Laboratoire 15: Liste générique

Durée du laboratoire: 4 périodes. A rendre le jeudi 21 avril 2016, au début de la séance de laboratoire.

1. Classes List et Iterator

1. Définir la classe `List` générique doublement chaînée permettant de stocker des listes d'objets ou de pointeurs sur des objets. Celle-ci devra entre autres proposer les fonctionnalités suivantes:
   - Constructeur sans paramètres,
   - Constructeur de copie,
   - Surcharge des opérateurs `=` (affectation) et `[]` (accès à un élément de la liste),
   - Méthode `size()` rendant le nombre d'éléments de la liste,
   - Méthodes d'insertion `insert(T o)` (en début de liste) et `append(T o)` (en fin de la liste),
   - Méthodes de suppression `remove(int index)` et `remove(T o)`,
   - Méthodes `begin()` et `end()` rendant un itérateur référençant le début ou la fin de la liste,
   - Méthode `find(T o)` permettant de rechercher un élément dans la liste et rendant un itérateur positionné sur le premier élément correspondant dans la liste ou, sinon, sur la fin de la liste.
2. Définir égalemnent la classe générique `Iterator` permettant de parcourir des listes. Celle-ci devra entre autres proposer les fonctionnalités suivantes:
   - Surcharge des opérateurs `++` et `--` permettant de passer à l'élément suicant ou précédent de la liste,
   - Surcharge de l'opérateur `*` afin de pouvoir obtenir l'élément de la liste sur lequel est placé l'itérateur,
   - Surcharge des opérateurs `==` et `!=` pour comparer la position de deux itérateurs.

Exemple d'utilisation de ces classes:

```
int main()
{
  List<string> l;

  l.append("un");
  l.append("deux");
  l.append("trois");

  for (List<string>::Iterator it = l.begin(); it != l.end(); it++)
    cout << *it << " ";
  cout << endl;
}
```

2. Travail à effectuer

Implémenter les classes `List` et `Iterator` et définir un programme testant toutes les fonctionnalités de ces clases.

Il devra utiliser une liste d'objets (p.ex. des `string`) ainsi qu'une liste de pointeurs sur des objets instanciés dans différentes classes d'une hiérarchie donnée (p.ex. des `Person`) . Pour cette dernière s'assurer du fonctionnement correct du mécanisme de liaison dynamique.