TRAVAUX DIRIGÉS 3

Ce TD porte sur le chapitre « Introduire de nouveaux types ».
Il suppose les notions du chapitre « Les Bases du C++ » assimilées, entre autres les tableaux, les pointeurs, les fonctions et les instructions de contrôle.

Exercice 1:

Soit une "classe d'énumération" **Jour**, débutant à la valeur 1, dont les énumérateurs sont les jours de la semaine, nommés par les trois premières lettres en français.

(a) Déclarer cette énumération.

(b) Définir cette énumération.

Exercice 2:

De la même façon, soit une "classe d'énumération" **Mois**, commençant aussi à la valeur 1.

(a) Déclarer **Mois**.

(b) Est-il possible de définir **Mois** en respectant exactement l'énoncé ? Pourquoi ? Définir **Mois** en rectifiant les énumérateurs problématiques (et uniquement ceux-ci).

(c) Nous avons choisi d'utiliser une "classe d'énumération". Si nous avions utilisé une "énumération simple", quel problème aurions-nous rencontré ?

Exercice 3:

À la suite des deux exercices suivants : Soit une structure **Date**, contenant un **Jour**, un **Mois** et un entier représentant l'année.

(a) Définir cette structure.

(b) Écrire l'instruction déclarant une variable de type **Date** ainsi qu'un pointeur sur le type **Date** pointant sur la variable précédente.

(c) On souhaite afficher la valeur de l'énumérateur contenu dans le membre de type **Jour**. On suppose "iostream" inclus et l'espace de nom "std" utilisé. Écrire la ligne d'instruction affichant cette valeur pour :

i. la variable de type **Date**

ii. le pointeur sur un type **Date**

(d) Pouvez-vous prévoir ce qui va s'afficher pour chaque ligne ? Donner un exemple. Justifier.

(e) On souhaite définir des valeurs par défaut pour les membres de la structure **Date** :

i. Initialiser **Jour** avec la représentation de Lundi

ii. Initialiser **Mois** en utilisant la conversion de l'entier 6 en **Mois**.

iii. Initialiser l'année à la valeur entière "1900".

Exercice 4:

Soit une fonction `affiche_jour()` qui, prenant un **Jour** en argument, affiche à l'écran une chaîne de caractères contenant le nom du jour dans une langue choisie (en argument).

(a) Définir une "classe d'énumération" **Langue** contenant les énumérateurs "francais" et "anglais".

(b) Indiquer comment représenter une "chaîne de caractères" en utilisant les types fondamentaux, sans la STL.

(c) Définir des expressions constantes listant la traduction des jours. (Utiliserez des "C-array" et le type const char, vous testerez les autres solutions en TP)

(d) Écrire une instruction qui, à partir d'une instance de **Jour**, affecte à une variable, de type int, l'indice correct dans l'un des tableaux de traduction.

(e) Donner la déclaration complète de la fonction `affiche_jour` prenant en argument un **Jour** et une **Langue**. La langue par défaut sera le français. Prendre soin de la déclaration en prenant en compte le caractère constant (ou pas) des différents types manipulés.

(f) Avec tous ces éléments, définir la fonction `affiche_jour`

(g) Soit une instance de **Date** "d" et un pointeur de **Date** "ptd" sur "d", écrire pour chacun les appels de cette fonction pour afficher le jour contenu dans la date, en français puis en anglais.

(h) On souhaite surcharger la fonction `affiche_jour()` pour utiliser le type **Date**, écrire la définition de la surcharge pour le type puis pour son pointeur.

(i) En prenant en compte le contexte d'utilisation de la structure **Date** – en plusieurs langues – quelle méthode d'initialisation des énumérations dans l'exercice 3 partie (e) paraît la plus pertinente ?

Exercice 5:

Soit une classe **Livre**, elle contient un titre, un auteur, une date d'achat, une date de dernier emprunt et son statut d'emprunt. Le statut d'emprunt sera représenté avec un bool. Les chaînes de caractères seront représentées par le type **string** disponible dans <string> puis dans l'espace de nom std.

