TP C++: Serie 3

Exercice 1: polymorphisme

Rappel:

Il ne faut pas confondre la redéfinition et la surdéfinition :
– Une surdéfinition (ou surcharge) permet d'utiliser plusieurs méthodes qui portent le même nom au sein d'une même classe avec une signature différente.
– Une redéfinition (overriding) permet de fournir une nouvelle définition d'une méthode d'une classe ascendante pour la remplacer. Elle doit avoir une signature rigoureusement identique à la méthode parente.
Un objet garde toujours la capacité de pouvoir redéfinir une méthode afin de la réécrire ou de la compléter.
Le polymorphisme représente la capacité du système à choisir dynamiquement la méthode qui correspond au type de l'objet en cours de manipulation.
On voit donc apparaître ici le concept de polymorphisme : choisir en fonction des besoins quelle méthode appeler et ce au cours même de l'exécution.
Le polymorphisme est implémenté en C++ avec les fonctions virtual et l'héritage.

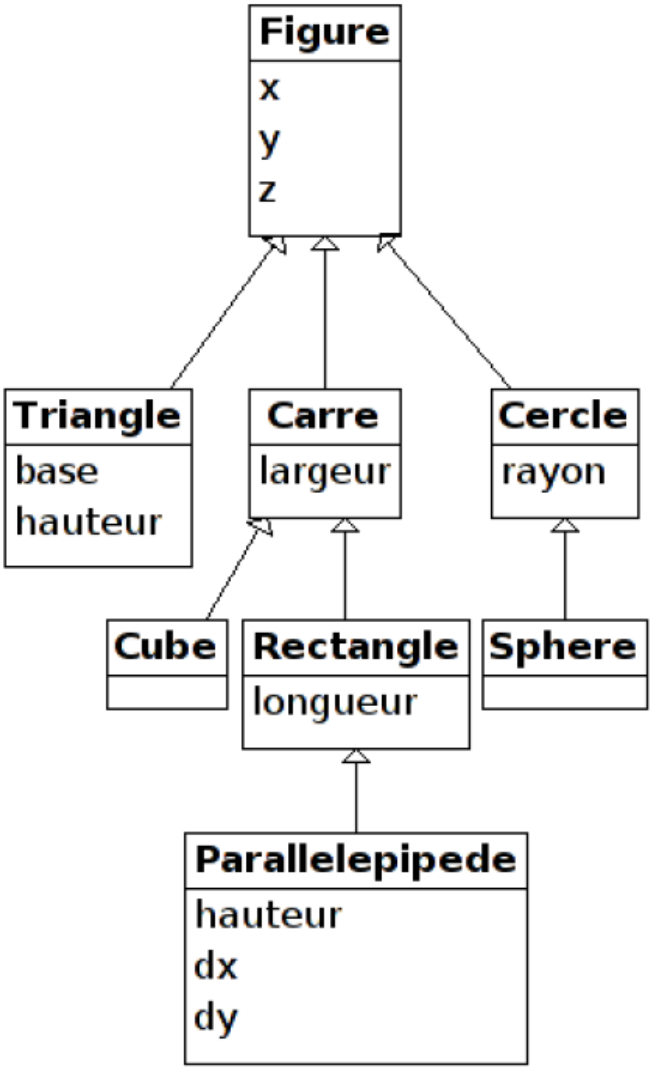
Énoncé:

1. Définir une classe de base pour toutes nos figures : la class Figure qui a trois protected attributs de type double x, y, et z.
2. Créer le constructeur d'initialisation pour la classe Figure, ainsi que la fonction afficher().
3. Créer la fonction description() qui return la chaîne de caractères "Figure".
4. Définir sous forme de classe la figure (héritage public du Figure) géométrique du Triangle avec les attributs protected double base, cote1, cote2, et hauteur.
5. Redéfinir la fonction afficher() et la fonction description() qui doit retourner la chaine de caractère "Triangle".
6. Définir sous forme de classe la figure (héritage public du Figure) géométrique du Carre avec l'attribut protected double largeur.
7. Redéfinir la fonction afficher() et la fonction description() qui doit retourner la chaîne de caractères "Carré".
8. Créer un program de test:

```
Je suis un Triangle
Mes attributs sont : 0 0 0 5 8
Je suis un Carré
Mes attributs sont : 0 0 0 4
```

9. Créer une fonction afficherInfos() qui prend n'importe quel figure en paramètre et affiche les informations et la description exact pour l'objet qu'on passe en paramètre.
10. On propose de traiter au moins le cas des rectangles, cercles, triangles, carrés, sphères, parallélépipèdes rectangles, cubes, ... mais cette liste n'est pas limitative.

Chaque classe devra permettre de mémoriser les données qui permettent de définir une instance, par exemple la longueur des deux cotés d'un rectangle, le rayon de la sphère, etc. Chaque instance devra disposer quand cela a un sens :
– d'une méthode double perimetre() qui renvoie la valeur du périmètre de l'instance sur laquelle on l'appelle ;
– d'une méthode double aire() qui renvoie l'aire de la surface de l'objet concerné ;
– d'une méthode double volume() qui renvoie le volume de la forme concernée.
On considérera que lorsque ces trois notions ne sont pas définies mathématiquement, alors la valeur renvoyée sera nulle. Par exemple, le volume d'un carré sera nul, et le périmètre d'une sphère également.



11. Rendre la classe Figure abstrait en rendons la méthode volume() virtuelle pure.

Formules:

perimetre du cercle: $2\pi r$ et aire du cercle: πr^2.
volume du sphere: $4/3\ \pi r^3$ et aire du sphere: $4\pi r^2$.

Correction: https://github.com/hm43/ExercicesTpCpp.git