INF1010

Programmation Orientée-Objet

Travail pratique #5
Fonctions et classes génériques, librairies STL

Objectifs : Permettre à l'étudiant de se familiariser avec les concepts de foncteur, de fonction et de classe générique et avec les librairies standards.

Remise du travail : mardi, 29 mars 2016, 8h AM

Références : Notes de cours sur Moodle & Chapitres 11 à 14, 16 et 20 du livre Big C++ 2e éd.

Documents à remettre : Les fichiers .cpp et .h complétés réunis sous la forme d'une archive au format .zip.

Directives : Directives de remise des Travaux pratiques sur Moodle

Les en-têtes (fichiers, fonctions) et les commentaires sont obligatoires.

Les travaux dirigés s'effectuent obligatoirement en équipe de deux personnes faisant partie du même groupe.

Veuillez suivre le guide de codage

Travail à réaliser

Dans ce TP nous poursuivrons l'implémentation de notre bibliothèque en y incorporant de nouveaux éléments, en améliorant la structure utilisée et en optimisant le code.

Lors des TPs précédents, il était possible de remarquer qu'il y avait énormément de redondance dans le code, les mêmes sections qui se répètent dans plusieurs méthodes. Nous pouvons maintenant appliquer les nouvelles notions vues dans le cours pour bien bénéficier des avantages des fonctions et classes génériques et de la robustesse des algorithmes de la librairie STL.

La librairie STL est comme une boite à outil qui est à la disposition d'un programmeur. Elle lui permet d'éviter à réinventer la roue et elle lui simplifie la vie. Il faut juste apprendre à utiliser ses outils, afin de les maitriser et de s'en servir d'avantage.

Grâce à ces concepts, ce TP vous permettra d'améliorer la gestion de la bibliothèque et d'offrir une solution plus optimisée. Nous allons remplacer les vecteurs par des gestionnaires qui permettent de gérer l'ajout, le retrait et la recherche des différents objets de la bibliothèque. Pour cela il nous faut trois gestionnaires, un pour les *Abonne*, un autre pour les *ObjetEmpruntable* et un dernier pour les *Emprunt*.

Vous devez implémenter les classes *Gestionnaire* et *Bibliotheque* ainsi que les foncteurs *MemeElement*, *Empruntable*, *RechercheObjetEmpruntable*, *DetruireEmprunt* et *TrieParTitre*. Vous devez aussi modifier les deux classes *Professeur* et *DVD*.

Prenez le temps de bien lire l'énoncé dans son intégralité avant de vous lancez dans le travail.

- Tout au long du TP, faites attention de toujours vérifier que vos itérateurs ne sont pas nuls et que vos conteneurs ne sont pas vides.
- L'utilisation de boucles for ou while de la forme for(int i; i <vec.size(); i++) est interdit. Vous devez utiliser les algorithmes lorsque possible, et les boucles for/while en utilisant les itérateurs.

Classes inchangées

La classe Abonne a été modifiée par rapport aux TPs précédents, vous pouvez vérifier les modifications apportées dans les fichiers fournis, vous n'avez pas à modifier cette classe.

L'opérateur == de la classe Emprunt a été surchargé pour vous, afin que vous puissiez implémenter les classes et les foncteurs génériques et afin accomplir les tâches demandées dans ce TP.

Les classes suivantes restent inchangées par rapport au TP4. : ObjetEmpruntable, ObjetNumerique, LivreNumerique, Etudiant, EtudiantBaccalauréat et Livre.

Classe *Gestionnaire*

La classe Gestionnaire est une classe générique permettant de contenir des objets de type quelconque (qu'on appellera ici type T).

- Cette classe ne devrait pas prendre plus d'une séance de TP à être implémentée.
- Elle devrait être implémentée seulement avec des algorithmes ou des méthodes du conteneur, aucune boucle n'est acceptée.
- Elle ne devrait pas donner accès direct au conteneur, c.-à-d. on n'implémente pas des méthodes obtenir et modifier.

Gestionnaire contient seulement l'attribut suivant :

- Conteneur (list) de pointeurs *d'objets*.

