Génie Logiciel TP2 : amies, copies et co.

L'objectif de ce TP est de créer plusieurs classes interdépendantes organisées dans des fichiers d'en-têtes (extension `.hpp`) et sources (extension `.cpp`) et d'utiliser le mot-clé **`friend`** pour donner des accès à des membres privés de ces classes.

On veut implémenter des classes pour manipuler des matrices et des vecteurs en dimension 3 et y appliquer quelques opérations comme un mise à l'échelle d'un vecteur, le produit scalaire et le produit matrice $\times$ vecteur. La visibilité de ces classes sera artificiellement restreinte pour forcer l'utilisation du mot-clé **`friend`**. On devra ensuite se contenter de l'interface développée pour implémenter la méthode de la puissance qui permet d'approcher la valeur propre de valeur absolue maximale d'une matrice (sous certaines conditions). Enfin, on envisagera de généraliser ces implémentations en dimension quelconque.

1 Vecteur en dimension 3

a) On commence par implémenter, dans des fichiers `vecteur.hpp` et `vecteur.cpp`, la classe `Vecteur` qui modélise un vecteur de $\mathbb{R}^3$. Cette implémentation sera composée de :
   - un tableau **privé** qui contiendra les coefficients du vecteur,
   - un **constructeur par défaut** qui permet également de spécifier les coefficients du vecteur,
   - un **constructeur par copie**,
   - un **destructeur**,
   - une méthode publique **affiche** qui affiche le vecteur dans la sortie standard.
   - un **operateur d'assignation** (de signature `Vecteur& operator= (Vecteur const& v)`),

   L'absence d'accesseurs pour les coefficients est ici volontaire dans le cadre de l'exercice.
   Remarque : n'hésitez pas implémenter la méthode `affiche` immédiatement après avoir implémenter le constructeur par défaut afin de pouvoir tester rapidement votre implémentation.

b) Ajoutez une méthode publique `echelle` qui, étant donné un facteur réel, met à l'échelle le vecteur courant (multiplie chaque coefficient par le facteur). Testez.

c) Ajoutez dans les mêmes fichiers, une fonction **amie** nommée `produit` qui, étant données deux vecteurs, renvoie leur produit scalaire.

Question bonus : remplacez la fonction `produit` et la méthode `echelle` par des surcharges de l'opérateur `*`.

2 Matrice carrée en dimension 3

a) On continue avec l'implémentation, dans des fichiers `matrice.hpp` et `matrice.cpp`, de la classe `Matrice` qui modélise une matrice de $\mathcal{M}_{3,3}(\mathbb{R})$. Cette implémentation sera composée de :
   - un tableau **privé** qui contiendra les coefficients de la matrice,
   - un **constructeur par défaut** public qui permet également de spécifier les coefficients de la matrice,
   - un **constructeur par copie**,
   - un **destructeur**,
   - une méthode publique `affiche` qui affiche la matrice dans la sortie standard.

b) Ajouter une méthode publique `produit` qui étant donné un vecteur, renvoie le produit matriciel de la matrice par ce vecteur. Quelle **amitié** faut-il déclarer pour que cette méthode puisse fonctionner ? Testez.

Questions bonus : remplacez cette méthode `produit` par une fonction amie acceptant une matrice et un vecteur. Vous pouvez également la remplacer par une surcharge de l'opérateur `*`.

3 La méthode de la puissance

On considère une matrice diagonalisable A dont la **valeur propre dominante** λ_i est de multiplicité un, c'est-à-dire que $|\lambda_i| > |\lambda_j|$ pour tous $j \neq i$.

Étant donné un vecteur initial $w^{(0)}$ n'appartenant pas au sous-espace caractéristique généré par les autres vecteurs propres $v_{j\neq i}$, alors la suite récurrente $(w^{(n)})_{n\in\mathbb{N}}$ définie par :

$$\forall n \in \mathbb{N}, \quad w^{(n+1)} := \frac{Aw^{(n)}}{\|Aw^{(n)}\|}$$

est convergente et

$$\lim_{n\to+\infty} \frac{\langle w^{(n)}, Aw^{(n)}\rangle}{\langle w^{(n)}, w^{(n)}\rangle} = \lambda_i.$$

a) Dans des fichiers `puissance.hpp` et `puissance.cpp`, implémentez une fonction `puissance` qui, étant donnée une matrice, un vecteur initial et un nombre d'itérés, calcule ce même nombre d'itérés de la suite $w^{(n)}$ et renvoie l'estimation obtenue de la valeur propre dominante.

Attention : cette fonction ne devra utiliser que l'interface publique proposé par `Vecteur` et `Matrice`.

b) Testez avec la matrice

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & -9 \end{bmatrix}$$

et un vecteur initial de votre choix.

4 Dimension arbitraire

Nous allons maintenant considérer des vecteurs et des matrices en dimension n quelconque.

a) la classe `Vecteur` aura maintenant deux constructeurs :

i. un constructeur par défaut qui accepte la taille du vecteur,

ii. un constructeur qui accepte la taille du vecteur et une fonction qui a un indice associe une valeur. Cette fonction sera utilisée pour **remplir le vecteur**.

b) mettez à jour les fonctions ou méthodes `produit` (et `echelle`) et testez votre implémentation en calculant un produit scalaire en **dimension 5**, puis en **dimension 3** pour calculer la valeur propre dominante de la matrice de l'exercice précédent.

c) **Bonus :** la classe `Matrice` aura maintenant deux constructeurs :

i. un constructeur par défaut qui accepte les **deux dimensions** de la matrice,

ii. un constructeur qui accepte les dimensions de la matrice et une fonction qui a un couple d'indices (i, j) associe une valeur. Cette fonction sera utilisée pour **remplir la matrice**.

Remarque : pour une matrice de dimension $m \times n$, vous pouvez allouer un tableau de longueur mn et utiliser la bijection $(i, j) \mapsto in + j$ pour accéder aux coefficients. Vous pouvez sinon créer un tableau de tableaux.

d) **Bonus :** mettez à jour les fonctions ou méthodes `produit` (et `echelle`) et testez votre implémentation en **dimension 5** pour calculer la valeur propre dominante de la matrice définie par $(i, j) \mapsto ij+1$ (indices commençant à 0).