Introduction aux classes en C++

Loïc Joly, Michel Simatic, Amina Guermouche

Télécom SudParis

4 Mai 2018

Les classes

1. Définition :

```
class Toto{
  private:
  /* liste d'attributs */
  public:
  /* liste des fonctions */
  }
```

Constructeurs

1. Constructeur par défaut : à définir lorsque la valeur par défaut a un sens

```
class Toto{
  private:
  /* liste d'attributs */
  public:
  Toto(){}
  }
```

2. Tout ce qui n'est pas initialisé par le constructeur prend les valeurs par défaut données par C++
3. Constructeur :

```
class Toto{
  private:
     int att1;
  public:
  Toto(int attribut1) : att1(attribut1){}
}
```

Surcharge des opérateurs (1/3)

1. operator toto(A a, B b) permet de faire a toto b qui est équivalent à operator toto(a,b)
2. +=, = () et [] sont forcément redéfinis à l'intérieur de la classe car ils ont accès aux attributs privés (et peuvent ainsi les modifier).
3. Il est préférable de redéfinir + à l'extérieur de la classe. Sinon l'opérateur ne prend qu'un seul argument. Un des problèmes qui peut se poser est que s'il y a un moyen de faire a + 1 avec a d'un type défini par l'utilisateur, faire 1 + a va tenter d'utiliser le + d'un entier (ce qui provoquera une erreur de compilation) alors que a+1 prend le + de la classe de a. Définir + à l'extérieur de la classe permettra de faire la surcharge dans les deux cas (en faisant 2 sucharges différentes).

Surcharge des opérateurs (2/3)

4. Il est préférable de définir == à l'extérieur de la classe. Une des raisons est que les objets de type const ne peuvent pas être comparé :

```
class Toto{
      bool operator==(const Toto &a)
       {
         return true; /* resultat bidon*/
       }
};
int main()
{
  Toto a; const Toto b;
  bool c = (b == a); /* erreur de compilation
       car == n'est pas defini avec l'
      operande de gauche de type const*/
}
```

Surcharge des opérateurs (3/3)

5. Remarque : Il est possible de résoudre le problème en ajoutant un const :

```
class Toto{
  bool operator==(const Toto &a) const
  {
    return true;
  }
};
```

Cependant, cette solution est dépendante de l'implémentation interne de la classe qui pourrait changer