TD 4 : Structures pointeurs et tableaux

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 (*Résolution d'équation partie 2*)

1. Définir une structure `Complex` permettant de représenter des nombres complexes.
2. Écrire une fonction `int resoudre(float a, float b, float c, Complex *res1, Complex *res2)` qui donne les solutions de l'équation $ax^2 + bx + c = 0$.
   Cette fonction retourne le nombre de solution de l'équation, sachant que les solutions obtenues peuvent être complexes.

Exercice 2 (*Nombres rationnels*)

1. Declarer une structure `NmbRationnel` qui representera les nombres rationels. Cette structure sera composée de deux entiers representant respectivement le numérateur (`num`) et le dénominateur `den`.
2. Écrire une procédure `afficheNum` qui affichera sous forme "numérateur/dénominateur" un nombre rationnel passé en paramètre.
3. Écrire une procédure `void addMult(NmbRationnel *num1, NmbRationnel *num2)` qui prend en paramètre deux pointeurs sur `nmbRationnel`. Cette fonction modifie les nombres rationnels pointés de la manière suivante :
   — Le nombre rationnel pointé par `num1` devient le **produit** des nombres pointés par `num1` et `num2`. (note : On ne cherchera pas à simplifier le nombre rationnel obtenu)
   — Le nombre rationnel pointé par `num2` devient la **somme** des nombres pointés par `num1` et `num2`. (note : On ne cherchera pas à simplifier le nombre rationnel obtenu)
   — tout nombre rationnel avec un denominateur à 0, génèrera un nombre avec un dénominateur à 0, même dans le cas de la somme.
4. Créer une fonction `NmbRationnel* creerRationnel()` qui va demander à l'utilisateur de saisir un numérateur et dénominateur en faisant attention à limiter la valeur du dénominateur à une valeur strictement positive. Le signe du nombre rationnel sera donc stocké dans le numérateur.
5. Écrire un programme principal qui
   — demande à l'utilisateur de saisir deux `NmbRationnel`
   — affiche sous forme "numérateur/dénominateur" leur somme et leur produit.

Exercice 3 (*gestion d'une classe*)

1. Définir une structure `Etudiant` qui comportera les champs suivants :
   — Le nom de famille de l'étudiant.
   — Son prénom.
   — Son numero de groupe.
   — Un tableau de 6 notes (les 6 notes d'Informatique de l'année).
2. Écrire une fonction constructeur de la structure `Etudiant`. Cette fonction demande à l'utilisateur le nom/ prénom de l'étudiant, son numéro de groupe et rempli aléatoirement le tableau de notes.
3. Écrire une procédure permettant d'afficher les différents champs d'un `Etudiant` passé en paramètre.
4. Tester les deux fonctions précédentes en créant un `Etudiant` dans la fonction principale et en remplissant puis affichant ses champs.
5. On désire gérer une petite promo d'étudiants. Demander à l'utilisateur l'effectif de la promotion et créer un tableau d'`Etudiant` dont la taille correspond à la valeur saisie. Remplir les champs de tous les étudiants puis afficher les informations de chacun d'entre eux.
6. Écrire une fonction `float moyGroupe(Etudiant tab[], int groupe)` qui retourne la moyenne des étudiants appartenant au groupe passée en paramètre.
7. Écrire une procédure `void verifMaj(Etudiant tab[], int groupe)` qui vérifie que le nom de famille des étudiants du tableau passé en paramètre commence bien par une majuscule. Si ce n'est pas le cas on remplace par une majuscule la première lettre.

8. Écrire une procédure pour trier les étudiants du tableau dans l'ordre alphabétique
   — Dans un premier temps en ne les triant que par la première lettre de leur nom.
   — Dans un second temps en considérant leur nom complet.
   On pourra utiliser un des algorithmes de tris de tableau écrits précédemment.

Exercice 4 (*Graphique pyramidal*)

Nous souhaitons afficher les statistiques de fréquentation d'un site internet en fonction de l'heure de la journée, et ce, pour des utilisateurs avec mobiles ou avec ordinateurs.

1. Créer une structure de type `Stats` contenant 1 entier `heure` entre 0 et 23, ainsi que 2 entiers : `mobile` et `ordinateur`, contenant une valeur entre 0 et 100 inclus (les pourcentages affichés)
2. Créer une fonction qui créer une liste de structures `Stats` pour chaque heure de la journée (donc 24) et remplir les champs `mobile` et `ordinateur` avec des valeurs aléatoires
3. Créer une fonction qui prend une liste de `Stats` en paramètre et affiche le graphique pyramidal à la manière de l'exemple ci-dessous

```
------------------------------------------------------------------
| 100%        MOBILES        0% |  | 0%      ORDINATEURS      100% |
------------------------------------------------------------------
|                          #### |23| ##                           |
|                     ########## |22| ########                     |
|                         ######## |21| ##############               |
|                         ######## |22| ##################           |
|            ################### |21| #########                    |
|                         ######## |20| ############                 |
|                       ######### |19| ##                           |
|                  ################ |18| #                            |
|                         ######## |17| #                            |
|                        ######### |16|                              |
|                         ######## |15|                              |
|                  ############### |14| ############################  |
|                      ########### |13| ##############################|
|                           ##### |12| ######################       |
|                               # |11| ###############              |
|                               # |10| ######                       |
|                               # | 9| ##########                   |
|                                 | 8| ######                       |
|                           #### | 7| ####################         |
|                           ##### | 6| ##########                   |
|                               # | 5| ############                 |
|                            #### | 4| #######                      |
|                             ### | 3| ##########                   |
|                              ## | 2| #####                        |
|                             ### | 1| ##                           |
|                          ####### | 0| #####                        |
------------------------------------------------------------------
```