Héritage et classe abstraite, application aux matrices

Exercice 1 Le but de cet exercice est d'appliquer l'héritage aux matrices. Une représentation de ce que l'on veut est présenté sur la Figure 1.

ATTENTION :

1. le mot clé **const** est mal placé dans le diagramme.
2. les fonctions ne sont pas mentionnées dans le diagramme.
3. les opérateurs **operator()** sont virtuels, et sont purs dans la classe **BMatrix**.

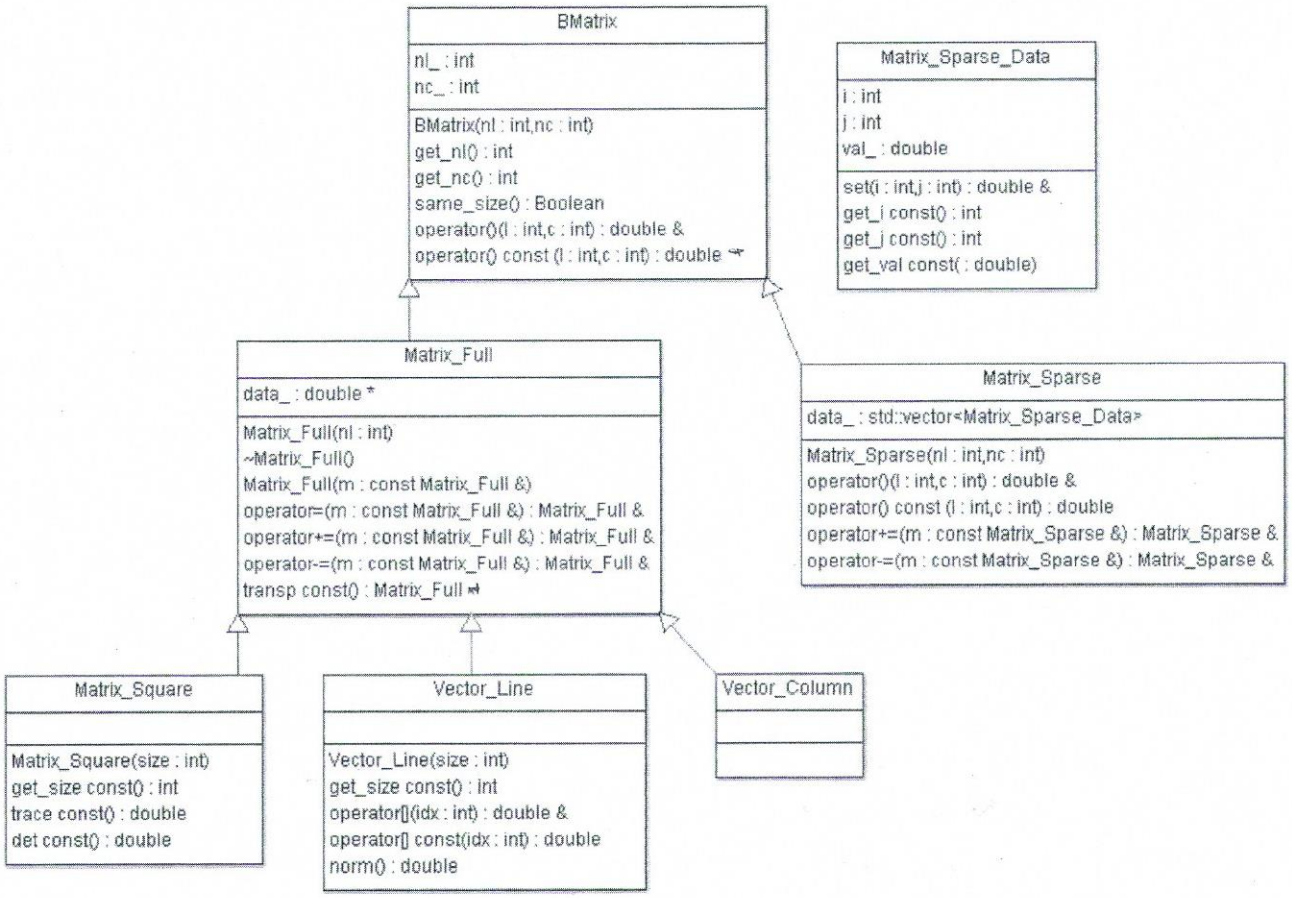

FIGURE 1 – Diagramme de classes

On écrira un fichier d'en-tête par classe, et un fichier source par classe pour l'implémentation des méthodes, ainsi qu'un fichier de test avec la fonction **main()**.

