TP C++ n°2 : Héritage - Polymorphismme

Faouz Hachim Jérôme Hue

12 décembre 2019

1 Description des classes

Notre programme comporte 4 classes : *Trajet*, *TrajetSimple*, *TrajetComp* et *Catalogue*. Vous trouverez ci dessous la description détaillée de chacune des classes. Une explication plus détaillée de chacune des méthodes est fournie dans le code.

1.1 La classe Trajet

Cette classe comporte seulement deux attributs : une ville de départ *villeDepart*et une ville d'arrivée *villeArr*. Les deux classes TrajetComp et TrajetSimple héritent de cette classe. Il est impossible pour un utilisateur de créer un trajet (donc ni simple ni composé) mais cette classe n'est pas abstraite. Concernant les méthodes, on a défini un constructeur par défaut, un constructeur par copie, un constructeur prenant en paramètre deux villes, un destructeur, des accesseurs et mutateurs ainsi qu'une méthode d'affichage, *Afficher()*.

1.2 La classe TrajetSimple

La classe TrajetSimple hérite de la classe Trajet, et comporte comme attribut supplémentaire un moyen de transport. Les méthodes de cette classe sont son constructeur par défaut, un constructeur prenant en paramètre trois chaînes de caractères pour les villes de départ, d'arrivée, ainsi qu'un moyen de transport. De plus on a défini une méthode d'affichage, un mutateur pour le moyen de transport et une méthode *creerTrajetSimple()* qui demande à l'utilisateur de saisir successivement les 3 attributs d'un Trajet simple puis fait appel au constructeur de la classe.

1.3 La classe TrajetComp

La classe TrajetComp hérite de la classe Trajet et comporte en plus un tableau de pointeurs de trajets. Il est spécifié dans les consignes de TP de n'implémenter un Trajet composé uniquement avec des trajets simples, mais nous avons fait en sorte que l'on puisse créer un trajet composé de trajets composés. En plus du constructeur par défaut, d'un constructeur prenant en paramètres une ville de départ et d'arrivée et initialisant un tableau de 50 trajets et du destructeur, on trouve une méthode *creerTrajetComp* qui demande à l'utilisateur de saisir successivement un nombre n de trajets qu'il aura spécifié au début de la méthode. Ces trajets sont ajoutés grâce à une méthode *addTrajet()*. Enfin, on trouve une méthode d'affichage.

1.4 Graphe d'héritage

Le graphe d'héritage est fourni en figure 1.

1.5 La classe Catalogue

Cette classe permet de stocker des trajets dans un tableau dynamique de pointeurs de trajets. Elle comporte donc un tableau dynamique de trajets, la taille de ce tableau ainsi que le nombre de trajets actuellement présents dans le tableau. Lorsque le tableau est rempli, on double sa taille. Hormis les constructeurs et destructeurs, on trouve une méthode d'affichage, une méthode d'ajout de trajet au catalogue et une méthode de recherche de trajet. Enfin, il existe des accesseurs pour chaque attribut de la classe ainsi que deux mutateurs pour enlever un trajet du catalogue, qui sont utilisés pour l'exploration.

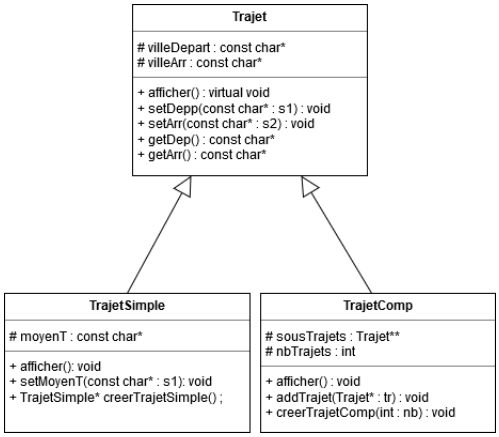

FIGURE 1 – Graphe d'héritage

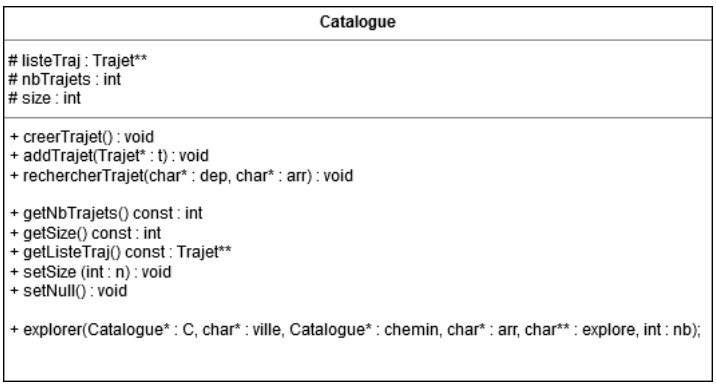

FIGURE 2 – La classe Catalogue

2 Structure de données et état de la mémoire

2.1 Structure de données

Pour gérer la collection ordonnées, on utilise un tableau dynamique. Son principal avantage est qu'il permet un accès rapide a un élément i du tableau. De plus, il n'est pas spécifié dans les consignes du TP d'implémenter une fonction permettant de supprimer un trajet du tableau, ce qui est l'un des principaux désavantages d'un tableau dynamique par rapport à une liste chaînée. Reste alors comme désavantage le fait d'avoir à copier l'intégralité du tableau lorsque celui-ci est rempli. En pratique, le tableau étant rempli à la main, cela importe peu puisque on n'atteindra pas souvent la capacité maximum du tableau.

