TP de C++ n°6

L'3

Thème 1 : Héritage ou composition ?

On rappelle que le choix entre héritage et composition se fait sur le critère suivant

Lorsqu'un objet de la classe Y est en relation avec un objet de la classe X par la relation :

- est_un, il s'agit **d'héritage** : Y hérite de X :

```
class Y : public X
{
      // membres private de Y
      public:
            Y():X(){…}
            ~Y(){…}
            …
}
```

- a_un, il s'agit de **composition** :

```
class Y
{
      // membres private de Y
      X x;
public:
      Y() : x(){…}
      ~Y(){…}
      …
};
```

- Ecrire 3 classes A,B et C telle que : C hérite de B et contient un objet de classe A. Ecrivez les constructeurs et destructeurs et déterminer l'ordre d'appel des constructeurs et destructeurs lorsqu'un objet de classe C est créé.

- Ecrire les classes véhicule_roulant, voiture, vélo, roue, moteur, volant en leur appliquant les relations d'héritage ou de composition selon le cas.

  Vous définirez également les prototypes de quelques méthodes pour chaque classe.

Thème 2 : Méthodes héritées

- Ecrire les classes `personne` et `etudiant` en utilisant des méthodes de même nom pour la saisie, l'affichage (vous pouvez surcharger les opérateurs << et >>). Ecrivez tous les accesseurs et mutateurs sur les membres des deux classes.

La classe personne doit contenir un champ `nom` de classe `Nstring` (développée pour le thème de synthèse sur la table de Hashage).

Est-il possible d'écrire :

```
int main()
{
      etudiant e("Gilbert");
      cout << e.getNom() << endl;
}
```

où `getNom()` est une méthode de la classe personne ?

Vérifiez le résultat obtenu.

On souhaite déterminer si l'héritage concerne les opérateurs et le constructeur par recopie. Pour la classe personne, écrivez un constructeur par recopie (ce qui devrait déjà être fait…), l'opérateur = et l'opérateur +, qui prend un entier et ajoute cet entier à l'age de la personne.

Vérifiez parmi ces méthodes celles qui sont héritées par la classe étudiant. Définissez dans la classe étudiant celles qui ne sont pas héritées.

Thème 3 : polymorphisme

Le polymorphisme est souvent associé à l'utilisation de classes abstraites. Nous allons, pour l'exemple qui suit, utiliser des formes géométriques simples dans le plan.

Soient les classes suivantes à utiliser : `triangle, polygone, forme, cercle, ellipse, point, segment, rectangle.`

En considérant qu'une forme est une partie du plan qui est fermée, indiquez les relations d'héritage et de composition qui existent entre les classes citées. Y a-t-il une ou des classes abstraites ?

Parmi les méthodes appliquées aux formes, on doit trouver : `afficher(), zoomer(), rotation(), translation()`. Ces méthodes doivent elles être qualifiées de `virtual` ?

Définissez les prototypes de ces méthodes.

Rédigez ensuite toutes les classes. Pour l'instant, la méthode `dessiner()` se contentera d'afficher les coordonnées des points des formes concernées. On doit également voir le nombre de triangles, de cercles, de rectangles définis.

Dans un programme, créer un tableau de pointeurs vers des formes et appliquez quelques transformations à des éléments de ce tableau. En quoi le polymorphisme est-il utile ?