TD Informatique 08 : Tableaux

Consignes générales : N'oubliez pas pour ce TD comme pour les suivants de vous creer un répertoire consacré au TD et d'enregister vos codes desssus.

On rappelle que les commandes à taper dans le terminal pour compiler puis éxécuter votre programme C :
— Pour compiler : *gcc -o nom_executable nom_programme.c*
— Pour exécuter : *./nom_executable*
Il est conseillé de toujours écrire l'algorithme en pseudo-code avant de passer sur machine.

Exercice 1 (*Fonctions de base*)

1. Définir une constante SIZE dont la valeur est 10.
2. Dans le programme principal, déclarer un taleau d'entiers de taille SIZE.
3. Écrire une procédure `printArray` qui prends deux arguments :
   — Un tableau d'entiers
   — Un entier qui correspond à la taille de ce tableau.
   Cette procédure doit afficher les éléments du tableau. La tester avec le tableau défini précédement. Qu'obtient-on ?
4. Écrire une procédure `fillArray` qui prend les même arguments que la procédure `printArray`. Cette procédure doit permettre de remplir le tableau avec des valeurs saisies par l'utilisateur.
5. Tester les procédures en remplissant puis affichant le tableau.

Exercice 2 (*Gestion de notes*)
Nous allons écrire un programme pour gérer les notes d'une classe de 25 élèves.

1. Définir une constante `SIZE` de valeur 25.
2. Dans le programme principal, déclarer un tableau de 25 cases qui servira à stocker les notes des étudiants.
3. Récupérer la procédure de l'exercice précédent permettant d'afficher un tableau. La modifier si besoin pour qu'elle puisse afficher le tableau de cet exercice.
4. Récupérer la procédure `fillArray` et la modifier si besoin pour qu'elle puisse remplir le tableau avec des valeurs aléatoires entre 0 et 20. Rappel : En langage C, on génère des nombres aléatoires avec la fonction `rand()` qu'il faut initialiser avec la commande `srand( time( NULL ) )`. Pour ce faire, il faut inclure les bibliothèques `time.h` et `stdlib.h`.
5. Écrire une fonction `averageArray(int tab[], int size)` qui retourne la moyenne des éléments d'un tableau passé en paramètre. Tester la fonction en affichant la moyenne de la classe.
6. Écrire une fonction `maxArray(int tab[], int size)` qui retourne l'élément maximum contenu dans le tableau passé en paramètre.
7. Écrire une fonction `minArray(int tab[], int size)` qui retourne l'élément minimum contenu dans le tableau passé en paramètre.
8. Afficher les valeurs minimum et maximum du tableau de notes.
9. Écrire une fonction `searchElmt(int tab[], int size, int elmt)` qui parcours le tableau tab et recherche si la valeur `elmt` est dans le tableau. Cette fonction retourne -1 si l'élément recherché n'est pas dans le tableau et le numéro de l'indice où se trouve l'élément s'il est dans le tableau (si l'élément existe plusieurs fois, on renverra l'indice le plus petit). Vérifier avec cette fonction si un étudiant a eu la note de 15.
10. Écrire une fonction `badMarks(int tab[], int size)` qui compte le nombre d'éléments inférieurs à 7 dans le tableau passé en paramètre. Afficher le nombre de notes inférieures à 7 dans le programme principal.

Exercice 3 (*tableau positif*)

1. Récupérer du TP précédent ou écrire à nouveau la fonction qui retourne la valeur absolue d'un nombre passé en paramètre.
2. Dans votre programme principal, déclarer un tableau d'entiers de size 7.
3. Remplir le tableau à l'aide de valeurs saisies par l'utilisateur.
4. A l'aide de la fonction de valeur absolue, modifier les cases du tableau pour qu'elles ne comportent que des éléments positifs. Afficher le tableau pour vérifier.

Exercice 4 (*Pair/impair*)

1. Déclarer un tableau `tab_total` de size 20 et le remplir avec des valeurs aléatoires entre 5 et 10. Afficher le tableau.
2. Écrire une fonction `evenArray(int tab[], int size)` qui retourne le nombre de valeurs paires présentes dans `tab`.
3. Créer deux autres tableaux `tab_even` et `tab_odd` qui vont être remplis de la manière suivante :
   — `tab_even` va prendre toutes les valeurs paires de `tab_total`
   — `tab_odd` va prendre toutes les valeurs impaires de `tab_total`
4. Afficher les tableaux `tab_even` et `tab_odd` (uniquement les valeurs utiles mais pas forcément la totalité des tableaux).

Exercice 5 (*Inversement de tableaux*)

1. Dans le programme principal, déclarer un tableau de taille 15 et le remplir avec des valeurs aléatoires entre 0 et 10. Afficher le tableau.
2. On veut "inverser" ce tableau, c'est à dire que la première valeur sera la dernière, la seconde l'avant dernière etc.
   — Inverser le tableau grâce à un tableau intermédiaire.
   — Inverser le tableau sans utiliser de second tableau.

Exercice 6 (*tableau sans zéros*)

1. Dans le programme principal, déclarer un tableau de taille 15 et le remplir avec des valeurs aléatoires entre 0 et 4. Afficher le tableau.
2. Écrire une procédure `removeZeros(int tab[], int size)` qui parcourt un tableau, et met les éléments qui valent zéro "à la fin du tableau".
   Par exemple le tableau : 1, 3, 0, 4, 0, 0, 2 deviendra 1, 3, 4, 2, 0, 0, 0.
   Vérifier la procédure en la testant avec le tableau de la question 1.