Analyse et Programmation Orientées Objets / C++

Fonctions et méthodes

Les fonctions du langage C (1)

➡ Déclaration et définition en C++

- Hors du **corps de la déclaration de toute classe**

- Ne sont jamais des fonctions membres
- Possibilité de paramètres formels de type Classe
- Possibilité de retour de type Classe
- Possibilité de paramètres effectifs objets
- Autres règles identiques aux usages antérieurs

Les fonctions en langage C (2)

➡ Appel des fonctions en C++

- Usage interdit de l'opérateur d'application .

- Règles identiques aux usages antérieurs
- Possibilité d'utiliser des pointeurs de fonctions

Les méthodes d'instance (1)

- **Déclaration des méthodes en C++**
  - Dans le **corps de la déclaration de la classe**
  - Section publique ou privée
  - Fonctions membres
  - Déclarées par prototypes
  - Cas particuliers des constructeurs et des destructeurs
  - Autres règles identiques aux fonctions du langage C

Les méthodes d'instance (2)

➡ Définition des méthodes en C++

- Dans un module indépendant

- Usage de l'Opérateur de Résolution de Portée ::
- Exceptionnellement dans le corps de la classe
- Cas particulier du constructeur par défaut
- Autres règles identiques aux fonctions du langage C

Les méthodes d'instance (3)

➡ Invocation des méthodes en C++

- **Par l'opérateur d'application . sans exception**

- Obligatoirement sur une instance de la classe
- Partout où peut être appelée une fonction en langage C
- Cas particulier des constructeurs
- Cas particulier des destructeurs
- Autres règles identiques aux fonctions du langage C

Les méthodes de classe (1)

➡ **Définition et intérêt**

- Fonctions membres indépendantes de toute instance
- Pas d'usage possible du pointeur *this*
- Permettent d'initialiser **des attributs de classe**

Les attributs de classe (1)

➡ Intérêt et déclaration

- Attributs partagés par toutes les instances

- Existent indépendamment des instances de la classe
- Initialisés à 0 par défaut
- Usage du **spécificateur *static* en préfixe**

Les attributs de classe (2)

➡ **Moyens d'accès**

- Par l'opérateur de résolution de portée ::

- Par l'opérateur d'application .
- Possibilité d'initialisation hors d'un corps de fonction

Les méthodes de classe (2)

➡ **Déclaration en C++**

- Dans le corps de la déclaration de la classe

- Usage du **spécificateur *static* en préfixe**
- Autres règles identiques à celles des fonctions

Les méthodes de classe (3)

➡ **Définition des méthodes en C++**

- Dans un module indépendant

- Usage de l'Opérateur de Résolution de Portée ::
- Exceptionnellement dans le corps de la classe
- Autres règles identiques aux fonctions du langage C
- Pas de répétition du spécificateur *static*

Les méthodes de classe (4)

➡ Invocation en C++

- **Par l'opérateur :: suivi du nom de la méthode**

- Sur une instance de la classe par l'opérateur .
- Autres règles identiques aux fonctions du langage C

Edition de liens des méthodes (1)

- **Ligature statique**
  - **Mécanisme standard en langage C**
  - Application standard aux méthodes en C++
  - Génération d'un CALL direct par le compilateur
  - Problème posé par l'héritage des méthodes

Edition de liens des méthodes (2)

➡ **Ligature dynamique**

- **Mécanisme propre à l'extension C++**

- Permet de résoudre le problème de la surcharge
- Génération d'un **CALL indirect** par le compilateur
- Editeur de liens fixe la valeur du pointeur intermédiaire

Edition de liens des méthodes (3)

➡ **Le spécificateur *virtual***

- **Spécificateur propre à l'extension C++**

- Utiliser en préfixe d'une fonction à surcharger
- Met en œuvre la ligature dynamique de la méthode
- Généralise le polymorphisme

Le spécificateur virtual

➡ **Règles d'utilisation**

- **Un constructeur ne peut pas être virtuel**
- Un destructeur peut être virtuel

A utiliser systématiquement dans une classe parent quand la méthode a toutes les chances d'être surchargée dans la ou les classes dérivées

Les fonctions virtuelles pures

➡ **Définition**

- **Déclarées dans une classe sans y être définies**
- Syntaxe de la déclaration :

virtual void *NomFonction* ()=0;

Les classes abstraites

➡ Définition et utilisation

- **Contient au moins une fonction virtuelle pure**

- Classe pour laquelle on ne peut créer aucune instance
- Sert de base pour d'autres classes dérivées
- Exemple (en géométrie) : la classe Figure