TD 1 – Enregistrements / Structures

Dans ce TD, nous allons travailler sur des données géométriques (points, polygones, ...) dans un premier temps, puis sur des dates. Aucun de ces types n'est pris en charge nativement en C++ (ni dans les autres langages d'ailleurs). L'objectif de ce TP est donc de vous faire manipuler des structures (ou enregistrements) afin de vous familiariser à l'utilisation de type complexes.

1. Points

- Définir sous forme de structure le type **point** composé de deux **réels x** et **y**.
- Écrire la fonction **saisirPoint** qui demande à l'utilisateur de saisir au clavier les coordonnées d'un **point**, puis qui retourne ce **point**.
- Écrire la fonction **distancePoints** qui prend en entrée deux **points** et retourne la distance (**réel**) entre ces **points** (faire appel au sous-programme **racine_carree** qui retourne la racine carrée d'un nombre). Rappel : la distance entre deux points (x1,y1) et (x2,y2) est la racine carrée de $(x1-x2)^2 + (y1-y2)^2$.
- Écrire le programme principal qui, en faisant appel aux fonctions précédentes, fait saisir deux **points** et affiche la distance qui les sépare.

2. Disques

- Définir sous forme d'enregistrement le type **disque** composé de deux champs rayon (type **réel**) et centre (type **point**).
- Écrire la fonction **saisirDisque** qui demande à l'utilisateur de saisir au clavier les coordonnées du centre d'un disque puis son rayon, et qui retourne ce **disque**.
- Écrire la fonction **testInclusion** qui prend en entrée un **point** et un **disque** et qui détermine si le point est inclus dans le disque.
- Écrire le programme principal qui, en faisant appel aux fonctions précédentes, fait saisir un **point** et un **disque** et affiche un message indiquant si le premier est inclus dans le second.

3. **Polygones**
   - Définir sous forme d'enregistrement le type **polygone** composé de deux champs **nombre_sommets** (type **entier**) et **sommets** (un tableau de **points**).
     Remarques :
     - Par défaut on considère des polygones fermés – le dernier sommet est relié au premier.
     - On suppose qu'un polygone ne peut avoir plus de 100 sommets.
   - Écrire la fonction **saisirPolygone** qui demande à l'utilisateur de saisir au clavier un nombre de **sommets**, puis les coordonnées des sommets d'un **polygone** retourné par la suite.
   - Écrire la fonction **perimetre** qui prend en entrée un **polygone** et qui retourne son périmètre, c-à-d la somme des distances entre le premier sommet et le deuxième, le deuxième et le troisième, ... , le dernier et le premier.
   - Exécuter cette fonction pas à pas.
   - Écrire le programme principal qui, en faisant appel aux fonctions précédentes, fait saisir les coordonnées d'un **polygone** et affiche son périmètre.

4. **Dates**
   - Définir sous forme d'enregistrement le type date composé de trois **entiers jour**, **mois** et **annee**.
   - Écrire la fonction **testDates** qui prend en entrée deux **dates d1** et **d2** et qui retourne:
     - 1 si d1 est antérieure à d2
     - 0 si d1 est égale à d2
     - -1 si d1 est postérieure à d2
   - Écrire la procédure **afficheDatesOrdre** qui prend en entrée un **tableau de dates** et sa taille, et qui, en faisant appel à la fonction précédente, doit afficher le contenu du tableau dans l'ordre chronologique.
   - Exécuter cette procédure pas à pas.
   - Écrire le programme principal qui génère aléatoirement un tableau de 10 **dates** (utiliser la fonction **alea**) puis fait appel à la fonction précédente pour afficher ces dates dans l'ordre.