INF1010
Programmation orientée-objet

Travail pratique 2
Vecteur, surcharge d'opérateur, méthodes constantes, passage de paramètres et constructeur de copie

Département de génie informatique et logiciel

Polytechnique Montréal
29 janvier 2020

POLYTECHNIQUE MONTRÉAL

Objectifs

Permettre à l'étudiant de se familiariser avec la surcharge d'opérateurs, les vecteurs de la librairie STL, les méthodes constantes, le passage par paramètres, les constructeurs de copie et l'utilisation du pointeur ***this***.

Remise du travail

- Date :**11 février 2020 à 23h55**
- **Une note de 0 sera attribuée aux équipes qui remettent leur travail en retard.**
- Remettre uniquement **tous** les fichiers .cpp et .h sous une archive **.zip**
- Nom du fichier zip : **matricule1_matricule2_groupe.zip** où matricule1 < matricule2. Exemple : 1234566_1234567_1.zip

Références

- Notes de cours sur Moodle
- Livre Big C++ deuxième édition
- https://en.cppreference.com/w/

Directives

- Les travaux s'effectuent obligatoirement en équipe de deux personnes faisant partie du même groupe.
- Les entêtes de fichiers sont obligatoires.
- Les fonctions doivent être documentées.
- **Le guide de codage sur Moodle doit être suivi.**

Conseils

- Lisez tout le document avant de commencer!
- Ayez lu vos notes de cours!

Faites attention si vous partagez votre travail, car le plagiat sera pénalisé par **une note de 0**.

Venez poser vos questions sur Slack! Vous devez utiliser votre adresse @polymtl.ca.
Lien pour faire son compte : https://join.slack.com/t/inf1010-h20/signup
Lien du workspace : https://inf1010-h20.slack.com

Informations préalables

Le travail consiste à continuer l'application pour le programme de CinéPoly commencée au TP précédent en y intégrant les notions de vecteur, de surcharge d'opérateur, de méthodes constantes, de passage de paramètres et de constructeur de copie

Pour remplacer les tableaux dynamiques qui rendaient la gestion des dépenses et des utilisateurs difficile, ce TP fait appel aux vecteurs de la librairie STL, soit *std::vector*. Par ailleurs, pour faciliter les interactions avec les différents objets, la surcharge d'opérateurs sera utilisée.
Les vecteurs implémentés en C++ (STL) sont très pratiques : ce sont des tableaux dont la taille s'adapte dynamiquement. On peut y ajouter des éléments sans se préoccuper de la taille de notre vecteur étant donné que la gestion de la mémoire est automatique. Lorsqu'il vous est demandé d'utiliser un vecteur de la STL plutôt qu'un tableau dynamique pensez à **bien supprimer tous les attributs qui n'ont plus lieu d'être et d'adapter les méthodes qui ont besoin d'être mises à jour**.

Le langage C++ est un langage avec lequel il est possible de redéfinir la manière dont fonctionnent la plupart des opérateurs (arithmétiques (+, -, *, /), d'affectation, etc..) pour de nouvelles classes. Nous pouvons donc redéfinir le comportement de ces opérateurs afin qu'ils effectuent une nouvelle opération ou englobent plusieurs opérations pour ces nouvelles classes.

Pour vous aider, la solution du TP précédent est fournie. Vous n'avez qu'à implémenter les nouvelles méthodes décrites plus bas et supprimer les attributs qui n'ont plus lieu d'être. Les méthodes à modifier vous ont été indiquées.

ATTENTION : Tout au long du TP, assurez-vous d'utiliser les opérateurs sur les objets et non sur leurs pointeurs ! Vous devez donc déréférencer les pointeurs si nécessaire.

ATTENTION : Vous serez pénalisés pour les utilisations inutiles du mot-clé *this*. Utilisez-le seulement où nécessaire.

ATTENTION : Il est fortement conseillé d'utiliser les fichiers fournis, plutôt que de continuer avec vos fichiers du TP1.

Remarque : Pour plus de précision sur le travail à faire et les changements à effectuer, veuillez-vous référer aux fichiers .h

Travail à réaliser

On vous demande de compléter les fichiers qui vous sont fournis pour pouvoir implémenter le système décrit ci-dessous.

Classe Auteur

Cette classe représente un auteur.

