Laboratoire 3: Liste générique

Durée du laboratoire: 8 périodes. A rendre le jeudi 2 mai 2019, au début du cours.

1. Classes

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 avec une liste d'initialiseurs,
   - 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(const T& o)` (au début) et `append(const T& o)` (à la fin),
   - Méthodes de suppression `removeAt(size_t index)` et `remove(const T& o)`,
   - Méthodes `begin()` et `end()` rendant un itérateur référençant le premier élément de la liste ou le dernier élément la liste,
   - Méthode `find(const T& o)` permettant de rechercher un élément dans la liste et rendant l'indice du premier élément correspondant dans la liste ou, sinon, -1,
   - Affichage dans un flux de la liste et de son contenu.

   Remarque: gérer le fait qu'une liste puisse être déclarée conme constante (non modifiable).

2. Définir égalemnent la classe générique `GenericIterator` permettant de parcourir des listes. Celle-ci devra entre autres proposer les fonctionnalités suivantes:
   - Surcharge des opérateurs `++` et `--` permettant de passer à l'élément suivant 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 de l'opérateur `->` d'accès à une propriété,
   - 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;
  // Affichage: un deux trois

  const List<int> c = { 42, 3, 14 };
  for (List<int>::ConstIterator it = c.end(); it != c.begin(); --it)
    cout << *it << " ";
  cout << endl;
  // Affichage: 14 3 42
}
```

2. Travail à effectuer

Implémenter les types `List`, `GenericIterator`, `Iterator` et `ConstIterator` et définir un programme testant leurs fonctionnalités.

Il devra utiliser, entre autres, 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.

Rapport

- Diagramme de classes,
- Documentation de vos choix de conception,
- Expliquer les éventuels avantages et/ou inconvénients de votre conception des types gérant les itérateurs.