TP C++ n°3 : Gestion des entrées / sorties

Sauvegarde et restitution d'un catalogue de trajets

1. Introduction

L'objectif de ce TP est de vous familiariser avec l'utilisation des classes de gestion des entrées / sorties. Il s'agit d'utiliser des fichiers en écriture et en lecture pour **sauvegarder et recharger des collections d'objets hétérogènes**. On reprendra pour cela les classes manipulant les trajets vues lors du TP C++ n° 2.
Ce troisième TP d'initiation au langage C++ continue à mettre en œuvre les principes de développement d'un logiciel avec la notion d'interface et de réalisation de classe (avec application du Guide de Style C++). Il vous permettra également d'**affiner l'écriture de votre fichier** makefile (**paramétrage soigné** à l'aide de **variables** et **intégration d'un *pattern*** pour accroître la généricité du *makefile*, avec une gestion explicite des dépendances de fichiers).
Ce TP s'appuie sur tous les concepts abordés dans le cadre des modules IF-3-POO1 et IF-3-POO2 (sauf généricité et STL)

2. Problème

Le menu de l'application réalisée durant ce TP3 devra proposer, en plus des possibilités déjà existantes (saisie au clavier et ajout, affichage et recherche de trajets), le **chargement d'un fichier** constitué de trajets (en principe produit par sauvegarde au préalable). Ces trajets sont simples ou bien composés. Les trajets lus dans le fichier (lecture sélective possible) seront stockés dans la représentation interne que vous avez implémentée lors du TP2 et complèteront le catalogue existant.
À tout moment, à partir de votre menu, l'utilisateur devra pouvoir **sauvegarder le catalogue courant dans un fichier** (écriture sélective de trajets possible). Les critères de sélection à la lecture et/ou à l'écriture seront développés dans le cahier des charges. On pourra, par exemple, proposer la sauvegarde des seuls trajets simples du catalogue ou bien encore proposer la sauvegarde des trajets (simples ou composés) dont la ville de départ est choisie par l'utilisateur… La figure 1 (ci-dessous) résume simplement ce fonctionnement.

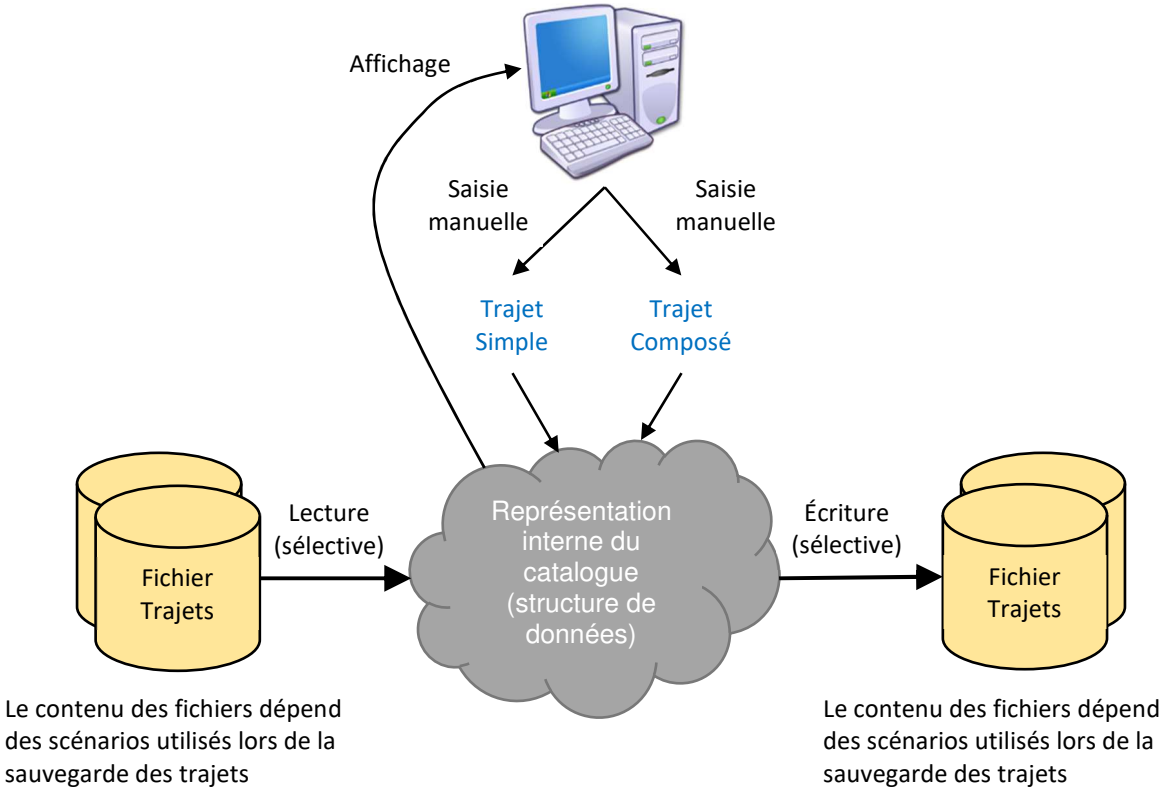


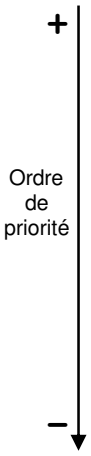
Figure 1. : Principe général de la sauvegarde et de la restitution de trajets