La méthode suivante doit être retirée :

- La méthode `afficher` doit être retirée car elle sera remplacée par l'opérateur <<

Les méthodes suivantes doivent être implémentées :

- L'opérateur << doit être implémenté pour remplacer la méthode `afficher`. **Vous devez permettre l'opération en cascade.**
- Opérateur == qui compare un `Auteur` avec un string. Par exemple, auteur == "Thomas". **L'opération doit pouvoir se faire dans les deux sens**. C'est-à-dire : auteur == "Thomas" et "Thomas" == auteur doivent tous les deux fonctionner. Cet opérateur doit retourner `true` lorsque le nom de l'auteur est égal au string.

Classe Film

Cette classe représente un film. Un film peut être inaccessible dans certains pays et peut aussi être restreint à un âge minimum**. Vous devez modifier les attributs afin de remplacer le tableau par un vecteur. Retirez tous les attributs que vous jugez dorénavant inutiles avec le remplacement du tableau par un vecteur. Il n'est plus nécessaire de faire de l'allocation dynamique pour le tableau de pays.**

La méthode suivante doit être retirée :

- La méthode `afficher` doit être retirée car elle sera remplacée par l'opérateur <<

Les méthodes suivantes doivent être modifiées :

- La méthode `ajouterPaysRestreint` doit être modifiée pour être adaptée au vecteur.

- La méthode `supprimerPaysRestreints` doit être modifiée pour être adaptée au vecteur.
- La méthode `estRestreintDansPays` doit être adaptée au vecteur.

La méthode suivante doit être implémenté :

- L'opérateur << doit être implémenté pour remplacer la méthode `afficher`. **Vous devez permettre l'opération en cascade.**

Classe GestionnaireAuteurs

Cette classe gère les auteurs. **Vous devez modifier les attributs afin de remplacer le tableau par un vecteur. Retirez tous les attributs que vous jugez dorénavant inutiles avec le remplacement du tableau par un vecteur**. Plusieurs méthodes doivent être changées à cause des attributs supprimés.

Les méthodes suivantes doivent être supprimées :

- La méthode `afficher` doit être retirée car elle sera remplacée par l'opérateur <<
- La méthode `ajouterAuteur` doit être retirée car elle sera remplacée par l'opérateur +=

Les méthodes suivantes doivent être implémentées :

- L'opérateur << doit être implémenté pour remplacer la méthode `afficher`. Attention, la classe `Auteur` ne possède plus de méthode `afficher`, vous devez utiliser son opérateur <<. **Vous devez permettre l'opération en cascade.**
- Opérateur += qui permet d'ajouter un auteur dans le vecteur d'auteurs si le maximum `NB_AUTEURS_MAX` n'est pas dépassé. Retourne `true` si l'auteur a été ajouter, `false` sinon. Cette méthode remplace la méthode `ajouterAuteur`.

Les méthodes suivantes doivent être modifiées :

- La méthode `lireLigneAuteur` doit utiliser l'opérateur += pour ajouter les auteurs
- `getNbAuteurs` doit être adapté au vecteur.
- `chercherAuteur` doit être adapté au vecteur. Elle doit aussi utiliser **l'opérateur == de la classe `Auteur`.**
- `chargerDepuisFichier` doit être adaptée au vecteur, **il faut vider le vecteur avant de charger les auteurs.**

Classe Librairie

Cette classe gère les films offerts par CinéPoly. Vous devez modifier les attributs afin de remplacer le tableau par un vecteur de **std ::unique_ptr de Film**. Retirez tous les attributs que vous jugez dorénavant inutiles avec le remplacement du tableau dynamique.

Les méthodes suivantes doivent être retirées :

- `AjouterFilm` car elle sera remplacée par l'opérateur +=
- `RetirerFilm` car elle sera remplacée par l'opérateur -=
- `Afficher` car elle sera remplacée par l'opérateur <<

Les méthodes suivantes doivent être implémentées :

- `Afficher` doit être remplacé par l'opérateur << et adapté au nouveau code de la classe film (qui n'a plus de méthode afficher).
- Le constructeur par copie doit être implémenté.
- L'opérateur = qui écrase les attributs de la librairie par les attributs de la librairie passé en paramètre. **Cet opérateur doit pouvoir être appelé en cascade.**
- L'opérateur += qui doit remplacer la méthode `ajouterFilm` et être adapté au vecteur et aux pointeurs intelligents. **Cet opérateur doit pouvoir être appelé en cascade.**
- L'opérateur -= qui doit remplacer la méthode `retirerFilm` et être adapté au vecteur et aux pointeurs intelligents. **Cet opérateur doit pouvoir être appelé en cascade.**

