Travaux pratiques n°7
Structures: Déclaration et utilisation

Objectifs

- Déclarer et manipuler des structures en C.

Partie 1: Compléments pratiques au langage C

L'instruction *typedef*:

Comme indiqué en cours l'instruction typedef permet de définir un nouveau type. Par exemple on peut définir le type entier à partir de *int* ou le type *string* à partir de *char*:

typedef int entier; // le type entier est équivalent à int
typedef char str40[41] ; // le type str40 est équivalent à un tableau de 40 char
En particulier, on peux éviter le mot *struct* dans les types structures en utilisant *typedef*.
Ainsi:

```
typedef struct
   {
     int x;
     int y;
   } point;
```

permet de définir le type *point* équivalent à *struct point* et de déclarer des variables de type *point*:
point p1,p2; // déclare deux variables de type point.

Vous avez aussi la possibilité de la faire en deux temps:

```
struct point
   {
     int x;
     int y;
   };
typedef struct point point;
```

Partie 2: Structures en langage C

Exercice 1.

Un polygone est constitué d'un ensemble de points (par exemple : 5 points pour un pentagone, 8 points pour un octogone, …).
Un polygone peut être constitué de 100 points au maximum.
Ainsi par exemple le polygone constitué des points (1,3) (2,4) (5,3) (6,2) (3,1) est un pentagone.
Écrire les procédures/fonctions en C suivantes:

pointEgal qui retourne 1 si deux points *p1* et *p2* ont les mêmes valeurs 0 sinon.

afficherPoly qui affiche les coordonnées des points successifs constituants un polygone.

initPoly qui initialise et retourne un polygone en demandant les valeurs successives des points le constituant.

concatPoly qui construit un nouveau polygone par concaténation de deux points: le premier point du second polygone devient le point suivant du dernier point du premier polygone dans le nouveau polygone construit.

soustrPoly qui construit un nouveau polygone par soustraction. Cette soustraction consiste à retirer les points du premier polygone qui apparaissent dans le second polygone.

interPoly qui construit un nouveau polygone par intersection de deux polygones. Cette intersection consiste à garder les points commun du premier et du second polygones dans l'ordre où ils apparaissent dans le premier.

deplPoly qui déplace le premier polygone en déplaçant chacun des points qui constitue le polygone d'un déplacement pour l'abscisse et d'un déplacement pour l'ordonnée.

Exercice 2:

Considérons un bateau dans un jeu de bataille navale, il a un nom ("bateau_1" par exemple), un type (croiseur, destroyer, sous-marin, etc...), une position sur la grille ((1,1) par exemple) de début et une position de fin ((1,5) par exemple), la longueur initiale du bateau (en nombre de cases), un état (touché x fois, coulé, ou en parfait état).

a) Donner une structure adéquate pour *bateau*.

b) Écrire une fonction en C qui a la position d'un tir et un bateau en entrée, qui teste le tir et modifie le bateau s'il a été touché et qui renvoie 0 si le tir a échoué, 1 si le bateau a été touché et 2 s'il a été coulé.

c) Dans une bataille navale à 2 joueurs, chaque joueur doit disposer d'un certain nombre de bateaux.
Écrire une fonction en C, *initBat* qui retourne un bateau défini par le type et la longueur qui sont donnés en entrée et par le nom et les positions début et fin qui sont demandées à l'utilisateur dans le corps de la fonction.