Travaux Dirigés de C++ n°3

Cours de programmation

—Licence d'informatique—

Des structures aux classes

Dans ce TD nous introduisons la notion de classe.

▸ **Exercice 1. (Structures et classes)** *Écrire une structure* `Point` *formée de deux* `double`*, (abscisse et ordonnée) et d'un nom, qui est un caractère. Écrire les fonctions :*

- `init`*, qui permet d'initialiser un point avec une abscisse, une ordonnée et un nom (utiliser les arguments par défaut) ;*
- `print`*, qui affiche des caractéristiques d'un point (nom, abscisse, ordonnée) ;*
- `translate`*, qui pratique la translation d'un point selon deux valeurs de type* `double` *(translation de l'abscisse et de l'ordonnée).*

On souhaite maintenant qu'une fois un point initialisé, il garde son nom pendant toute sa durée de vie. Ceci implique qu'aucune fonction (sauf `init`*) ne puisse changer le nom d'un point (on supposera que la fonction* `init` *n'est appelée qu'une fois sur un point pendant toute sa durée de vie). Modifiez la structure* `Point` *et les fonctions en conséquence.*

Reprendre la définition du type `Point` *en transformant la structure en une classe.*

▸ **Exercice 2. (Composition)** *Définir une classe* `Rectangle` *ayant deux champs de type* `Point`*, représentant les coins supérieur gauche et inférieur droit du rectangle (on suppose que les côtés des rectangles sont parallèles aux axes du repère). Vous définirez les méthodes :*

- `init` *qui construit un rectangle à partir de deux points ;*
- `init` *qui construit un rectangle à partir d'un point et de deux dimensions (largeur et hauteur) ;*
- `print` *qui affiche, entre crochets, les points haut gauche et bas droit définissant le rectangle ;*
- `translate` *qui effectue la translation d'un rectangle par rapport à deux valeurs de type* `double`*;*
- `contains` *qui teste si un point appartient au rectangle.*

Changez maintenant la représentation interne de la classe `Rectangle` *pour un point (de la classe* `Point` *représentant le coin en haut à gauche) et deux dimensions (de type* `double` *représentant la largeur et la hauteur du rectangle).*

▸ **Exercice 3. (Dépendance mutuelle)**

Soit la fonction principale suivante :

```
int main(void) {
  X x;
  Y y;
  x.m(y);
  y.m(x);
}
```

Définir les classes `X` et `Y`, qui contiennt toutes les deux une méthode `m`, de façon à ce que la fonction ci-dessus compile.

▸ **Exercice 4.**

Définir une classe `Droite`. Vous devrez choisir (en fonction des opérations qu'on devra définir sur la classe et de critères que vous jugerez judicieux) une représentation interne à la classe pour représenter les droites (par exemple, une droite est représentable par un vecteur et un point, ou bien deux points, ou bien deux réels a et b qui sont ceux de l'équation $y = ax + b$ de la droite, ou bien trois réels a et b et c qui sont ceux de l'équation $ax + by + c = 0$ de la droite). Vous définirez les méthodes suivantes :

- `init` *qui initialise une droite à partir des réels de son équation ;*
- `init` *qui initialise une droite à partir de deux points ;*
- `print` *qui affiche une représentation d'une droite ;*
- `intersection` *qui retourne l'intersection de deux droites ;*
- `contains` *qui teste l'appartenance d'un point à une droite ;*
- `parallelTo` *qui teste si cette droite est parallèle à une autre ou non.*