

TP n° 5 de C++

Liste chaînée dynamique en C++ - notion élémentaire d'itérateur Généricité sur une structure de donnée dynamique

- 1) Ecrire dans les mêmes fichiers (hpp et cpp) le code C++ pour 3 classes : Cellule (*Cell*), Liste (*List*) et un itérateur de liste (*ItList*). Ces classes ne seront pas implémentées sous forme de classe imbriquée mais on considère que la classe Cell a pour amie la classe List et la classe ItList (à la différence de la STL qui imbrique les classes – concept de « nested classes » qui n'est pas présent dans les concepts objets).

La cellule « Cell » possède un attribut de type entier, et un pointeur sur la cellule suivante (*_next*) (on considère une liste simplement chaînée).

La classe « List » possède comme attribut une tête (*_first*) et une fin de liste (*_last*). La fin de liste pointe sur le dernier élément de la liste (*_first = _last = NULL* quand la liste est vide).

Vous implémenterez des méthodes telles que celles proposées dans la STL (Standard Template Library). Pour l'instant, implémentez simplement des méthodes pour :

- ajouter un élément en tête (*push_front*) ou en fin (*push_back*)
- supprimer un élément en tête (*pop_front*) ou en fin (*pop_back*)
- tester si la liste est vide (*empty*)
- récupérer le nombre d'éléments (*size*).

Vous pouvez vous inspirer des prototypes utilisés par la bibliothèque standard, et pensez à tester vos méthodes au fur et à mesure.

La classe itérateur de liste (*ItList*) comporte un pointeur sur une cellule courante (*_cellPtr*). Vous proposerez des surcharges pour l'opérateur de déréférencement *** (qui donne l'objet pointé par l'itérateur), les opérateurs *++* (pré et postfixés), et les opérateurs de comparaison *==* et *!=*.

La classe « List » peut maintenant fournir accès à ses éléments grâce à ces itérateurs. Il lui faut pour cela une méthode *begin()* qui retourne un itérateur sur le début de la liste, et une méthode *end()* qui retourne un itérateur sur la fin de la liste (en fait, dès qu'un itérateur vaut *end()*, la fin de la liste est atteinte).

Codez et testez dans un programme principal les classes une à une en proposant les constructeurs que vous jugerez utiles.

- 2) Testez dans le programme principal deux instances d'itérateurs référant la même liste pour faire, par exemple, la comparaison d'éléments 2 à 2 au sein de cette liste et le parcours d'une liste dans le but d'afficher ses éléments un à un.
- 3) Codez ce patron de Liste de manière générique avec la notion de template.
- 4) **Optionnel** : Pour faciliter la manipulation de liste, ajoutez des méthodes à la classe « List » en vous inspirant de celles fournies par *std::list* (recherche, insertion, suppression, tri, inversion, etc.)