Classes (suite) [cm]
Support de Cours

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Version 20 mai 2018

Table des matières

C++ - Classes (suite)

Mots-Clés Conception Objet, Classe, Vecteur ■
Requis Axiomatique impérative, Classes ■
Difficulté • ○ ○ (30 min) ■

Introduction
Ce module complète la notion de classe (attributs/méthodes de classe) et introduit la classe `Vecteur`.

1 Membres de classe

1.1 Attribut de classe

Attribut de classe

Dit aussi **attribut partagé**, il est créé en **un seul exemplaire** et **partagé** par l'ensemble des objets de la classe. Il existe même lorsqu'aucune instance n'est déclarée et il peut être *privé* ou *public*.

Attribut de classe

```
// déclaration
class K
{
  static T attr;
} ;
// initialisation
T K::attr( ... )
```

Explication

Déclare un attribut de classe nommé `attr` de la classe `K`.

C++ : Instanciation d'un attribut de classe

Il doit être **initialisé à l'extérieur** de la classe.

1.2 Méthode de classe

Méthode de classe

Méthode qui peut être appelée **indépendamment** de tout objet (c.-à-d. à partir du nom de la classe et de l'opérateur de portée). Elle peut être *privée* ou *public*.

Méthode de classe

```
// prototype
class K
{
   static T meth(...);
} ;
// implémentation
T K::methode(...)
{
  ...
}
```

Explication

Déclare une méthode de classe nommée `meth` de la classe `K`.

Membres accessibles = Membres de classe uniquement

Une méthode de classe peut être appelée alors qu'aucun objet de la classe n'existe. Elle ne peut donc qu'accéder à d'autres méthodes/attributs de classe. Elle **ne peut utiliser** directement **ni** les attributs d'instances (non partagés), **ni** les méthodes d'instances (non partagées).

1.3 Exemple : Application typique

L'application typique des membres de classe est le compteur d'instances. Dans le cas par exemple d'une classe `Point`, on aura :

```
#ifndef POINT_HEADER
#define POINT_HEADER
class Point
{
  public:
    Point(int x, int y)
    : m_x(x), m_y(y){
       m_npoints += 1;
    }
    ~Point(){
       m_npoints -= 1;
    }
    int x() const{
      return m_x;
    }
    int y() const{
      return m_y;
    }
    void deplacerDe(int incrx, int incry);
    void deplacerVers(int x, int y);
    static int npoints(); //<- méthode de classe
  private:
    int m_x;
    int m_y;
    static int m_npoints; //<- attribut de classe
};
```

```
#include "Point.hpp"
// Definition de l'attribut de classe m_npoints
// notez que l'on ne repete pas le mot clé static
// la valeur d'initialisation est OBLIGATOIRE
int Point::m_npoints = 0;
void Point::deplacerDe(int incrx, int incry){
  m_x += incrx;
  m_y += incry;
}
void Point::deplacerVers(int x, int y){
  m_x = x;
  m_y = y;
}
int Point::npoints(){
  return m_npoints;
}
```

Explication
A FINIR..... ■

2 Classe Vecteur

2.1 Classe Vecteur

Classe Vecteur
Modélise les tableaux dynamiques, c.-à-d. dont la taille peut varier.

C++ : Classe Vecteur

- Pour l'utiliser : `#include <vector>`
- Déclaration d'un vecteur d'éléments de type `T` : `vector<T> v;`
- Déclaration et pré-réservation d'une taille a priori : `vector<T> v(n);`
- Déclaration, pré-réservation d'une taille a priori et initialisation de tous les éléments à une valeur : `vector<T> v(n,val);`
- Accès à l'élément `k` : `v[k]` (avec $k = 0..$)

C++ : Tableaux dynamiques multidimensionnels
Les tableaux à plusieurs dimensions sont des tableaux de tableaux et cela est valable aussi bien pour les tableaux de taille fixe que pour les tableaux dynamiques. Cependant la syntaxe de la déclaration est particulière : un espace **est obligatoire** entre les « > » fermants sous peine d'ambiguïté avec le symbole « >> » d'extraction de flux !.

```
vector<vector<... <T>...> > v; //<- ATTENTION
```

L'accès à un élément particulier s'écrit de façon similaire :

```
v[k1][k2]...[kM]; // avec 0<= ki < v[i].size()
```

2.2 Principales opérations

Les objets se manipulent essentiellement à travers des méthodes. Les méthodes qui suivent permettent d'obtenir la même chose qu'avec un tableau normal, mais en gagnant la possibilité d'ajouter une nouvelle case ou de supprimer une case existante.

- Taille (nombre d'éléments) : `v.size()`
- Prédicat de tableau vide : `v.empty()`
- Supprimer tous les éléments : `v.clear()`
- Supprimer le dernier élément : `v.pop_back()`
- Ajouter un nouvel élément à la fin : `v.push_back(val)`

3 Exemple : Manipulation d'un vecteur