cercle

1. Définir une structure `cercle` qui contient un point comme centre du cercle, le rayon du cercle ainsi que son nom ;
2. Définir la fonction `saisirCercle` qui permet de saisir un cercle ;
3. Définir la procédure `afficherCercle` qui permet d'afficher un cercle ;
4. Définir la méthode `translaterCercle(int int_deltaX, int int_deltaY)` qui permet de translater le cercle sur le plan ;
5. Définir la fonction `perimetreCercle` qui permet de calculer le périmètre d'un cercle ;
6. Définir la fonction `aireCercle` qui permet de calculer l'aire d'un cercle.

Rappel : Soit un cercle $\mathscr{C}$ de centre $O = (x, y)$ et de rayon r. Le périmètre $p_{\mathscr{C}}$ du cercle est $p_{\mathscr{C}} = 2 \times \pi \times r$ et l'aire $\mathscr{A}_{\mathscr{C}}$ du cercle est $\mathscr{A}_{\mathscr{C}} = \pi \times r^2$.

Triangle

1. Définir une structure `triangle` qui contient 3 points comme sommets du triangle ainsi que son nom ;
2. Définir la fonction `saisirTriangle` qui permet de saisir un triangle ;
3. Définir la procédure `afficherTriangle` qui permet d'afficher un triangle ;
4. Définir la méthode `translaterTriangle(int int_deltaX, int int_deltaY)` qui permet de translater un triangle sur le plan ;

5. Définir la fonction `perimetreTriangle` qui permet de calculer le périmètre d'un triangle ;
6. Définir la fonction `aireTriangle` qui permet de calculer l'aire d'un triangle.

Rappel : Soit un triangle T de sommets $A = (x_a, y_a), B = (x_b, y_b), C = (x_c, y_c)$ et de côté a, b, c. L'aire $\mathscr{A}_T$ du triangle est, d'après la formule d'Heron : $\mathscr{A}_T = \sqrt{s(s-a)(s-b)(s-c)}$ avec $s = \dfrac{a+b+c}{2}$.

Polygone

On considère qu'un polygone est simplement un tableau de points.

1. Définir la fonction `saisirPolygone` qui permet de saisir un polygone en fonction du nombre de sommets souhaités par l'utilisateur ;
2. Définir la procédure `afficherPolygone` qui permet d'afficher un polygone ;
3. Définir la méthode `translaterPolygone(int int_deltaX, int int_deltaY)` qui permet de translater un polygone sur le plan ;
4. Définir la fonction `perimetrePolygne` qui permet de calculer le périmètre d'un polygone ;
5. Définir la méthode `sauverPolygone` qui permet de sauvegarder un polygone dans un fichier.

Menu

Écrire une méthode qui va permettre à l'utilisateur de votre logiciel de choisir d'afficher une des figures géométriques implémentées ainsi que ses caractériques, et elle utilisera un type énumération `figure` défini par `Point`,`Cercle`, `Triangle` et `Polygone`.