TP8 POO - C++

gilles.scarella@univ-cotedazur.fr, simon.girel@univ-cotedazur.fr

NB: Pour bien comprendre le sujet de ce TP, il faut lire préalablement les pages 28 à 30 du cours 4 sur Moodle qui n'ont pas été montrées pendant la séance.

Consignes:

- Pour compiler le code, on exécutera dans le terminal
  g++ main.cpp MyVector.cpp -o MyVector
  Pour tester le code, après l'avoir compilé avec la méthode précédente, on exécutera dans le terminal
  ./MyVector
- Les déclarations devront être écrites dans le fichier *MyVector.hh*, les définitions dans *MyVector.cpp* et la fonction *main* contenant les exemples dans *main.cpp*.

Le but de ce TP est d'implémenter une classe permettant de modéliser un vecteur de $\mathbb{R}^n$, où n est quelconque, et de l'utiliser dans quelques exemples.

1 Attributs et constructeurs de la classe *MyVector*

- Dans le fichier *MyVector.hh*, déclarer la classe *MyVector*.

  Cette classe aura deux données membres privées:

  - $\_n$ qui contiendra la longueur du tableau contenu dans notre objet *MyVector*, de type entier non signé.
  - $\_vals$ qui sera un pointeur sur *double*.
- Déclarer un premier constructeur pour notre classe, prenant un entier non signé en paramètre et qui initialisera $\_n$ et qui allouera l'espace mémoire nécessaire pour $\_vals$. Il initialisera toutes les données de ce tableau à 0. Le code de ce constructeur devra être implémenté dans le fichier *MyVector.cpp*.
- Ecrire un deuxième constructeur, qui prendra cette fois un entier non signé et un pointeur sur *double* en paramètre. Il initialisera les deux données membres $\_n$ et $\_vals$ grâce aux deux paramètres.

  Attention : $\_vals$ doit contenir des valeurs égales à celles du tableau passé en paramètre mais ces deux pointeurs ne doivent pas pointer vers la même adresse mémoire...
- Tester les deux constructeurs dans une fonction *main*, définie dans un fichier *main.cpp* en utilisant simplement les deux constructeurs.

2 Destructeur de la classe

- Déclarer et implémenter le destructeur de cette classe.

3 Fonction membre *affiche*

- Ecrire une fonction *affiche*, fonction membre de la classe, qui effectuera un affichage de notre vecteur. Cette fonction sera utilisée dans la suite pour vérification.
- La tester dans la fonction *main* sur au moins une des variables précédemment créées.

4 Fonction membre *prod_scal*

- Ecrire une fonction membre *prod_scal* qui renverra le produit scalaire du vecteur instancié courant et d'un autre vecteur passé en paramètre. On supposera que les dimensions des deux vecteurs sont identiques.
- La tester dans la fonction *main*.

5 Fonction membre *resize*

- Ecrire une fonction membre *resize*, prenant un paramètre *newSize* de type entier non signé.

  Cette fonction modifiera la taille du tableau *_vals* ainsi que la valeur de *_n*, en ne renvoyant rien. On utilisera l'algorithme suivant:
  Si $newSize > \_n$, alors les nouvelles valeurs de *_vals* seront initialisées à 0.
  Si $newSize < \_n$, les dernières données de *_vals* seront détruites.
  Si $newSize == \_n$, on ne change rien.
- La tester dans la fonction *main* sur au moins deux cas pertinents.

6 Fonction membre *reduce*

- On se propose d'implémenter une fonction *reduce*, membre de la classe *MyVector*, prenant en paramètre un pointeur sur fonction prenant deux paramètres de type *double* et renvoyant une valeur de type *double*. Cette fonction retournera une valeur de type *double*.

  Plus précisément, on voudrait que la fonction *reduce* appliquée sur un objet *MyVector* v et sur la bonne fonction f retourne la somme s des

éléments du vecteur v de telle sorte que
(pour simplifier, on note n pour $\_n$ et v_i pour $v.\_vals[i]$)

$$
\begin{array}{l}
s = 0 \\
\text{for } i = 0 \text{ à } n-1 \\
\quad s = s + v_i = f(s, v_i)
\end{array}
$$

Par exemple, *reduce* appliquée au vecteur v de $\mathbb{R}^4$, $v = (1, 2, 3, 4)$, et à la fonction qui additionne deux nombres, renverra 10.

- La tester dans le *main* après avoir choisi la bonne fonction comme argument.

7 Fonction membre *myMediane*

- Déclarer et définir une fonction *myMediane*, membre de la classe *myVector*, qui ne prend aucun argument et qui renvoie un *double*. La valeur renvoyée correspond à la médiane du vecteur sauf si le vecteur est de taille 0 ou s'il n'est pas dans l'ordre croissant. Dans ce cas-là, la valeur renvoyée vaudra -1.

- La tester dans le *main*.

- [**Facultatif**] Déclarer et définir la fonction *myMedianeV2*, qui a le même but que *myMediane*, à l'exception que cette fonction renvoie la valeur médiane du vecteur dans le cas général. Un tri des éléments du vecteur est alors nécessaire (par exemple, vous pouvez utiliser l'algorithme de tri à bulles, expliqué sur *Wikipedia*, pour trier les éléments). La tester.