|  |  |  |  |
| --- | --- | --- | --- |
|  | [Choisir la date] |  | |
| |  |  | | --- | --- | |  |  | |  |  | |  |  | |  |  | | |  | |
| Projet Graphe  *Librairie de classe et de fonctions* | | | |
|  |  | | |  |  |  | | --- | --- | --- | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |
|  |  | | DUPAS Yoann |

Table des matières

[Présentation du sujet 2](#_Toc9165268)

[I. Modélisation UML 2](#_Toc9165269)

[A. CGraphe 3](#_Toc9165270)

[B. CSommet 4](#_Toc9165271)

[C. CArc 5](#_Toc9165272)

[D. CParser 5](#_Toc9165273)

[E. CException 6](#_Toc9165274)

[II. Evolution du projet 6](#_Toc9165275)

[III. Notice d’utilisation 6](#_Toc9165276)

Projet Graphe

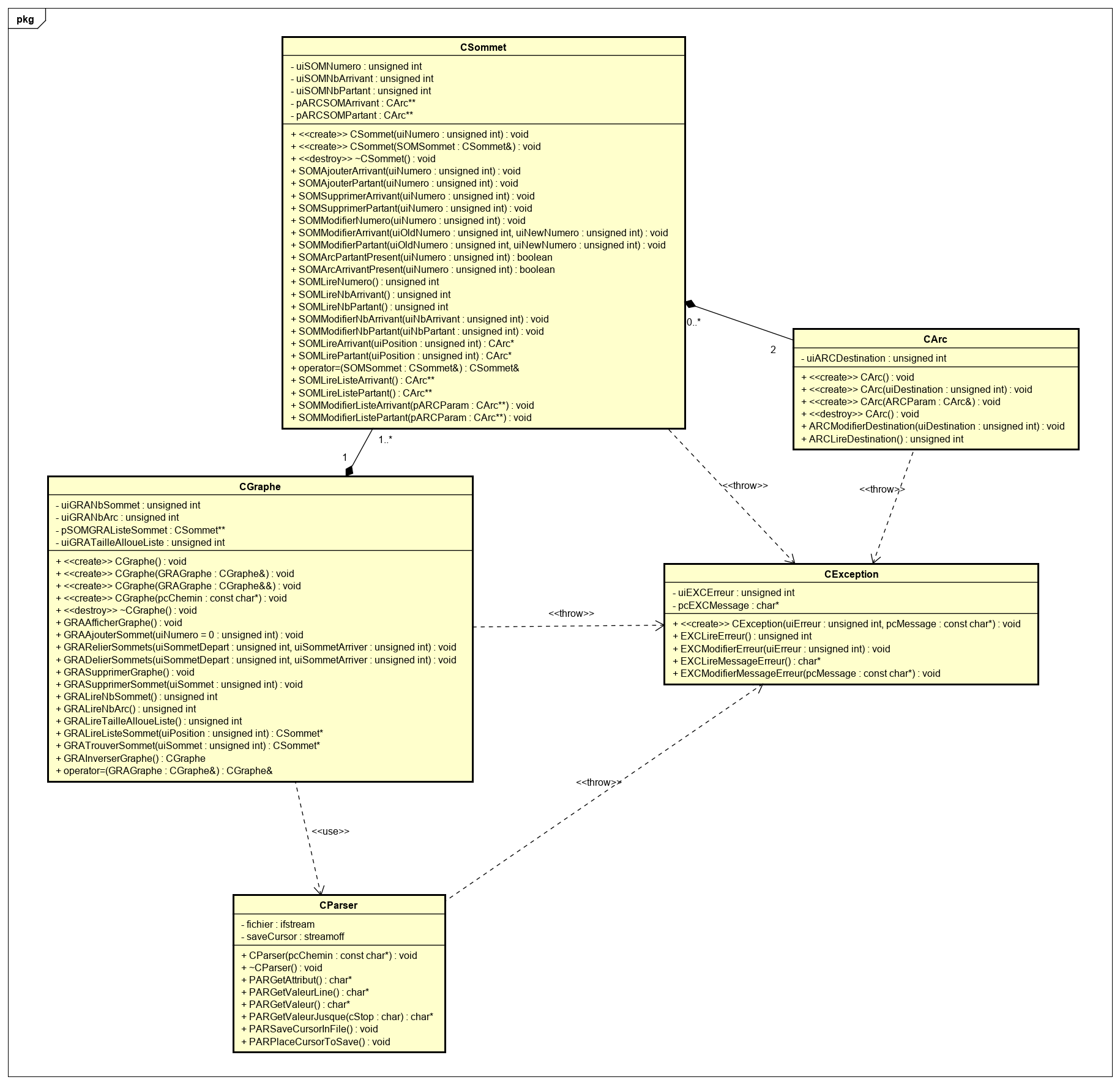
Librairie de classe et de fonctions

# Présentation du sujet

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.

# Modélisation UML

Voici le diagramme UML correspondant à notre projet final sur les graphes.



