PR-4207 Atelier génie logiciel : C++ et généricité

Thierry Géraud
theo@lrde.epita.fr

2015

Mise en bouche

• Écrire un programme qui affiche `hello world!`

• Quelles différences avec un `printf` du C, avec un `System.out.println` de Java ?

• Peut-on écrire un programme tel que :
  — le source ne contienne pas la chaîne `51`
  — l'exécutable contienne l'entier 51 ?

• Est-ce possible de faire calculer $n!$ au compilateur ? Si oui, comment ? Si non, pourquoi ?

1 Deux classes et des opérateurs

• Écrire une classe `vector` (les attributs imposés sont `n` la dimension du vecteur et `int* data` le buffer des coordonnées). On veut pouvoir écrire :
  — `vector v1(3), v2(3); // dim 3`
  — `v1[1] = 1`
  — `v2 = v1`
  — ... → `exo1/vector.hh`

• Écrire une classe `matrix` (les attributs imposés sont `nl` et `nc` les nombres de lignes et colonnes, et `int* data` le buffer des cellules). On veut pouvoir écrire :
  — `m(i, j) = 1`
  — ... → `exo1/matrix.hh`

• On veut pouvoir écrire :
  — `v = 2 * v + 1;`
  — `m = m1 * m2;`
  — `v = m * v;`
  — ... → `exo1/ops.hh`

• Quels sont les problèmes possibles de robustesse de ce programme ?

• Écrire une procédure qui remplit les cellules d'une matrice avec une valeur `a` ?
→ `exo1/fill_matrix.hh`

• Comment peut-on écrire une unique procédure pour remplir les cellules d'un vecteur ou d'une matrice avec une valeur `a` ? → `exo1/fill.hh`

2 Un "vrai" modèle objet

• Copier `exo1/vector.hh` et `exo1/matrix.hh` dans `exo2/`

• Faire que `vector` et `matrix` dérivent respectivement de `abstract_vector` et `abstract_matrix`.

• Est-ce que les méthodes abstraites vérifient bien le principe d'ouverture/fermeture ?

• Ajouter les classes concrètes `one_vector` (pour des vecteurs multiples de $\mathbf{1}$; attributs `n` et `int v`) et `dia_matrix` (pour des matrices diagonales ; attributs `n` la dimension et `vector v` les valeurs de la diagonale). → `exo2/one_vector.hh` et `exo2/dia_matrix.hh`

• Est-ce que la routine `exo1/fill.hh` est réutilisable ? Pourquoi / pourquoi pas ?

• On veut de nouveau pouvoir utiliser des produits "matrice $*$ vector" :
  — Quelles sont les différentes stratégies de mise en œuvre de cette fonctionnalité ?
  — Qu'est-ce qui est génant / qui pose problème ?
  — Quels sont les avantages et inconvénients des solutions ?

• On se pose d'autres questions :
  — Est-ce possible d'y arriver avec un unique code ?
  — Comment garantir que ce code est performant ?
  — Comment éviter les fuites de mémoire ?

• Maintenant on s'y colle... → `exo2/ops.hh`

• Faites planter votre programme sans que le compilateur ne puisse rien dire...

3 On itère

• Souvenez-vous du *design pattern* "itérateur"...

• Comment peut-on maintenant écrire une unique procédure pour remplir les cellules d'un vecteur ou d'une matrice avec une valeur `a` ? → `exo2/fill.hh`

• Pouvez-vous utiliser la routine `std::fill` de la bibliothèque standard du C++ ? Pourquoi ?

4 Finalement

• Quelles leçons peut-on tirer de ces exercices (en termes de génie logiciel d'une part et de concepts de programmation d'autre part) ?

• Quel est le lien (caché ?) entre les exercices et la mise en bouche ?