Marc-Antoine Abou Jaoude 2011421

Marc-André Lavoie 1944202

Quel est l'avantage d'utiliser une linked-list au lieu d'un vecteur si on veut supprimer ou rajouter un élément ? Quelle structure de données est plus avantageuse si on veut accéder à des éléments à des positions aléatoires ?

Une linked-list est beaucoup plus rapide quant à l'insertion ou la suppression d'éléments à des endroits aléatoires tandis que le vecteur est plutôt optimisé pour l'insertion à la fin du conteneur. En effet, une linked list est un conteneur où les éléments sont liés entre eux à l'aide de pointeurs. Ainsi, pour l'insertion d'un nouvel élément, il suffit de modifier les valeurs des pointeurs next et previous des éléments autour de la position d'insertion. Or, pour le vecteur, il devient nécessaire de décaler tous les objets suivants la position d'insertion ce qui peut être très coûteux au niveau performance. Par ce fait, il est beaucoup plus avantageux d'utiliser des linked list puisque c'est moins coûteux au niveau performance. Pour accéder à des éléments aléatoires, il est préférable d'utiliser un vecteur. En effet, le vecteur a un itérateur à accès aléatoire et c'est un conteneur à accès aléatoire ce qui est nécessaire pour accéder à des éléments à des positions aléatoires. Les deux seuls conteneurs à accès aléatoires de la STL sont deque et vector mais l'accès aléatoire est fait plus lentement avec le deque et par conséquent il est préférable d'utiliser un vector pour l'accès aléatoire.

Pourquoi est-ce que l'implémentation des classes génériques est dans .h et non pas séparée en .h et .cpp comme les classes normales ?

Le compilateur doit avoir accès à la définition et l'implémentation de la classe générique pour pouvoir produire une version spécifique de la classe. En effet, pour compiler un programme principal, le compilateur a seulement besoin de connaître les déclarations de classe et leur implémentation seront compilés séparément et ensuite les deux seront combinés pour créer un fichier exécutable. Or, pour les classes génériques, il n'est pas suffisant de les séparer puisque le compilateur doit s'assurer que toutes les méthodes implémentées dans la classe sont valides pour le type générique spécifié.