



Les classes en C++

Documents à rendre :

Pour **chaque question de chaque exercice**, vous devez rendre un **dossier** contenant un fichier d'entête (déclaration de la classe) et 2 fichiers sources (définition de la classe et fichier main.cpp). Vous devez également rendre un document LibreOffice Writer avec les réponses aux questions.

Travail à réaliser :

Exercice 1 : classe CPersonne

Dans cet exercice, une personne est caractérisée par :

- son âge,
- son nom,
- son poids,
- sa taille.

Q1) Dans cette 1ère version, **tous les membres** de la classe CPersonne (y compris ses attributs) doivent être **publics**. Déclarer et définir 3 méthodes :

- *Initialise()* : sert à donner une valeur à tous les attributs d'une personne;
- *Affiche()* : permet d'afficher toutes les caractéristiques d'une personne;
- *GetImc()* : méthode, sans affichage, qui renvoie l'indice de **masse corporelle** de la personne (poids divisé par la taille au carré).

Dans le fichier d'entête de la classe, vous devez **justifier soigneusement** :

- le type de chaque attribut;
- le prototype de chaque méthode :
 - type et valeurs possibles de la valeur de retour;
 - type et rôle de chaque paramètre;

Coder une fonction *main()* qui teste:

- l'accessibilité tous les membres de la classe (notamment des attributs) ;
- l'implémentation des méthodes.

Q2) Dans cette seconde version, vous devez **encapsuler** les données de la classe CPersonne. Pour chaque attribut, vous devez déclarer et définir une méthode *setxxx()* et une méthode *getxxx()*.

Modifier le code de la fonction *main()* afin de tester ces nouvelles méthodes.



Les attributs sont-ils privés ou publics ? Sont-ils accessibles à l'extérieur de la classe (c'est à dire dans les fonctions qui ne sont pas membres de la classe) ?

- Q3)** Ajouter dans la classe CPersonne :
- un constructeur **avec paramètres**;
 - et un destructeur.



A quoi sert le constructeur ? Que faire de la méthode *Initialise()* ?
Réaliser dans le constructeur et dans le destructeur des **affichages afin de "tracer" leurs appels**.

Modifier la fonction *main()* afin de tester toutes les méthodes de la classe.

Dans le fichier *main.cpp*, déclarer, définir et appeler une fonction indépendante, sans valeur de retour et sans paramètre, qui instancie la classe CPersonne.



Quand le constructeur d'une classe est-il appelé? Même question pour le destructeur ?

- Q4) Ajouter** dans la classe un constructeur sans paramètre. Ainsi, la classe disposera de 2 constructeurs, un avec paramètres et un autre sans paramètre. On dit alors que le constructeur (plus précisément le symbole qui lui est associé) est surdéfini : il possède ici 2 définitions.

Réaliser, entre autre, dans le constructeur sans paramètre un affichage différent (de celui effectué dans le constructeur avec paramètres) afin de "tracer" son appel.

Modifier la fonction *main()* afin d'appeler les 2 constructeurs.

- Q5)** Modifier la classe CPersonne afin de la munir d'un seul constructeur avec paramètres et des valeurs par défaut pour chaque paramètre.

Modifier la fonction *main()* afin d'illustrer tous les appels possibles de ce constructeur (en lui transmettant 4 paramètres, 3 paramètres, 2 paramètres, 1 paramètre et enfin 0 paramètre).



- Q6)** On souhaite ne pas surdimensionner le tableau de caractères qui contient le nom de la personne. Quelle solution proposez-vous?

Modifier en conséquence les différentes méthodes de la classe CPersonne (sans oublier le destructeur).
Réaliser une fonction *main()* qui teste toutes les méthodes de la classe.

Exercice 2 : classe CPoint

Déclarer, définir et tester une classe CPoint.

Cette classe doit encapsuler correctement 3 attributs :

- les coordonnées du point (abscisse et ordonnée de type réel);
- un compteur qui **compte le nombre de fois où un point est déplacé**.

Elle doit être **sans affichage et sans saisie** (i.e. : les méthodes de la classe CPoint doivent être sans affichage et sans saisie).

Elle doit être munie :

- d'un constructeur et d'un destructeur;
- d'une méthode qui permet de déplacer un point (**l'appel de cette méthode doit constituer le moyen pour l'utilisateur de modifier les coordonnées d'un point après sa création**);
- de méthodes permettant d'accéder aux attributs du point (**justifier vos choix**).

Dans le fichier d'entête de la classe, vous devez **justifier soigneusement** :

- le type de chaque attribut;
- le prototype de chaque méthode :
 - type et valeurs possibles de la valeur de retour;
 - type et rôle de chaque paramètre;

Exercice 3 : classe CDate

Le calendrier grégorien a été adopté en 1582. Il a permis de corriger la dérive séculaire du calendrier julien alors en usage en ajoutant un jour aux années bissextiles.

Depuis l'ajustement du calendrier grégorien, sont bissextiles les années :

- soit divisibles par 4 mais non divisibles par 100 ;
- soit divisibles par 400.

Donc, inversement, ne sont pas bissextiles les années :

- soit non divisibles par 4 ;
- soit divisibles par 100, mais pas par 400.

Cahier des charges :

Réaliser une classe CDate permettant d'**encapsuler** les caractéristiques (jour, mois, année) d'une date.

La classe CDate devra être **sans affichage et sans saisie**.

Elle devra permettre la gestion de toutes les dates postérieures ou égales au 1/1/1582. Elle devra être capable de **traiter correctement les années bissextiles**.

Elle devra disposer :

- d'un constructeur sans paramètre (qui initialise le ou les attribut(s) avec une date valide);
- d'un destructeur;
- de méthodes permettant d'**accéder en consultation** à la date :
 - sous la forme de 3 entiers;
 - ou sous la forme d'une chaîne de caractères aux formats JJ/MM/AAAA, J/M/AAAA, JJ/M/AAAA ou J/MM/AAAA.
- de méthodes permettant d'**accéder en modification** à la date. La nouvelle date pourra être fournie sous la forme de 3 entiers, ou sous la forme d'une chaîne de caractères (en respectant les formats précédents). Avant, toute modification du ou des attribut(s), ces méthodes doivent **contrôler la validité de la nouvelle date** : ainsi, le ou les attributs encapsulés dans un objet de cette classe correspondront toujours à une date valide. Il faudra prévoir un mécanisme permettant d'informer l'utilisateur de la validité (ou non validité) de la date transmise en paramètre.
- d'une méthode qui calcule le nombre de jours écoulés depuis le début de l'année en cours.
- d'une méthode qui permet de comparer 2 dates (le paramètre implicite de la méthode et une autre date transmise en paramètre). Quand on compare 2 dates, il y a 3 résultats possibles :
 - d1 antérieure à d2;
 - d1 postérieure à d2;
 - d1 égale à d2.

Dans le fichier d'entête de la classe, vous devez **justifier soigneusement** :

- le type de chaque attribut;
- le prototype de chaque méthode :
 - type et valeurs possibles de la valeur de retour;
 - type et rôle de chaque paramètre;

Réaliser une fonction *main()* qui teste toutes les méthodes de la classe.