Projet C++: Les graphes

Ecole Polytechnique de l'Université de Tours

Avril 2019

Préambule

Lors de ce projet tuteuré nous allons travailler sur les *graphes*.

L'objectif est de réaliser une librairie de classes et fonctions permettant de manipuler des graphes. La réalisation de ce projet va nécessiter l'utilisation de la méthodologie UML pour modéliser le logiciel à développer, ainsi que le langage C++ pour la mise en œuvre. Vous devrez à l'issue de ce projet rendre un rapport présentant le travail que vous avez réalisé. Ce projet est à réaliser dans un volume de 12h de TPs, volume réparti comme suit (ce n'est qu'une suggestion) :

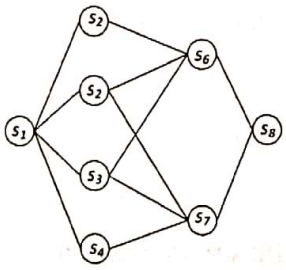
1. Modélisation UML : 3h pour la réalisation du diagramme de classes (seul diagramme qui vous est obligatoirement demandé dans ce projet). Ce temps inclus de la rédaction de votre rapport.
2. Développement C++ : 9h. Ce temps inclus de la rédaction de votre rapport.

Vous travaillez en binômes.

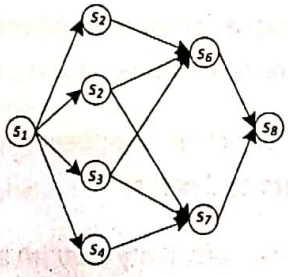
A l'issue de ces 12h vous devrez rendre votre rapport, les sources de votre projet ainsi qu'une version exécutable sous Windows et qui a été compilée dans le profil Release sous Visual Studio.

Spécifications fonctionnelles

Passons maintenant à la spécification du travail à réaliser. Tout d'abord qu'est-ce qu'un graphe ? Un graphe $G=[S,A]$ est défini par un ensemble de sommets S et un ensemble d'arrêtes A : une arrête a_{ij} relie deux sommets s_i et s_j du graphe. On dit dans ce cas que le graphe est non orienté. Si les arrêtes sont des arcs alors on parle d'un graphe orienté : l'arc a_{ij} relie le sommet s_i vers le sommet s_j.



- Un graphe non orienté -



- Un graphe orienté -

Un graphe *G* peut être très pratique pour représenter quelque chose. Par exemple, supposons que vous ayez une carte d'un quartier (comme dans la figure ci-dessous) : vous pouvez alors très simplement créer un graphe représentant cette carte ! Un carrefour est représenté par un sommet dans le graphe et une rue par une arrête.

- Exemple de création d'un graphe « routier » -

Il est possible de rajouter des informations, soit sur les arrêtes/arcs soit sur les sommets du graphe en fonction de ce que l'on veut faire. Par exemple, on peut rajouter sur les arrêtes du graphe exemple précédent la distance séparant deux (carrefours) sommets du graphe.

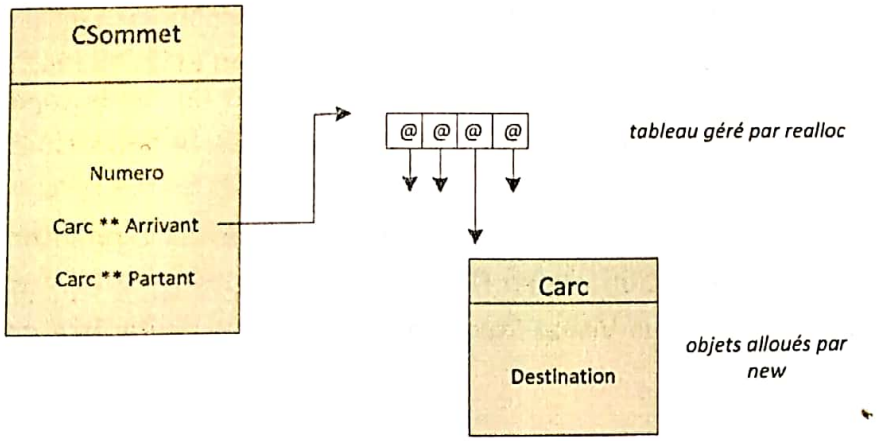
Dans ce projet tuteuré, nous allons mettre en place toutes les classes permettant de créer, modifier et accéder à un graphe orientés. Pour cela nous allons créer 4 classes :

1. La classe Cexception : qui permettra de lever des exceptions (vous pouvez reprendre celle que vous avez déjà développé dans le projet précédent),
2. La classe Carc : qui représentera des arcs dans le graphe,
3. La classe Csommet : qui représentera des sommets dans le graphe,
4. La classe Cgraphe : qui représentera un graphe.

