GAYET Lilian

FILLEUL Lucas

**Rapport Projet C++**

**Partie 3 - Les graphes**

# **Sommaire**

[**Sommaire**](#_61xa3i75sld8) **2**

[**Modélisation**](#_ghq3b2cquqtr) **3**

[Diagramme de classes](#_ms4but72800x) 3

[**Implémentation de la librairie**](#_fn1szhssinxd) **4**

[CGraphe](#_w10k23w8bgi8) 4

[CSommet](#_p979f8ut8rxz) 4

[CArc](#_lzr48wcxh2do) 4

[CException](#_e97ezdx0jngk) 5

[CParser](#_13vlmbwfs58x) 5

[COperation](#_o86fwfrom1gc) 5

[**Utilisation de la librairie**](#_9f38t1b0lzth) **6**

[**Annexes**](#_k37myeh7uas) **7**

# 

# **Modélisation**

## **Diagramme de classes**

Dans un premier temps, suite à la lecture du sujet, nous avons listé les classes qui nous semblaient importantes à la réalisation de ce projet ainsi que les attributs et méthodes de chacune. Après discussion et modification, nous avons finalisé le diagramme de classe et avons obtenus ceci : (En annexe dans un format plus grand et dans le dossier de rendus).

### 

### 

# 

# 

# **Implémentation de la librairie**

## **CGrapheOriente**

Afin d’implémenter la librairie, nous avons décidé que la création et la gestion d’un graphe orienté ne se ferait que grâce aux méthodes de **CGrapheOriente** . L’utilisateur ne dispose que des constructeurs de **CGrapheOriente** ainsi que de deux méthodes : GRAAjouterSommetArcs et GRAAfficherGraphe. Toute autre méthode est déconseillée d’utilisation si l’utilisateur n’a pas une connaissance complète de la librairie. Les méthodes misent à disposition possèdent une gestion d’erreur plus poussée et simplifient grandement l’utilisation de la librairie.

## **CGraphe**

La classe **CGraphe** permet de créer un graphe sans avoir la faculté d’être orienté. Cette classe hérite de la classe **CGrapheOriente** et dispose de toutes les méthodes de celle-ci dont certaines ont été surchargées comme par exemple le constructeur de confort prenant en paramètre un chemin vers un fichier et les méthodes d’ajouts d’arc (d’arrête).

## **CSommet**

La classe **CSommet** consiste à implémenter le principe de sommet dans un graphe. Cette classe est composée, comme attributs, de l’Id du sommet, d’une liste d’arcs entrants et d’une liste d’arcs partants. Mais aussi de nombre d’arcs arrivants et partants dans le but de pouvoir parcourir les listes d’arcs. L’utilisateur n’a pas vocation à instancier un objet **CSommet**, c’est pour cela qu’il n’existe pas de constructeur par défaut. C’est le graphe qui s’en chargera.

Il existe cependant un constructeur de confort comprenant en paramètres :

* l’Id du sommet
* une liste d’arcs arrivants
* une liste d’arcs partants

Ainsi qu’un constructeur de recopie.

Nous avons mis 3 méthodes de gestions des listes (reallocation, suppression) d’arcs en privé car ces fonctions ne sont utiles qu’en interne mais n’ont pas lieu d’être disponibles pour l’utilisateur.

## **CArc**

La classe **CArc** consiste à créer un objet qui permet de relier deux sommets. Son seul attribut porte la valeur du sommet auquel l’arc est associé. L’association de cet attribut n’a pas la même signification en fonction de où il se situe dans le sommet. Dans la liste des arcs arrivant du sommet, cet attribut porte l’id du sommet qui arrive vers le sommet en cours. Dans la liste des arcs partants du sommet, cet attribut porte l’id du sommet vers lequel le sommet en cours part ( se dirige ).

## **CException**

La classe CException permettra de lever et gérer des exceptions afin de prévenir des cas d’utilisations prévisibles mais anormaux. Toutes les erreurs portant un code d’erreur compris entre 100 et 199 signifie que l’erreur est issue de l'utilisation de la classe CArc, celles entre 200 et 299 de l'utilisation de la classe CSommet, entre 300 et 399 CGraphe, entre 400 et 499 CParseur et enfin celles entre 500 et 599 COperation.

