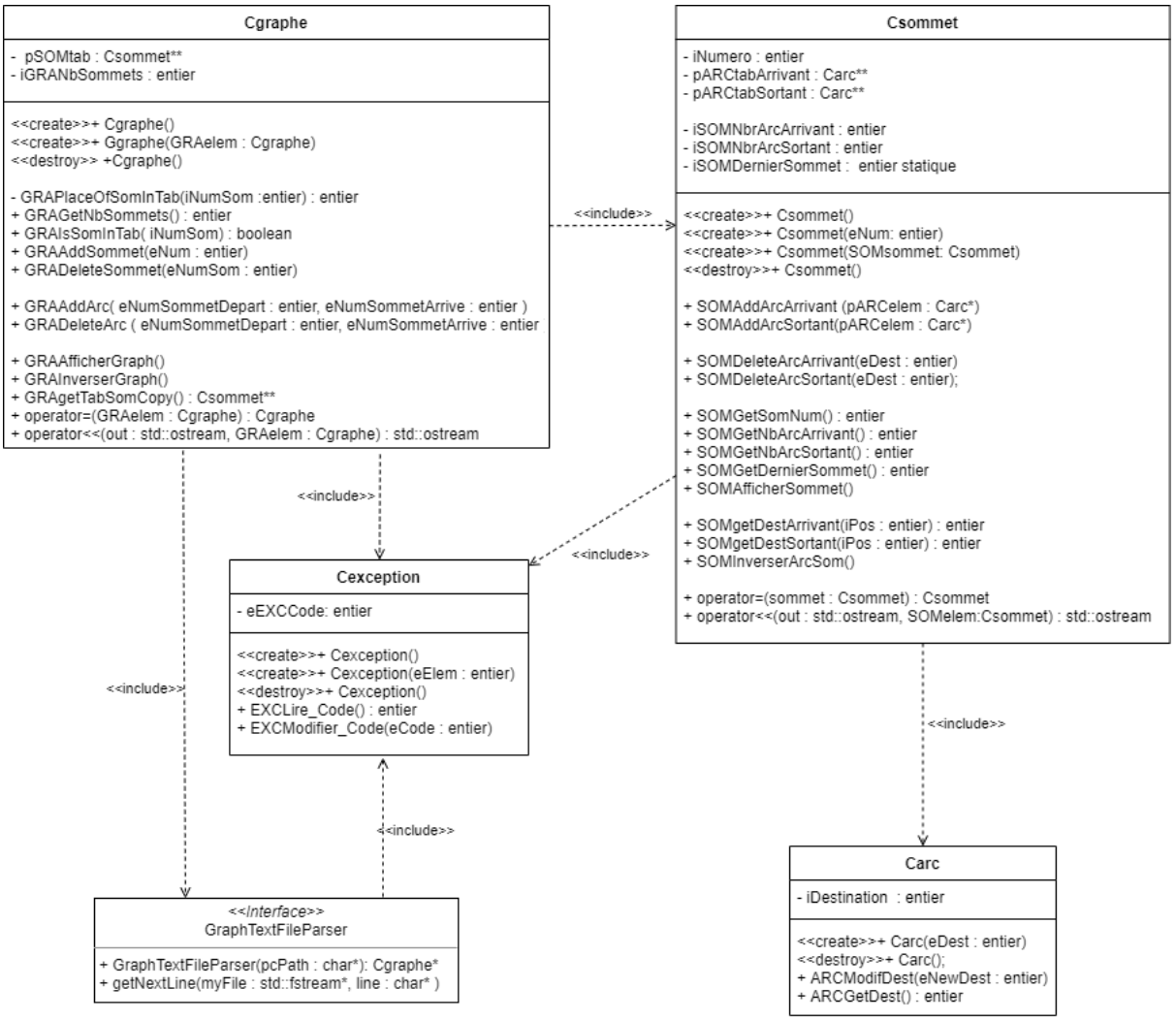
Rapport de projet Tutoré C++

Partie 2 - Graphes

Lilian Pouvreau
Corentin Esnault

Diagramme UML



Choix importants

Opérateur New/Delete

Nous avons décidé d'utiliser les opérateurs new et delete, car nous étions plus à l'aise avec ceux là, et car cela nous estimons que notre code peut ainsi effectuer la même chose que si nous avions utilisé realloc. Etant donné que nous copions des adresses lors d'ajouts/suppressions d'arcs/de sommets, nous ne perdons pas de temps à recréer des objets. De plus, la fonction realloc effectue la même chose que notre code.

