Département de génie informatique et logiciel

INF1010 – Programmation orientée-objet
TP2

Automne 2022

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Remise du travail

- Date : **6 Octobre 2022 avant 23h30**
- **Une note de 0 sera attribuée aux équipes qui remettent leur travail en retard.**
- Remettre **tous** les fichiers directement sous une archive **.zip** (sans dossier à l'intérieur et sans autres fichiers).
- ✂**Toute archive autre que zip ou remise ne respectant pas ces consignes sera refusée et se méritera une note de 0. Attention, une archive .7z ne compte pas comme une archive zip.** ✂
- Respectez le format de la remise! Nom de l'archive zip : **`matricule1_matricule2_groupe.zip`**
  Exemple : 1234567_1234568_1.zip
- Vous devez uniquement remettre les fichiers .cpp

Références

- Notes de cours sur Moodle et le livre Big C++

Directives

- Les travaux s'effectuent obligatoirement en équipe de deux personnes faisant partie du même groupe.
- Les fonctions que vous décidez d'ajouter au programme doivent être documentées.

Conseils

- Si vous avez des difficultés au cours du TP, rappelez-vous que de nombreux problèmes demandés sont assez couramment rencontrés en C++. Consulter la documentation sur cppreference, les notes de cours, et les forums sur Google peuvent vous donner de bonnes pistes de résolution!
- Veuillez utiliser la version v17 de C++.

✂Tout cas de plagiat sera automatiquement reporté au Comité d'examen de fraude (CEF) ✂

Spécifications générales

✂ Le non-respect des spécifications générales entraînera des pénalités. ✂

- Vous pouvez utiliser n'importe quel environnement autant que le compilateur soit C++17.
  - Des ressources sont disponibles pour vous aider à débuter avec votre environnement:
    - Visual studio: https://youtu.be/BqLw7RDnNew
    - Visual studio code: https://youtu.be/d5jy7YHLX0w
    - Xcode: https://youtu.be/oTeotPuDPbY

- **L'inclusion des fichiers d'en-tête (`.h`) doit être sensible aux majuscules et minuscules**. Malheureusement, sur Windows les fichiers ne sont pas sensibles aux majuscules et minuscule, donc des erreurs d'inclusions pourraient passer inaperçues. Relisez-vous bien.
- Toutes les méthodes doivent être définies dans le fichier d'implémentation (`.cpp`) **dans le même ordre** que leur déclaration dans le fichier d'en-tête (`.h`). Le non-respect de cette règle entraînera une pénalité au niveau du style.
- Utilisez le plus possible la liste d'initialisation des constructeurs. L'utilisation du corps des constructeurs à la place de la liste d'initialisation entraînera une pénalité de style. L'ordre des variables (attributs) dans la liste d'initialisation doit être la même que celle dans la liste des attributs de la classe dans le fichier d'en-tête.
- Suivez le guide de codage sur Moodle.
- Modifications/ajouts au guide de codage à respecter :
  - Vous pouvez utiliser le style d'accolades que vous désirez, **tant que vous êtes uniformes**. Sachez cependant qu'il est généralement attendu d'un(e) développeur(se) suivre la convention établie par le projet, ou la convention établie par la langue (si elle existe) en créant un nouveau projet. Dans le cas du TP fourni, les accolades ouvrantes sont sur leur propre ligne (style Allman).
  - Mettez un espace avant la parenthèse ouvrante des énoncés de contrôle comme if, for, while et switch, mais pas avant les parenthèses d'un appel de fonction. Ne mettez jamais d'espace tout juste après une parenthèse ouvrante ou tout juste avant une parenthèse fermante.
  - Le deux-points (`:`) et le signe égal (`=`) devraient être entourés d'espaces, sauf dans le cas des étiquettes de `case` et les spécificateurs d'accès (`private`, `protected` et `public`) qui ne devraient pas avoir d'espace avant le deux-points. Un espace devrait toujours suivre les éléments de ponctuation comme la virgule (`,`), le deux points (`:`) et le point-virgule (`;`).
  - N'utilisez pas `NULL` ou `0` pour les pointeurs, mais bien `nullptr`. Dans le cas de test de `nullptr` avec un pointeur, soyez explicite : préférez `if (p != nullptr)` à `if (p)`.

- N'utilisez pas de `else` après un `return`.

