Mohamed FAYALA
Youssef SIDHOM

Compte Rendu de TP

TP C++ N°2: HÉRITAGE - POLYMORPHISME

Sommaire :

1. DESCRIPTION DÉTAILLÉE DES CLASSES

Notre classe Catalogue est la classe qui nous sert comme interface console avec l'utilisateur qui lui permet de choisir une des opérations suivantes :

- Afficher tous les trajets du catalogue
- Ajouter un trajet simple au catalogue
- Ajouter un trajet composé au catalogue
- Effectuer une recherche simple d'un trajet entre deux villes
- Rechercher une combinaison de trajets possible entre deux villes à partir des trajets du catalogue.

Pour stocker les différents trajets du catalogue on utilise une Linkedlist de trajets. Chaque cellule de la liste chaînée stocke un pointeur vers un Trajet et un pointeur vers une cellule (la cellule suivante qui compose notre liste chainée).

Afin de faciliter l'usage à notre utilisateur, nous avons décidé d'organiser l'affichage de notre Catalogue selon L'Ordre alphabétique des villes de départs. Cette opération est réalisée avant chaque affichage du Catalogue.

La classe Trajet est une classe abstraite, ancêtre des d eux classes filles TrajetSimple et TrajetComposé. Ces deux dernières classes héritent de Trajet, ce qui nous permet de les manipuler dans une liste hétérogène de trajets sans faire la distinction entre les deux. Nous avons choisi de caractériser tout Trajet par une ville de départ et une ville d'arrivée. Un TrajetSimple a donc comme attributs 'arrivee' et 'depart' et en plus un moyen de transport 'transport'. Un TrajetCompose a comme attributs 'arrivee' et 'depart' héritées du Trajet ainsi qu'une liste chaînée de trajets qui contient les étapes (trajets simples ou composés) qui le composent. Pour chaque ajout d'une étape à un trajet composé on vérifie que la ville d'arrivée du dernier trajet de la liste et la ville de départ du trajet que nous souhaitons ajouter correspondent, et nous modifions aussi la ville d'arrivée attribué à ce trajet composé.

2.DESCRIPTION DÉTAILLÉE DE LA STRUCTURE DE DONNÉE

La structure de données que nous avons choisi d'utiliser pour gérer notre catalogue ainsi que notre Trajet Compose est une liste chaînée. Une liste chaînée étant composée par des cellules qui contiennent des informations et un pointeur vers la cellule suivante, on a commencé par créer la classe Cell. Cell a comme attributs data de type pointeur vers un Trajet et next de type pointeur vers Cell. Le premier attribut pointe donc vers le Trajet qu'on veut stocker à l'intérieur de la cellule et next pointe vers la cellule suivante dans la liste chaînée.

Une liste chaînée a comme attribut ‘racine’ qui est la première cellule de la Linkedlist qui va pointer vers la cellule suivante jusqu’à la fin. La fin de la liste est une cellule dont l’attribut next pointe vers *nullptr.* Un affichage d’une liste chaînée est un parcours de la liste et un appel à la fonction Affichage () de chaque trajet.

La structure est utilisée dans la classe Catalogue et la classe TrajetCompose mais avec la possibilité d’ordonner la collection de deux façons différentes : par ordre d’insertion pour respecter la sémantique d’un parcours ou par ordre alphabétique des villes de départ en appelant la méthode tri () de Linkedlist dans la méthode Affiche () d’un Catalogue.