Création à partir d'un fichier

Ce constructeur est à part des autres car il est long et complexe - puisqu'il contient un parser et un constructeur - et cela améliore la lisibilité de l'ensemble. De plus, il est différent des constructeurs "classiques" vus en cours et fait lui-même appel aux constructeurs définis dans la classe Cgraphe.

Lecture du Fichier/Parser

Concernant la lecture d'un fichier texte contenant des informations sur un graphe à créer, nous avons eu différents choix à réaliser.

Tout d'abord, nous avons décidé que si une lecture d'un fichier ne se passait pas correctement, nous arrêtions le programme plutôt que d'essayer de continuer avec un graphe erroné. Par exemple, si le nombre de sommets trouvé est différent de celui annoncé, nous levons une exception et ne continuons pas à lire le fichier. Ce choix nous permet de ne pas travailler avec des graphes incomplets, ou des erreurs faisant planter le programme.

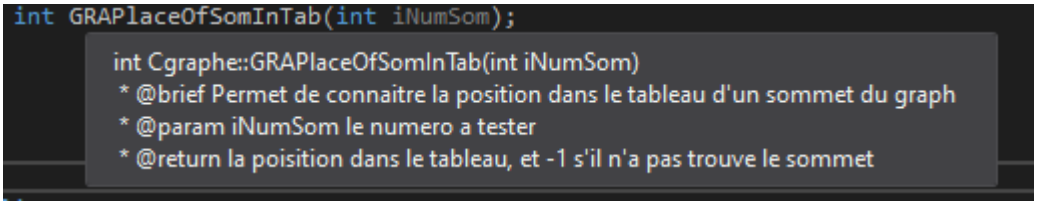
Un autre choix important est que le format du fichier doit être suivi à la lettre. S'il y a des sauts de ligne en trop, ou des espaces non nécessaires, et que ceux-ci font planter le programme, c'est à l'utilisateur de modifier son fichier pour correspondre au format. Nous avons ainsi donné en précondition de la fonction le format attendu du fichier.

Nous essayons cependant de détecter et auto-corriger certaines erreurs. Nous sommes capables de détecter s'il y a des sauts de lignes ajoutés entre deux lignes, ou nous pouvons retirer les espaces non nécessaires, grâce à "atoi", du moment que ces espaces ne sont pas en plein milieu d'un nombre. Ces erreurs seront corrigées automatiquement par des fonctions.

Nous pouvons également détecter si le nombre d'arc ou de sommets ne correspond pas à celui énoncé en début de fichier. Enfin, les fonctions de Cgraphe peuvent également détecter si un sommet est créé plusieurs fois, s'il n'existe pas lors de l'ajout d'un arc, etc.. Toutes ces erreurs lèveront une exception.

Ecriture des commentaires

Concernant les commentaires, et plus précisément les descriptions des fonctions, nous avons utilisé la convention doxygen c++ (c'est-à-dire utiliser "///" pour par exemple avoir "/// @brief une description de fonction"). Nous nous sommes rendu compte que cette convention ne fonctionnait pas du tout avec visual studio 2017, malgré l'installation d'une extension faite pour cela. Nous avons donc fait le choix d'avoir une convention plus proche du C (exemple : /** puis * @brief une description de fonction et enfin */) afin que Visual studio soit capable de l'interpréter pour permettre une écriture du code bien plus agréable ressemblant à cela :



Ajout d'un un arc

Pour ajouter un arc au graphe, on donne le numéro des sommets qu'il relie dans l'ordre du sens de la flèche. Cela améliore la lisibilité du code et correspond au mieux à la modélisation d'un graphe.