Notre librairie se décompose en 5 classes principales :

* CGraphe
* CSommet
* CArc
* CParser
* CException

## CGraphe

C’est la classe qui représente un graphe.

On retrouve quatre attributs dans un objet CGraphe :

* uiGRANbSommet : indique le nombre de sommet dans le graphe
* uiGRANbArc : indique le nombre d’arc dans le graphe
* uiGRATailleAlloueListe : indique la taille alloué pour la taille de la liste des sommets
* pSOMGRAListeSommet : contient la liste des objets sommets que compose le graphe.

On retrouve quatre constructeurs et un destructeur dans la classe CGraphe :

* CGraphe() : C’est le constructeur par défaut. Il permet d’allouer la taille de la liste des sommets à un et initialise les attributs.
* CGraphe(const char \* pcChemin) : Ce constructeur permet de construire un objet à partir d’un fichier texte suivant une mise en forme spécifique permettant de représentant un objet CGraphe.
* CGraphe(CGraphe& GRAGraphe) / CGraphe(CGraphe&& GRAGraphe) : Ce sont les constructeurs de recopie avec la norme C++11 qui permettent la recopie d’objet CGraphe par référencement à gauche et à droite.
* ~CGraphe() : C’est le destructeur. Il permet de détruire un objet correctement en vidant la liste des sommets contenue dans l’objet.

On dispose de cinq accesseurs en lecture dans la classe CGraphe :

* GRALireNbSommet() : renvoit un entier non signé correspondant à l’attribut uiGRANbSommet
* GRALireNbArc() : renvoit un entier non signé correspondant à l’attribut uiGRANbArc
* GRALireTailleAlloueListe() : renvoit un entier non signé correspondant à l’attribut uiGRATailleAlloueListe
* GRALireListeSommet(unsigned int uiPosition) : renvoit un pointeur vers un objet CSommet qui se trouve à la position uiPosition dans pSOMGRAListeSommet
* GRATrouverSommet(unsigned int uiSommet) : renvoit un pointeur vers un objet CSommet qui à pour numéro uiSommet dans pSOMGRAListeSommet

On dispose d’un accesseur en écriture dans la classe CGraphe :

* GRAAjouterSommet(unsigned int uiNumero = 0) : permet d’ajouter un nouveau sommet dans la liste des sommets avec un numéro prédéfinit par l’utilisateur où !!!!!!!!!!!!!!!!!!!!!!!!!!!!

La classe CGraphe dispose aussi d’autre méthodes comme :

* GRAAfficherGraphe() : permet d’afficher les numéros de sommets ainsi que les liaisons entre les sommets. Par exemple : Graphe1 : [1,2,3,4,5] [(1,2);(2,3);(4,1);(4,5)]
* GRARelierSommets(unsigned int uiSommetDepart, unsigned int uiSommetArriver) /GRADelierSommets(unsigned int uiSommetDepart, unsigned int uiSommetArriver) : permet de créer des liens, représenter par des arcs, entre les sommets
* GRAInverserGraphe() : permet d’inverser le sens des liens entre les sommets, l’arrivé devient le départ et le départ l’arrivé.
* GRASupprimerSommet(unsigned int uiSommet) : supprime le sommet
* GRASupprimerGraphe() : vide le graphe de tous les sommets

Et enfin la classe possède une surcharge d’opérateur = qui permet d’assigner le graphe au graphe passé en paramètre.

## CSommet

C’est la classe qui représente un sommet du graphe. Son existence est lié à un objet CGraphe.

Un objet CSommet possède cinq attributs :

* uiSOMNumero : indique le numéro du sommet commençant à partir de 1
* uiSOMNbArrivant : indique le nombre d’arc arrivants
* uiSOMNbPartant : indique le nombre d’arc partants
* pARCSOMArrivant : liste de pointeur vers des objets CArc qui partent du sommet
* pARCSOMPartant : liste de pointeur vers des objets CArc qui arrivent vers le sommet

On retrouve deux constructeurs et un destructeur :

* CSommet(unsigned int uiSommet) : Constructeur de confort qui permet de créer un sommet avec un numéro uiSommet
* CSommet(CSommet& SOMSommet) : Constructeur de recopie
* ~CSommet() : Destructeur par défaut

On dispose de sept accesseurs en lecture :

* SOMLireNumero() : retourne le numéro du sommet
* SOMLireNbPartant() : retourne le nombre d’arc partant
* SOMLireNbArrivant() : retourne le nombre d’arc arrivant
* SOMLireArrivant(unsigned int uiPosition) : retourne un pointeur sur l’arc à la position uiPosition dans la liste des arcs arrivants
* SOMLirePartant(unsigned int uiPosition) : retourne un pointeur sur l’arc à la position uiPosition dans la liste des arcs partants.
* SOMLireListeArrivant() : retourne la liste des arcs arrivants
* SOMLireListePartant() : retourne la liste des arcs partants

Ainsi que de neuf accesseurs en écriture :