Gestionnaire implémente seulement les méthodes suivantes :

- Un constructeur par défaut.
- Un destructeur.
- Une méthode *ajouterElement()* qui reçoit un pointeur de l'élément en paramètre et qui permet de l'ajouter seulement s'il n'est pas déjà dans le conteneur. Si l'ajout a été fait, la méthode renvoie true, false sinon.
- Une méthode *retirerElement ()* qui reçoit un pointeur de l'élément en paramètre et qui permet de le retirer du conteneur. Si le retrait a été fait, la méthode renvoie true, false sinon.
- Une méthode générique sur le predicat, *retirerContenu()* qui prend un prédicat unaire (Predicate&) en paramètre, retire l'objet ou les objets dans le conteneur qui vérifie le prédicat et retourne true si au moins un élément est retiré, false sinon. Un prédicat unaire une fonction qui a un seul paramètre et retourne un booléen.
- Une méthode générique sur le prédicat *trouverElement()* qui prend un prédicat unaire (Predicate&) en paramètre, recherche l'objet qui vérifie le prédicat et retourne un pointeur de type T. Si aucun objet n'est retrouvé, un pointeur nul est retourné sinon.
- Une méthode *trouverElement()* qui est une surcharge de la méthode précédente, elle prend un élément de type T reçu en paramètre, recherche l'objet et retourne true si l'élément est trouvé, false sinon.
- Une méthode générique sur le prédicat, *trouverContenu()* qui prend un prédicat unaire (Predicate&) en paramètre, recherche tous les objets qui vérifient le prédicat et retourne une liste de pointeur de type T. Si aucun objet n'est retrouvé, une liste vide est retournée sinon.

Pour vous aider à tester cette classe, vous pouvez utiliser le fichier maingestionnaire.cpp.

Foncteur *MemeObjet*

Écrire un foncteur générique binaire <*T,P*> pour les algorithmes de la librairie STL qui permet de trouver un objet identique dans le conteneur.

Ce foncteur est un prédicat générique qui contient un attribut pointeur de *type P* et qui retourne *true* si l'objet de type *T* en paramètre est équivalent à l'attribut de pointeur type P.

Indice : Pour vérifier l'équivalence d'un emprunt c.-à-d. vérifier le matricule et la cote en même temps l'opérateur == est surchargé avec un paramètre de type Pair. Donc l'attribut de Type P pourra contenir une variable de type soit *Pair<string,string>, string ou le type de l'élément*.

Les opérateurs == sont déjà implémentés pour vous, tous les objets nécessaires surchargent cet opérateur, vous pouvez consulter le code fourni si nécessaire.

Foncteur *Empruntable*

Écrire un foncteur prédicat unaire pour un algorithme de la librairie STL qui permet de compter les emprunts de l'abonne et qui permet de vérifier si l'objet a été déjà emprunté.

Ce foncteur contient les attributs suivants :

- Un string matricule
- Un string cote
- Un compteur
- Un booléen

Indice : Le prédicat (Foncteur) est passé par copie dans les algorithmes STL, si vous voulez récupérer les résultats du foncteur, il faut que les attributs du foncteur pointent sur des variables externes.

Foncteur *RechercheObjetEmpruntable*

Écrire un foncteur prédicat unaire pour un algorithme de la librairie STL qui permet de chercher un string dans les objets empruntables. Il va rechercher les différents éléments empruntables dans le gestionnaire d'objet de la bibliothèque qui contiennent l'information désirée. Il retourne true si l'information a été trouvée, false sinon.

Foncteur *DetruireEmprunt*

Écrire un foncteur pour un algorithme de la librairie STL qui permet de détruire et retirer tous les emprunts de la bibliothèque qui sont contenus dans le gestionnaire d'emprunt.

Foncteur *TrieParTitre*

Ce foncteur est un prédicat binaire qui permet de comparer les titres de deux ObjetEmpruntable, il retourne true si les titres sont en ordre alphabétique, false sinon.

Classe *Professeur*

Les attributs et méthodes liées au TP4 restent inchangées, sauf si il est nécessaire pour apporter les modifications suivantes à la classe Professeur.

L'attribut suivant doit être modifié :

- *list* de *string* qui contient les écoles.

La méthode suivante doit être modifiée :

- L'opérateur << doit afficher les différentes écoles où le professeur enseigne en ordre alphabétique inverse. En premier, il s'agit de trier en ordre décroissant et ensuite, on affiche.

Classe *DVD*

Les attributs et méthodes liées au TP4 restent inchangées, sauf si il est nécessaire pour apporter les modifications suivantes à la classe *DVD*.

L'attribut suivant doit être modifié :

- *list* de *string* qui contient les acteurs.