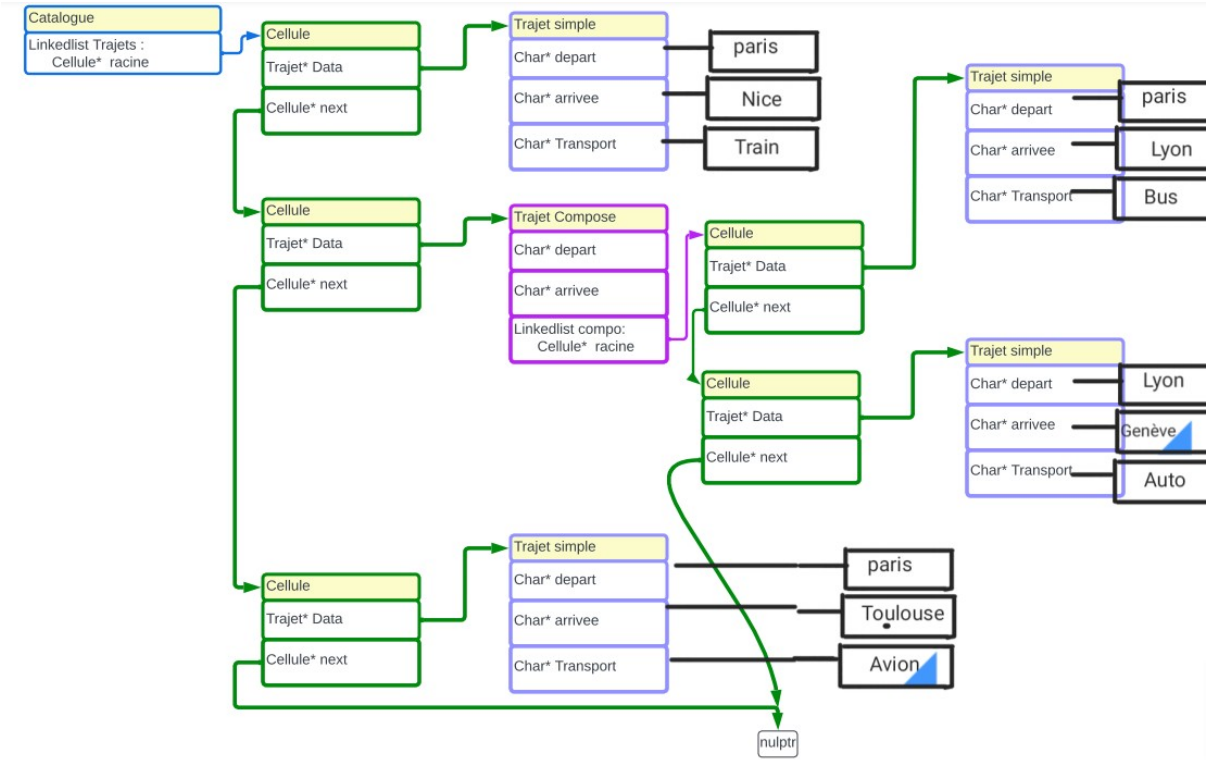

Figure 1: Schéma de la représentation en mémoire

3. AXES D'AMÉLIORATION

- Optimisation du format d'entrée des données :

Notre Class Catalogue permet à n'importe quel utilisateur d'ajouter et de chercher un trajet et d'afficher tout le contenu du Catalogue. Dans une approche un peu plus professionnelle, modifier le Catalogue doit être réservé seulement à l'administrateur. Donc l'un des axes d'amélioration serait de limiter l'accès à certaines fonctionnalités du Catalogue selon le statut de l'utilisateur (admin/particulier).

On pourra aussi travailler sur les noms de villes ou les types de transport. En effet, pour le moment nous n'avons pas un système qui vérifie la véracité des données rentrées, il est possible de rentrer un trajet qui va de Clément à Marie en roue libre Pour cela on pourra utiliser une base de données pour les noms de villes et un simple enum pour les moyens de transport.

La création d'un trajet composé peut être aussi améliorée. En effet, pour le moment pour créer un trajet composé on ne peut que le composé à partir de nouveaux trajets simples. On pourrait tout d'abord permettre à l'utilisateur de rajouter un trajet composé comme étant une étape d'un trajet composé, mais aussi on devrait proposer à l'utilisateur la possibilité de rajouter un trajet simple ou composé qui existe déjà dans le Catalogue.

- Optimisation de la structure de données :

L'intérêt primaire de ce Catalogue est de chercher le trajet qui convient le plus à l'utilisateur, de plus L'offre de trajets d'une société de transport est rarement changée. A part, pour l'initialisation des trajets qui nécessite d'ajouter des masses de données, la fonctionnalité de notre Catalogue reste majoritairement la recherche.

Ce qui nous amène à notre deuxième problème lié à l'optimisation. Nous utilisons une structure de données (listes chaînées) qui est plus adaptée à l'ajout de trajets plutôt qu'à la recherche. En effet un ajout nous coûtera environ O(1) ce qui est très optimal, cependant un simple parcours de la liste pour réaliser une recherche simple nous coûte O(n), cela sans oublier le cas d'une double boucle utiliser par exemple dans la fonction de tri qui va nous coûter jusqu'à $O(n^2)$.

Ce problème pourra être résolu avec les tableaux. Avec, un tableau on va résoudre le problème du parcours de la liste des trajets qui serait d'une complexité O(1), sauf que la complexité de l'ajout atteindra le O(n) mais comme on utilise plus le Catalogue pour chercher que pour ajouter cela à notre avis reste mieux en termes d'optimisation. Enfin, le seul problème des tableaux c'est qu'une fois le tableau plein faudra créer un nouveau tableau, chose très complexe si on gère beaucoup de données.

Donc il faudrait une stratégie propice afin d'avoir assez d'espace pour éviter de réallouer le tableau sans pour autant prendre de la place inutile dans la mémoire.

- Optimisation de l'interface graphique :

Dernière optimisation à envisager reste celle de l'interface graphique, malgré qu'elle soit la plus insignifiante d'un point vue technique, elle reste d'une importance primordiale à notre Catalogue.

L'intérêt est de présenter à l'utilisateur une interface plus développée, pour le moment on reste sur un modèle très basique avec un simple affichage de lignes sur terminale.

Seul problème des tableaux c'est qu'une fois le tableau plein faudra créer un nouveau tableau, chose très complexe si on gère beaucoup de données, Donc il faudrait une stratégie propice afin d'avoir assez d'espace pour éviter de réallouer le tableau sans pour autant gaspiller la mémoire.