Programmation orientée objet

Variables et fonctions statiques

Georges Abou Khalil - École Polytechnique de Montréal

Plan

- Variables globales statiques
- Variables locales statiques
- Fonctions globales statiques
- Attributs statiques
- Méthodes statiques

Mot clef « static »

- **Tout élément statique est déclaré en C/C++ à l'aide du mot clef « static »**
- Le sens du mot-clef « static » diffère selon le type d'élément auquel il est associé

Variables globales statiques

- Rappelons qu'une variable globale est une variable déclarée à l'extérieur de toute fonction et est donc visible dans toutes les fonctions
- Une variable globale statique est une variable globale qui est uniquement visible à l'intérieur du fichier dans lequel elle est déclarée
- Elle ne peut donc pas être exportée vers d'autres fichiers
- Dans le cadre de ce cours, ce type de variable n'est pas utilisé

Variables locales statiques

- **Une variable locale déclarée statique n'est pas détruite lors de la sortie de la fonction**
- Ainsi, lorsque la fonction est appelée à nouveau, la valeur de la variable statique est celle qu'elle avait à la fin du dernier appel de la fonction
- Elle se comporte donc comme une variable globale, mais n'est visible qu'à l'intérieur de la fonction où elle est déclarée
- La déclaration d'une variable statique se fait de la manière suivante :

```
static type nom = valeurInitiale;
```

Variables locales statiques (suite)

- Reprenons l'exemple de la suite de Fibonacci :

```
int fibonacci()
{
    static int n1 = 0;
    static int n2 = 1;
    int temp = n1 + n2;
    n1 = n2;
    n2 = temp;
    return n1;
}
```

Variables locales statiques (suite)

Appel	Valeur de n1	Valeur de n2
1er	0	1
2ème	1	1
3ème	1	2
4ème	2	3

Fonction globales statiques

- Une fonction globale statique, comme une variable globale statique, est uniquement visible à l'intérieur du fichier dans lequel elle est déclarée
- Dans le cadre de ce cours, ce type de fonction n'est pas utilisé

Attributs statiques

- Un attribut statique est un attribut qui est partagé par toutes les instances de cette classe
- Ainsi, il en existe une et une seule version, et sa valeur est la même pour tous les objets de la classe
- Un tel attribut est appelé *attribut de classe*

Attributs statiques (suite)

- Les attributs statiques d'une classe sont créés au démarrage du programme. Ils **existent donc même si aucune instance de la classe n'est créée**
- Ils peuvent être accédés comme un attribut normal (opérateurs . et **->**) ou bien directement, à l'aide de l'opérateur **::**
- Comme tout attribut, un attribut statique possède une visibilité (public, private ou protected)
- Les attributs statiques doivent être déclarés dans le fichier cpp et on peut alors leur spécifier une valeur initiale

Attributs statiques - Exemple

- Classe.h :

```
class Classe
{
  private:
    static int varStatique_;
};
```

- Classe.cpp :

```
int Classe::varStatique_ = 0;
```

Méthodes statiques

- Une méthode statique est une méthode qui n'opère pas sur des objets
- Une telle méthode peut être appelée à partir d'un objet (opérateurs . et `->`) ou à l'aide de l'opérateur `::`
- Même si une méthode statique est appelée à partir d'un objet, elle ne possède pas de pointeur **`this`**
- Les méthodes statiques sont appelées méthodes de classe

Méthodes statiques (suite)

- Comme les méthodes statiques n'ont pas de pointeurs **`this`**, elles se comportent comme des fonctions globales
- Les méthodes statiques ont cependant l'avantage d'avoir accès à tous les éléments de la classe, comme toute autre méthode
- Les méthodes statiques sont donc généralement utilisées pour manipuler des attributs statiques qui sont privés (ou protégés) et donc non accessibles en dehors de la classe (et de ses dérivées)

Méthodes statiques - Exemple

- Un exemple simple d'utilisation de méthodes et attributs statiques est l'ajout d'un compteur d'instances
- À chaque fois qu'un objet est créé, un compteur est incrémenté (dans le constructeur)
- À chaque fois qu'un objet est détruit, un compteur est décrémenté (dans le destructeur)
- Le compteur est un attribut privé statique de la classe, initialisé à 0
- Il existe une méthode statique pour retourner la valeur du compteur

Méthodes statiques - Exemple

- Employee.h :

```
class Employee
{
public:
  Employee(/*...*/) { compteur_++; /*...*/}
  ~Employee() { compteur_--; /*...*/}
  static int getCompteur() { return compteur_; }
private:
  static int compteur_;
};
```

Tous les objets de cette classe partageront le même attribut `compteur`

- Employee.cpp :

```
int Employee::compteur_ = 0;
```

Méthodes statiques - Exemple

- Main.cpp :

```
int main()
{
   cout << Employee::getCompteur() << endl;
   Employee* ptr1 = new Employee();
   cout << Employee::getCompteur() << endl;
   Employee* ptr2 = new Employee();
   cout << Employee::getCompteur() << endl;
   delete ptr2;
   cout << Employee::getCompteur() << endl;
   delete ptr1;
   cout << Employee::getCompteur() << endl;
}
```

Méthodes statiques - Exemple

- Affichage :

```
0
1
2
1
0
```