UQAC

Université du Québec
à Chicoutimi

**Département d'informatique et mathématique**

---

# PROGRAMMATION ORIENTÉE OBJET

## 8PRO128

# TP N°2

***Hiver 2024***

**Groupe 01**

**Chargée de cours :** Asma Zouaghi

**Courriel :** azouaghi@uqac.ca

**Objectif :** Développer et mettre en œuvre trois classes Heure, Date et DateH pour représenter l'heure et la date. Ce travail de programmation comprend la conception des interfaces de ces classes, l'implémentation de leurs méthodes, et leur test via la fonction principale (main). Ce projet a pour but de vous faire :

- ✓ appliquer les principes de l'héritage dans la conception de classes ;
- ✓ implémenter des constructeurs paramétrés pour initialiser les objets avec des valeurs spécifiques, tout en validant ces valeurs selon des critères prédéfinis ;
- ✓ utiliser la surcharge d'opérateurs pour faciliter la comparaison intuitive entre objets.

**Détails du travail pratique :**

1. Implémenter une classe **Heure** contenant :

   a. trois membres donnés de type entier représentant le format heure, minute et seconde, en statut protégé ;

   b. un constructeur paramétré acceptant trois entiers (heure, minute, seconde) avec 0 comme valeur par défaut. Ce constructeur vérifie la validité des valeurs (heure : 0-23, minute et seconde : 0-59) et les remplace par 0 si elles ne sont pas valides ;

   c. surdéfinir l'opérateur > pour comparer deux instances Heure, retournant true si la première instance représente une heure plus tardive que la deuxième, et false sinon ;

   d. une fonction membre *affiche* qui permet d'afficher l'heure au format hh:mm:ss en utilisant les fonctions std::setw et std::setfill pour le formatage.

      *NB :* Pour afficher des zéros en plus, vous pouvez utiliser des éléments des librairies standard de C++ comme illustré ci-dessous :

```
# include < iostream >
# include < iomanip >
using namespace std;
// Affichage de 03 au lieu de 3
cout << stw(2) << setfill('0') << 3;
```

2. Implémenter une classe **Date** contenant :

   a. trois membres données de type entier pour le jour, le mois et l'année, en statut protégé ;

   b. un constructeur paramétré acceptant trois entiers (jour, mois, année) qui vérifie la validité des valeurs fournies, y compris la vérification de l'existence du jour.

      *NB :* Lors de la vérification de la validité du jour, considérez tous les mois ayant 28 ou 30/31 jours selon le cas, sans prendre en compte les règles spécifiques aux années bissextiles.

3. Implémenter une classe **DateH** qui hérite à la fois d'Heure et de Date :

   a. utiliser l'héritage multiple pour combiner les fonctionnalités des classes Heure et Date ;

   b. le constructeur de DateH doit accepter six arguments entiers (jour, mois, année, heure, minute, seconde) et utiliser les constructeurs des classes de base pour initialiser les objets ;

   c. ajouter une fonction membre *affiche* spécifique à DateH qui affiche la date suivie de l'heure ;

   d. surdéfinir l'opérateur > pour la classe DateH afin de comparer deux instances, où la comparaison prend en compte à la fois la date et l'heure.

4. Implémentez le programme principal :

   a. Tester la classe ***Heure*** :

      - ✓ créer deux instances de la classe Heure h1 et h2 avec différentes heures, minutes et secondes. Assurez-vous d'inclure des cas où les valeurs sont en dehors des limites acceptables pour voir si elles sont correctement ajustées à 0 ;
      - ✓ afficher l'heure pour chaque instance créée en utilisant la méthode affiche ;
      - ✓ comparer les deux instances de Heure en utilisant l'opérateur > et afficher le résultat.

b. Tester la classe ***Date*** :
   - ✓ Créer deux instances de la classe Date d1 et d2 avec différentes combinaisons de jours, mois et années. Inclure des cas où les dates sont invalides pour vérifier le traitement des erreurs ;
   - ✓ afficher la date pour chaque instance. (Vous devrez peut-être ajouter une méthode d'affichage simple dans la classe Date pour ce test.)

c. Tester la classe ***DateH*** :

   - ✓ créer deux instances de DateH dh1 et dh2 en utilisant une combinaison de valeurs pour les jours, mois, années, heures, minutes et secondes ;
   - ✓ afficher la date et l'heure pour chaque instance de DateH en utilisant la méthode affiche ;
   - ✓ comparer les deux instances de DateH en utilisant l'opérateur > et afficher le résultat. Assurez-vous de tester des cas où les dates sont différentes ainsi que des cas où les heures diffèrent mais les dates sont les mêmes.

d. Quelques tests supplémentaires :

   - ✓ créer l'instance DateH j1(1, 1, 2020, 0, 0, 0); // 1er janvier 2020 à minuit ;
   - ✓ créer l'instance DateH j2(31, 12, 2019, 23, 59, 59); // 31 décembre 2019 juste avant minuit ;
   - ✓ comparer j1 et j2 avec l'opérateur > et afficher le résultat (devrait être true) ;
   - ✓ convertir j1 et j2 en instances de Heure (si la conversion est possible) et comparer les deux en utilisant l'opérateur >. Afficher le résultat (devrait être false car les deux représentent la même heure du jour).

**Critères d'évaluation :** Réflexion et efforts démontrés dans le développement du code.

**Rendu :** Le code doit être écrit en C++ et soumis sous forme de fichiers *.cpp* et *.h*.

**Date de Rendu :** Le 02 mars 2024 – avant 12h00.

Bonne chance ☺ !