C++ - Feuille TP n°4 Des vecteurs pour stocker plusieurs valeurs (3 séances)

NB : Les codes sources utiles pour ce TP sont disponibles sur Moodle

Objectifs

1. Écrire dans des vecteurs
2. Lire les valeurs stockées dans un vecteur
3. Parcourir un vecteur de différentes manières
4. Modifier un vecteur

Consignes générales pour les exercices sur machine

1. Lisez entièrement chaque question avant de commencer à écrire du code
2. Préparez votre jeu d'essai
3. Écrivez votre code
4. Commentez votre code
5. Testez votre code (affichez le vecteur si vous l'avez modifié même si ce n'est pas demandé) et corrigez si besoin. Indiquez vos tests en commentaire à la fin du fichier.
6. Passez à la question suivante.

TRAVAIL à RENDRE : Déposer sur Moodle le fichier contenant votre code et vos tests à la fin de chaque exercice**.**

Exercice 1 : Manipulations élémentaires d'un vecteur

a) Recopier depuis Moodle le programme **football.cpp** dans votre répertoire de travail.

b) Compléter le programme pour faire afficher le nom de la joueuse mémorisé dans la première case du vecteur **equipe** de 2 façons : avec l'indice et en utilisant la fonction **front**. Même question avec la dernière case (avec l'indice et en utilisant la fonction **back**)

c) Afficher la liste complète des joueuses contenues dans le vecteur **equipe**. NB : cette boucle sera réutilisée à chaque question pour vérifier que le contenu du vecteur est correct.

d) Même question mais on affichera depuis la fin du vecteur (du dernier jusqu'au premier).

e) **Ajout** : Compléter ce programme pour pouvoir **ajouter** des nouvelles joueuses dans l'équipe. Le nombre de joueuses à ajouter est demandé à l'avance. Les nouvelles joueuses seront placées en fin de vecteur. Afficher pour vérifier.

f) **Recherche** : Complétez le programme pour permettre **de tester la présence d'une joueuse donnée** dont le nom est saisi par l'utilisateur.
Le programme devra afficher « joueuse inconnue » ou « joueuse présente dans la case xx du vecteur » selon le résultat (avec xx représentant l'indice). Dans ce cas afficher l'indice et la valeur à cet indice pour vérifier.
Quel type de boucle faut-il utiliser ? Cf amphi d'algo (utiliser un booléen **`trouve`** pour déterminer si la joueuse est présente dans le tableau. Au départ la joueuse n'est pas trouvée, donc **`trouve`** est initialisé à **`false`**. Il est utilisé pour le test de boucle, et mis à jour dans la boucle.

g) **Remplacement** : si la joueuse était présente, compléter le programme pour permettre de **remplacer** la joueuse par une autre. (Vérifier en affichant le contenu du vecteur en utilisant la boucle écrite en question c)).

h) **Normalisation** : afin de faciliter les recherches, il est souhaitable de stocker les chaines de caractères dans un format normalisé. Dans cet exercice on décide que les noms des joueuses sont stockés **en majuscules** dans le vecteur **`equipe`**.
Modifier le programme pour ajouter les instructions permettant de **convertir** le nom de la joueuse **en majuscules** partout où cela est nécessaire. Tester.

Rappels :

- En C++ une chaine de caractères mémorisée dans une variable de type string peut être manipulée comme un vecteur de caractères avec l'opérateur [ ]. Ainsi pour la chaine :

  ```
  string nom="toto";
  ```

  `nom[0]` par exemple désigne le premier caractère (ici **`'t'`**) ,

  et **`nom`**[**`nom.length()-1]`** désigne le dernier caractère (ici **`'o'`**).

- On utilisera la fonction **`toupper`** pour qui retourne un caractère majuscule (nécessitant l'inclusion du fichier **cctype)**. Par exemple si **nom** est une string **`nom[2]=toupper(nom[2])`** met le 3ème caractère de la chaine **`nom`** en majuscules.

Déposer sur Moodle le programme correspondant aux questions a) à h) avec les essais

Exercice 2 : Vecteurs de réels

NB : à partir de la question b) les essais seront constitués avec différents vecteurs, initialisés dès leur déclaration.

a) En utilisant de poly d'algo : écrire un programme **vecteurReels.cpp** qui permet de saisir une suite de nombres réels et de les stocker dans un vecteur **tab**. La lecture des nombres doit s'arrêter après que la valeur STOP (-999 par exemple) a été saisie. STOP n'est pas mémorisé dans le vecteur. Afficher le contenu du vecteur pour vérifier.
N'oubliez pas le jeu d'essais en commentaires.

b) Commentez votre premier programme **vecteurReels.cpp.**
Ecrivez maintenant un programme qui déclare un vecteur **tab** de réels. Initialiser le tableau à sa déclaration avec les valeurs {10,15,9}.
Compléter pour ensuite calculer et afficher la moyenne des valeurs mémorisées dans **tab**. Tester avec différents vecteurs pour constituer le jeu d'essai. Penser à tous les cas.

c) Compléter pour afficher **l'indice et la valeur** du plus grand élément mémorisé dans **tab**. Quelques indications : en vous inspirant des TD/TP précédents, commencer par calculer **la valeur** du max. Tester. Ensuite réfléchir aux modifications à faire pour déterminer non pas le max, mais l'**indice de la case** qui contient le max (penser qu'à partir d'un indice, on peut retrouver la valeur). Tester avec différents tableaux. Penser à tous les cas. Enfin compléter pour déterminer également l'indice (et donc la valeur) du min. Tester.

d) On souhaite séparer en deux tableaux distincts **tabNeg** et **tabPos** respectivement les valeurs négatives et positives contenues dans **tab**. Afficher les contenus des tableaux pour vérifier.

```
Contenu de tab : 1.2, -2.0, 13.4, 14.5, -13.6
Contenu de tabNeg :  -2.0 , -13.6
Contenu de tabPos :   1.2, 13.4, 14.5
```

e) Compléter le programme pour supprimer toutes les valeurs nulles du vecteur tab. NB : il n'est pas nécessaire de conserver l'ordre des valeurs. On peut ainsi procéder en remplaçant chaque valeur nulle par la dernière valeur du vecteur, puis retirer la dernière valeur. N'oubliez pas d'afficher le tableau avant et après…

```
Contenu de tab avant :      tab: 1.0, 0, -2.2, 13.6, 0, 0, 14.5, -13.8
Contenu de tab après : 1.0, -2.2, 13.6, 14.5, -13. 8
```

f) **indirection** : reprendre la question 4 pour cette fois-ci mémoriser les **indices** (et non les valeurs) des cases du tableau tab qui contiennent les valeurs négatives (resp. positives). Utiliser ces tableaux pour afficher ensuite l'ensemble des valeurs négatives, suivies de l'ensemble des valeurs positives de tab.

```
Contenu de tab : 1.2, -2.0, 13.4, 14.5, -13.6
Contenu indNeg : 1, 4
Contenu de indPos : 0, 2, 3
```

On affichera :

```
Valeurs négatives : -2.0, -13.6, 1.2
Valeurs positives : 13.4, 14.5
```