3. Cahier des charges – Définition du problème

Votre application, conçue lors du TP C++ n° 2, devra être complétée pour intégrer de nouvelles fonctionnalités :

1. **Le chargement d'un catalogue de trajets (lecture d'un fichier)**, c'est-à-dire l'ajout de trajets simples ou composés, au catalogue existant, à partir d'un fichier de trajets (choix possible du fichier en entrée). Ce chargement pourra être sélectif.
2. **La sauvegarde du catalogue courant (écriture dans un fichier)** : à tout instant, il faudra être en mesure de sauvegarder dans un fichier, le catalogue courant (trajets simples ou composés). Cette sauvegarde pourra être sélective.

La sauvegarde ou la restitution de trajets devra proposer plusieurs scénarios dont l'objectif est de sélectionner les trajets à sauvegarder dans un fichier ou à charger dans le catalogue courant :



1. **Sans critère de sélection** : dans le cas de la sauvegarde, tous les trajets (simples ou composés) du catalogue courant devront être sauvés dans un fichier. C'est une sauvegarde totale des objets hétérogènes du catalogue courant dans un fichier. Pour le chargement, il faudra partir d'un fichier existant (sauvegarde au préalable) et charger la totalité (pas de critère de sélection) de son contenu (trajets simples et composés) dans le catalogue courant ;
2. **Selon le type des trajets** : dans ce scénario, la sauvegarde et la restitution des trajets dépendront du type de trajet. Selon le choix de l'utilisateur (simple ou composé), il ne faudra retenir que des trajets simples ou des trajets composés dans les opérations de sauvegarde ou de restitution ;
3. **Selon la ville de départ et / ou la ville d'arrivée** : dans ce scénario, la sélection des trajets à sauver ou à restituer se fait selon la ville de départ et / ou la ville d'arrivée. Toutes les configurations devront être envisagées ;
4. **Selon une sélection de trajets** : dans ce dernier scénario, la sauvegarde ou la restitution de trajets s'appuiera sur une notion d'intervalle de trajets `[n,m]`. L'indice `n` de cet intervalle indique le premier trajet que l'utilisateur souhaite sauver ou restituer et l'indice `m` de l'intervalle correspond au dernier trajet (les 2 bornes sont toujours dans l'intervalle). La valeur `m – n + 1 ≥ 1` définit le nombre de trajets à sauver ou à restituer.

Cette description des scénarios **reste volontairement floue**. Par conséquent, il faudra complètement spécifier ces scénarios pour être capable de trouver **le meilleur format de fichier pour votre fichier de sauvegarde / restitution**. C'est **le premier travail à réaliser** dans le cadre de ce TP3 et il est déterminant pour la suite. Il faut définir avec beaucoup de soin ce format de fichier notamment pour faciliter la relecture des trajets sauvegardés. Votre **réflexion sur le format du fichier** de sauvegarde / restitution de trajets devra impérativement **prendre en compte tous les scénarios** alors que **la réalisation appliquera l'ordre de priorité tel que défini dans le sujet**. Enfin, **la gestion des fichiers** (noms des fichiers, existence…) est également volontairement floue et devra être parfaitement spécifiée.

Note : Dans tous les cas de figure, lors d'un chargement (quel que soit le scénario), les nouveaux trajets **seront rajoutés** aux trajets déjà existants dans le catalogue courant.

4. Réalisation

Par rapport aux deux TP C++ précédents, l'objectif est de maîtriser la gestion des flux d'entrées / sorties, mais aussi de consolider les connaissances en algorithmie et programmation de base en C++. La gestion de la mémoire, ayant été validée précédemment, ne devrait pas poser de problème.

Attention : il n'y a plus de contraintes particulières de réalisation dans ce TP.

Tout ce qui a été déjà vu dans les 2 cours IF-3-POO1 et IF-3-POO2 est utilisable pour réaliser ce TP. Cependant, **il ne s'agit pas de reprendre l'existant et de l'améliorer** (intégration de la surcharge d'opérateurs et/ou de la généricité et/ou de la STL…) **mais de consacrer la totalité du temps du TP à la gestion des flux d'entrées et de sorties et au nouveau cahier des charges** ! Les entrées / sorties déjà présentes dans le TP2 seront améliorées et vérifiées (notamment l'état des flux). Si nécessaire, **l'usage de la classe `string` est autorisé**.

Note :

Comme pour les premiers TP C++, la bonne gestion de la mémoire dynamique sera validée par l'outil **`valgrind`**. (cf. TP IF-3-OP et IF-3-POO1). Au fur et à mesure de l'avancée de votre réalisation, il faudra s'assurer de l'absence de fuite de mémoire et de la bonne gestion des pointeurs.

5. Récupération des informations

Comme pour les TP précédents, il faut utiliser les squelettes de classe présentés et utilisés lors des cours. Si nécessaire, l'accès aux squelettes est possible sur Moodle :
Accueil /Mes cours/Informatique /Informatique /IF-3 /IF-3-POO2 /TP3 de C++ (TP1 de POO2) /SQUELETTES