Héritage et classe abstraite, application aux matrices - 2

Exercice 1 Nous allons ajouter une classe dérivant de la classe abstraite et qui implémente des matrices creuses : nous allons utiliser 3 tableaux **val_**, **idx_** et **start_** comme membres de donnée. Pour chaque coefficient non nul, la valeur sera conservée dans le tableau **val_**, ordonné par indice de ligne croissant, puis par indice de colonne croissant. En même temps, on remplit le tableau **idx_** avec l'indice de la colonne du coefficient. Les deux tableaux ont donc la même taille. Enfin, le tableau **start_** sera tel que **start_[i]** sera l'indice de début de la i-ème ligne dans les tableaux précédents. Ainsi, le nombre de coefficients non nuls de la ligne **i** sera **start_[i+1] - start_[i]**.
Ainsi on ne stocke que les coefficients non nuls (d'où un gain de place en mémoire).

1. Implémenter les méthodes et fonctions classiques des matrices (pleines) pour les matrices creuses.
2. Implémenter dans la classe **Matrix_Sparse** la méthode **Matrix full_get() const** qui renvoie la matrice pleine associée à une matrice creuse.
3. Implémenter dans la classe **Matrix** la méthode **Matrix_Sparse sparse_get() const** qui renvoie la matrice creuse associée à une matrice pleine.

Exercice 2 Reprendre le TP précédent en ré-écrivant les classes dans le namespace **va** qui implémente les matrices pleines avec un **std : :valarray**, et mesurer la différence de vitesse des multiplications matricielles entre les deux implémentations (utiliser la fonction **gettimeofday()** pour ceci).