

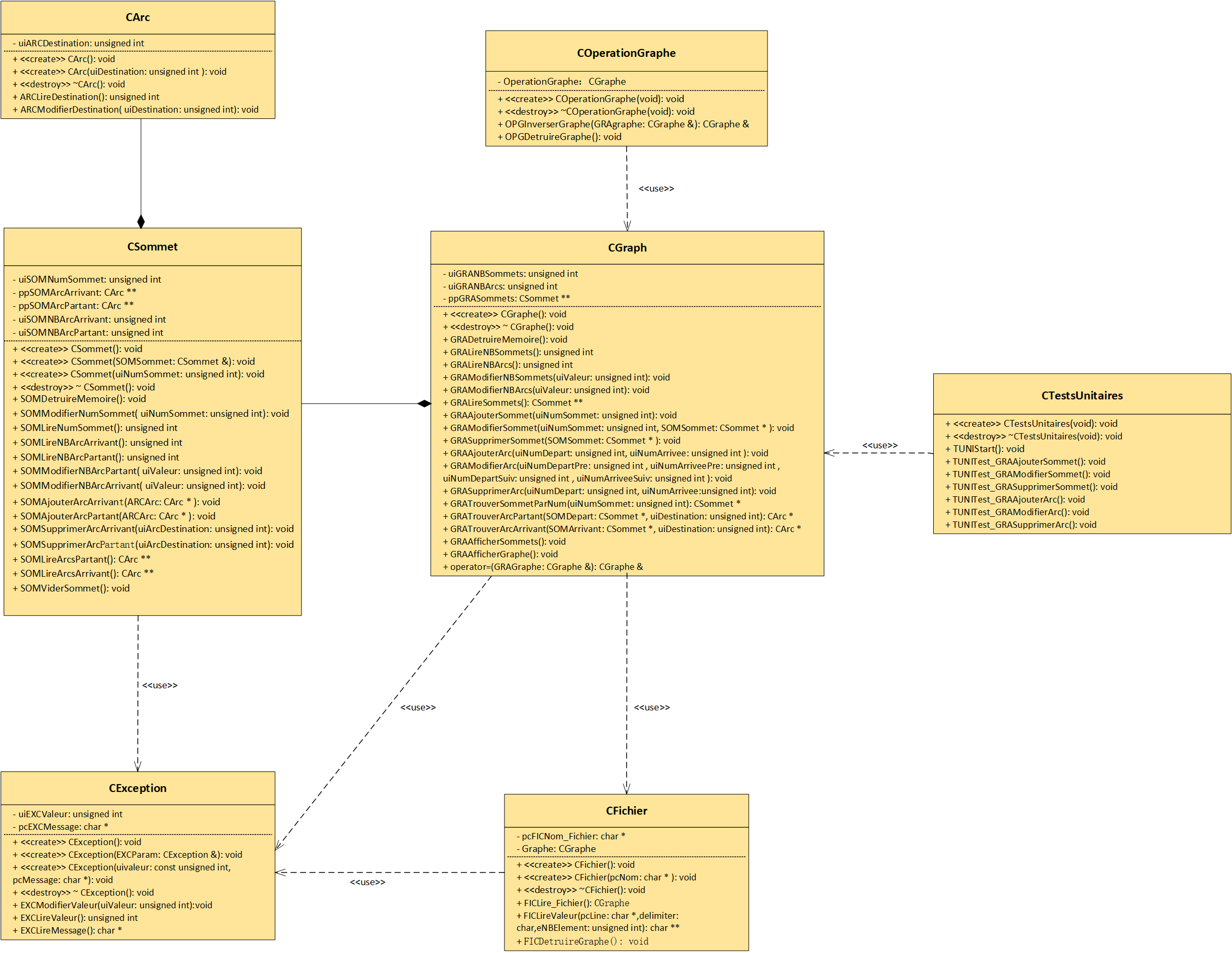
Le Projet Graphe – Rapport

*Élèves : FENG Jiaming, MENG Qingling – DI3 Mundus*

# Pour le 27 mai 2018

**Important:** Dans notre projet, nous avons utilisé **VS2012** pour programmer.

**1 Le diagram de la classe de UML**

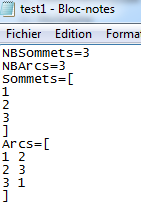
****

**2 Un manuel d'utilisation de librairie**

**2.1 Classe CFichier**

CFichier est une classe qui permet de lire un fichier texte et créer le graphe associé.