On pensera à utiliser le namespace **ensiie** ainsi que les exceptions quand cela est nécessaire. De plus, il faudra réutiliser le plus possible le code qui a été déjà écrit pour le factoriser le plus possible.

Pour les fonctions mathématiques, on inclura le fichier *cmath*. Les fonctions **double pow(double a, double b)**, **double sqrt(double x)** et **double fabs(double x)** renvoie respectivement a^b, $\sqrt{x}$ et $|x|$.

La description des classes est la suivante :

1. La classe **BMatrix** : classe abstraite
   - Membres de données : **nl_** et **nc_** pour le nombre de lignes et de colonnes des matrices.
   - Méthodes (toutes définies en-ligne)) :
     - **int get_nl() const** et **int get_nl() const** renvoyant les valeurs des membres de données.
     - La méthode **bool same_size(const BMatrix &)** qui teste si la matrice en paramètre a la même taille que l'objet courant.
     - L'opérateur **double operator()(int l, int c)** qui est virtuel pur (surchargé pour la lecture et l'écriture d'un élément de la matrice).
2. La classe **Matrix_Full** : hérite de la classe **BMatrix**.
   - Membre de données : **double *data_**, stockant les éléments de la matrice.
   - Méthodes :
     - Les constructeur, destructeur, constructeur de copie et **operator=**.
     - Les 2 **operator()**.
     - **Matrix_Full & operator+=(const Matrix_Full & m)**.
     - **Matrix_Full & operator-=(const Matrix_Full & m)**.
     - **Matrix_Full transp() const** pour la transposée.
   - Fonctions :
     - Les opérations algébriques : addition, soustraction et multiplications de 2 matrices, ainsi que la multiplication d'une matrice avec un scalaire et la multiplication d'une matrice avec un vecteur (vecteur ligne à gauche et vecteur colonne à droite).
     - L'opérateur de flux **operator«** pour afficher une matrice.
3. La classe **Matrix_Square** : hérite de la classe **Matrix_Full**.
   - Méthodes :
     - Le constructeur **Matrix_Square(int size)**.
     - l'accesseur **int get_size()** qui renvoie la taille de la matrice.
     - Une méthode **const** qui renvoie la trace d'une matrice.
     - Une méthode **const** qui renvoie le déterminant d'une matrice.
4. La classe **Vector_Line** : hérite de la classe **Matrix_Full**.
   - Méthodes :
     - Le constructeur **Vector_Line(int size)**.
     - l'accesseur **int get_size()** qui renvoie la taille du vecteur.
     - Les 2 **operator[]** pour accéder en lecture et en écriture aux éléments du vecteur.
     - La méthode **norm(double p)** qui renvoie la norme p du vecteur, en gérant le cas particulier $p = 2$.
   - La fonction **operator*** qui renvoie le produit scalaire de deux vecteurs.
5. La classe **Vector_Column** : hérite de la classe **Matrix_Full**, même implémentation que **Vector_Line**.
6. La classe **Matrix_Sparse** (matrice creuse) : hérite de la classe **BMatrix**. La stratégie est d'avoir une classe **Matrix_Sparse_Data** ayant trois membres de données : **i_, j_, val_** pour la ligne, la colonne et la valeur d'un élément de la matrice. La matrice creuse est alors une liste déléments de type **Matrix_Sparse_Data**, spécifiant uniquement les éléments de la matrice qui sont non nuls. Cela permet d'optimiser la consommation de la mémoire.
   - Membres de données : **data_** de type un **std : :vector** de **Matrix_Sparse_Data** (ce n'est pas le plus optimal et sera amélioré plus tard).
   - Méthodes :
     - Le contructeur **Matrix_Sparse(int nl, int nc)**.
     - toutes les méthodes de **Matrix_Full**.
   - Fonctions : toutes celles de **Matrix_Full**.