Mise en contexte

Maintenant que vous avez fait l'implémentation des différentes classes lors du dernier TP, vous devez désormais vous familiariser avec la classe conteneur vector de la STL, la surcharge des opérateurs, des constructeurs copie et les fonctions globales amies. Dans ce TP, vous aurez donc à faire l'implémentation des surcharges de manière à faire passer tous les tests.

Aperçu des classes du projet

- ***Date***: Classe qui représente une date ;
- ***Joueur*** : Classe qui représente un joueur d'une équipe;
- ***Équipe*** : Classe qui représente l'ensemble des joueurs de l'équipe.

Travail à réaliser

Implémenter les classes en complétant les méthodes manquantes des fichiers .cpp fournis.

Classe Date

La classe représente une date ayant les attributs suivants : Année, Mois et jour.

Il faut implémenter les méthodes suivantes :

- Constructeur de copie.
- Opérateur < . Compare deux dates en fonction des années, mois et ensuite des jours.
- Operateur + . Additionne des jours à la date courante. Il faut ajuster le mois et l'année en conséquence.
- Operateur+= Additionne des jours à la date courante. Il faut ajuster le mois et l'année en conséquence.
- Operateur++. L'opérateur préfix additionne un jour à la date courante. Il faut ajuster le mois et l'année en conséquence.

La fonction globale amie : operator<<.

Classe Joueur

Classe qui représente un joueur d'une équipe.

Cette classe contient les attributs suivants : le nom du joueur, le nombre de matchs, le nombre de buts marqués, et le nombre d'assists.

Il faut implémenter

- la méthode suivante : operator < compare le nombre de buts ou le nombre d'assists.

- la fonction globale amie : operator<<

Classe Équipe

Classe qui représente l'ensemble des joueurs de l'équipe.

Cette classe contient les attributs suivants :

- Un nom (string).
- Un nombre de victoire (entier).
- Un tableau vector de share pointeurs à des joueurs.
- La date de création de l'équipe.

Il faut implémenter toutes les méthodes de la classe en particulier

- Constructeur de copie.
- La surcharge de l'opérator =.
- getNombresJoueurs() retourne le nombre de joueurs dans l'équipe.
- getJoueur() retourne le joueur dont le nom est passé en paramètre. Retourne un null pointer si le joueur n'existe pas dans l'équipe.
- operator +() qui ajoute un joueur dans le tableau vector intelligent si le joueur n'existe pas.
- operator-() qui supprime un joueur s'il existe dans l'équipe. Si le joueur n'existe pas, on ne fait rien.
- operator++(). L'opérateur préfix incrémente le nombre de victoires de l'équipe.

La fonction globale amie operator << qui affiche le nom de l'équipe, les joueurs et le nombre de victoires.

main.cpp

Le programme principal roule une série de tests. L'affichage devrait correspondre à ce qui suit :

```
Test  1: Getters de la classe Equipe                                    : OK
Test  2: constructeur par copie de la classe Equipe                     : OK
Test  3: operateur = de la classe Equipe                                : OK
Test  4: operateur + de la classe Equipe                                : OK
Test  5: operateur - de la classe Equipe                                : OK
Test  6: operateur ++ de la classe Equipe                               : OK
--------
Total pour la section: 5/5  arrondi au dixieme

Tests pour classe Joueur:
--------
Test  7: Getters de la classe Joueur                                    : OK
Test  8: Setters de la classe Joueur                                    : OK
Test  9: operateur < de la classe Joueur                                : OK
Test 10: operateur << de la classe Joueur                               : OK
--------
Total pour la section: 3/3  arrondi au dixieme

Tests pour classe Date:
--------
Test 11: operateur < de la classe Date                                  : OK
Test 12: operateur + de la classe Date                                  : OK
Test 13: operateur += de la classe Date                                 : OK
Test 14: operateur ++ de la classe Date                                 : OK
--------
Total pour la section: 2/2  arrondi au dixieme
```

Correction

La correction du TP se fait sur 20 points :

- [3 pt] Compilation du programme sans avertissement.
- [4 pt] Exécution du programme.
- [10 pt] Comportement exact de l'application.
- [2 pt] Qualité et documentation du code.
- [1 pt] Absence de fuites de mémoire.

Relisez bien les spécifications générales et les consignes de remise au début du TP avant de remettre votre travail!