Pour utiliser le CFichier, il suffit de créer un objet CFichier en passant le nom du fichier en paramètre et Il ne lire que le fichier dans un format spécifié.

Il ne peut pas distinguer majuscule et minuscule.Et la syntaxe doit être correcte.

**2.2 Classe CGraphe**

Cette classe représente un graphe.Un graphe peut être déclaré seulement par constructeur par défault.L’utilisateur peut ajouter des sommets par le fonction void GRAAjouterSommet(unsigned int uiNumSommet);

On peut ajouter un sommet dont le numéro est “uiSOMNumSommet”.

Et puis l’utilisateur peut ajouter les arcs:

void GRAAjouterArc(unsigned int uiNumDepart, unsigned int uiNumArrivee);

On peut ajouter un arc entre les deux sommets dont le numéro du sommet est “uiNumDepart” et “uiNumArrivee”. Et c’est un arc uiNumDepart --->uiNumArrivee.

Pour modifier le numéro d’un sommet, on peut utiliser la fonction suivant:

void GRAModifierSommet(unsigned int uiNumSommet, CSommet \* SOMSommet);

L’utilisateur peut changer le numéro d’un objet CSommet à uiNumSommet.

Et le fonction se charge de modifier les arcs associées.

Si l’utilisateur voudrait supprimer un sommet, appelez la fonction:

void GRASupprimerSommet(CSommet \* SOMSommet);

Passez un objet CSommet qui sera supprimé en paramètre, et puis tous les arcs associé et le sommet sont supprimé.Il libére également la mémoire.

La méthode pour modifier un arc fonctionne commes ci-dessous:

void GRAModifierArc(unsigned int uiNumDepartPre, unsigned int uiNumArriveePre, unsigned int uiNumDepartSuiv, unsigned int uiNumArriveeSuiv);

1111111

2341111

2341112

2341113

2341114

**uiNumDepartPre=2**

**uiNumArriveePre=4**

1111111

2341111

2341112

2341114

2341113

**uiNumDepartSuiv=1**

**uiNumArriveeSuiv=3**

GRAModifierArc()

Pour supprimer un arc, utilisez le fonction suivant:

void GRASupprimerArc(unsigned int uiNumDepart, unsigned int uiNumArrivee);

passez le numéro du sommet où l’arc part comme le premier argument et le numéro du sommet où l’arc arrive comme le deuxième argument

La librairie posséde la méthode pour afficher tous les informations du graphe

L’utilisateur peut aussi utiliser les méthodes pour trouver un sommet ou un arc existant si besoin.

**2.3 Classe CException**

On définit 9 codes d’erreur pour indiquer les types d’erreur.

#define ERREUR\_OUVERTURE\_FICHIER 1

#define ERREUR\_ALLOCATION 2

#define ERREUR\_SYNTAXIQUE 3

#define ERREUR\_VIDE\_GRAPHE 4

#define ERREUR\_SOMMET\_DOUBLE 5

#define ERREUR\_SOMMET\_INEXISTANT 6

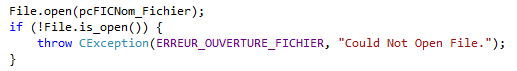
#define ERREUR\_ARC\_DOUBLE 7

#define ERREUR\_ARC\_INEXISTANT 8

#define ERREUR\_ARGUMENT 9

L’utilisateur peut principalement utiliser le constructeur suivant pour initialiser les exceptions.L’utilisateur peut aussi lire et modifier le message et le code d’erreur.

CException(const unsigned int uivaleur, char \* pcMessage);



**2.4 La fonction principale**

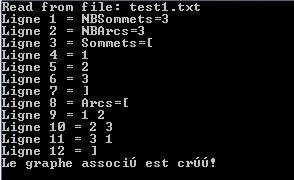
Pour UN nom de fichier passé en paramètre, on crée un objet de type CFichier et lire le fichier en utilisant le méthode "**FICLire\_Fichier()”**, et on crée le graphe associé et affiche le graphe.

