Rapport - Projet Matrice

COUCHOUD Thomas COLEAU Victor

29 mars 2017

Table des matières

Chapitre 1

Présentation du sujet

L'objectif de ce projet est de réaliser une librairie de classes et de fonctions permettant de manipuler des matrices.

Chapitre 2

Architecture

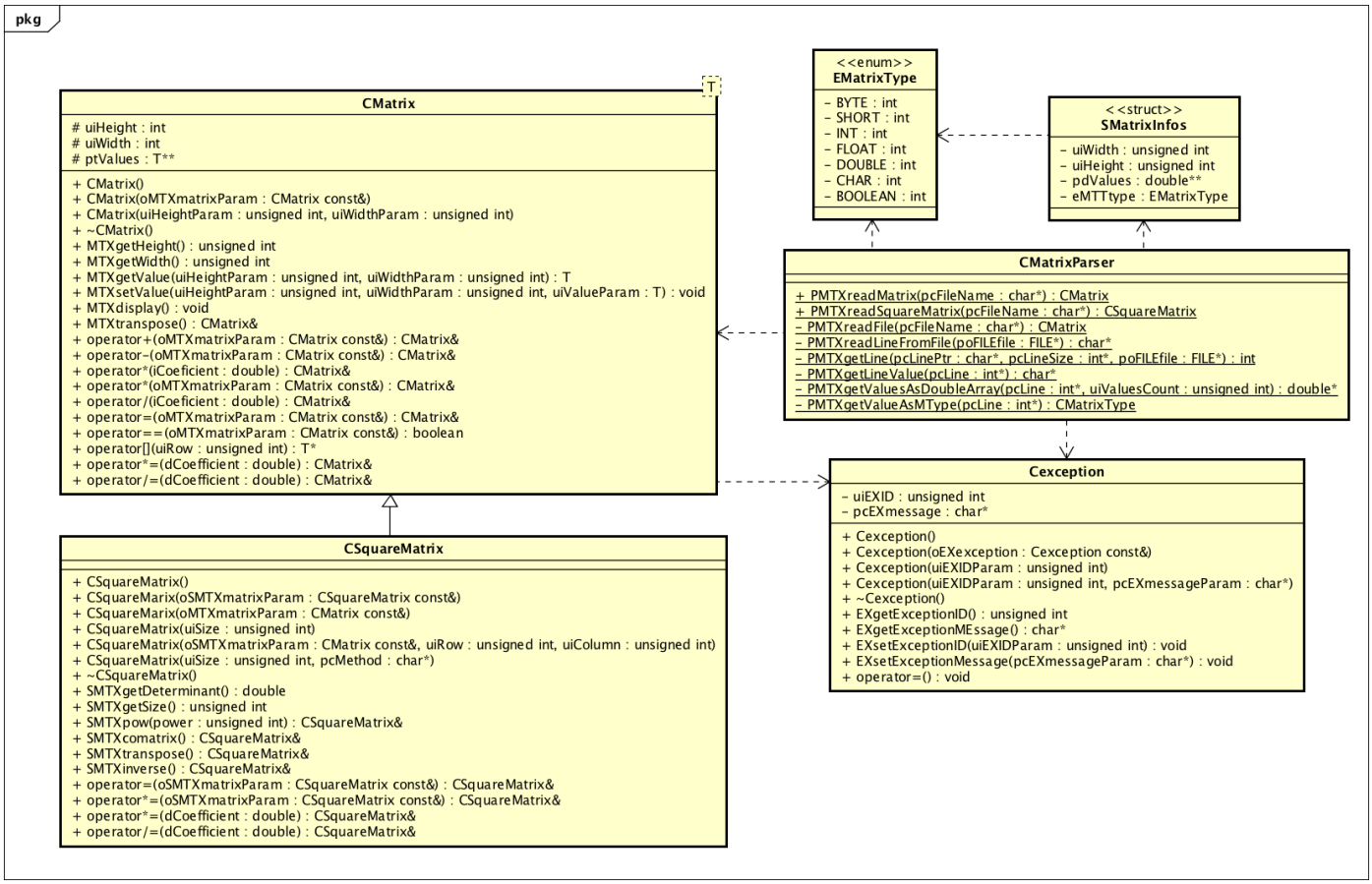

FIGURE 2.1 – Diagramme de classes

La classe centrale de notre projet est CMatrix. Celle-ci représente une matrice. Y sont stockées les informations suivantes :

- La hauteur représentant le nombre de lignes de la matrice.
- La largeur représentant le nombre de colonnes de la matrice.
- Un tableau 2D contenant l'ensemble des valeurs de la matrice.

De plus, cette classe contient l'ensemble des méthodes demandées telles que :

- L'opérateur d'addition avec une deuxième matrice en renvoyant une nouvelle.
- L'opérateur dde soustraction avec une deuxième matrice en renvoyant une nouvelle.
- L'opérateur de multiplication par un scalaire (à droite et à gauche) renvoyant une nouvelle matrice.