(a) Définir cette classe en supposant tout le travail des exercices précédents contenu dans l'en-tête "date.h".

(b) Vous préciserez le nom des fichiers contenant les instructions pour les questions suivantes.

- Définir un constructeur initialisant chaque membre par une valeur donnée en argument.
- Déclarer l'usage du constructeur par défaut "par défaut".
- Pourquoi est-il nécessaire d'ajouter cette déclaration ?

(c) On souhaite maintenant que l'auteur soit stocké comme un pointeur, rectifier la déclaration de **Livre**, écrire son destructeur. Pourquoi est-il nécessaire de le faire ?

(d) Rectifier le constructeur déclaré plus haut, il prendra l'auteur par valeur et crée une nouvelle instance. Pourquoi est-il conseillé de retirer le constructeur par défaut "par défaut" maintenant ?

(e) Nous allons ajouter des valeurs par défauts pour les membres de la classe **Livre**, le pointeur sera initialisé sur nullptr.

i. Spécifier ces valeurs dans la déclaration. Peut-on réactiver le constructeur par défaut "par défaut" ? Justifier.

ii. Spécifier ces valeurs dans un constructeur par défaut. Cette solution est-elle différente de la précédente ? Est-elle à privilégier ?

Exercice 6:

Nous supposons avoir choisi la version de **Livre** contenant un pointeur pour l'auteur, des valeurs par défaut définies à la déclaration et deux constructeurs : par défaut et en passant tous les arguments dont l'auteur par valeur.
Cette classe sera déclarée dans "livre.h" et définie dans "livre.cpp".

Nous souhaitons créer une classe **Etagere**, représentant un rayon de bibliothèque, conteneur de livres, organisé par étage. Chaque étage peut contenir 10 livres, l'étagère contiendra 3 étages. L'étagère possédera un code de rayonnage composé d'un caractère et d'un numéro entier.

(a) Nous ne souhaitons pas détailler l'étage, son contenu sera décrit à l'aide de "C-array". Quel est le type le plus pertinent pour ce membre ? (Considérez qu'un livre peut tout à fait changer d'étagère...)

(b) Définir la classe **Etagere**, préciser les instructions pour utiliser **Livre**.

(c) Écrire un constructeur par défaut. L'étagère sera vide dans ce cas et situé en A-1.

(d) Écrire le destructeur. Est-il nécessaire ? Dans les deux cas, justifier et souligner les problèmes. Proposer une implémentation élégante si besoin.

(e) Un constructeur par copie est-il présent ? Même question pour l'opérateur d'affectation.

(f) Nous choisissons de définir une "copie" d'une étagère par la création d'un nouveau rayon de bibliothèque contenant un exemplaire supplémentaire de chaque livre. Par convention nous lui affecterons la position Z-0. Est-il nécessaire de définir les fonctions de copie ? Si oui, les exprimer.

(g) À l'interface de la classe **Livre** décrite dans l'énoncé, on ajoute une fonction retournant un pointeur de **string** pour récupérer l'auteur :

```
1 // dans livre.h
2 string* get_auteur(){ return auteur_;}
```

Étudier les instructions suivantes :

```
1  Date d;
2  Etagere* g = new Etagere;
3  Livre* book = new Livre("La craie", "Papy matheux", d, d, true);
4
5  g->contenu[0][0] = book;
6
7  Etagere h;
8  h = *g;
9  cout << *(h.contenu[0][0]->get_auteur()) << endl;
10
11 delete g;
12 cout << *(h.contenu[0][0]->get_auteur()) << endl;
13
14 delete book;
15 cout << *(h.contenu[0][0]->get_auteur()) << endl;
```

i. Qu'affiche la l. 9 ?

ii. Qu'affiche la l. 12 ? Pourquoi ?

iii. Qu'affiche la l. 15 ? Justifier.