Après, on crée un objet de “**COperationGraphe”** pour faire l’opération de inverser les arcs du graphe. S’il y a une erreur de lecture ou des opérations , on attraper une exception.

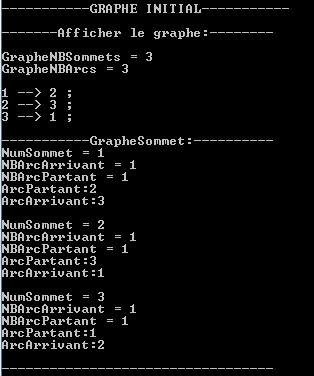
En fin, on fait les tests unitaires et puis détecte des fuite mémoire.

**2.4.1 La présentation**

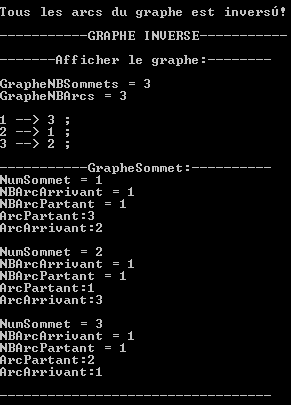
Lire le contenu d’un graphe dans un fichier texte.



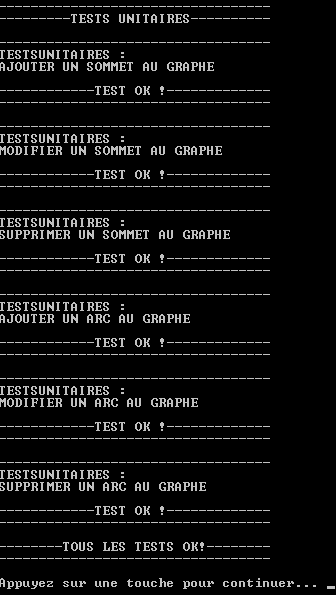
Afficher le graphe



Afficher le graphe inversé



Afficher les r**é**sultats de tests unitaires



**3 Choix de programmation**

**3.1 Vérification de la mémoire**

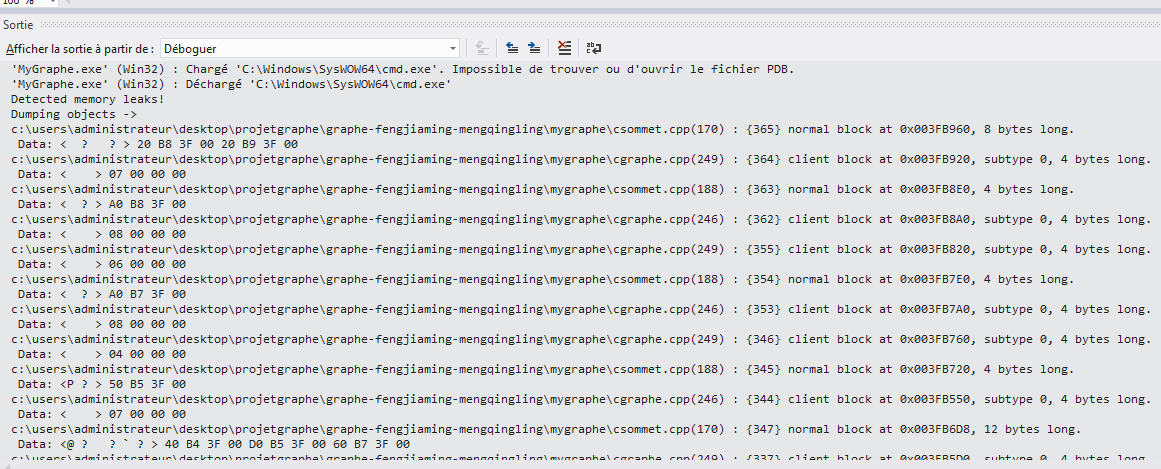
Nous avons vérifié l’état de la mémoire de notre programme grâce à la bibliothèque crtdbg.h fournie par VisualStudio, afin d’éviter les fuites mémoire. Pour utiliser cette bibliothèque, il faut mettre un fichier d'en - tête “DetectMemoryLeak.h”.