La méthode suivante doit être modifiée :

- L'opérateur << doit afficher les acteurs du *DVD* en ordre alphabétique inverse.

Classe *Bibliotheque*

La classe *Bibliotheque* est celle qui fait le lien entre toutes les classes précédentes.

Cette classe doit être implémentée à nouveau, le fichier de définition est fourni, il ne reste qu'à compléter les méthodes demandées.

Bibliotheque contient les attributs suivants :

- Gestionnaire *d'abonnés*.
- Gestionnaire d'*ObjetEmpruntable*.
- Gestionnaire d'*Emprunt*.

Bibliotheque implémente les méthodes suivantes :

Indication : Il y a une nouvelle méthode *trierEmprunt*() à ajouter. Les méthodes demandées sont identiques à celles présentes dans le TP4, c'est **seulement leur implémentation qui devra être modifiée**. Puisque les attributs de cette classe sont des conteneurs de classe Gestionnaire, il faut penser à utiliser les méthodes de celle – çi, ainsi que les foncteurs déjà définis. Pour les boucles, il faut impératif d'utiliser les itérateurs.

- Un constructeur par défaut.
- Un destructeur (foncteur *DetruireEmprunt)*.
- Une méthode *trouverAbonne ()* qui prend un matricule (*string*) en paramètre, elle doit créer le foncteur *MemeObjet*, recherche l'abonné dans le gestionnaire et retourne un pointeur de type *Abonne*. Si aucun abonné n'est retrouvé, un pointeur *nul* est retourné sinon.
- La méthode *ajouterAbonne ()* qui permet d'ajouter l'*abonné* reçu en paramètre dans le gestionnaire approprié.
- La méthode *retirerAbonne ()* qui permet de retirer l'abonné en utilisant le matricule reçu en paramètre. La méthode doit créer le foncteur *MemeObjet*. Avant de le retirer, il faut retourner tous les objets qu'il a empruntés. La méthode retourne donc un booléen *true* s'il est retiré, sinon *false*.
- Une méthode *trouverObjetEmpruntable ()* qui prend une cote (*string*) en paramètre, elle doit créer le foncteur *MemeObjet*, recherche l'objet empruntable dans le gestionnaire et retourne un pointeur de type *ObjetEmpruntable*. Si aucun objet empruntable n'est retrouvé, un pointeur *nul* est retourné sinon.
- La méthode *ajouterObjetEmpruntable ()* qui permet d'ajouter l'*ObjetEmpruntable* reçu en paramètre dans le gestionnaire approprié.
- La méthode *retirerObjetEmpruntable()* qui permet de retirer l'objet empruntable en utilisant la cote (string) reçue en paramètre, elle doit créer le foncteur *MemeObjet*. L'objet est retiré seulement s'il n'est pas emprunté. La méthode retourne donc un booléen *true* s'il est retiré, sinon *false*.
- La méthode *rechercherObjetEmpruntable()* qui prend en paramètre une chaîne de caractères (*string*), elle doit créer les foncteurs *RechercheObjetEmpruntable* et *TrieParTitre* .
    - Cette méthode va rechercher les différents éléments empruntables de la bibliothèque qui contiennent l'information désirée en **utilisant la méthode appropriée de la classe gestionnaire**.

    - La méthode doit trier les objets par ordre alphabétique selon leur titre à l'aide du foncteur et de l'algorithme de tri avant de les afficher.
    - Pour chaque élément trouvé, les informations seront affichées à l'aide de la fonction opérator <<.
    - Si rien n'a été trouvé, affichez un message.
- La méthode *emprunter()* qui prend en paramètres le matricule d'un abonné, la cote d'un objet empruntable et la date de retour associée. Cette méthode doit retourner une valeur booléenne indiquant si l'emprunt a été fait ou non.
    - Elle doit s'assurer que le nombre d'emprunts ne dépasse pas la limite par type d'abonné.
    - Elle doit diminuer le nombre d'objets associés disponibles.
    - Elle doit utiliser le foncteur Empruntable .
- La méthode *retourner()* qui prend en paramètres le matricule d'un abonné, la cote d'un objet empruntable. Cette méthode doit retourner une valeur booléenne indiquant si l'emprunt a été retourné ou non.
    - Elle doit utiliser le foncteur *MemeObjet* pour trouver l'emprunt.
    - Elle doit augmenter le nombre d'objets associés disponibles.
    - Elle doit détruire l'objet de la mémoire après son retrait du gestionnaire.
- La méthode *trierEmprunt ()* qui prend en paramètre un pointeur d'abonné et retourne une *map*, cette méthode doit chercher et trier les emprunts en ordre alphabétique selon les titres des objets empruntés. Elle utilise le foncteur *MemeObjet.* Cette méthode est appelée par la méthode infoAbonne().
- La méthode *infoAbonne()* qui prend en paramètre un matricule d'abonné et affiche les informations qui le concerne en utilisant l'opérateur << approprié.
    - Cette méthode doit déterminer de quelle sous-classe fait partie l'abonné (typeid).
    - Une fois déterminé, la méthode doit faire un *dynamic_cast*<> pour obtenir un pointeur de la classe appropriée, dans le but d'appeler l'opérateur <<.
    - Finalement, la méthode doit afficher les emprunts retournés par la méthode *trierEmprunt().*
- Les opérateurs += , -=, >> restent inchangés, ils sont identiques au TP précédent.

