TP C++ 6 : Héritage

M1 Mathématiques Appliquées

2019-2020

Remarques préliminaires :

- Il est fortement recommandé de valider les questions de manière incrémentale à l'aide d'un programme principal. Pour ce faire, il est conseillé de "tracer" son code en affichant des messages lors d'appels à des fonctions, en vérifiant que des valeurs devant être modifiées l'ont effectivement été, etc.
- On codera les classes dans des fichiers à part (.h pour la déclaration, .cpp pour la définition) et on utilisera un fichier de test par exercice.

Exercice 1 : Tableaux bornés

Le but de cet exercice est de construire des tableaux de réels flottants dont les valeurs se trouvent dans un intervalle donné.

1. Implémenter une classe `Tableau` similaire à celle vue en cours, où la taille du tableau sera un membre donnée. La classe contiendra :

   - un constructeur prenant un entier en paramètre, qui allouera dynamiquement la mémoire pour un tableau de la taille de l'entier;
   - un destructeur qui libèrera la mémoire;
   - un constructeur de recopie;
   - l'opérateur d'affectation `=` redéfini.

   Elle surdéfinira également l'opérateur `[]` pour qu'il permette d'accéder aux éléments du tableau par référence, et ainsi autoriser la modification de ces éléments.

2. Implémenter une classe `TableauBorne` dérivée de la classe `Tableau`. En plus des membres donnés de la classe `Tableau` (qui devront être accessibles dans la classe `TableauBorne`), cette classe comportera deux membres donnés de type `float` correspondant aux bornes de l'intervalle dans lequel devront se trouver les éléments du tableau. On définira donc un constructeur pour cette classe prenant un entier et deux flottants en arguments.

3. Dans la classe `TableauBorne`, surdéfinir l'opérateur `[]` pour que la modification des valeurs du tableau ne soit plus autorisée via cet opérateur. L'opérateur `[]` de la classe `Tableau` ne devra pas être accessible pour un utilisateur de la classe `TableauBorne`.

4. Dans la classe `TableauBorne`, surdéfinir l'opérateur `()` pour permettre la modification des valeurs du tableau. Cet opérateur prendra en argument un entier correspondant à l'indice de l'élément à modifier dans le tableau, et un flottant correspondant à la nouvelle valeur. Si l'indice dépasse la taille du tableau, ou que la valeur n'est pas dans l'intervalle de valeurs acceptées, le tableau ne devra pas être modifié.

Exercice 2 : Heures et dates

1. Implémenter une classe `Heure` contenant :

   - Trois membres données de type entier représentant le format heure/minute/seconde, en statut protégé (`protected`);
   - Un constructeur prenant trois entiers en argument avec 0 comme valeur par défaut, qui vérifie que ces entiers aient des valeurs admissibles et les remplace par 0 sinon (par exemple, une heure devra être un entier entre 0 et 23);
   - Une fonction membre `affiche` qui affiche l'heure sous le format hh:mm:ss.
     NB : Pour afficher des zéros en plus, on pourra 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. Surdéfinir l'opérateur `>` de sorte à pouvoir l'appliquer à deux variables de type `Heure` et renvoyer `true` si la première représente une heure plus tardive que l'autre, et `false` sinon.

3. Implémenter une classe `DateH` qui dérive de la classe `Heure` contenant :

   - Trois membres données de type entier représentant le jour, le mois et l'année;
   - Un constructeur à 6 arguments entiers, qui vérifie que les valeurs données sont admissibles et que le jour encodé existe bien[1];
   - Une fonction membre `affiche` qui affiche le jour et l'heure correspondant à l'objet.

4. Surdéfinir l'opérateur `>` pour l'appliquer à deux variables de type `DateH`. Tester l'exemple suivant (en adaptant si besoin l'initialisation à votre implémentation du constructeur) :

```
DateH j1 (1,1,2020,0,0,0); %1er janvier 2020 a minuit
DateH j2 (31,12,2019,23,59,59); %31 decembre 2019 juste avant minuit
bool b= j1>j2;% Doit renvoyer true
Heure j3=j1;
Heure j4=j2;
b = j3>j4; % Doit renvoyer false
```

[1] Pour simplifier, on pourra ignorer les années bissextiles.

5. Question complémentaire : Reprendre l'exercice en mettant maintenant les membres données de la classe `Heure` en statut `private`.