Auteurs : Rim BENZEKRI, Adam CHELLAOUI

TP1 C++ (POO2) : Gestion des entrées / sorties

I. Fichier démo

I.1. Description détaillée du format

Chaque ligne commence par un **$** pour un **trajet simple** ou un **@** pour un **trajet composé**. Aussi, chaque ligne se termine par un **%**.

Pour un **trajet simple**, nous avons choisi le format suivant :

$(ville de départ),(ville d'arrivée),(nombre de trajets):(ville de départ),(moyen de transport),(ville d'arrivée),%

Nous pouvons noter que le **nombre de trajets** est **toujours égal à 1** pour un **trajet simple**.

Pour un **trajet composé**, nous avons choisi le format suivant :

@(ville de départ),(ville d'arrivée),(nombre de trajets) :(ville de départ),(moyen de transport),(ville de 1ère escale),(ville de 1ère escale),(moyen de transport),(ville de 2ème escale),(ville de 2ème escale),(moyen de transport),…..,(ville d'arrivée),%

N.B. Quand un catalogue est sauvegardé dans un fichier, la syntaxe est exactement la même que dans le fichier démo, ainsi, un catalogue sauvegardé peut également être inséré comme nouveau catalogue.

I.2. Fichier demo.txt

$Lyon,Bordeaux,1:Lyon,Train,Bordeaux,%

@Lyon,Paris,2:Lyon,Bateaux,Marseille,Marseille,Avion,Paris,%

$Lyon,Paris,1:Lyon,Auto,Paris,%

II. Conclusion

II.1. Problèmes marquants

Nous avons plusieurs fois changé le format du fichier, nous avons d'abord pensé à commencer chaque ligne par un S ou un C pour différencier les trajets simples et complexes, cependant, nous avons eu des bugs puisque ce serait plus difficile de différencier ces lettres de celles qui constituent une ville. Nous avons donc jugé que ce serait plus facile d'utiliser des symbole puisqu'une ville n'en contient pas.

Nous avons finalement opté pour un $ pour les trajets simples puisque ce symbole ressemble au S, et @ pour les trajets composés, nous aurions voulu utiliser un € pour les trajets composés mais ce n'est pas un caractère. Nous avons conscience que ça aurait été plus claire d'utiliser des chaines de caractères pour différencier les trajets, comme par exemple trajet_simple, trajet_composé, mais l'utilisation d'un seul caractère était beaucoup plus simple à coder.

Nous n'avons pas eu de gros problème, mais nous avons passé un peu plus de temps sur la méthode chargerVoyage qui permet d'insérer dans le catalogue des trajets en fonctions des villes de départ et d'arrivée. Finalement, nous avons simplement dû remplacer un sauter par un getline pour régler le problème.

II.2. Axes d'évolution

Nous avons réussi à produire un programme fonctionnel qui réponde aux demandes du cahier des charges. Cependant, certaines fonctions, bien que non obligatoires, le rendrait peut-être plus intéressant.

Nous n'avons par exemple pas intégré de méthode qui vérifie l'unicité d'un trajet, ainsi, si l'utilisateur rentre deux fois le même trajet, il apparaitra bien deux fois dans le catalogue. Nous avons jugé que cette méthode reste secondaire, et que le programme serait plus rapide sans.