PoP C++ Série 4

H1 :
1) opérateurs bit à bit (*bitwise operators*)
2) usage de static à l'échelle d'un module

H2 : MOOC MOOC Introduction à la programmation orientée objet (en C++)
Série semaine 3: constructeur/ destructeur
Surcharge des opérateurs

Première partie : opérateurs bit à bit

Exercice 1.1(niveau 0) : **intérêt des opérateurs bit à bit et extraction d'un groupe de 5 bits.**

Le document séparé introduit les concepts de masque et de décalage.

Exercice 1.2 (niveau 1) : **fonctions de manipulation à l'échelle du bit ou groupe de bits.**

Chacune des fonctions suivantes doit vérifier que les paramètres reçus sont corrects avant d'effectuer la manipulation demandée. Afficher un message en cas d'erreur. Tester les fonctions sur des exemples simples vous permettant de les valider rapidement.

a) Ecrire une fonction **f1** recevant en paramètre un entier non-signé **n** et un second entier non-signé **index** compris entre 0 et 31 et qui renvoie la valeur du bit de rang index dans n.

b) Ecrire une fonction **f2** recevant en paramètre :
- un entier non-signé **n**
- un entier non-signé **index** compris entre 0 et 31
- une **val** entière 0 ou 1

et qui modifie **n** en affectant le bit de rang **index** à la valeur **val**.

c) Ecrire une fonction **f3** recevant en paramètre :
- un entier non-signé **n**
- un entier non-signé **witdh** compris entre 1 et 32
- un entier non-signé **shift** compris entre 0 et 32-width
- une **val** entière non-signée comprise entre **0** et $\mathbf{2^{width}-1}$

et qui modifie **n** en insérant **val** sur les bits situés entre les rangs **shift** et **shift+width-1**

Seconde partie : static à l'échelle d'un module

Exercice 2.1(niveau 0) : **module student gérant un ensemble de variables Student.**

Le document séparé illustre l'usage de static pour un type concret mis en œuvre avec une structure.

Exercice 2.2 (niveau 2) : gestion d'un agenda

On testera les modules **date** et **event** au fur et à mesure des questions avec un module de niveau supérieur **test** qui inclut leur interface et appelle les fonctions exportées.

a) Méthode de travail : Commencer par écrire le fichier **makefile** et les **fichiers .cc et .h dans une version minimale** qui compile pour produire l'exécutable **test**, avant de préciser les classes **date** et **event** avec les questions suivantes. On mettra en oeuvre les header guard pour les interfaces des modules (série3 PoP).

b) module **date** gérant une classe **Date** de calendrier
Ecrire un module responsable d'une classe Date avec 3 attributs privés : **jour, mois, annee.**

Un seul constructeur sera défini pour initialiser simultanément les 3 champs en vérifiant que la date est correcte pour le calendrier grégorien (la série1 fourni le code source qui effectue ces vérifications). Un message est affiché en cas d'erreur.

c) Une méthode publique de calcul du *nombre de jours entre deux dates* sera aussi fournie (cf série1).

d) Ecrire une surcharge des opérateurs == et != pour la classe **Date**

e) module **event** gérant une classe **Event** et un ensemble d'instance de ce type dans un tableau dynamique **agenda** restant confidentiel au niveau du module event.
La classe Event contient les attributs privés : **nom** et **lieu** de type string, et **date** de type Date.

Un constructeur doit permettre d'initialiser une instance à partir de la valeur de tous les attributs.

f) Ecrire une surcharge des opérateurs == et != pour la classe **Event**

g) Une méthode publique **add_to_agenda** doit permettre d'ajouter la valeur d'une variable de type Event à l'ensemble des Event mémorisés dans **agenda** au niveau du module **event**. Les conditions à remplir sont les suivantes :
- L'agenda ne doit pas contenir déjà cet **Event**
- il ne doit pas y avoir d'Event déjà prévu pour le même lieu à la même *date plus ou moins 6 jours*
Un message d'erreur est affiché en cas d'erreur.

h) Une méthode publique doit permettre d'afficher l'Event sur lequel elle est appelée.
Une méthode de classe doit permettre d'afficher l'ensemble des Event mémorisés dans l'agenda.