Analyse des containers

Mélodie KOUY et Ségolène LAZARUS

Qu'est-ce qu'un container ?

Un container c'est une structure qui permet de stocker des données, des valeurs. Il gère l'espace de stockage des données et possède des méthodes permettant de manipuler ces données.
L'avantage des containers, c'est qu'ils gèrent automatiquement la mémoire nécessaire pour stocker objets et valeurs, et offrent des performances optimales. De plus, ils font partie de la bibliothèque standard c++, ce qui permet de trouver facilement des informations pour les utiliser.
Nous allons maintenant vous présenter plus en détails des containers list, vector, deque et map.

List

Ce sont des listes doublement chaînées, c'est-à-dire que les données sont stockées indépendamment les une des autres.

Très pratique lorsqu'on veut supprimer ou insérer des données, elle permet de facilement le faire sur les débuts et les fins de tableau. Par exemple, on peut l'utiliser dans le cas d'une pile d'objets.

Elle contient beaucoup de méthodes lui permettant de parcourir le tableau sans utiliser d'index (avec push_back et push_front qui sont utilisés respectivement pour ajouter des données à la fin et au début de la liste, et pop_back et pop_front pour les supprimer).

Avantages	Inconvénients
- Insertion, déplacement et suppression rapide à partir de n'importe quelle position - itération pas affectée par l'ajout ou la suppression de données	- Prend plus de mémoire - On est obligé de parcourir la liste depuis le début ou depuis la fin pour récupérer un élément (au milieu par exemple)

Vector

Le Vecteur est un tableau dont la mémoire se réalloue automatiquement et dont la taille est dynamique. Il possède plusieurs fonctions pour ajouter (push_back), insérer (insert), supprimer (erase) ou encore pour inverser des données (swap).

Avantages	Inconvénients
- pas besoin de se soucier de l'allocation de la mémoire	- utilise beaucoup de mémoire pour gérer dynamiquement les données

- taille dynamique - facilité d'accès aux éléments du tableau	

Deque

Deque est l'acronyme de **d**ouble-**e**nded **que**ue. C'est un conteneur de séquence de taille dynamique qui peut être agrandi ou rétréci aux deux extrémités.
Assez similaire avec vector, il est cependant plus efficace dans un deque de rajouter des éléments au début du tableau ainsi que pour travailler sur de longues séquences, tandis que la réallocation d'un vector demande plus de ressources.
Sa principale différence avec Vector est sa manière d'utiliser la mémoire. Deque pourra disperser tous ses éléments dans différents espaces de la mémoire, et ainsi ne conservera que les données pour accéder directement à ses dits éléments.

Avantages	Inconvénients
- allocation et libération dynamique, pas besoin de le faire manuellement - insertion et suppression rapide au début et à la fin	- les pointeurs ne sont stockés pas à la suite les uns des autres - moins pratique si on travaille dans le milieu du tableau

Map

Map fonctionne avec un système qui associe des clés avec des valeurs ; chaque clé est unique. On peut insérer, supprimer, se déplacer dans le tableau rapidement. Avec find par exemple, on peut chercher une valeur dont on a le nom de la clé, et celle-ci nous sera retournée très rapidement. Les éléments du tableau sont triés automatiquement en fonction de la clé. Map fonctionne sous la forme d'un arbre binaire de recherche (BinarySearchTree).

Par exemple, on peut l'utiliser pour stocker des numéros de téléphone. La clé sera le nom du propriétaire du numéro et la valeur sera le numéro en lui-même.

```
myMap["Alice"] = 0606060606;
cout << "Numéro d'Alice : " << myMap["Alice"] << endl;
```

Avantages	Inconvénients
- la clé suffit à retourner rapidement la valeur recherchée - clés toujours dans un ordre spécifique puisque triées automatiquement - suppression et insertion efficaces	- on ne peut pas modifier une clé - itération difficile puisque les éléments sont triés en fonction de la clé - peut utiliser beaucoup d'espace mémoire