Elle comporte les méthodes suivantes :

* EXCLireCodeErreur
* EXCLireMessageErreur
* EXCModifierErreur

EXCLireCodeErreur permet, comme son nom l’indique, de renvoyer le code d’erreur levé.

EXCLireMessageErreur, quand à elle, “traduit” le code d’erreur en un message d’erreur afin d’expliquer à l’utilisateur la raison de la levée d’exception.

EXCModifierErreur est un accesseur d’écriture du code d’erreur mais il n’est pas utilisé au sein de notre code. Cependant, il pourrait s’avérer utile lors d’une nouvelle implémentation du code.

## **CParser**

La classe **CParseur** permet de créer un objet **CParseur** qui portera en attributs un chemin vers un fichier, un flux vers le fichier et deux séparateurs voulus par l’utilisateur. Le premier séparateur, sépare le libellé de la valeur dans le fichier. Le second est le séparateurs entre deux couples (libellé,valeur). L’unique méthode de cette classe est PARLireLibVal. Cette méthode lie une ligne du fichier si la précédente a été totalement lue, sépare les valeurs et écrit aux adresses passées en paramètres les valeurs trouvées. Cette classe n’a aucune connaissance du format de fichier qu’elle lit et de l’objet qui l’appel. Cela permet à cette classe d’être générique et de ne subir aucune modification dans le but d’une réutilisation dans un autre projet.

## 

## **COperation**

La classe COperation permet d’effectuer des opérations plus complexe sur des graphes. Cette classe contient actuellement uniquement la méthode d’inversion d’un graphe. Elle pourra évoluer en y ajoutant d’autres opérations sur les graphes. L'opération d’inversion, comme demandé, créer, à partir d’un graphe passé en paramètre, un nouveau graphe où tous les arcs sont inversés.

# **Utilisation de la librairie**

Afin d’utiliser la librairie, vous avez accès aux méthodes suivantes :

* Constructeur par défaut de CGraphe qui permet d’instancier un graphe vide :

**appel** : *CGraphe graphe = CGraphe();*

*CGraphe \* pgraphe = new CGraphe();*

* Constructeur de recopie de CGraphe qui permet d’instancier un graphe à partir d’un autre déjà créer :

**appel** : *CGraphe graphe = CGraphe(autregraphe);*

*CGraphe \* pgraphe = new CGraphe(autregraphe);*

* Constructeur de confort de CGraphe qui permet d’instancier un graphe à partir d’un fichier :

**appel** : *CGraphe graphe = CGraphe(cheminverslefichier);*

*CGraphe \* pgraphe = new CGraphe(cheminverslefichier);*

* Méthode GRAAjouterSommetArcs qui permet d’ajouter un sommet avec ou sans arcs ou d’ajouter des arcs a un sommet :

**appel** : *graphe.GRAAjouterSommetArcs(param1, param2, param3);*

*pgraphe->GRAAjouterSommetArcs(param1, param2, param3);*

*param1 : Un entier non signé qui portera ou porte l’id du Sommet*

*param2 : Une liste d’entiers non signés correspondants aux identifiants de sommet vers lesquels le sommet portant l’id param1 pointera (arc entre param1 et les ids de param2). La liste doit porter comme dernière valeur de liste la valeur NULL.*

*param3 : Une liste d’entiers non signés correspondants aux identifiants de sommet vers lesquels ils pointent vers le sommet portant l’id param1 (arc entre les ids de param2 et param1). La liste doit porter comme dernière valeur de liste la valeur NULL.*

* Méthode GRAAfficherGraphe qui permet d’afficher dans la console le graphe :

**appel** : *graphe.GRAAfficherGraphe();*

*pgraphe->GRAAfficherGraphe();*

Exemple d’utilisation :

*1. CGraphe graphe = CGraphe();*

*2. graphe.GRAAjouterSommetArcs(1);*

*3. graphe.GRAAjouterSommetArcs(2, {1, NULL}, NULL);*

*4. graphe.GRAAjouterSommetArcs(1, NULL, {2, NULL});*

*5. graphe.GRAAfficherGraphe();*

# **Annexes**