**A. Description des classes**

Une classe Abstraite **Trajet**.

Pas de constructeur,

3 méthodes virtuelles pures.

- 2 pour obtenir les villes d'arrivée et de départ du trajet

- Une pour afficher le trajet sur la console

Une classe **TrajetSimple** enfant de **Trajet**

3 attributs privés (char\*) : *villeDepart*, *villeArrivee* et *transport*

Une classe **TrajetListe** représentant une liste de Trajets

Deux structures internes.

**Element**, représentant un élément de la liste chainée (contenant un **Trajet**), lié à l'élément suivant

**IElement**, représentant un élément d'ittération de la liste chainée, utilisé uniquement pour les foreach(s)

Des méthodes pour obtenir le premier / le dernier **Trajet** de la liste

Des méthodes pour ajouter / supprimer des trajets dans la liste

Une méthode de libération mémoire spéciale

Des méthodes système utilisées pour les foreach(s)

Une classe **TrajetCompose** enfant de **Trajet**

Un attribut privé représentant la liste des trajets composant le trajet global

Une méthode permettant de construire pas à pas le trajet composé (Ajoute())

Une classe **Catalogue** regroupant tous les trajets disponibles

Un attribut privé représentant la liste des trajets composant le catalogue

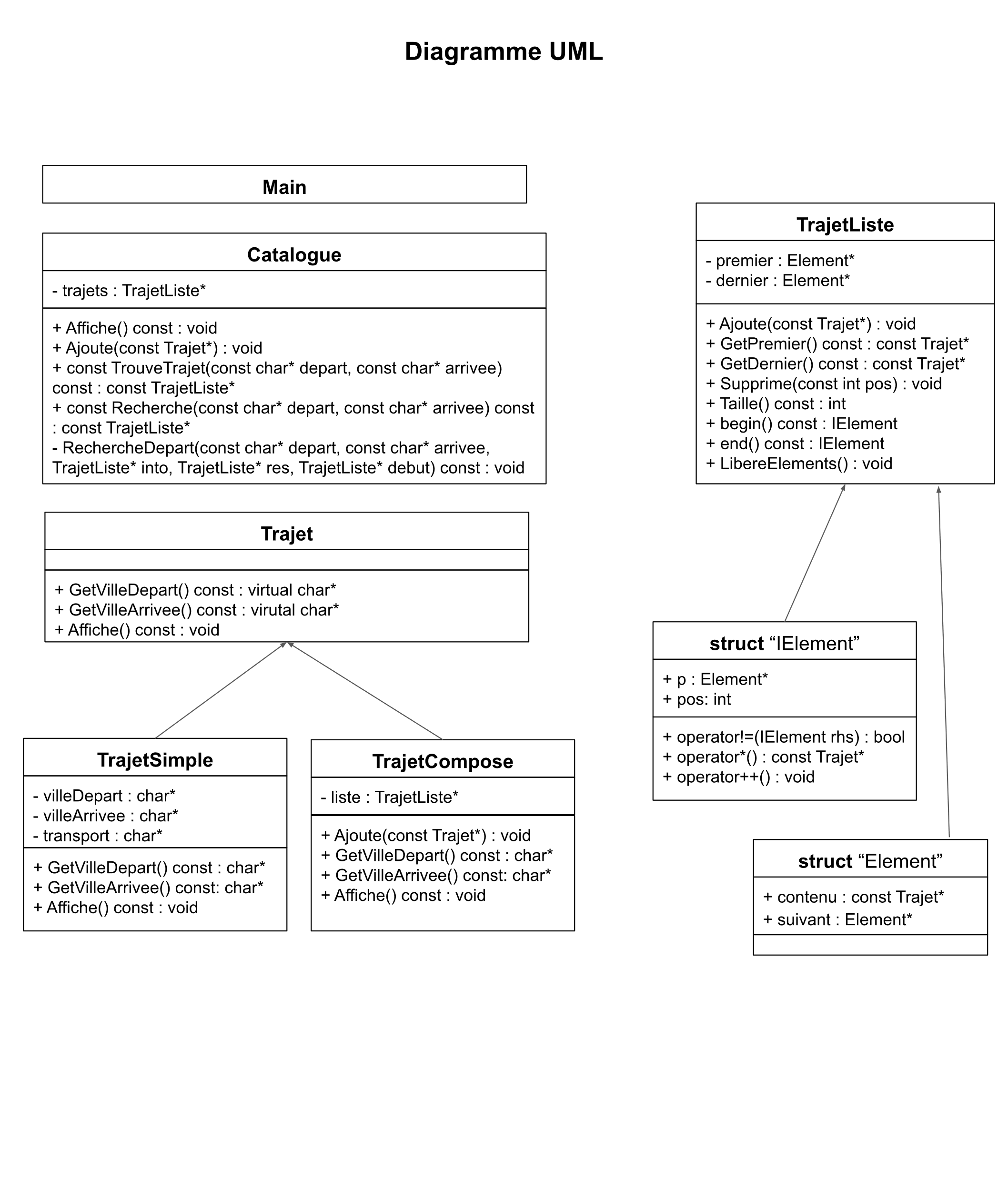
Une méthode d'affichage

Une méthode construction (Ajoute())