Lien entre les sommets

Afin d'éviter toute confusion, nous considérons qu'un seul arc peut faire le lien entre deux sommets. Si l'on souhaite rajouter un arc entre deux sommets déjà reliés, une exception est levée.

Numéro de sommet

Puisque chaque sommet est identifié par un numéro, nous avons ajouté une variable statique iSOMDernierSommet qui correspond au numéro attribué au dernier sommet créé. Ainsi, lorsque le constructeur par défaut est appelé, il attribue au sommet qu'il crée le numéro iSOMDernierSommet+1, et iSOMDernierSommet est mis à jour.

Opérateur de flux de sortie

Afin de faciliter l'affichage du/des graphes pour l'utilisateur, nous avons décidé d'ajouter une surcharge de l'opérateur de flux de sortie "<<". Ainsi, plutôt que d'appeler la fonction d'affichage, on peut écrire facilement plusieurs graphes/sommets/arc dans la console, ou encore dans un fichier texte par exemple.

Mise en place de tests

Durant ce projet, nous avons essayé de mettre en place un projet de test, afin de créer des tests permettant de vérifier rapidement le fonctionnement de nos classes. Ces tests ne sont pas complets, mais permettent tout de même de vérifier certaines fonctions de bases.

Manuel d'utilisation

Carc

Attributs

iDestination	Entier	Numéro du sommet de destination de l'arc (qui n'indique pas forcément le sens)

Méthodes

Carc(int iDest);	Constructeur de Carc	Carc ARCtest(2); Carc* pARCtest = new Carc(2);
~Carc();	Destructeur de Carc	delete pARCtest;
void ARCModifDest(int	setter pour modifier la	ARCtest.ARCModifDest(3);

iNewDest);	destination de l'arc	
int ARCgetDest();	getter pour l'attribut de destination	ARCtest.ARCgetDest();

Csommet

Attributs

iNumero	Entier	Numéro du sommet
pARCtabArrivant	pointeur de Carc	Tableau de Carc arrivant au sommet
pARCtabSortant	pointeur de Carc	Tableau de Carc sortant du sommet
iNbArcArrivant	Entier	Nombre totals d'arcs arrivant du sommet
iNbArcSortant	Entier	Nombre totals d'arcs sortant du sommet
iDernierSommet	Entier Statique	Permet de garder le numéro du dernier sommet créé pour connaître le prochain numéro disponible.

Méthodes

Csommet();	Constructeur par défaut	Csommet SOMtest(); Csommet* pSOMtest=new Csommet();
Csommet(int iNum);	Constructeur de confort	Csommet SOMtest(5);
Csommet(Csommet& SOMsommet);	Constructeur de recopie	Csommet SOMcopy(SOMtest);
~Csommet();	Destructeur de Csommet	delete pSOMtest;
void SOMAddArcArrivant(int iDest);	Permet d'ajouter un arc arrivant au sommet	SOMtest.SOMAddArcArrivant(2);
void SOMAddArcSortant(int iDest);	Permet d'ajouter un arc sortant au sommet	SOMtest.SOMAddArcSortant(2);
void SOMDeleteArcArrivant(int iDest);	Permet de supprimer un arc arrivant au sommet	SOMtest.SOMDeleteArcarrivant(2);
void	Permet de supprimer un arc	SOMtest.SOMDeleteArcSort