2.2 État de la mémoire

2.2.1 Rappel du jeu de test

Le jeu de test est composé des trois trajets suivants :

a) Trajet simple **TS1** : De Lyon à Bordeaux en Train.

b) Trajet Composé **TC2** : De Lyon à Paris en Auto.

c) Trajet simple **TS3** : De Lyon à Marseille en Bateau puis de Marseille à Paris en Avion.

2.2.2 État de la mémoire

La figure 3 représente l'état de la mémoire après création des trois trajets des jeux de test.

3 Conclusion

Au cours de ce TP, les principales difficultés rencontrées auront été sans surprise relatives à la manipulation des pointeurs, notamment lors de la création d'un tableau composé, d'autant plus avec la contrainte de n'avoir aucune erreur ni fuite de mémoire dans valgrind. La principale idée d'amélioration est la création d'une structure dédiée (avec `typedef struct`) pour le tableau dynamique comportant les trajets qui est en l'état géré "à la volée" dans la méthode *addTrajet()*.

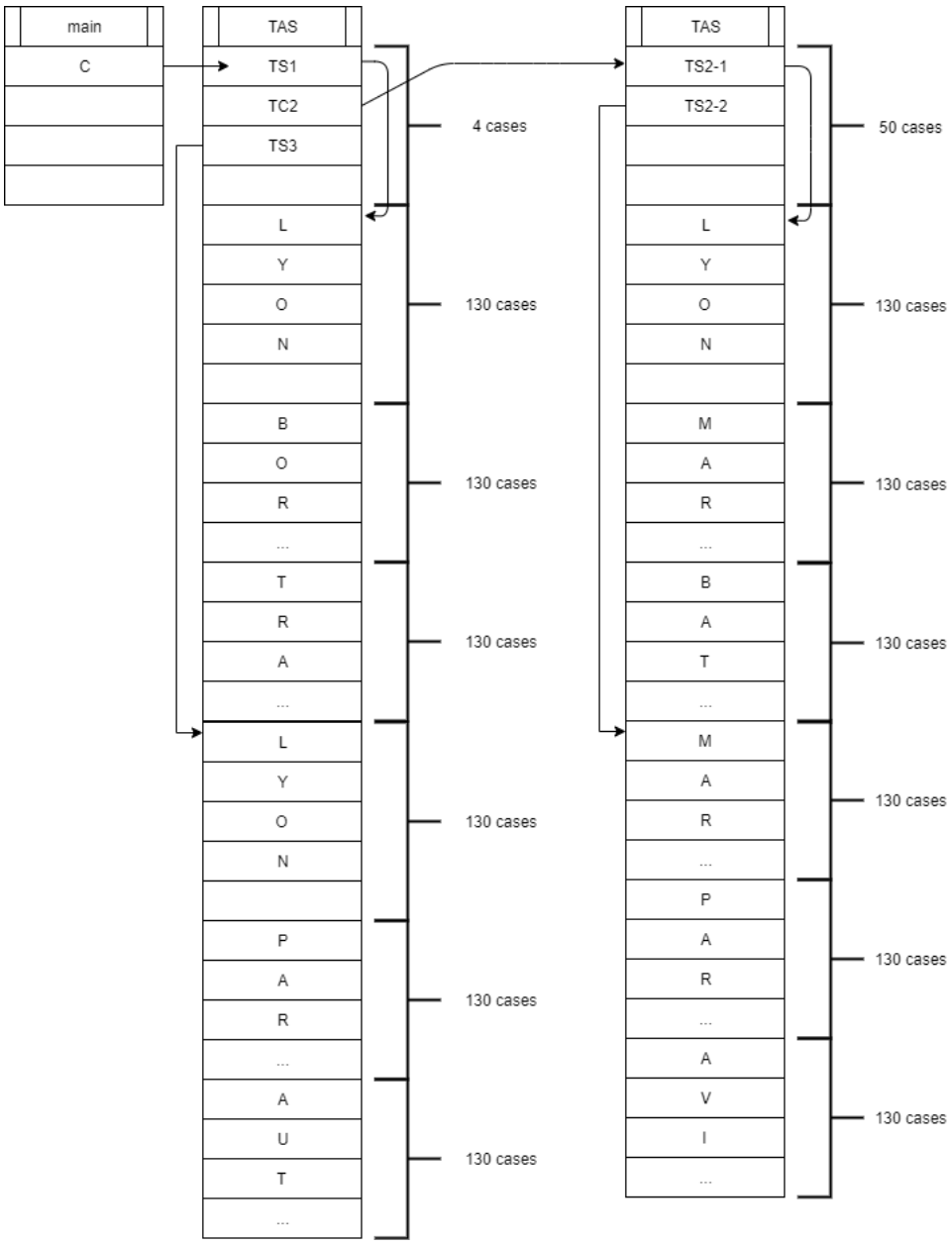

FIGURE 3 – Etat de la mémoire après création des jeux de test