Préparation du TP6

Résumé des Points Clés des Programmes

1. Utilisation d'un `vector<>::iterator` et d'un `set<>::iterator`

Points Clés :
- **Itérateurs** : Utilisation des itérateurs pour parcourir les éléments d'un `vector` et d'un `set`.
- **Affichage** : Affichage des éléments des conteneurs en utilisant des itérateurs.
- **Génération de nombres aléatoires** : Utilisation de `rand()` pour générer des nombres aléatoires.
- **Templates** : Utilisation de templates pour créer une fonction générique d'affichage.

2. Test de la capacité d'un `vector`

Points Clés :
- **Capacité du vecteur** : La capacité d'un vecteur est la taille de la mémoire allouée pour stocker les éléments.
- **Augmentation de la capacité** : La capacité du vecteur augmente généralement par un facteur de 2 lorsque cela est nécessaire.
- **Affichage** : Affichage de la capacité et de la taille du vecteur chaque fois que la capacité change.

3. Utilisation d'un tas pour trier un vecteur (version out-of-place de heapsort)

Points Clés :
- **File de priorité** : Utilisation de `priority_queue` pour trier les éléments.
- **Templates** : Utilisation de templates pour créer des fonctions génériques de tri.
- **Affichage** : Affichage des vecteurs avant et après le tri.
- **Comparaison** : Utilisation de foncteurs (`std::greater` et `std::less`) pour définir l'ordre de tri.

4. Le modèle `std::map`

Points Clés :
- **Map** : Utilisation de `std::map` pour stocker des paires clé-valeur.
- **Itérateurs** : Utilisation des itérateurs pour parcourir les éléments de la map.
- **Affichage** : Affichage des paires clé-valeur triées par clé.
- **Comparaison de paires** : Comparaison de paires en utilisant les opérateurs de comparaison.

5. Algorithmes et structures de données

Points Clés :
- **Inclusion de plages** : Vérification si une plage d'éléments est incluse dans une autre plage.
- **Tri par tas** : Utilisation de `priority_queue` pour implémenter un algorithme de tri par tas.
- **Templates** : Utilisation de templates pour créer des fonctions génériques de tri et de manipulation.
- **Échange d'éléments** : Échange de deux éléments pointés par des itérateurs.
- **Affichage** : Affichage des résultats des tests et des opérations.

6. Test de la capacité de différents conteneurs

Points Clés :
- **Conteneurs** : Test des performances de `vector`, `deque`, et `list` en termes de temps nécessaire pour insérer des éléments.
- **Temps** : Mesure du temps CPU écoulé en utilisant la fonction `clock()`.
- **Fichier de sortie** : Écriture des résultats dans un fichier `plot.txt` pour être utilisés par un outil de traçage (comme Gnuplot).
- **Génération de nombres aléatoires** : Utilisation de `rand()` pour générer des nombres aléatoires.

Points Clés Généraux

- **Conteneurs** : Utilisation des conteneurs. *Voir chapitre 5 page 111 ainsi que les exemples 5.3 page 118..*
- **Itérateurs** : Utilisation des itérateurs pour parcourir et manipuler les éléments des conteneurs.

- **Templates** : Utilisation de templates pour créer des fonctions génériques.
- **Files de priorité** : Utilisation de `priority_queue` pour implémenter des algorithmes de tri.
- **Affichage** : Affichage des résultats des tests et des opérations.
- **Comparaison** : Utilisation de foncteurs pour définir l'ordre de tri et comparer des paires.
- **Performance** : Mesure des performances des différents conteneurs en termes de temps nécessaire pour insérer des éléments.

Pour tout complément:
https://cplusplus.com/reference/