

TP3 C++ – M2 ISIFAR

Classes . . .

M. de Falco [defalco@pps.jussieu.fr] – Site de Chevaleret, Bureau 5A16

19 octobre 2009

Préliminaires

Rappels sur les classes

Rappel de la syntaxe des déclarations (Foo.h) :

```
#ifndef FOO_H
#define FOO_H

class Foo {
public:
    Foo(); // constructeur
    tr methode(ta1 , . . . , tan);
private:
    t variable;
};
```

```
#endif // FOO_H
```

et des définitions :

```
#include "Foo.h"
```

```
Foo::Foo() {
    // corps du constructeur
}
```

```
tr Foo::methode(ta1 arg1 , . . . , tan argn) {
    // corps de la methode
}
```

1 Classe Point

1.1 Structure basique

Réalisez une classe Point qui permettra de représenter un point du plan. On prévoira :

1. Un constructeur recevant en argument les coordonnées (type `double`) d'un point.
2. Une méthode (encore appelée fonction membre) `translate` effectuant une translation par rapport à un autre point passé en argument.
3. Une méthode `print` affichant le point sur la sortie standard.
4. Des méthodes `abscisse` et `ordonnée`.

Les coordonnées du point seront privées. On séparera déclaration et définition des méthodes.

Écrivez un petit programme de test.

1.2 Ajout de fonctionnalités

Ajoutez à la classe précédente (en définissant des arguments adaptés) :

1. `scale` qui effectue une homothétie.
2. `rotate` qui effectue une rotation.
3. `rho` et `theta` qui renvoie les coordonnées polaires.
4. Écrire une fonction `invert` qui inverse le point vu comme un nombre complexe.
5. `droite` qui étant donné un autre point renvoie l'équation de la droite passant par ce point depuis l'instance. On définira pour cela une classe `Droite` représentant une équation de droite.
6. Ajoutez à la classe `Droite` une méthode `isIn` qui prend un point et vérifie si il est sur la droite.

A chaque étape il faudra bien sûr écrire des tests.

1.3 Classe Point3D

Reprendre l'exercice précédent en rajoutant une dimension. La classe `Droite` sera remplacée par une classe `Hyperplan`.