TP - Matrices (C++)

Contraintes et Remise du TP

- **Chaque classe doit correspondre à un couple de fichier hpp/cpp séparé** (sauf cas particulier : classe template, classes très petites ...)
- Il n'est pas demandé d'utiliser un outil de documentation particulier (Doxygen, etc.) mais indentez et commentez votre code.
- De même, il n'est pas demandé de tests untaires, mais testez votre code dans un main. (L'utilisation d'un framework de test – Boost.Test, Catch, Google Test – donnera lieu à un bonus.)
- La correction tiendra compte de la briéveté des méthodes que vous écrivez (évitez les fonctions de plus de 25 lignes) ; n'hésitez pas à découper une méthodes en plusieurs sous-méthodes (privées) plus courtes.
- Les noms de classe commencent par une majuscule.
- Les noms de méthodes et d'attributs commencent par une minuscule.
- La convention de nommage des accesseur est getNomAttribut() et setNomAttribut(...).
- Le code source et les éventuels autres documents (texte, diagrammes, etc.) doivent être envoyés sous forme d'une archive tar.gz.
- Le rendu est à déposer sur le Moodle au plus tard 15 minutes après la fin du TP.

Barème

- **Classe Matrice** - 15 points (fonctionnalités 12, documentation et gestion d'erreurs 3)
- **Classe Vecteur** - 5 points (fonctionnalités 4, documentation et gestion d'erreurs 1)

Sujet

Le but de ce TP est de programmer une petite bibliothèque d'algèbre linéaire autour d'une classe Matrice équipée des opérateurs courants (+, -, *, etc.).

Le code devra être documenté par des commentaires et testé dans un main. Les erreurs (accès en dehors de la matrice, opérations entre matrices de mauvaises dimensions, trace d'une matrice non carrée, etc.) devront être détectées et lancer des exceptions (comme par exemple std :: runtime_error).

Classe Matrice

Pour cette première partie, nous travaillerons avec des matrices de double. Voici ce que doit être capable de faire votre classe :

- constructeur qui construit une matrice nulle d'une taille donnée ;
- accès en lecture et écriture à la case (i, j) via l'opérateur () ;
- copie entre deux matrices (via l'opérateur = et le constructeur de copie) ;
- multiplication, addition et soustraction entre matrices ;
- multiplication, addition et soustraction d'un scalaire (ajouter un scalaire = l'ajouter à tous les coefficients de la matrice) ;
- trace d'une matrice carrée (somme des coefficients diagonaux) ;
- comparaison de matrices via l'opérateur == ;
- Méthode de classe renvoyant une matrice identité d'une taille donnée ;
- Méthode de classe renvoyant une matrice aléatoire (valeurs dans [0; 1]) d'une taille donnée ;

Vous devez par ailleurs écrire (en dehors de la classe) une fonction du type : **std :: ostream& operator<<(std::ostream&, const Matrice& m)** permettant d'afficher la matrice dans un flux ostream.

Les tests unitaires ne sont pas demandés, mais vous devez fournir un main démontrant les capacités de votre classe.

Classe Vecteur

Dans un second temps, vous devez programmer une classe Vecteur (Matrice colonne) dérivée de Matrice qui ajoute les opérations suivantes :

- produit scalaire ;
- produit vectoriel (pour les vecteurs de taille 3) ;
- norme ;
- accès en lecture/écriture à l'élément i avec l'opérateur [].

Produit Matrice/Vecteur

Votre code devrait "naturellement" permettre de réaliser des produits matrice-vecteur, l'objet Vecteur étant considéré comme un objet Matrice (colonne) et pouvant être traité par l'operator * de Matrice (subsomption).

Cependant, le résultat de cette opération est un également un objet Matrice (colonne), et non un objet Vecteur. Ajoutez une autre fonction operator * à la classe Matrice, qui permet de réaliser des produits matrice-vecteur et renvoyant un objet Vecteur.