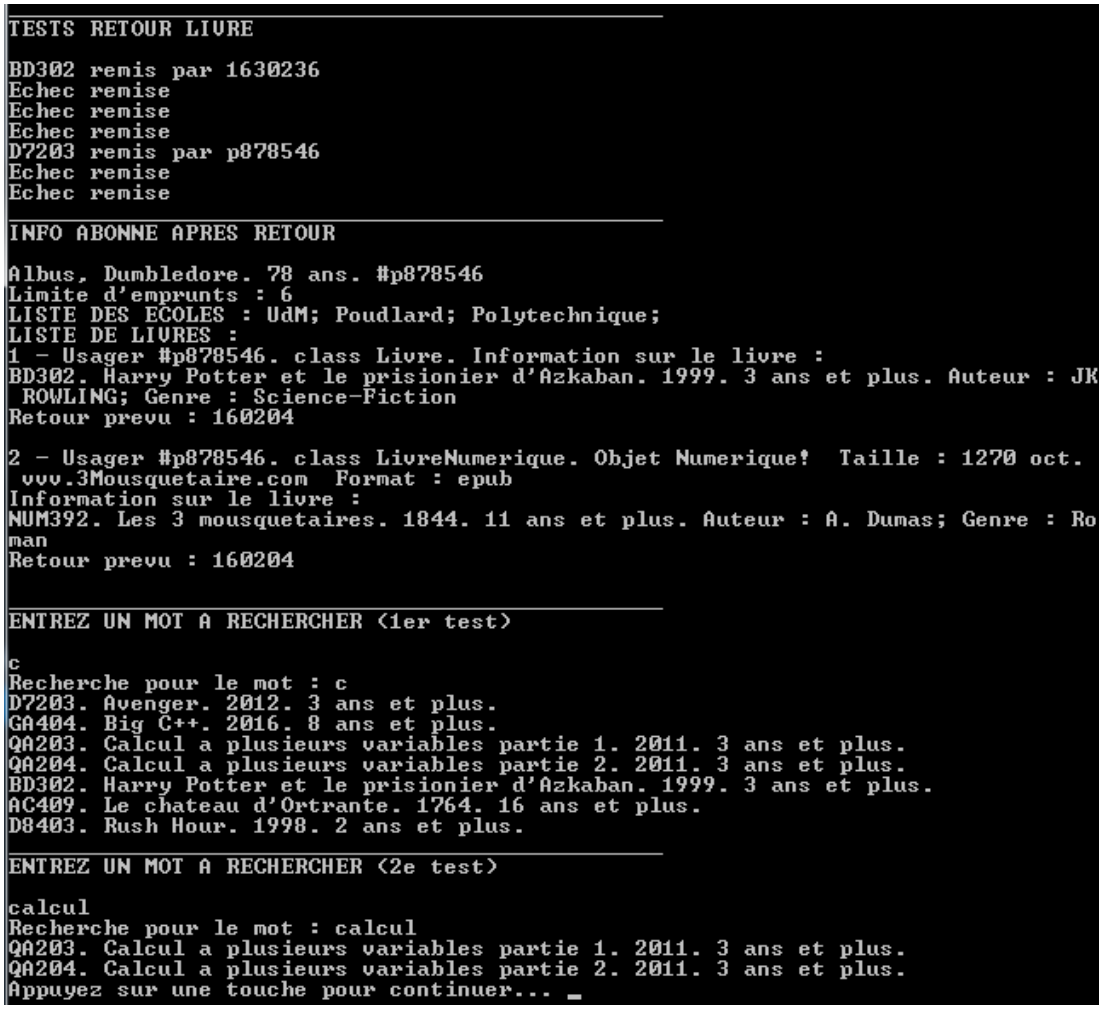
Main.cpp

Le programme principal contient des directives à suivre pour tester les différentes méthodes que vous devez implémenter dans les différentes classes. Vous n'avez pas à compléter le main, assurez-vous juste du bon fonctionnement des méthodes appelées.

Le résultat final devrait être similaire à ce qui suit :

```
_____________________________________________
CREATION DES ABONNES ET DES OBJETS EMPRUNTABLES

Albus, Dumbledore. 78 ans. #p878546
Limite d'emprunts : 6
LISTE DES ECOLES : UdM; Poudlard; Polytechnique;
Information sur le Dvd :
D8403. Rush Hour. 1998. 2 ans et plus. Realisateur : Al; Acteurs : Jackie Chan;
Chris Tucker;
Information sur le Dvd :
D7203. Avenger. 2012. 3 ans et plus. Realisateur : Nick Fury; Acteurs : Thor; Ir
on Man; Hulk; Captain America; Black Window;
_____________________________________________
AJOUT DES DIFFERENTS ELEMENTS A LA BIBLIOTHEQUE

Echec d'ajouter tony, car il existe deja
tony est retire
tony est ajoute a nouveau
Echec d'ajouter Le chateau d'Ortrante, car il existe deja
Le chateau d'Ortrante est retirÚ
Le chateau d'Ortrante est ajoutÚ a nouveau
_____________________________________________
CREATION DES OBJETS NUMERIQUES

Objet Numerique!  Taille : 1270 oct.  vvv.3Mousquetaire.com  Format : epub
Information sur le livre :
NUM392. Les 3 mousquetaires. 1844. 11 ans et plus. Auteur : A. Dumas; Genre : Ro
man

Objet Numerique!  Taille : 1270 oct.  vvv.3Mousquetaire.com  Format : epub
Information sur le livre :
NUM393. Les 3 mousquetaires 2. 1846. 11 ans et plus. Auteur : A. Dumas; Genre :
Roman
