Tp 4 C++. Master 1 Physique

1 Les classes en tant que agrégat de données (suite)

Exercise 1

On reprend la classe `Vecteur` du TP3.

1. Munissez-la :
   — d'un destructeur qui libère correctement la mémoire. Faites de sorte que ce destructeur signale sur la console qu'il a été appelé.
   — d'une méthode `affiche()` qui affiche la taille du vecteur et la valeur de ses éléments.
   — d'un constructeur par recopie, qui, lui aussi, signale qu'il est appelé. Prendre les précautions nécessaires quant à la modification de la taille du tableau.
2. Surchargez l'opérateur « = » de deux manières différentes :

```
Vecteur& operator= (const Vecteur &);
Vecteur  operator= (const Vecteur  );
```

   Quelle est la différence entre ces deux façons de faire. Expliquer exactement ce qui se passe dans les deux cas.

Exercise 2

Écrire une fonction « amie » (`friend`) de la classe vecteur qui reçoit en arguments deux vecteurs, et compare leurs dimensions. Ainsi, elle retourne :
— -1 si la taille du premier vecteur est plus petite que celle du second
— 0 si les deux tailles sont égales
— 1 si la taille du premier vecteur est plus grande que celle du second

2 Les classes en tant que composant logiciel

Le but de cette section est de poser les structures de base permettant de manipuler des figures géométriques. Pour cela, on va définir une classe générique `Figure` dont vont dériver les classes `Segment`, `Cercle`, `Rectangle`, etc...
On pourra s'inspirer des classes présentées en cours. On vous rappelle qu'il faudrait créer deux fichiers pour chaque classe, un fichier d'en-tête (`.hpp`) et un fichier source (`.cpp`).

Exercise 3

Écrire une classe `Figure` ne contenant qu'un constructeur, un destructeur et une méthode `print` affichant « Je suis une figure ». C'est la classe de base qui va servir de point de départ pour les autres classes ; elle va contenir toutes les caractéristiques communes aux classes qui en dérivent.

Exercise 4

1. Écrire une classe `Segment` qui dérive de `Figure` et contenant deux instances (données membres) de la classe `Point` (voir TP3) : les deux points correspondent aux extrémités du segment.
2. Munissez-la d'un constructeur qui initialise les deux points.
3. Surchargez la méthode `print` pour qu'elle affiche « Je suis un segment » et les coordonnées des extrémités.

Exercise 5

1. Écrire une classe `Cercle` qui dérive de `Figure` et contenant deux instances de la classe `Point` (le centre et un point sur le cercle).
2. Comme précédemment, surchargez la méthode `print`.
3. Considérez l'extrait de programme suivant :

```
int main()
{
  Figure *a, *b;
  a = new Figure();
  b = new Cercle();
  a->print();
  b->print();
  return 0;
}
```

   a) Pourquoi le compilateur nous laisse-t-il référencer un objet de type `Cercle` par un pointeur de type `Figure` ?
   b) Est-ce que c'est sans risque ?
4. Exécutez le programme ci-dessus et notez ce qu'il affiche. Ensuite, déclarez les méthodes `print` en tant que méthodes virtuelles. Exécutez le programme et comparez avec le résultat précédent, expliquer ce qui s'est passé.