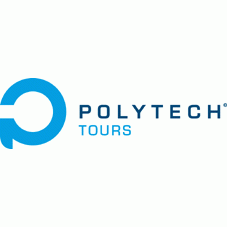
PERVEYRIE Hugo DI3 Groupe 2

DURET Romuald

Compte-rendu de projet :

# Librairie de matrice (C++)



18/04/2021

# Introduction :

Dans ce rapport vous trouverez tout le nécessaire à la compréhension de notre démarche sur la conception et la réalisation de ce projet. Par ailleurs, comme vu précédemment, il y a une partie consacrée à l’utilisation de la librairie afin d’accompagner tout utilisateur qui voudrait l’utiliser.

Tout d’abord, la première partie vous permettra de comprendre notre réflexion sur la conception du projet. Ce travail en amont permit de pouvoir travailler dans de bonnes conditions et d’une façon structurée.

Ensuite, la seconde partie expliquera comment s’est déroulé le développement de la librairie. Nous parlerons donc de la prise en main de l’IDE, les difficultés que nous avons rencontrées ainsi que des différentes méthodes de développement que nous nous sommes employés à utiliser.

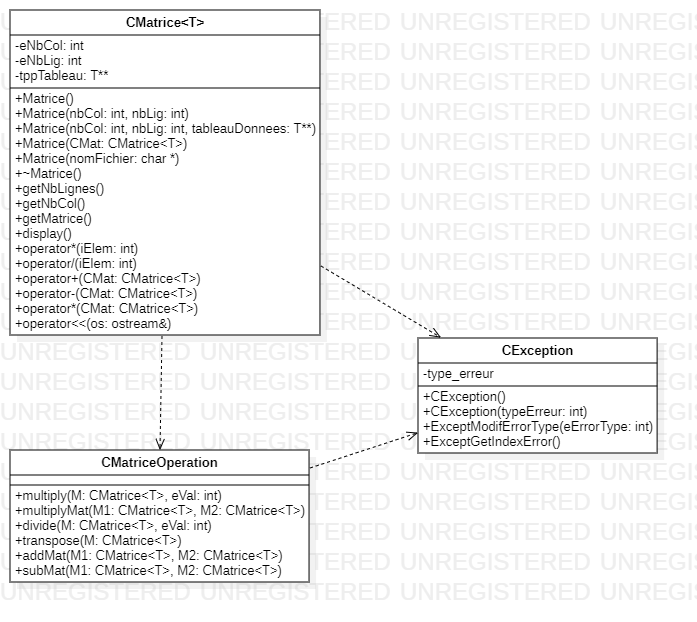
Enfin, nous conclurons sur ce que ce projet nous a apporté par rapport à nos connaissances personnelles et celles vu tout au long de ce semestre à Polytech.

# I – Conception

Pour la conception de la structure du projet nous avons décidé de séparer les opérations élémentaires de la matrice tels que les accesseurs, constructeurs et surcharges de toutes les autres opérations. Pour cela nous avons créé une interface CMatriceOperation qui contient toutes les méthodes sur le traitement des matrices. Cette séparation permet une bien meilleure lisibilité du code que l’utilisation d’une seule classe CMatrice contenant toutes ces opérations.

Nous avons aussi ajouté une classe CException qui nous permet de lever nos propres exceptions concernant le traitement des matrices.

Vous pouvez retrouver ci-dessous le diagramme de classe du projet :



# II – Développement

Pour le développement de ce projet, il était important de choisir des outils adéquats pour sa réalisation. Tout au long de l’année on nous a appris à compiler manuellement ou en utilisant le site Repl.it donc sans utiliser d’IDE, or pour ce projet, l’utilisation de Visual Studio 2017 nous était imposé. De plus, Visual Studio 2017 proposant des outils Git, il aurait été dommage de ne pas les utiliser. Ainsi tout notre projet est disponible sur Git Hub. L’utilisation du git fut pratique car nous a permis de travailler sur des fichiers différents sans se soucier de la synchronisation du projet.

Pour ce qui est du développement même, nous nous sommes beaucoup inspirés de l’exemple vu en cours du CArbreBinaire pour l’organisation. En effet, presque classe possède son .h, nous avons également profité des cours de Génie Logiciel et d’Algorithme Orientée Objet pour respecter au mieux les normes de nommage ainsi que les bonnes manières pour développer. Nous avons donc essayé au mieux d’appliquer ces méthodes en tentant d’anticiper pour chaque fonction les potentiels problèmes qui pourraient être gérés en pré ou postcondition.

La gestion de ces problèmes s’est faite via la classe CException qui permet de lever des exceptions personnalisées. Ainsi à travers notre analyse en amont nous avons pu anticiper la plupart des erreurs que l’utilisation de la librairie sur des matrices de même type peut engendrer. Cette classe possède 2 attributs : l’indice d’erreur et le tableau des messages d’erreurs. La classe CMatrice possède de nombreuses constantes correspondant à chaque erreur possible (*Exemple : CMatriceEmptyFile vaut 1 et correspond à l’erreur d’un fichier vide*). Chaque indice d’erreur est classé, par exemple, les constantes de 1 à 10 correspondent à la création d’une matrice via un fichier, à partir de 51, elles correspondent à des erreurs liées aux calculs. Donc, il y a une section try-catch dans le main qui dès qu’il y une CException qui est levée dans CMatrice elle arrive dans le main qui s’occupe de chercher via l’indice d’erreur le message associé dans le tableau de message d’erreurs.

# Conclusion :

Pour conclure, ce projet nous a vraiment poussé à utiliser toutes les connaissances vues en cours et même tout au long de l’année ce qui malgré la difficulté nous a permis de développer facilement ou du moins sans rencontrer de gros problème.