Rapport - Projet Graph

COUCHOUD Thomas COLEAU Victor

4 mai 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 graphs. Celle-ci devra nous permettre de stocker en mémoire des graphs ainsi que leurs composants, à savoir des sommets et des arcs. De plus, il devra être possible d'effectuer des actions basiques telles qu'ajouter ou supprimer des éléments.

Enfin, il devra être possible de générer des graphs en important leur structure depuis des fichiers texte externes formatés comme indiqué dans le sujet.

Chapitre 2

Architecture

La classe centrale de notre projet est CGraph. Celle-ci représente un graph. Y sont stockées les informations suivantes :

- Le nombre de sommets contenus dans le graph.
- L'indice du plus grand sommet du graph (choix expliqué par la suite).
- Un tableau 1D contenant des pointeurs sur les objets sommets.

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

- L'ajout d'un sommet au graph.
- La suppression d'un sommet du graph.
- La vérification de l'existance d'un sommet dans le graph.
- L'ajout d'un arc (orienté) entre deux sommets existants.
- La suppression d'un arc entre deux sommets.
- La modification d'un arc par redirection de son sommet d'arrivé.
- La vérification de l'existance d'un arc dans le graph.
- L'ajout d'un lien (arc dans les deux sens) entre deux sommets.
- L'inversion du graph (invertion de tous les arcs).
- Le nettoyage du graph.
- L'affichage du graph.
- L'opérateur d'addition ajoutant un sommet.
- L'opérateur de soustraction supprimant un sommet.
- L'opérateur d'affectation duplicant un graph dans un autre.

Deux autres classes se distinguent par leur importance : CVertex (un sommet) et CArc (un arc).

La classe CVertex contient les informations suivantes.

- L'indice du sommet.
- Le nombre d'arcs entrants.
- Un tableau 1D de pointeurs sur les arcs entrants.
- Le nombre d'arcs sortants.
- Un tableau 1D de pointeurs sur les arcs sortants.

Cette classe contient des méthodes servants principalement à modifier les arcs contenus.

- L'accesseur à l'indice du sommet.
- L'accesseur au nombre d'arcs entrants.
- L'accesseur au nombre d'arcs sortants.
- L'ajout (par la création) d'un arc entrant.
- La suppression d'un arc entrant.
- La modification d'un arc entrant.
- L'ajout (par la création) d'un arc sortant.

— La suppression d'un arc sortant.
— La modification d'un arc sortant.
— La vérification de l'existance d'un arc entrant.
— La vérification de l'existance d'un arc sortant.
— L'affichage des arcs entrants.
— L'affichage des arcs sortants.
— L'invertion des arcs.
— L'opérateur d'affectation duplicant le contenu d'un sommet dans un autre.

La classe CArc ne contient que peu d'informations.
— L'indice du sommet pointé par l'arc.

Elle contient les méthodes suivantes :
— L'accesseur à l'indice de l'arc.
— Le mutateur de l'indice de l'arc.
— L'opérateur d'affectation duplicant un arc.

Afin de pouvoir lire des graphs externes au programme, nous avons implémenté une classe statique CGraphParser. Cette dernière propose différentes fonction permettant la lecture d'un fichier. Nous avons par exemple la possibilité de lire une ligne, la découper selon un caractère etc. Grâce à celle-ci, le constructeur prenant en paramètre un nom de fichier va pouvoir lire de manière plus aisée le ficher graph.

