Examen de C++ - ISIMA 2ème année – Tronc commun – Décembre 2010

Documents autorisé : une feuille A4 recto-verso manuscrite

Questions indépendantes

1) Déclarer une classe A qui possède comme attributs d'instances un objet de classe B et un pointeur sur un objet de classe C (0,5 pt). A quels types de relations cela correspond-t-il en UML (1 pt).

2.a) Déclarer une classe A avec une méthode virtuelle draw() qui affiche "A", puis une sous-classe B qui hérite de A et redéfinit cette méthode draw() pour afficher "B". Pour la classe B, nous avons le constructeur par défaut (1 pt).

2.b) Qu'affiche le code ci-dessous (0,5 pt)

```
B unB ;
unB.draw();
```

2.c) Déclarer une fonction « *affiche* » qui accepte en paramètre un objet de classe A (avec le mode de passage des paramètres par défaut en C++). Au sein de cette méthode un appel à la méthode *draw*() à partir de cet objet. (0,5 pt)

2.d) Qu'est-ce qui est affiché si on effectue un appel du type : `affiche(unB)` (0,5 pt)

2.e) Modifier la fonction pour qu'elle accepte une référence sur un objet de la classe A. (0,5 pt)
Est-ce que l'affichage lors de l'appel précédent est changé ? (0,5 pt)

3) Ecrire une fonction template « *clone* », qui accepte une référence constante sur un objet de type « T » et retourne un pointeur de type « T ». Le corps de la fonction retourne un pointeur sur un objet T simplement en faisant une allocation dynamique avec appel au constructeur de la classe « T » qui utilise la référence passée en argument. Préciser quel est le type de constructeur appelé. Ecrire ensuite une fonction *main* qui déclare et initialise une variable entière i à 5, puis déclare un pointeur sur un entier pClone qui est initialisé en appelant la fonction *clone* précédente avec l'entier i, puis la fonction main détruit le clone pointé. (3 points)

4) Le code suivant retourne l'erreur : `main.cpp: In static member function 'static void A::afficher()': Line 4: error: invalid use of member 'A::_valeur' in static member function compilation terminated due to -Wfatal-errors.`

Donner une explication de l'erreur rencontrée. La traduction ne suffit pas (1 pt).

- Proposer une correction pour que ce code compile sans modifier le *main* (1 pt)
- Proposer un nouveau code – classe et *main* – pour garder la méthode afficher en l'état (2 pts)

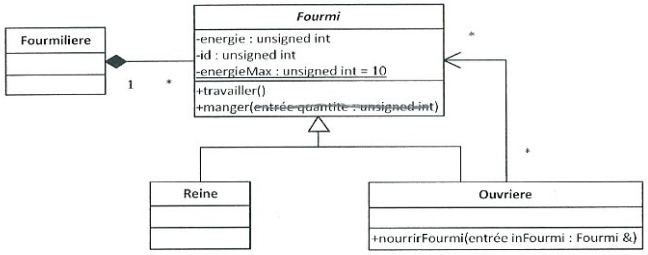
```
1  #include <iostream>
2
3  class A {
4     int _valeur;
5   public:
6     A() { _valeur = 0; }
7     static void afficher() { std::cout << _valeur << std::endl; }
8  };
9
10 int main(int, char **) {
11
12   A a;
13   a.afficher();
14
15   return 0;
16 }
```

5) Le code suivant donne le message d'erreur ci-dessous. Modifier et expliquer votre modification (1 pt)

```
In function 'void afficher(const A&)': Line 16: error: no matching function for call to
'A::afficher() const' compilation terminated due to -Wfatal-errors.
```

```
1        #include <iostream>
2
3        class A {
4           int i ;
5           int * pi;
6         public:
7           A() { i = 0; pi = &i; }
8           void afficher() { std::cout << "A" << " " << i << std::endl; }
9           void modifier(int j) const {
10             *pi = j;
11          }
12          const A * give() const { return this; }
13       };
14
15       void afficher(const A & a) {      a.afficher(); }
16
17       int main(int, char **)
18       {
19          A a;
20
21          afficher(a);
22          a.modifier(3);
23
24          return 0;
25       }
```

Problème : 7 points



1 - Implémenter la classe abstraite « Fourmi » qui possède trois attributs avec les hypothèses que vous jugerez nécessaires et en évitant la duplication de code

- Un identifiant unique.
- La quantité d'énergie qu'il reste à la fourmi.
- La quantité maximale d'énergie d'une fourmi (par souci de simplicité, cette valeur sera identique pour l'ensemble des catégories de fourmi).

Elle possède également deux méthodes :

- Une méthode *travailler()*. La finalité de cette méthode est de permettre, pour la classe Reine et la classe Ouvrière :
  - ✓ d'afficher à l'écran l'identifiant de la fourmi,
  - ✓ de décrémenter l'énergie de la fourmi
  - ✓ d'afficher le travail effectué par une fourmi (chaque type de fourmi réalise un travail différent). Le travail est « pondre » pour une reine et « construire/chercher à manger » pour une ouvrière.
- Une méthode *manger()* commune à l'ensemble des fourmis remonte le niveau d'énergie de la fourmi à son maximum.

2 - La classe Reine et la classe Ouvrière sont des implémentations concrètes de la classe Fourmi. Redéfinir, lorsque c'est nécessaire, les méthodes des classes mères et implémenter la méthode *nourrirFourmi()* qui appartient à la classe Ouvrière et appelle la méthode manger() de l'instance de Fourmi passé en paramètre.

3 – On veut maintenant disposer d'une classe Fourmilière composée d'un nombre variable de fourmis (une fourmi ne peut appartenir qu'à une seule fourmilière, les fourmis d'une fourmilière ont la même durée de vie que la fourmilière).
Indiquer quelles méthodes vous implémenteriez pour assurer qu'aucune fuite mémoire ne puisse apparaître dans notre application ? Justifiez vos choix.

La fourmilière est dotée d'un compteur de larves qui est incrémenté à chaque ponte et d'une méthode éclore() qui permet à 10 larves au plus de se transformer en fourmi (9 ouvrières et 1 reine si possible et si on a moins de 10 larves il n'y a pas de reine). Les nouvelles fourmis sont des membres à part entière de la colonie. Une fourmilière ne peut se créer spontanément et le premier individu de la fourmilière est une reine (fécondée).

Implémentez la fourmilière.

Bonus STL sur 2 points

1. Définissez une classe *MyPair*, stockant deux instances de types templates différents : une clé et une valeur. Vous ne définirez pas de méthodes particulières pour cette classe.

2. Définissez une classe *MyMap* contenant une liste STL d'instances de *MyPair*. Comme précédemment, vous ne définirez pas de méthodes particulières pour cette classe.

3. Déclarez un type *iterator* pour la classe *MyMap* en réutilisant les itérateurs de la classe *std::list*. Vous veillerez à ce qu'il puisse être utilisé comme suit :

```
MyMap<int, string>::iterator  it;
```