* SOMAjouterPartant(unsigned int uiNumero) : ajoute un arc à la liste des arcs partant avec le numéro correspondant au sommet de destination
* SOMAjouterArrivant(unsigned int uiNumero) : ajoute un arc à la liste des arcs arrivant avec le numéro correspondant au sommet de destination
* SOMModifierNumero(unsigned int uiNumero) : modifie le numéro du sommet par le nouveau numéro de sommet en paramètre
* SOMModifierPartant(unsigned int uiOldNumero, unsigned int uiNewNumero) : modifie la destination de l’arc partant par le nouveau numéro uiNewNumero
* SOMModifierArrivant(unsigned int uiOldNumero, unsigned int uiNewNumero) : modifie la destination de l’arc arrivant par le nouveau numéro uiNewNumero
* SOMModifierNbArrivant(unsigned int uiNbArrivant) : modifie le nombre d’arc arrivant par le nombre uiNbArrivant
* SOMModifierNbPartant(unsigned int uiNbPartant) : modifie le nombre d’arc partant par le nombre uiNbPartant
* SOMModifierListeArrivant(CArc \*\* pARCParam) : remplace la liste d’arc arrivant par la liste en paramètre
* SOMModifierListePartant(CArc\*\* pARCParam) : remplace la liste d’arc partant par la liste en paramètre

La classe dispose aussi de deux méthodes de vérification :

* SOMArcPartantPresent(unsigned int uiNumero) : vérifie si oui ou non l’arc de destination uiNumero dans la liste des arcs partant est présent
* SOMArcArrivantPresent(unsigned int uiNumero) :vérifie si oui ou non l’arc de destination uiNumero dans la liste des arcs arrivant est présent

Ainsi que deux méthodes de suppression :

* SOMSupprimerArrivant(unsigned int uiNumero) : supprime l’arc de destination uiNumero de la liste des arcs arrivant
* SOMSupprimerPartant(unsigned int uiNumero) : supprime l’arc de destination uiNumero de la liste des arcs partant

La classe dispose aussi d’une surcharge d’opérateur = qui permet d’assigner un sommet à un autre sommet.

## CArc

C’est une classe qui représente un arc. Son existence n’a de sens qu’avec un objet CSommet.

Elle ne dispose que d’un seul attribut :

* uiARCDestination : nombre qui correspond au sommet de destination de l’arc

Ses méthodes sont :

* CArc() : constructeur par défaut
* CArc(unsigned int uiDestination) : constructeur de confort qui initialise uiARCDestination à uiDestination
* CArc(CArc& ARCParam) : constructeur de recopie par défaut
* ~CArc() : destructeur par défaut
* ARCModifierDestination(unsigned int uiDestination) : remplace uiARCDestination par la nouvelle destination uiDestination
* ARCLireDestination() : accesseurs qui retourne uiARCDestination

## CParser

La classe CParser est un analyseur syntaxique.

Un objet CParser contient deux attributs :

* Fichier : un objet ifstream correspondant au pointeur vers le fichier
* saveCursor : un objet streamoff qui permet d’enregistrer un emplacement du curseur

La classe possède un constructeur de confort :

* CParser(const char\* pcChemin) : il permet d’ouvrir le fichier à partir de l’adresse passée en paramètre

Un destructeur qui permet de fermer proprement le fichier et une liste de méthode qui permet de récupérer les valeurs du fichier :

* PARGetAttribut() : permet de retourner l’attribut à l’emplacement du curseur
* PARGetValeurLine() :
* PARGetValeur() :
* PARGetValeurJusque(char cStop) :

Ainsi que des méthodes permettant de manipuler les curseurs :

* PARSaveCursorInFile() :
* PARPlaceCursorToSave() :

## CException

Cette classe permet de gérer les exceptions du projet.

La classe possède les attributs suivants :

* pcEXCMessage (const char \*) : C’est un message d’erreur qui permet de nous indiquer plus d’information sur les erreurs
* uiEXCErreur (unsigned int) : C’est le numéro d’erreur défini par fichier pour chaque erreur potentielle.

Ainsi qu’un constructeur de confort :

* CException(unsigned int uiErreur, const char\* pcMessage) : il permet de créer l’objet CException avec un numéro d’erreur et un message

De deux accesseurs en lecture :

* EXCLireErreur() : retourne le numéro d’erreur
* EXCLireMessageErreur() : retourne le message d’erreur

Et de deux accesseurs en écriture () :

* EXCModifierErreur (unsigned int uiErreur) : modifie le numéro de l’erreur
* EXCModifierMessageErreur(const char\* pcMessage) modifie le message d’erreur

# Evolution du projet

Pour pouvoir réaliser ce projet nous avons commencé par le développement de la classe CArc puis de CSommet et enfin de CGraphe. Nous avons récuperer la classe CException et CParser du projet de matrice et modifier la classe CParser afin de la rendre plus réutilisable en la rendant plus neutre par rapport au projet.

# Notice d’utilisation