ISIMA 2ème année, Examen de Génie Logiciel à Objet 2003 durée 2 heures

Première partie du Partiel de C++ - (1h - documents de cours non autorisés)

1. Question de Génie Logiciel (4 points) – F3/F4/F5

a) Expliquer la différence entre vérification (du fonctionnement) et validation (du produit). Quel est l'impact au niveau du développeur ? du client ?

b) Comment mettre en œuvre une stratégie de vérification ? Quels sont les intérêts des langages orientés objets dans ce domaine ?

c) Discuter les avantages et inconvénients de la surcharge d'opérateurs

2. Questions de compréhension (6 points) – (F1/F2/F3/F4/F5) justifier les réponses

a) Qu'est-ce qu'une classe virtuelle ? une classe abstraite ?

b) Pourquoi le destructeur d'une classe virtuelle doit être virtuel ?

c) Pourquoi l'opérateur de copie/affectation retourne-t-il `*this` ?

d) Donner un exemple d'opérateur dans lequel on force la création d'un objet (duplication) sur le type de retour.

Quelle en est la raison ?

e) Quelle est la différence entre agrégation par référence et agrégation par valeur ?

Quelles sont les implications dans l'implémentation d'une classe utilisant l'un ou l'autre ?

f) Dans quels cas peut-on se passer de définir des constructeurs dans une classe fille ?

Peut-on définir un constructeur virtuel ?

Seconde partie du Partiel de C++ - (1h - documents de cours autorisés)

1. Problème (10 points) – F1/F2/F3/F4/F5

Soit une classe : " `class Objet { } ;` " dont on pourrait dériver toutes les classes écrites par l'utilisateur.

a) Proposer le code source C++ d'une ou plusieurs classes pour un composant logiciel permettant de gérer un arbre binaire. Chaque feuille de l'arbre comporte un pointeur à droite, un pointeur à gauche et un pointeur sur un " `Objet` ". Vous proposerez une liste de méthodes qui permettent d'utiliser cet arbre convenablement. (3 pts) (vous n'implémenterez pas les méthodes dans cette question, seules les déclarations de la ou des classes suffisent)

b) Quelle technique du C++ devrait-on utiliser pour obtenir un arbre qui manipule des types indifférents (pas seulement des entiers) et qui puisse contrôler que les éléments d'un même arbre soient de même type ? (1 pt)

c) Modifiez le code proposé en (a) pour utiliser la technique de codage du (b). (1 pt)

d) Proposer une implémentation d'une fonction membre de la classe ou d'une des classes du (a) qui permette d'effectuer un parcours de tous les éléments de l'arbre en appliquant une méthode virtuelle à chaque objet rencontré. Cette méthode virtuelle doit être passée en paramètre par un pointeur argument de la fonction membre. On doit supposer que cette méthode est disponible dans l'interface de la classe "`Objet`". (2 pt)

e) Pourquoi peut-il être nécessaire de définir une classe itérateur séparée de la classe arbre et pour quels types de méthodes ? (1 pt)

f) Implémenter les méthodes suivantes : recherche d'un objet et insertion d'un objet. (2 pts). On suppose que les objets sont triés de gauche à droite dans l'arbre (G < D) et que l'opérateur de comparaison '<' est disponible.

Questions (4 points) – F1/F2

a) Déclarer une classe `A` avec une fonction virtuelle `f` (ne retournant rien et n'acceptant aucun paramètre, écrire également la déclaration du destructeur virtuel de cette classe et ne détaillez rien d'autre. Ce n'est pas une fonction virtuelle pure, on suppose donc que la classe `A` implémente cette fonction virtuelle. (1 pt)

b) Déclarer une classe `B` qui dérive de la classe `A` en « public », vous redéclarerez la fonction virtuelle `f`. (1 pt)

c) Ecrire l'implémentation de la méthode `f` dans la class `B`, la seule partie du corps de la fonction que l'on vous demande d'écrire est le rappel de la méthode virtuelle `f` telle qu'elle est implémentée dans la classe `A`. Il est courant, dans une méthode virtuelle, de rappeler la méthode de la classe de base avant d'y ajouter un comportement. (1 pt)

d) Clonage d'objets : écrire une fonction template « `clone` », qui accepte une référence constante « `x` » sur un objet de type « `T` » et retourne un pointeur de type « `T` ». Le corps de la fonction retourne ce pointeur simplement en faisant une allocation dynamique avec appel au constructeur de la classe « `T` » qui utilise la référence `x`. (1 pt)