On appelle la fonction suivante, à la fin du main():

**\_CrtDumpMemoryLeaks()**

Qui affichera dans la fenêtre de debug un résumé des fuites mémoire.

Il peux aussi donner les numéros de lignes de code qui correspondent aux fuites de mémoire. Donc on peux résoudre les problèmes de mémoire.



Il y a beaucoup de problèmes de mémoire, pendant qu’on crée des sommets et des arcs donc on rédige le fonction **“GRADetruireMemoire()”** dans la classe CGraphe pour libérer les sommets et rédigele fonction **“SOMDetruireMemoire()”** dans la classe CSommet pour libérer les arcs dans la liste des arcs partant et arrivant.

Et on appele le fonction **“SOMDetruireMemoire()”** dans le fonction “**GRADetruireMemoire()”.**

void CGraphe::GRADetruireMemoire(){

...

ppGRASommets[uiBoucleI]->SOMDetruireMemoire();

...

}

Pour l’objet Graphe temporaire qu’on utilise dans la classe CFichier pour stocker le graphe associé d’un fichier. On rédige le fonction **“FICDetruireGraphe()”** qui appelle le fonction **“GRADetruireMemoire()”** pour detruire l’objet Graphe de CFichier. Et c’est le même procédé pour l’objet graphe dans la classe “**COperationGraphe”.**

**3.2 Classe CoperationsSurGraphe**

C’est précisément à cause de problèmes de mémoire qu’on va obtenir un nouveau graphe quand on inverse un graphe, donc on choisit de créer un nouveau classe **“COperationGraphe”** pour faire l’opération “InverserGraphe”, au lieu de rédiger le fonction dans la classe CGraphe.

**3.3 Passage d’arguments par référence**

Lors de la programmation, nous avons également rencontré des problèmes de mémoire quand on rédige le constructeur de recopie. En cas de extraire le graphe depuis un fichier, on utilise le passage d’arguments par valeur pour retourner le graphe lues dans le fichier, et le compilateur appelle **le constructeur de recopie** automatiquement qui cause une fuite de mémoire. Pour ça, on change de utilise le **passage d’arguments par référence**. La question est résolue.

**3.4 Realloc et Malloc**

Nous avons choisi utiliser des tableaux dynamiques qui gérés par malloc / realloc pour gérer la liste des arcs dans la classe CSommet et la liste des sommets dans la classe CGraphe. Et les pointeurs sur les objets sont gérés avec new / delete.

**4 La façon du projet**

**4.1Classe CException**



Figure 1 classe CException

Dans la classe **CException :**

Il y a 2 attributs : **unsigned int uiEXCValeur, char \* pcEXCMessage**

1) **unsigned int uiEXCValeur :** indiquer le numéro d’erreur.

2) **char \* pcEXCMessage** :indiquer le message d’erreur.

L’utilisateur peut construire un objet de la classe CException , lire et modifier le code d’erreur et le message aussi.

**4.2 Classe CFichier**



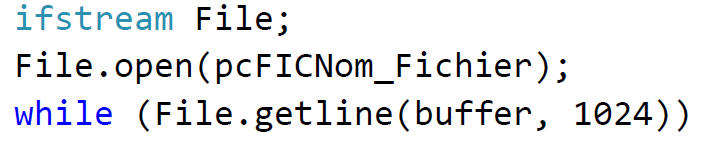
Figure 2 classe CFichier

Dans la classe **CFichier :**

Il y a deux attribut : **char \* pcFICNom\_Fichier** représentant le nom de fichier. Et **CGraphe Graphe** est pour stocker l’objet qui est lu à partir du fichier **pcFICNom\_Fichier** et faciliter la libération de la mémoire.

