Construction Objets avancée

Giuseppe Lipari

February 18, 2018

1 Instructions

Vous devez rendre sur github le code demandé avec un fichier README qui contient :

- Vos noms ;
- Pour chaque question :
  - Si vous avez réussi à coder les fonctionnalités demandées
  - La liste de tests de régression correspondants à la question

2 TP 4: Templates

Le but de ce TP est de prendre confidence avec la programmation de *templates*.

L'intersection de deux ensembles A et B est l'ensemble C qui contient tous les éléments qui sont présent dans le deux ensembles:

$$C = A \cap B = x | x \in A \wedge x \in B$$

L'union de deux ensembles A et B est l'ensemble C qui contient tous les éléments qui sont présent dans un de deux ensembles:

$$C = A \cup B = x | x \in A \vee x \in B$$

L'ensemble C ne contient pas de doublons.

Vous devez écrire ces deux fonctions de manière la plus générale possible.

2.1 Question 1 : fonctions sur vecteurs

Écrire deux fonctions, `set_intersection` et `set_union` que, à partir de deux vecteurs d'entiers, retournent un vecteur que contiens l'intersection (l'union) de deux vecteurs d'entiers.

```
vector<int> set_intersection(const vector<int> &a, const vector<int> &b);
vector<int> set_union(const vector<int> &a, const vector<int> &b);
```

Tester les deux fonctions.

(Vous pouvez supposer que les deux vecteur en entrée ne contient pas de doublons).

2.2 Question 2: utilisons les iterateurs

Généraliser les fonction pour prendre comme paramètres des *iterator*. Le résultat doit être écrit sur un iterator aussi.

```
void set_intersection(vector<int>::const_iterator a_begin,
        vector<int>::const_iterator a_end,
        vector<int>::const_iterator b_begin,
        vector<int>::const_iterator b_end,
        vector<int>::iterator c_begin);
```

La fonction sera utilisé comme dans le programme suivant:

```
vector<int> vec_a = {1, 2, 3, 4};
vector<int> vec_b = {3, 4, 5, 6};
vector<int> vec_c;

set_intersection(begin(vec_a), end(vec_a), begin(vec_b), end(vec_b),
            back_inserter(vec_c));
```

La fonction union a le même prototype, et on l'utilise de la même manière.
Testez les deux fonctions.

2.3 Question 3: template

Généralisez les fonctions développé dans la question 2 en utilisant des templates.

Testez les fonctions sur des vecteurs et des listes d'entiers.

2.4 Question 4: objects

Créer de vecteurs et des listes d'objets de type `MyClass`, et essayez de faire des intersections et des unions avec les fonctions développées dans la Question 3. Vérifiez que tout est correct.

Essayez de faire l'intersection d'un vecteur d'objets `MyClass` avec un vecteur d'entiers. Quel est le résultat ? Qu'est-ce qu'il passe si on échange l'ordre des paramètres ?

Modifier la classe `MyClass` pour empêcher la conversion automatique de int à `MyClass`