LA CLASSE MATRICE

Dans le cadre de ce second exercice, vous allez vous entrainer à développer des classes en langage C++ puis à les utiliser. Ainsi la partie conceptuelle vue en cours vous paraitra (surement) plus limpide…

Dans le cadre de cet exercice, vous allez vous pencher sur la création d'une classe dont l'objectif est de simplifier la manipulation de matrices. Pour cela, on s'intéressera à la définition de la classe nommée « Matrice» permettant de réaliser quelques-unes des opérations élémentaires.

PARTIE 1 :

Dans un premier temps, une « Matrice » sera composée de N x M données entières. Pour vous simplifier la tâche (ou pas) vous stockerez les informations contenues dans la matrice au sein d'un tableau monodimensionnel alloué dynamiquement.

Votre classe devra fournir les fonctionnalités suivantes à l'utilisateur :

- Un constructeur par défaut permettant de créer une matrice de N colonnes et M lignes ;
- Un destructeur ;
- Un accesseur et un mutateur pour accéder aux éléments de la matrice (get/set) ;
- Des accesseurs pour permettre la lecture des dimensions de la matrice (width, height) ;
- Un accesseur et un mutateur pour lire et écrire une donnée à la position (x,y) dans la matrice (get, set) ;
- Une fonction membre permettant d'afficher de manière TRES propre la matrice à l'écran (show);
- Une méthode permettant d'additionner une donnée à l'ensemble des éléments de la matrice (add) ;
- Une méthode permettant de soustraire une donnée à l'ensemble des éléments de la matrice (sub) ;
- Une méthode permettant de multiplier une donnée avec l'ensemble des éléments de la matrice (mult) ;
- Une méthode permettant d'additionner deux matrices entre elles si leurs tailles sont compatibles (add), cette fonction retournera une nouvelle matrice contenant le résultat de l'addition;
- Une méthode permettant de multiplier deux matrices entre elles (mult), cette fonction retournera une nouvelle matrice contenant le résultat de la multiplication ;

QUESTION 1 – STRUCTURE DE LA CLASSE

Ecrivez la structure de la classe répondant aux besoins énoncés.

QUESTION 2 – LES METHODES

Ecrivez le code source des méthodes contenues dans la classe.

QUESTION 3 – LE PROGRAMME MAIN

Ecrivez un programme permettant de vérifier le bon fonctionnement de votre création.

QUESTION 4 - TESTEZ SUR MACHINE VOTRE SOLUTION

Pour pouvoir tester votre code chez vous ou à l'école (en salle E210 par exemple), vous avec besoin d'avoir un éditeur de texte (gedit, emacs, …) et un compilateur (GCC). A l'aide de l'éditeur, écrivez tout vos codes dans un seul fichier nommé matrice.cpp et ensuite compiler le tout avec :

```
g++ matrice.cpp –o matrice -Wall
```

Ensuite il ne vous reste plus qu'à exécuter le programme ainsi généré.

QUESTION 5 – EXTENSION DES FONCTIONNALITES

Ajoutez les fonctionnalités suivantes pour rendre l'utilisation de votre classe Matrice plus sympathique :

- Une fonction membre permettant de cloner une Matrice. Vous utiliserez le prototype de méthode suivant :

```
Matrice* clone( ) ;
```

- Ajoutez la même fonctionnalité mais à l'aide cette fois ci d'un nouveau constructeur.
- Ajoutez une méthode permettant d'appliquer une rotation de 90° au contenu de la matrice (rotate90).
- Ajoutez des méthodes permettant de calculer le minimum (min), le maximum (max), de la somme (sum) et de la moyenne (avg) des éléments contenus dans une matrice.
- Une méthode permettant d'afficher dans le flux de sortie les données mémorisées dans l'objet. Pour cela vous devrez définir en dehors de la classe une fonction possédant le prototype suivant :

```
std::ostream& operator<<(std::ostream& out, const Matrice& lhs)
```

Vous devrez prendre soin d'utiliser le mot clé const sur vos méthodes

- Faites de même pour les opérateurs +, -, *, ==, != , etc. Pour vous aider, vous trouverez le prototype des fonctions à déclarer ainsi que quelques explications sur les pages suivantes :

http://en.cppreference.com/w/cpp/language/operators

http://www.tutorialspoint.com/cplusplus/cpp_overloading.htm

Modifiez votre programme principal afin de vérifier le bon fonctionnement de l'ensemble des nouvelles fonctionnalités que vous avez intégré dans votre classe.

PARTIE 2 :

Une fois que l'ensemble des travaux à réaliser dans la première partie de cet exercice a été validé, nous allons nous attacher à la seconde partie qui va considérer la gestion des matrices particulières.

- Développez une classe MatriceCarrée dont le constructeur ne prend qu'un seul paramètre N.
- Développez une classe MatriceIdentité qui sera par définition carrée et dont la valeur des éléments à l'initialisation sera 0 sauf sur la diagonale.

PARTIE 3 :

Pour rendre votre classe vraiment intéressante et réutilisable, il est indispensable de :

- Rendre le format des données configurable à la compilation (=> template).
- De gérer les problèmes à l'exécution plus proprement (=> exception)

Adaptez vos codes sources afin de prendre en compte ces 2 points.