Introduction au C++

Qualificateur *const*

Objet *const*

- Principe
  - (a) un objet déclaré *const* est *non-modifiable* une fois initialisé.
  - (b) il doit être initialisé dès sa déclaration.

```
int i ;                 // OK    i est non-const.
const int j ;           // KO !! cf. (b)

const int k = 2 ;       // OK    cf. (b)
k++;                    // KO !! cf. (a)
```

- Intérêt
  - Dès qu'une variable est déclarée *const*, le compilateur:
    - traque toute instruction de modification de la variable
    - détecte une erreur si c'est le cas.

Pointeur sur objet *const*

- Principe
  - (a) un objet déclaré *const* est *non-modifiable* une fois initialisé.
  - (b) il doit être initialisé dès sa déclaration

```
const int *pi = &i; // OK (non-const → const)
i++;                // OK i est non-const.
(*pi)++;            // KO !! cf. (a)
```

- const_cast ??? uniquement C++ (pas C).
- Pointeur *const* sur objet (l'objet n'étant pas forcément *const*)

Méthode *const (1)*

- Principe
  - une méthode est déclarée *const* lorsqu'on souhaite exprimer qu'elle ne modifie pas l'objet *this*.
  - Une méthode *const* peut être appelée sur un objet *const* ou non-*const*.
  - *Mais seule une méthode const peut être appelée sur un object const.*

- La déclaration de méthode *const* protège le programmeur
  - Exemple

```
class Rectangle {
public:
   Rectangle( double dx,
              double dy );
   bool isSquare() ;

private:
   double _dx ;
   double _dy ;
};
```

```
bool Rectangle::isSquare() {
   return (_dx = _dy) ;
};
```

Erreur possible: affectation au lieu de

```
bool Rectangle::isSquare() const {
   return (_dx == _dy) ;
};
```

L' erreur est détectable si on déclare *const*.

Méthode *const* (2)

- La déclaration de méthode *const* protège le programmeur:
  - le compilateur vérifie que l'implémentation de la méthode ne modifie pas l'objet *this*:
    - soit directement
    - soit par appel de méthode non-*const*

```
class A {
private:
   meth1() const;
   meth2();
   int i;
};
```

const est placé dans le *header*.

const est rappelé dans le *source*.

```
void A::f() const {
   i++;            // KO !!
   meth1();        // OK
   meth2();        // KO !!
}
```

- Remarque:
  - Dans une méthode non-*const* d'une classe A, *this* est de type « *A* * ».
  - Dans une méthode *const* d'une classe A, *this* est de type « *const A* * » .
- Recommandations
  - Déclarer *const* sur toute méthode qui ne modifie pas *this*.
  - Les *accesseurs* sont quasiment toujours *const*.