Dans votre projet on supposera qu'aucune information particulière n'est stockée au niveau des sommets et des arcs.

Dans la classe Carc, il n'y a qu'un seul attribut : le numéro du sommet destination (nommons le *Destination*).

Dans la classe Csommet, il y a trois attributs : le numéro du sommet, la liste des arcs partants du sommet et la liste des arcs arrivants au sommet. Ces deux listes d'objets devront être gérées dynamiquement. Je vous propose l'implémentation suivante :



Ainsi, pour un arc réel du graphe il faudra donc créer **2** objets de la classe Carc : l'un sera inséré dans la liste des Partant du sommet de départ et l'autre sera inséré dans la liste des Arrivant du sommet d'arrivé. Prenons l'exemple du graphe routier ci-dessus et supposons qu'il soit orienté ; considérons l'arc (s_1 ;s_2). Pour créer cet arc en mémoire il faudra donc créer deux objets : *ARCa1* de *Destination* 2 et *ARCa2* de *Destination* 1. L'objet *ARCa1* sera inséré dans la liste des arcs *Partant* de l'objet correspondant au sommet s_1. De la même façon, *ARCa2* sera inséré dans la liste des arcs *Arrivant* de l'objet correspondant au sommet s_2. L'inconvénient d'une telle méthode est que la place mémoire occupée par un graphe est grande puisque l'on double tous les arcs en mémoire. L'avantage est que les parcours dans le graphe vont être accélérés. On y gagnera en vitesse de traitement au détriment de la mémoire.

La classe graphe va tout simplement contenir un tableau de Csommet qui seront les sommets du graphe.

La librairie que vous devez fournir doit permettre :

1. De créer des objets graphe,
2. De faire des opérations « élémentaires » sur un graphe,
    a. Ajouter/modifier/supprimer un sommet au graphe,
    b. Ajouter/modifier/supprimer un arc du graphe,
    c. Afficher à l'écran un graphe.

Par ailleurs, vous êtes libre de proposer de nouvelles fonctionnalités tant qu'elles relèvent du confort d'utilisation de votre code.

Contraintes techniques

Pour la réalisation de votre projet vous devrez respecter les quelques contraintes énoncées ci-dessous :

1. Les variables, types, classes, fonction et méthodes devront respecter les conventions de nommage,
2. Vous devez mettre en œuvre les bonnes pratiques du Génie Logiciel telles qu'elles ont pu être présentées lors des cours d'Algorithmes Orientés Objets et de Langages Orientés Objets,
3. Vous devrez intégrer la gestion des exceptions dans votre programme : utilisez pour cela la classe Cexception vue en TPs,
4. Vous travaillerez sous Visual Studio et votre projet devra être ***application console win32***,

En plus des classes nécessaires pour réaliser ce qui a été demandé ci-dessus à propos des graphes, vous devrez développer une fonction qui permet de lire le contenu d'un graphe dans un fichier texte et créer un objet graphe en mémoire pour stocker les données lues dans le fichier. Le format du fichier texte (imposé) est donné ci-dessous et illustré dans le cadre d'un exemple : il repose sur l'utilisation de quelques balises.

NBSommets=<Nombre_de_Sommets_du_graphe>
NBArcs=< Nombre_d_arcs_du_graphe >
Sommets=[
Numero=<Numéro_sommet >
Numero=<Numéro_sommet>
...
Numero=<Numéro_sommet>
]
Arcs=[
Debut=<Sommet_départ_arc>, Fin=<Sommet_arrivée_arc>
Debut=<Sommet_départ_arc>, Fin=<Sommet_arrivée_arc>
...
Debut=<Sommet_départ_arc>, Fin=<Sommet_arrivée_arc>
]

Un exemple :

NBSommets=3
NBArcs=3
Sommets=[
Numero=1
Numero=2
Numero=3
]
Arcs=[

Debut=1, Fin=2
Debut=2, Fin=3
Debut=3, Fin=1
]

La fonction principale de votre projet devra alors faire l'algorithme suivant :

- Pour un nom de fichier passé en paramètre, lire le fichier et créer le graphe associé,
- Afficher le graphe à l'écran,
- Inverser tous les arcs du graphe pour obtenir un **nouveau** graphe,
- Afficher ce nouveau graphe à l'écran.

Eléments à fournir à l'issu du projet

A l'issu du projet vous devrez fournir :

- Un rapport présentant les modèles UML que vous avez pu construire, la façon dont votre code a été développé et une présentation des choix que vous avez pu être amené à réaliser, …. Plus tout ce qui est nécessaire pour la compréhension de votre projet.
- Les sources complètes (avec le projet sous Visual Studio),
- Un exécutable fonctionnant sous Windows et compilé en mode Release.