Cette méthode **CGraphe & FICLire\_Fichier()** lisant le fihcier et les stocker dans un objet de CGraphe. La définition est dans le fichier CFichier.cpp,.La façon est ci-desous :

La lecture du fichier est réalisée à l’aide de la librairie fstream et par utilisation de getline() et de <ifstream> :



On lit les lignes du fichier en utilisant “getline” par une boucle et on distingue chaque ligne par variable fileligne. Pour chaque ligne du fichier, on utilise le méthode suivant:

char \*\* FICLireValeur(char \* pcLine, char cDelimiter, unsigned int uiNBElement);

On sépare une ligne par “cDelimiter” et on stocke chaque élément dans un tableau de type char \*\* qui va retourner à le fonction FICLire\_Fichier().

Sur les deux premières lignes, on sépare une ligne par ‘=’.

Les lignes contient des éléments du graphe est divisé par ‘ ’.

En fin, on retourne la référence du graphe.

**4.3 Classe CArc**

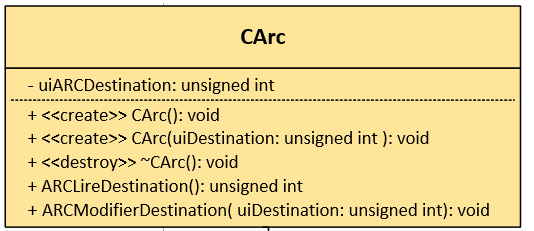


Figure 3 classe CArc

La classe CArc gèrant des informations d’un arc dans un graphe. Dans la classe CArc :

Il y a juste un attribut unsigned int uiARCDestination, qui représente le numéro du sommet sur un arc.

L’utilisateur peut construire un objet de la classe CArc , lire et modifier le numéro du sommet sur un arc aussi.

**4.4 Classe CSommet**



Figure 4 classe CSommet

La classe CSommet gèrant des informations d’un sommet dans un graphe. Dans la classe CSommet, il y a 5 attributs :

**unsigned int uiSOMNumSommet**: le numero du sommet

**CArc \*\* ppSOMArcArrivant**: un tableau de la classe CArc dont le point final est ce sommet

**CArc \*\* ppSOMArcPartant**: un tableau de la classe CArc dont le point depart est ce sommet

**unsigned int uiSOMNBArcArrivant**: le nombre des arc arrivants

**unsigned int uiSOMNBArcPartant** : le nombre des arc partants

Les listes de Carc sont des tableaux gérés dynamiquement par malloc() /realloc() / free(), contenant des pointeurs vers des Carc. Ces pointeurs sont gérés par les opérateurs new et delete du C++. Cela permet une plus grande flexibilité de notre classe quant à l’ajout et à la suppression d’arcs.

L’utilisateur peut obtenir le numéro du sommet, mais ne peut pas modifier le numéro du sommet. On permet l’utilisateur de obtenir et modifier le nombre des arc arrivants d’un sommet et le nombre des arc partants d’un sommet.

Hors cela l’utilisateur peut obtenir , modifier et supprimer un arc de la liste (arrivant ou partant).

**4.5 Classe CGraphe**



Figure 5 classe CGraphe

La classe CGraphe gèrant un graphe. Dans la classe CGraphe, il y a 3 attributs :

**unsigned int uiGRANBSommets**:le nombre de sommets de ce graph

**unsigned int uiGRANBArcs** : le nombre de arcs de ce graph

**CSommet \*\*ppGRASommets** : un tableau de sommets de ce graph.

L’utilisateur dispose de méthodes permettant d’ajouter, de rechercher et supprimer des sommets, ainsi que de méthodes permettant de rechercher, ajouter, modifier et supprimer des arcs. Lorsqu’un sommet est détruit, tous les arcs associés à ce sommet sont détruits.

On permet à afficher des sommets d’un graphe et afficher d’un graphe.

En raison de la nature des attributs de la classe Cgraphe, une surcharge de l’opérateur d’affectation prenant un CGraphe en paramètre est définie.

**4.6 Classe COperationGraphe**



Figure 6 classe CoperationGraphe

La classe COperationGraphe est pour réaliser inverser un graphe. Dans la classe COperationGraphe, il y a un unique attribut **CGraphe OperationGraphe.**

Dans cette classe, on peut inverser un graphe.

**4.7 Classe CTestsUnitaires**



Figure 7 classe CtestsUnitaires

La classe CTestsUnitaires est pour tester des opérations Ajouter, Modifier et Supprimer du sommet et du arc.