Les méthodes suivantes doivent être modifiées :

- Le constructeur doit être adapté aux attributs retirés et à l'ajout du vecteur.
- Le destructeur doit être adapté au fait que le tableau dynamique est maintenant un vecteur.
- La méthode `ChercherFilm` doit être adaptée aux pointeurs intelligents.
- La méthode `getNbFilms` doit être adaptée au vecteur.
- La méthode `chargerRestrictionsDepuisFichiers` doit être adaptée au changement dans les attributs.
- La méthode `lireLigneFilm` doit utiliser l'opérateur +=
- La méthode `trouverIndexFilm` doit être adaptée au vecteur.
- La méthode `supprimerFilms` doit être adaptée au vecteur (le vecteur doit être vide après l'appel à cette méthode) et elle doit aussi être adaptée aux pointeurs intelligents. N.B cette méthode ne s'occupe plus de reset le nombre de film des auteurs, cette fonctionnalité a été bouger dans le destructeur de Film.

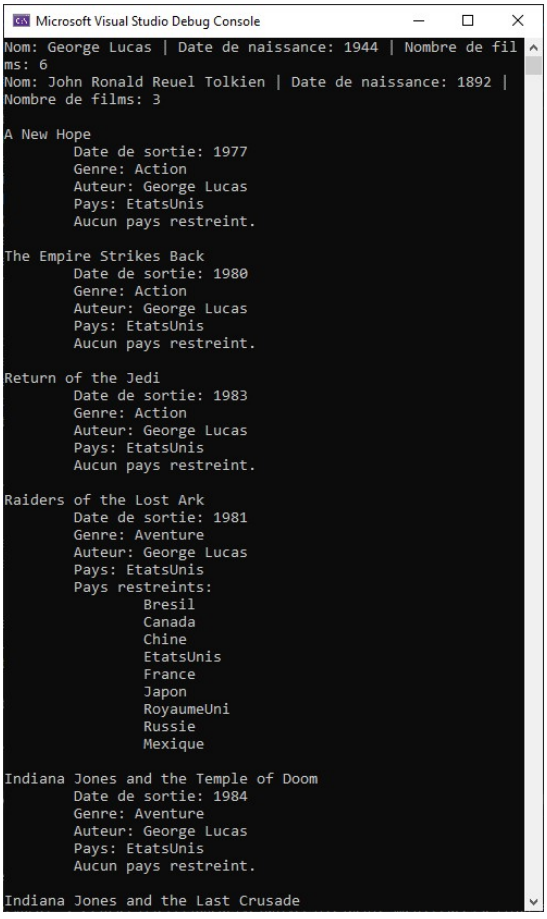
Classe Utilisateur

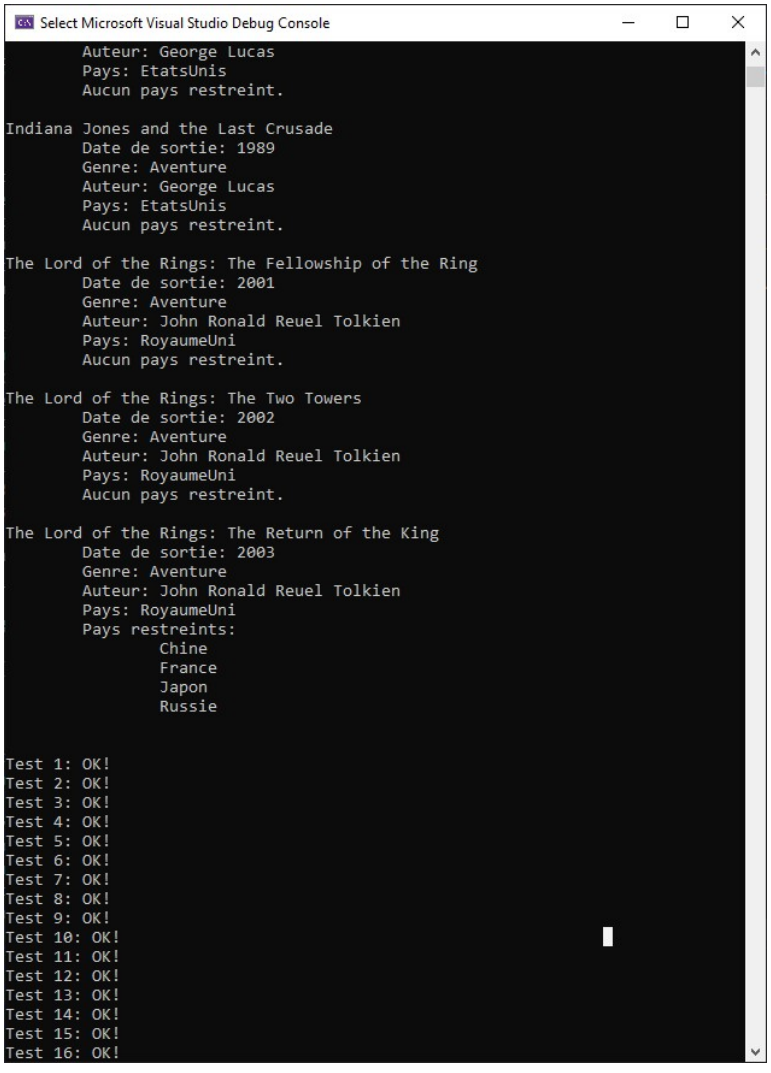
Cette classe représente un utilisateur de CinéPoly. Cette classe n'a pas de changement

