Analyse et Programmation Orientées Objets / C++

Types et classes génériques

Générique (1)

➡ **Définitions de l'adjectif (Larousse)**

Qui appartient au genre, à tout genre.

Se dit d'un mot dont le sens englobe toute une catégorie d'êtres ou d'objets (Déf. en linguistique).

Nom commun de toutes les espèces du même genre (Déf. en biologie).

Elément d'un ensemble pris sous sa forme générale (Déf. en math).

Générique (2)

➡ **Définition technique (Informatique)**

Dont le type et/ou la composition n'est pas arrêté.

Dont la génération de code exécutable est paramétrée.

Dont la composition et/ou l'organisation peut varier en cours d'exécution (avec ou sans run time !).

Objectifs de la "généricité"

- **Retarder la description complète d'un modèle**

  Modèle de données (collections, classes).

  Modèle de traitement (fonctions, méthodes).

Moyens techniques déjà étudiés

- **Modèles de données**

  Usage des pointeurs banalisés (**void***)

- **Modèles de traitement**

  Usage des pointeurs de fonctions

  Usage de la surcharge (niveau moindre)

Type générique (ou meta type)

- **Définition (C++)**

  Type de données qui peut être fixé postérieurement à la définition des structures qui le contiennent.

  Usage du spécificateur **template** (patron) pour formaliser le caractère générique d'un type.

Usage du spécificateur template

- **Syntaxe générale**

  Fonction dont l'un au moins des paramètres formels est de type générique.

  Mécanisme d'instanciation des **templates** lors de la génération de code.

Fonction générique

- **Définition (C++)**

  Fonction dont l'un au moins des paramètres formels est de type générique.

  Mécanisme d'instanciation des **templates** lors de la génération de code.

Classe générique (ou meta classe)

- **Définition (C++)**

  Classe dont l'un des attributs ou l'une des méthodes au moins est générique.

Exemple de la classe Pile

➡ Spécifications fonctionnelles

Gestion d'une file LIFO

- Taille maximum fixée à l'exécution
- Indépendance vis à vis du type des objets gérés
- Contrôle de la validité
- Accesseurs cardinal et taille
- Méthodes empiler, depiler et vider
- Libération mémoire
- Gestion intégrée des erreurs

Types génériques (1)

➡ **Définition (C++)**

Type de données qui peut être fixé postérieurement à la définition des structures qui le contiennent.

Usage du spécificateur **template** (patron) pour formaliser le caractère générique d'un type.

Synonyme : meta type

Types génériques (2)

➡ **Syntaxe générale**

template < **class|typename** *nom* [=type]
[, **class|typename** *nom* [=type]]
>

Types génériques (3)

- **Exemples**

template < class E, typename F, *class* G=*int* >

E, F et G sont trois types génériques.

Classes génériques (1)

- **Définition (C++)**

  Classe dont l'un des attributs ou l'une des méthodes au moins est générique.

  Synonyme : meta classe ou patron de classe

Fonctions génériques (1)

- **Définition (C++)**

  Fonction dont l'un au moins des paramètres formels ou le retour est de type générique.

  Mécanisme d'instanciation des **templates** lors de la mise en oeuvre.

  Aucune génération de code avant instanciation.

Instanciation des templates

➡ Définition (C++)

A la première instanciation d'une classe générique, tous ses types génériques doivent être instanciés pour l'objet à construire (en types prédéfinis ou définis).

Instanciation **implicite** ou **explicite**

L'instanciation provoque la génération de code du constructeur invoqué.

Instanciation implicite (1)

Réalisée automatiquement par le compilateur quand le type des paramètres effectifs de l'instance créée définissent sans aucune ambiguïté le type cible.

Instanciation implicite (2)

```
template  < class G >
class Paire {
private :
G    m_A;
G    m_B;

public :
---


};
```

Instanciation implicite (3)

```
void main () {
Paire d1(1, -2);
Paire d2(-7.1f, 1.0f);
    ---
}
```

Instanciation explicite

Le programmeur fixe explicitement le type cible lors de la création de l'objet (instance d'une classe générique).

Génération de code (1)

- **Attention**

  En l'absence du contexte d'utilisation de la classe, le compilateur ne peut pas générer le code (*.obj) des fonctions *génériques*.

  Le type <**G**> des éléments manipulés reste inconnu jusqu'à la création de chaque instance de la classe générique !

Génération de code (2)

- **Plusieurs codes binaires**

  Chaque fonction virtuelle donne lieu à la génération d'un code binaire (*.obj) distinct pour instance du type <G>.

  Cette difficulté est inhérente au concept de template.

Génération de code (3)

➡ Compilation séparée (1)

Les fonctions et les méthodes génériques s'accommodent mal de la compilation séparée.

Solution trouvée avec la notion d'exportation de classe (mot clé **export**).

Vérifier la capacité de chaque environnement de programmation à gérer la spécification **export**.

Génération de code (4)

- **Compilation séparée (2)**

  Une solution alternative consiste à générer un fichier *.obj unique.

  Cette solution ne nécessite pas obligatoirement d'abandonner la répartition des codes sources dans plusieurs fichiers distincts.

  Usage de la directive include étendue aux fichiers sources.

Génération de code (5)

- **Compilation séparée (3)**

  Il faut contourner l'impossibilité d'inclusion multiple de la déclaration des attributs d'une classe C++.

  Mettre en œuvre les **directives conditionnelles** du préprocesseur C(#ifndef, #endif).

Compilation conditionnelle (1)

- **Définition d'une variable de compilation**

  # define _PILE_

  Variable _PILE_ choisie au gré du programmeur.

  Définition à placer en tête du fichier Pile.h (après les directives include)

Compilation conditionnelle (2)

- **Contrôle d'une variable de compilation**

```
# ifndef _PILE_
# include "Pile.h "
# endif
```

Définition à placer en tête des fichiers *.cpp

Classes génériques (2)

Interactions avec les autres concepts

- Une classe générique peut être dérivée
- Une classe générique peut être composée
- Une classe générique peut être abstraite
- Une classe générique peut être déclarée amie
- Une classe non générique peut contenir des fonctions génériques

Fonctions génériques (2)

Interactions avec les autres concepts

- Une fonction générique peut être surchargée
- Surcharge par fonction normale ou générique
- Une fonction générique peut être déclarée amie
- Une fonction virtuelle ne peut pas être générique
- Une fonction inline peut être générique
- Les destructeurs ne peuvent pas être génériques

La classe Pile – V2 (12)

```
G Pile <G>:: empiler (G pE) {
   if (nok()) throw -2.0;
   if (pE==NULL) throw -2.1;
    if (m_sommet == taille) throw -3.0;
    m_pT [m_sommet++] = pE;
    return pE;
}
```

La classe Pile – V2 (13)

```
G Pile <G>:: depiler () {
    if (nok()) throw -2.0;
    if (m_sommet == 0) throw -3.0;
    return  m_pT [--m_sommet] ;
}
```

Au-delà des classes génériques

➡ **Que reste t-il de non paramétrable en C++?**

Classe C++ dont le nombre et le type des attributs pourraient varier en cours d'exécution : rêve ou réalité ?

Classe C++ dont le nombre de méthodes pourraient varier en cours d 'exécution : rêve ou réalité ?

Bibliographie

- **Cours C/C++ Christian Casteyde**
- **Cours C/C++ Bruce Eckel**
- **http://rperrot.developpez.com/articles/c/genericite/**