- L'opérateur de division par un scalaire renvoyant une nouvelle matrice.
- L'opérateur de multiplication par une autre matrice en renvoyant une nouvelle.
- L'affichage d'une matrice dans la console.
- Le calcul et la création de la transposée d'une matrice sans changer l'originale.
- L'opérateur d'affection copiant le contenu d'une matrice dans une autre.
- L'opérateur de comparaison.
- Les opérateurs *= et /= avec un scalaire ou une matrice. Ceux-ci modifient la matrice en cours.
- L'opérateur parenthèse prennant deux paramètres et renvoyant la valeur contenue dans la case demandée.

Nous avons ensuite fait hériter à cette classe une autre classe appelée CSquareMatrix représentant elle aussi une matrice mais carrée. Celle-ci contient les mêmes informations que sa classe mère mais propose des méthodes supplémentaires spécifiques aux calculs mathématiques réservés aux matrices carrées :

- Calcul du déterminant.
- Calcul de l'inverse.
- Calcul de la comatrice.
- Calcul des puissances d'une matrice.

Afin de pouvoir lire des matrices externes au programme, nous avons implémenté une classe statique CMatrixParser. Cette dernière lira un fichier texte du format indiqué et renverra au choix, une matrice ou une matrice carrée. Pour réaliser cela, cette dernière s'appuie sur une énumération et une structure. L'énumération eMatrixType permet de faire la transition entre le type de la matrice écrite dans le fichier et le type du language. La structure sMatrixInfo stocke de manière temporaire les informations de la matrice lues dans le fichier.

Chapitre 3

Choix de codage

Le choix de créer une classe CSquareMatrix héritée de CMatrix vient des nombreuses opérations mathématiques propres aux matrices carrées. Nous aurions put implémenter celles-ci directement dans la classe des matrices quelconques mais nous aurions alors du vérifier à chaque appel si la matrice avait le même nombre de lignes et de collones et lever une exception en cas d'échec. Grâce à ce choix, nous évitons une grand nombre de vérifications sans pour autant géner les calculs normaux puisque des matrices carrées sont aussi des matrices simples et donc peuvent être multipliées, transposées, etc..

Chapitre 4

Tests effectués

Afin de pouvoir valider le fonctionnement de notre code, nous avons réalisé de nombreux tests. Certains ont été réalisés de manière manuelle mais un bon nombre d'entre eux ont été écrits sous forme de "test unitaires". Ceux-ci sont présents dans les fichiers CXXXTest.cpp qui vont respectivement tester leur classes XXX.

Nous sommes conscients que les tests sont très minimalistes et ne couvrent pas tout, cependant cela est déjà un bon départ pour vérifier un fonctionnement "normal".

Commençons par CExceptionUnit :

- Tests des différents constructeur impliquant l'ID de l'exception ainsi que les getters et setters de celui-ci.
- Tests des différents constructeur impliquant le message de l'exception ainsi que les getters et setters de celui-ci.

Ceux-si sont très généralistes mais couvre relativement bien les différentes utilisations d'un objet de cette classe.

Intéressons nous maintenant à CMatrixUnit :

- Test des différents constructeurs ainsi que les getters concernant le nombre de lignes et le nombre de colonnes.
- Tests des getters et setters sur les valeurs de la matrice.
- Tests des différentes opérations :
- — Transposée d'une matrice.
  - Multiplication par une coefficient.
  - Division par un coefficient.
  - Division par le coefficient 0 renvoyant une exception.
  - Test d'égalité de matrices.
  - Test d'affectation de matrices.
  - Test de l'opérateur ()
  - Test de l'opérateur () avec coordonnées invalides renvoyant une exception.
  - Test de la somme de deux matrices.
  - Test de la sous traction de deux matrices.
  - Test du produit de deux matrices.
  - Test du produit de deux matrices incompatibles renvoyant une exception.
  - Test de l'opérateur *= avec un coefficient.
  - Test de l'opérateur *= avec une matrice.
  - Test de l'opérateur *= avec une matrice incompatible renvoyant une exception.
  - Test de l'opérateur /= avec un coefficient.
  - Test de l'opérateur /= avec un coefficient 0 renvoyant une exception.

Ces tests nous semblent assez complet sur les matrices. Cependant chaque test n'est effectué qu'une seule fois ce qui peut remettre en doute l'efficacité du test.

Passons ensuite aux tests contenus dans CSquareMatrixUnit :

— Test des différents constructeurs ainsi que les getters concernant la dimension de la matrice.
— Tests des getters et setters sur les valeurs de la matrice.
— Tests des différentes opérations :
— — Test du calcul du déterminant.
  — Test du calcul d'une comatrice.
  — Test du calcul d'une matrice inverse.
  — Test du calcul d'une matrice inverse avec déterminant nul, renvoyant une exception.
  — Calcul de la puissance d'une matrice.

Chapitre 5

Conseils d'utilisation