main.cpp

Le fichier main.cpp vous est fourni et ne devrait pas avoir à être modifié pour la remise. Une série de tests sont fournis dans ce fichier pour vous aider à vérifier le comportement de votre programme. Vous pouvez désactiver des blocs de tests en changeant les `#ifdef true` pour des `#ifdef false` afin de compiler le programme avant d'avoir tout terminé. Si un test échoue, allez voir le but du test dans la fonction main.

Sortie attendue





Fichiers de lecture

Trois fichiers de lecture vous sont fournis. `auteurs.txt`, `films.txt` et `restrictionsPays.txt`. Ces fichiers sont utilisés pour tester votre programme et vos méthodes de lecture des fichiers. Assurez-vous de mettre les fichiers à un endroit où votre programme peut les lire.

Compilation

Sous Windows : Faites une solution Visual Studio.

Sous Linux et MacOS: Un Makefile vous est fourni. Mettez tous les fichiers `.h` sous `TP1/include`, mettez tous les fichiers `.cpp` sous `TP1/src` et mettez le Makefile directement dans `TP1`. `make -C path/to/TP1 all` compile le programme. `make -C path/to/TP1 run` roule le programme.

Fuites de mémoire

Sous Windows :

Le fichier `debogageMemoire.h` est présent pour vous aider à vérifier s'il y a des fuites de mémoire dans votre code. Exécutez la solution en débogage pour qu'il fonctionne. Si une fuite de mémoire est détectée, un message sera affiché dans la fenêtre de sortie (Output) de Visual Studio. **Compilez avec -W4.**

Sous Linux :

Utiliser Valgrind : Lancer la commande ci-dessous :

```
valgrind --tool=memcheck --leak-check=yes ./PATH_VERS_PROGRAMME
```

Pour installer valgrind : `sudo apt install valgrind`, `sudo pacman -S valgrind`, etc. Si aucune fuite n'est détectée, vous devriez voir la ligne :

```
All heap blocks were freed -- no leaks are possible.
```

Spécifications générales

- Utilisez la liste d'initialisation pour l'implémentation de vos constructeurs.
- Ajouter le mot-clé *const* chaque fois que cela est pertinent.
- Ajouter des références dans les paramètres lorsque cela est pertinent

- Documentez le code source.
- Ne remettez que les fichiers .h et .cpp lors de votre remise
- Les entêtes de fichiers sont obligatoires
- Les fonctions doivent être documentées. Suivez les exemples des fonctions déjà documentées.
- Les fonctions doivent être implémentées dans le même ordre que la définition de la classe.
- **Le guide de codage sur Moodle doit être suivi.**
- **Bien lire le barème de correction ci-dessous.**

Correction

La correction du TP1 se fait sur 20 points :

- [4 points] Compilation du programme (le programme ne doit pas avoir d'avertissements).
- [2 points] Exécution du programme.
- [2 points] Comportement exact des méthodes du programme.
- [2 points] Surcharge correcte des opérateurs.
- [2 points] Utilisation correcte des vecteurs.
- [1 points] Utilisation correcte du mot clé `const`.
- [2 points] Utilisation correcte du mot clé `this`
- [1 points] Passage de paramètres adéquat.
- [4 points] Documentation du code et bonne norme de codage.