_____________________________________________
TESTS D'EMPRUNTS

1630236 a reussi 2 emprunt(s) !
1630236 a rate 5 emprunt(s) !
1782965 a reussi 4 emprunt(s) !
1782965 a rate 3 emprunt(s) !
1865487 a reussi 5 emprunt(s) !
1865487 a rate 2 emprunt(s) !
p878546 a reussi 3 emprunt(s) !
p878546 a rate 4 emprunt(s) !
```

```
_____________________________________________
INFO ABONNE AVANT RETOUR

Albus, Dumbledore. 78 ans. #p878546
Limite d'emprunts : 6
LISTE DES ECOLES : UdM; Poudlard; Polytechnique;
LISTE DE LIVRES :
1 - Usager #p878546. class Dvd. Information sur le Dvd :
D7203. Avenger. 2012. 3 ans et plus. Realisateur : Nick Fury; Acteurs : Thor; Ir
on Man; Hulk; Captain America; Black Window;
Retour prevu : 160204

2 - Usager #p878546. class Livre. Information sur le livre :
BD302. Harry Potter et le prisionier d'Azkaban. 1999. 3 ans et plus. Auteur : JK
 ROWLING; Genre : Science-Fiction
Retour prevu : 160204

3 - Usager #p878546. class LivreNumerique. Objet Numerique!  Taille : 1270 oct.
 vvv.3Mousquetaire.com  Format : epub
Information sur le livre :
NUM392. Les 3 mousquetaires. 1844. 11 ans et plus. Auteur : A. Dumas; Genre : Ro
man
Retour prevu : 160204
```



Correction

La correction du TP5 se fera sur 20 points. Voici les détails de la correction:

- (10 points) Compilation et exécution exacte du programme.
- (3 points) Utilisation exacte des conteneurs et des boucles.
- (3 points) Utilisation adéquate des algorithmes et des foncteurs.
- (2 points) Utilisation adéquate de la classe générique et du foncteur générique.
- (2 points) Documentation du code et norme de codage.