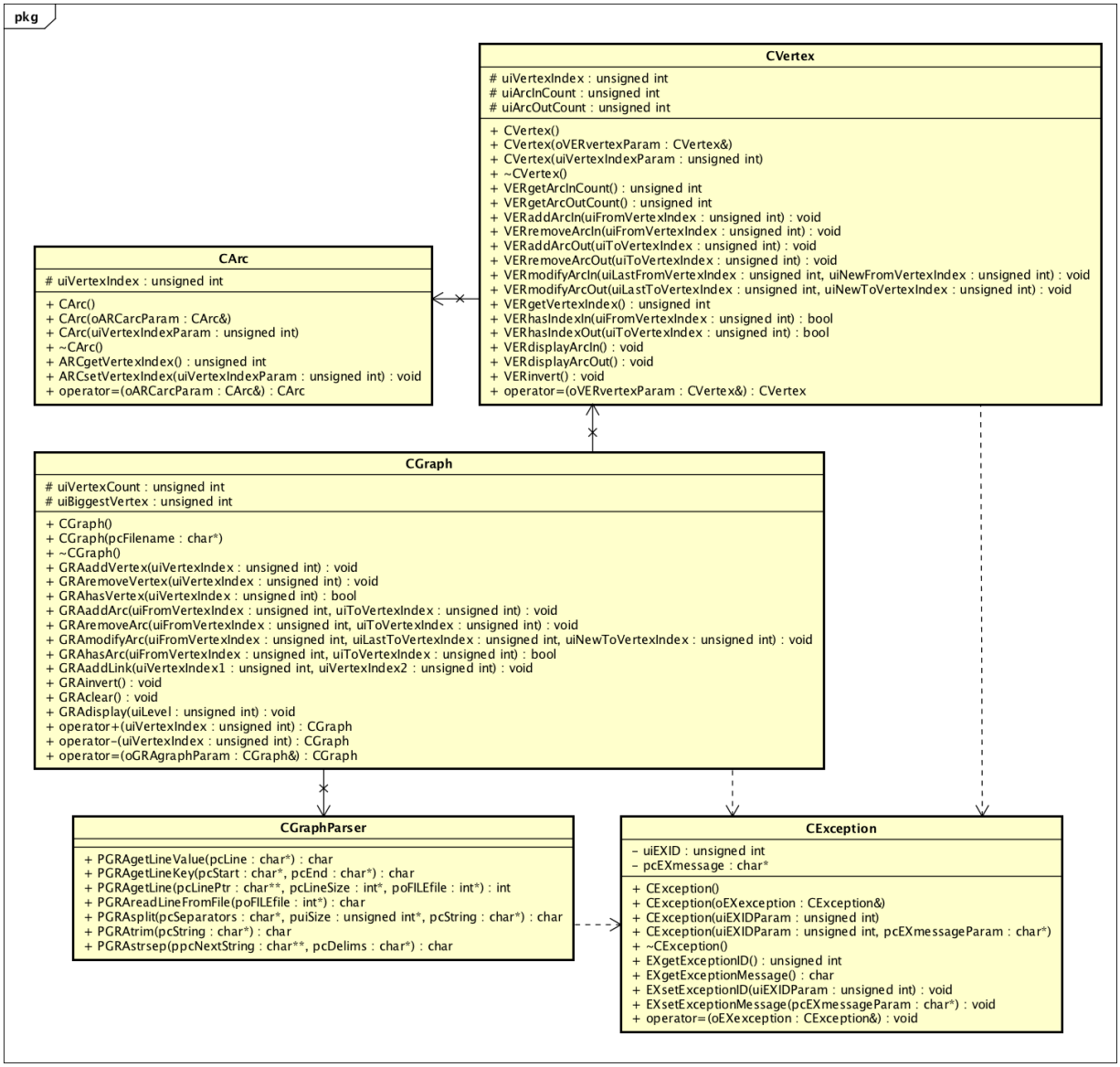

FIGURE 2.1 – Diagramme de classes

Chapitre 3

Choix de codage

Avant tout chose, nous avons prit la décision de créer un fichier nommé "utils.h". Celui-ci contient des macros permettant de compiler sous Visual Studio et un système Unix (utilisation des méthodes propres à chaque compilateur). De plus, il contient un certain nombre de macros utilitaires, notamment MMALLOC et RREALLOC allouant l'espace mémoire demandé tout en vérifiant le bon déroulement.

Un point important est l'absence d'accesseurs des objets CVertex et CArc depuis un graph. De cette façon, l'utilisateur ne peut pas modifier un sommet ou un arc appartenant à un graph en appelant les méthodes de CVertex ou de CArc directement sur l'objet en question. Il est obligé de passer par les méthodes de CGraph qui appeleront elles-mêmes les sous-méthodes nécessaires.
Cela évite qu'un sommet (ou un arc) soit ajouté, modifié ou supprimé sans que l'objet graph n'en soit informé ce qui évite un état incohérent du graph..

Un deuxième choix important est le stockage des sommets dans le graph. Plutôt que d'ajouter à la suite tous les sommets dans un tableau (stockage par ordre d'arrivé), nous posons chaque sommet à la position du tableau égale à l'indice du sommet moins 1. Ainsi le sommet 1 est à la position 0, le 10 à la position 9, etc..

Concernant le parser, nous avons fait le choix de séparer les méthodes "utilitaires" dans une classe CGraphParser. En effet de cette manière cette classe pourra être réutilisée par la suite pour lire des fichiers similaire. La partie propre a l'interprétation d'un fichier graph se fait dans le constructeur CGraph.