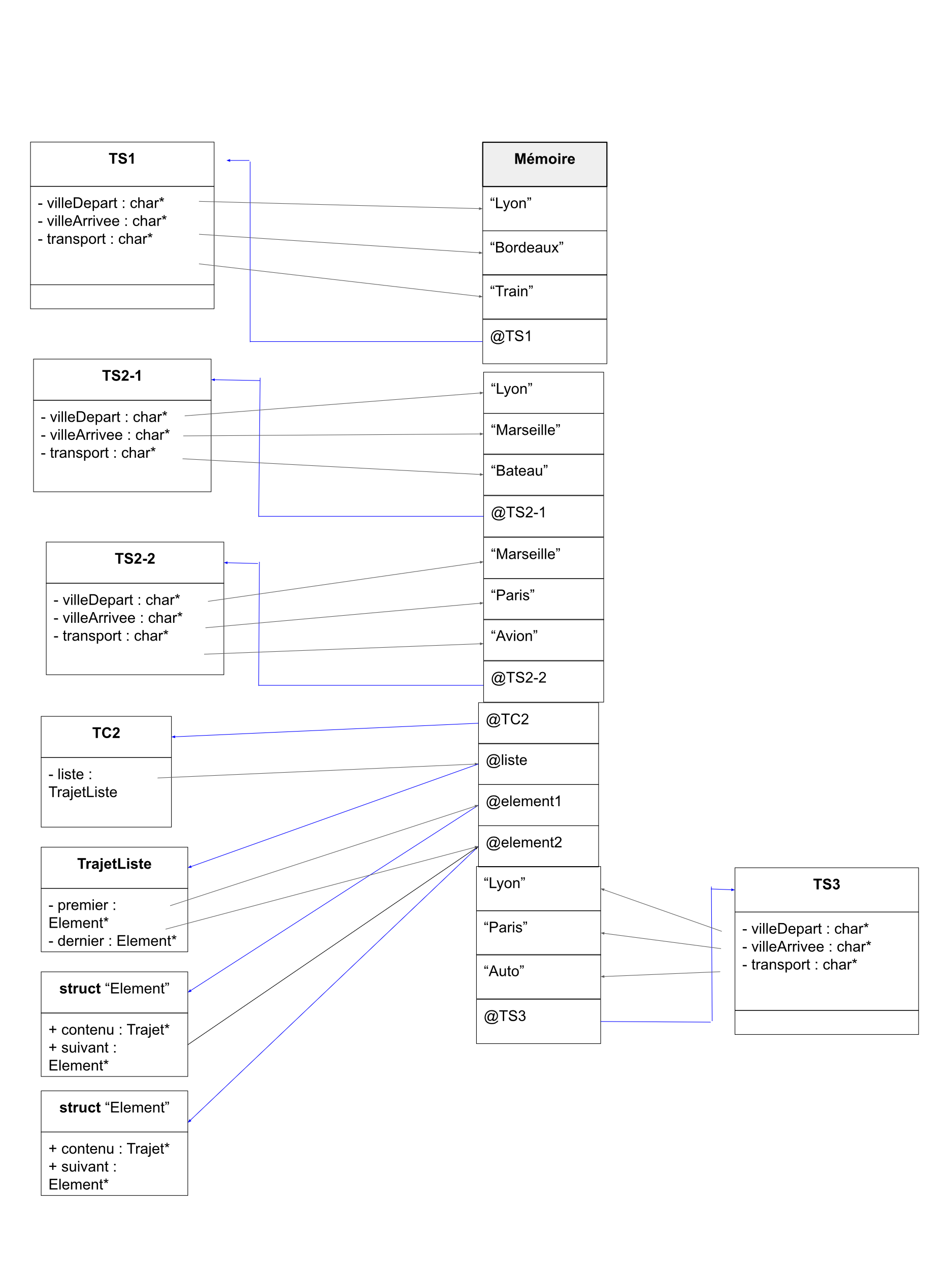
Deux méthodes publiques de recherches (soit simple soit avancée)

Une méthode privée utiliser pour la recherche avancée

Voir UML à la suite :

****

**B. La description détaillée de la structure de données :**

****

**C. Listing des classes**

**D. Conclusion**

Dans ce tp, nous avons rencontré principalement deux problèmes.

Premièrement, nous avons implémenté la recherche avancée par récursivité. Mais nous avions un overflow lorsque la recherche contenait une boucle. Afin de palier à ce problème, nous avons proposé différentes solutions. Soit effectuer un appel récursif seulement sur une ville que l’on n’a pas encore étudiée. Soit passer en paramètre les trajets restants possibles (c’est à dire non utilisés et ne créant pas de boucles). Nous avons finalement réalisé la deuxième méthode.

Le deuxième problème que nous avons rencontré est le valgrind. Nous avions quelques fuites de données. Après de longues recherches, nous avons réussi à quasiment toutes les solutionnées, seule la recherche avancée pose encore parfois problème.

Axes d’évolutions :

- Ajouter différente notion sur chaque trajet : Distance, Temps, Coût. Ce qui permettrait, à l’aide d’une nouvelle recherche, de donner à l’utilisateur les trajets les plus intéressants pour lui suivant la notion qu’il choisit.

- Ajouter une interface graphique (sans console)

**E. Utilisation du programme**

Une fois l'exécutable lancé, un menu des actions disponibles sera affiché, voici un récapitulatif :

"Ajout" -> Permet d'ajouter un trajet simple ou composé au catalogue

"Affichage" -> Affiche le contenu du catalogue sur la console

"Recherche" -> Recherche (de la manière souhaitée\*) un trajet dans le catalogue

"Quitter" -> Clos proprement le programme

\*recherche simple ou avancée