TP : Standard Template Library

Exercice 1 : la classe vector

Créer une classe Etudiant contenant son nom, son age, la formation suivie et un identifiant unique (entier). Prévoir une fonction (non membre) d'affichage pour les données de la classe.

Instancier un objet *cours1* de la classe ***vector*** pour stocker des éléments de type Etudiant. Remplir *cours1* avec quatre ou cinq étudiants dont vous créerez les données directement dans votre code.
Instancier un objet *cours2* et lui affecter les valeurs présentes dans *cours1*.

Parcourir *cours1* et afficher son contenu en utilisant ***for_each*** avec la fonction externe d'affichage de la classe Etudiant. Faire de même en remplacant la fonction par un foncteur (on utilisera une classe différente d'Etudiant pour cela).
Enfin, parcourir et afficher le contenu de *cours2*, cette fois en utilisant un itérateur et une boucle for.

Trier maintenant *cours1* en utilisant l'algorithme ***sort*** avec comme relation d'ordre une fonction de comparaison d'Etudiant travaillant sur les identifiants. Trier ensuite *cours2* en utilisant l'algorithme ***sort*** avec comme relation d'ordre un foncteur pour comparer deux Etudiant en se basant sur leur âge cette fois. Réafficher le contenu des deux cours après le tri pour en vérifier la validité.

Ajouter un nouvel étudiant à *cours1*, mais en l'insérant de manière à respecter l'ordre préétabli (on utilisera un itérateur pour se déplacer jusqu'à la bonne position).

Exercice 2 : la classe CsgTree

En reprenant les classes CSG de la séance précédente déjà implantés, créer à présent une classe CsgTree, qui contiendra et gérera la création et l'évolution de notre arbre csg. Parmi ses membres on trouvera un objet de la classe ***set***, contenant des pointeurs sur les racines de nos arbres csg. Préciser lors de cette déclaration un foncteur pour comparer deux éléments en les classant selon leur identifiant (un entier unique par noeud).
De la meme manière, on déclarera également un ***set*** contenant des pointeurs sur les feuilles de l'arbre csg.

Répondre aux questions suivantes :
- Pourquoi choisir dans la STL précisément la classe ***set*** pour contenir les racines et les feuilles ?
- Pourquoi maintenir un ***set*** pour les feuilles ET pour les racines ?

Créer dans CsgTree les méthodes de création/gestion de l'arbre csg : ajout de primitives graphiques et joindre des noeuds existants par une opération csg.

Dans l'interface nous travaillerons avec les identifiants des noeuds et non leur adresse dans la mémoire.
Question : Comment faire pour récupérer efficacement l'adresse d'un noeud dont on connait seulement l'identifiant ?
Si on veut éviter une recherche, on peut utiliser une structure adéquate pour stocker cette correspondance identifiant ↔ adresse. Quelle classe de la STL permettrait cela ?
La déclarer dans CsgTree et prévoir la ou les méthodes devant l'accompagner pour obtenir l'adresse à partir de l'identifiant, mais aussi l'identifiant à partir de l'adresse.