SOMDeleteArcSortant(int iDest);	sortant au sommet	ant(2);
int SOMGetSomNum();	Récupère le numéro de sommet actuel	SOMtest.SOMGetSomNum();
int SOMGetNbArcArrivant();	Donne le nombre d'arcs arrivants	SOMtest.SOMGetNbArcArrivant();
int SOMGetNbArcSortant();	Donne le nombre d'arcs sortants	SOMtest.SOMGetNbArcSortant();
int SOMGetdernierSommet();	Récupère le numéro du dernier sommet créé	SOMtest.SOMGetDernierSommet();
void SOMAfficherSommet();	Affiche toutes les informations du sommet	SOMtest.SOMAfficherSommet();
int SOMgetDestArrivant(int iPos);	Renvoie la destination de l'arc arrivant à la position iPos dans le tableau	SOMtest.SOMgetDestArrivant(0);
int SOMgetDestSortant(int iPos);	Renvoie la destination de l'arc sortant à la position iPosdans le tableau	SOMtest.SOMgetDestSortant(0);
void SOMInverserArcSom();	Permet d'inverser tous les arcs du sommet.	SOMtest.SOMInverserArcSom();
Csommet& operator=(Csommet &sommet);	Surcharge de l'opérateur egal	SOMtest = SOMtest2;
std::ostream& operator<<(std::ostream& out, Csommet& SOMelem);	surcharge du flux de sortie pour un affichage simplifié de la structure	std::cout<< SOMtest;

Cgraphe

Attributs

pSOMtab	Pointeur de Csommet	Un tableau de Csommet
iNbSommets	Entier	Compteur de nombre de sommets

Méthodes

Cgraphe();	Constructeur par défaut de Cgraphe	Cgraphe GRAtest(); Cgraphe* pGRAtest = new Cgraphe();
Cgraphe(Cgraphe& GRAelem);	Constructeur de recopie	Cgraphe GRAcopy(GRAtest);
~Cgraphe();	Destructeur de Cgraphe	delete pGRAtest;
int GRAPlaceOfSomInTab(int iNumSom);	Permet de connaître la position dans le tableau d'un sommet du graph	GRAtest.GRAPlaceOfSomInTab(1) ;
int GRAGetNbSommets();	getter pour le nombre de sommets dans le graphe	GRAtest.GRAGetNbSommets();
bool GRAIsSomInTab(int iNumSom);	Permet de savoir si un numéro est un sommet du graph	GRAtest.GRAIsSomInTab(1);
void GRAAddSommet(int iNum);	Permet d'ajouter un Sommet dans le graphe, sans aucun Arcs	GRAtest.GRAAddSommet(1);
void GRADeleteSommet(int iNumSom);	Permet de supprimer un Sommet dans le graphe, ainsi que tous ses arcs (et les arcs dans les sommets reliés à celui-ci)	GRAtest.GRADeleteSommet(1);
void GRAAddArc(int iNumSommetDepart, int iNumSommetArrivee);	Permet d'ajouter un arc entre deux sommets	GRAtest.GRAAddArc(1,2);
void GRADeleteArc(int iNumSommetDepart, int iNumSommetArrivee);	Permet de supprimer un arc entre deux sommets	GRAtest.GRADeleteArc(1,2);
void GRAAfficherGraph();	Affichage du graph oriente, en affichant tous les sommets existants, ainsi que les arcs entrants et sortants de ce sommet	GRAtest.GRAAfficherGraph();
void GRAInverserGraph();	Fonction afin d'inverser tous les arcs du graphs, entre tous les sommets	GRAtest.GRAInverserGraph();
Csommet** GRAgetTabSomCopy(void);	Permet de récupérer une Copie du tableau des sommets	GRAtest.GRAgetTabSomCopy();
Cgraphe&	Opérateur = surchargé	GRAcopy = GRAtest;

operator=(Cgraphe& GRAelem);		
std::ostream& operator<<(std::ostream& out, Cgraphe& GRAelem);	surcharge du flux de sortie pour un affichage simplifié de la structure	std::cout << GRAtest;

File parser

Méthodes

Cgraphe* GraphTextFileParser(char* pcPath)	Crée un graphe orienté à partir d'un nom de fichier/chemin vers un fichier	GraphTextFileParser(graphe.txt)
inline void getNextLine(std::fstream* myFile, char* line)	Fonction pour passer a la prochaine ligne non vide du fichier. Lance une exception si une fin de document est trouvée.	getNextLine(&myFile, line)