Chapitre 4

Tests effectués

Afin de pouvoir valider le fonctionnement de notre code, nous avons réalisé de nombreux tests. Certains ont été réalisés manuellement 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 classe 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". De plus chaque test n'est effectué qu'une seule fois ce qui peut mettre en doute l'efficacité de ceux-ci.

Commençons par CExceptionUnit :

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

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

Intéressons-nous maintenant à CArc :

- Test de la création d'un arc sans paramètres.
- Test de la création d'un arc avec une destination et vérification de l'accesseur associé.
- Test de la modification de la destination d'un arc.

Passons ensuite aux tests contenus dans CVertex :

- Test de la création d'un sommet sans paramètres.
- Test de la création d'un sommet avec un ID et vérification de l'accesseur associé.
- Test de l'ajout d'un arc entrant.
- Test de la fonction indiquant si un arc entrant existe.
- Test de l'ajout d'un arc entrant dupliqué.
- Test de la suppression d'un arc entrant.
- Test de la suppression d'un arc entrant n'existant pas.
- Test de la modification d'un arc entrant.
- Test de la modification d'un arc entrant n'existant pas.
- Test de l'ajout d'un arc sortant.
- Test de la fonction indiquant si un arc sortant existe.
- Test de l'ajout d'un arc sortant dupliqué.
- Test de la suppression d'un arc sortant.
- Test de la suppression d'un arc sortant n'existant pas.
- Test de la modification d'un arc sortant.
- Test de la modification d'un arc sortant n'existant pas.
- Test de l'opérateur =.
- Test du constructeur de recopie.

Etudions dès à présent CGraph :

- Test de l'ajout d'un sommet.
- Test de la fonction indiquant si un sommet existe.
- Test de la suppression d'un sommet.
- Test de la suppression d'un sommet n'existant pas.
- Test de l'opérateur +.
- Test de l'ajout d'un arc.
- Test de la suppression d'un arc.
- Test de l'ajout d'un arc avec une destination inexistante.
- Test de la suppression d'un arc avec une destination inexistante.
- Test de l'ajout d'un arc avec un départ inexistant.
- Test de la suppression d'un arc avec un départ inexistant.
- Test de la modification d'un arc.
- Test de la modification d'un arc avec des sommets inexistants.
- Test de l'ajout d'un lien.
- Test de la suppression d'un sommet contenant des arcs entrants/sortants.
- Test de l'opérateur =.
- Test du constructeur prenant un fichier en paramètre.

Enfin les derniers tests sont effectués dans CGraphParser :

- Test de la fonction pour obtenir la partie valeur.
- Test de la fonction pour obtenir la partie clef.
- Test de la fonction permettant de couper un string selon des délimiteurs.
- Test de la fonction permettant de renvoyer un tableau après une découpe par délimiteur.
- Test de la fonction permettant de raccourcir un string en retirant les caractères non utiles en début et fin de string.

Chapitre 5

Conseils d'utilisation

Le programme a été conçu et compilé de sorte que l'exécutable puisse prendre en arguments des fichiers sources de matrice formatés comme défini dans le sujet.

Suite à cela, il va exécuter les instructions suivantes :

- Demander à l'utilisateur un coefficient.
- Calculer et afficher les résultats de la multiplication de chaque matrice par le coefficient précédent.
- Calculer et afficher les résultats de la division de chaque matrice par le coefficient précédent.
- Calculer et afficher le résultat de la somme de toutes les matrices.
- Calculer et afficher le résultat de la somme alternée de toutes les matrices.
- Calculer et afficher le résultat du produit de toutes les matrices.

Si une exception est levée pour n'importe lequel de ces points, le message d'erreur est affiché et le programme s'arrête instantanément.

Pour toute autre utilisation, la classe Main devra être modifier par l'utilisateur, deux choix sont possibles.

Une matrice peut être lue d'un fichier grâce aux fonctions statiques "PMTXreadMatrix" et "PMTXreadSquareMatrix" prenant chacune en paramètre le chemin relatif d'un fichier source de matrice. La première renverra un objet de la classe CMatrix quelles que soient les réelles dimensions de la matrice lue alors que la seconde créera une matrice carrée (ou lèvera une exception si la matrice lue n'est pas carrée). Les deux lèveront des exceptions si un problème survient lors de la lecture.

D'une seconde manière, une matrice peut être créée directement depuis son constructeur mais celle-ci sera remplie de 0. Nous pouvons par la suite utiliser le setter adéquat pour modifier chaque valeur de la matrice a sa guise. A savoir qu'un constructeur identité est disponible pour les matrices carrées.

Pour toute information supplémentaire sur les méthodes, se référer aux cartouches d'entête présents dans les fichiers .h.