TP 1 informatique 2 : Structures

Ne pas oublier de créer un répertoire pour placer les codes de ce TP !

Toutes les fonctions écrites doivent être testées dans le programme principal !

Exercice 1 (*Un peu de géomètrie*)
Un point dans un espace 2D est caractérisé par son abscisse et son ordonnée.

1. Déclarer et définir une structure `Point`.
2. Ecrire une fonction `constructeur_Point()` qui retourne un Point dont les attributs ont été saisis par l'utilisateur.
3. Ecrire une procédure `affiche_Point( Point x)` qui affiche les attributs du Point passés en paramètres sous forme `(abscisse, ordonnée)`.
4. Dans le programme principal, définir deux variables de type `Point`. Les définir et les afficher en utilisant les deux fonctions précédentes.
5. Ecrire une fonction `ajouter_Point(Point a, Point b)` qui retourne un `Point` dont les coordonnées sont la somme des coordonnées des points a et b.
6. Écrire une fonction `longueur( Point a, Point b)` qui retourne la longueur du segment `ab`.
   Indication : On pourra utiliser `sqrt()` qui calcule la racine carrée. Cette fonction est dans la bibliothèque `math.h`. Si cette bibliothèque est incluse dans le code il faut rajouter `-lm` à la suite de la ligne de commande de compilation.
7. Écrire une fonction `estCarre( Point a, Point b, Point c, Point d)` qui retourne 1 si les 4 points passés en paramètre forment un carré.
8. Écrire une fonction `sontAlignes( Point a, Point b, Point c)` qui retourne 1 si les trois `Points` passés en paramètres sont alignés (appartiennent à la même droite).

Exercice 2 (*Gestion de date*)
Une date peut être écrite sous forme jour/mois/année.
Pour simplifier cet exercice on pourra considérer que tous les mois ont 30 jours.

1. Déclarer et définir une structure `Date` composée de trois entiers représentant respectivement le jour le mois et l'année.
2. Écrire une fonction constructeur `constructeur_Date()` qui retourne une variable de type `Date` dont les attributs auront été saisis par l'utilisateur. Si les valeurs saisies sont incorrectes, on redemandera la saisie.
3. Écrire une procédure `affiche_Date(Date d)` qui affiche les attributs de la Date passée en paramètre sous forme `jour/mois/année`.
4. Tester les deux fonctions précédentes en déclarant une variable de type `Date` dans le programme principal et en l'affichant.
5. Écrire une fonction `ajoute_jour(Date a, int jour)` qui ajoute un nombre de jours à la date passée en paramètre et retourne la nouvelle date obtenue.
6. Écrire une fonction `enlever_jour(Date a, int jour)` qui retire un nombre de jours à la date passée en paramètre et retourne la nouvelle date obtenue.

Exercice 3 (*combat !*)
Dans cet exercice, nous allons simuler un combat épique de ninjas.

1. Voici les caractéristiques d'un Ninja :
   — son attaque
   — sa défense
   — son esquive
   — ses points de vie
   — son nom, que l'on définira comme un tableau de caractères.
   Toutes les caractéristiques autres que le nom peuvent être représentées par des réels. Définir la structure `Ninja`.
2. Dans le programme principal, déclarer deux variables `Ninjas`. Les valeurs de leurs champs seront initialisées avec des valeurs aléatoires telles que :
   — 5.0 <= attaque <= 10.0
   — 2.0 <= défense <= 4.0

— 0.1 <= esquive <= 0.2
— 80.0 <= vie <= 100.0

Le nom sera saisi par l'utilisateur grâce à la commande `scanf("%s",NomVariableNinja.nom)` (pas de & !)

3. Créer une procédure `combat(Ninja a, Ninja b)` qui prend en paramètres deux variables de type `Ninja`. Un combat se déroule de la manière suivante : le Ninja a attaque, puis le Ninja b. Le combat se déroule jusqu'à ce qu'un des combattants s'écroule (ses points de vie tombent à 0). Un round se passe comme suit :
   — le programme effectue un jet de valeur aléatoire entre `0.0` et `1.0`
   — si cette valeur est plus petite ou égale à la valeur d'esquive du ninja défenseur, alors il ne se passe rien
   — sinon le programme récupère la valeur d'attaque de l'attaquant, il lui soustrait la valeur de défense du défenseur (en faisant attention à limiter la valeur à 0 : il ne faut pas descendre en-dessous.) : c'est la valeur des dégâts.
   — le programme réduit les points de vie du défenseur du montant des dégâts calculés précédemment.

   Ajouter des affichages dans cette fonction pour que l'on puisse suivre la manière dont se déroule le combat. Afficher le gagnant du combat à la fin de la procédure.
4. Lancer le combat entre vos deux Ninjas déclarés dans le programme principal.

Exercice 4 (*Anniversaire Automatique*)

1. Définir une structure `Personne` qui va contenir :
   — Une chaîne de caractères `prenom`
   — Une `Date date_naissance` (vous pouvez récuperer la structure Date définie à l'exercice 2).
2. Ecrire la fonction constructeur de `Personne` qui va récupérer les valeurs saisies par l'utilisateur. On pourra utiliser la fonction `constructeur_Date()` définie à l'exercice 2.
3. Déclarer un tableau de 10 `Personne` dont les valeurs seront saisies grâce à la fonction de la question précédente.
4. Ecrire une fonction/procédure `bonAnniversaire` qui prend en paramètre un tableau de 10 `Personnes` et une `Date`. Cet algorithme doit afficher "Bon anniversaire à " suivi du prénom des `Personne` dont l'anniversaire est à la Date passée en paramètre.