Compte Rendu Tp C++

Gestion entrée sortie

A) Introduction

Après avoir travaillé principalement de façon orienté objet et manipulé les pointeurs lors du premier Tp, nous allons ici détailler le travail que nous avons fournis à gérer les flux d'entrées et sorties de notre classe,. Ceci à été réaliser au travers d'une gestion de fichier de sauvegarde simple, ce qui permettra d'avoir une durée de vie du catalogue plus élevée que celle de la durée du programme.

B) Description détaillée du format du fichier

Idée du format général du Fichier

Etant donné que nous avons deux types d'objets, nous devons trouver un moyen de représenter ces deux différents objets de façon simple et facile d'accès. Pour cela nous avons décidé de créer une méthode de sauvegarde dans chaque classe.

Exemple du contenu du fichier :

```
s|Lyon|Bordeau|Train
!
c|2
s|Lyon|Marseille|Bateau
s|Marseille|Paris|Avion
!
s|Lyon|Paris|Avion
!
```

Trajet Simple

Exemple de ligne : *s*|*Lyon*|*Bordeau*|*Train*

Le format est le suivant :
s|VILLE_DEPART|VILLE_ARRIVE|MOYEN_TRANPORT

Cette représentation très simple nous permet d'obtenir toutes les informations nécessaires à propos du trajet en une ligne. Nous voyons que nous utilisons le séparateurs '|'. Cette façon de séparer les différents attributs de l'objet nous permet d'utiliser facilement la fonction générique getline() qui admet un paramètre permettant de spécifier un séparateur.

Trajet Composé

Un trajet composé est d'abords annoncé avec la ligne :

c|NOMBRE_TRAJET

NOMBRE_TRAJET étant le nombre de trajets contenus dans le trajet composé. On écrit ensuite ligne par ligne le détail de tous les trajets contenus dans le trajet composé en utilisant le format du trajet simple.
exemple :

```
c|2
s|Lyon|Marseille|Bateau
s|Marseille|Paris|Avion
```

Rôle du '!'

Celui-ci à un objectif très simple, il nous permet de distinguer chaque trajet global de notre fichier. Il est particulièrement utile lorsque nous devons filtrer par type de trajet. Cela nous permet de sauter facilement un trajet composé sans se soucier de compter le nombre de trajet. De plus, cela permet d'avoir un fichier de sauvegarde assez clair pour la lecture humaine.

C) Implémentation

Après s'être décidé du format du fichier, nous avons dû commencer à réfléchir à la façon dont nous allions implémenter le fichier.

Nous avons procédé à quelques ajout de méthodes dans les classes trajets. Tout d'abords nous avons doté Trajet de la méthode abstraite Save(). Cette dernière renvoie un string sous le format de sauvegarde du fichier. Il est contient une ligne pour les trajets simple et plusieurs lignes pour les trajets composés. De plus, nous avons rajouté une méthode getType pour savoir si nous enregistrons un trajet simple ou composé. Cela nous évite de devoir faire des dynamiques cast et de gérer les erreurs renvoyées pour connaître le type de trajet.

Pour que notre fichier main reste clair et ne soit pas trop rempli, nous avons choisi de créer quatres nouveaux fichier :

loadFuctions.h / loadFunctions.cpp

saveFunctions.h / saveFunctions.cpp

qui contiennent respectivement les fonctions nécessaires au chargement et à la sauvegarde des fichiers. Pour les fonctions de chargement, on trouve une fonction principale, loadFromFile, qui permet d'afficher le menu de chargement. Cette fonction appelle une des trois autres fonctions contenu dans le même fichier. Dans le cas du chargement de tous les trajets et du chargement d'un intervalle on utilise la même fonction loadAll mais sans spécifier d'intervalle particulier dans le cas du chargement de tout le fichier. On utilise les fonctions loadByTrajectType et loadByCityName pour charger les trajets du fichier selon le type ou selon la ville de départ et/ou arrivée. Les fonctions pour sauvegarder fonctionnent de la même manière avec la fonction menu saveToFile et les trois fonctions saveAll, saveByTrajectType et saveByCityName.

D) Conclusion

Problèmes rencontrés

- Il y a eu au tout début l'adaptation à l'application développée. Il a fallu comprendre celle-ci. Cette compréhension est nécessaire pour pouvoir améliorer le programme.
- Par la suite nous avons constaté que nous ne pouvions pas avoir des noms de villes avec des espaces. Pour résoudre ce problème, nous avons utilisé getline() à la place des " cin >> ".
- Étant donné que nous n'avons pas utlisé les strings lors du premier TP, nous avons dû nous familiariser avec ces derniers ainsi que leur méthode telle que std::string::c_str();

Axes d'évolution

- Nous pourrions ajouter des trajets composés de trajets composés dans le fichier.
- Nous pourrions gérer aussi les permissions des fichiers, pour empêcher l'écriture sur les fichier sauvegarder.
- Peut être serait-il judicieux d'empêcher la modification en chiffrant les données écrites, ou pour de la ajouter de la confidentialité.
- Faire une vérification avant écriture pour savoir si nous pouvons bien écrire dans le fichier, par exemple si nous n'avons pas les droits d'écriture sur ce fichier.