TP 6

Dans le cadre de l'UE, vous êtes amenés à programmer en **C++** en utilisant les références pour réaliser vos passages de paramètres ainsi que les constructeurs, les destructeurs et les surcharges de l'opérateur d'affectation. Néanmoins, il sera important que vous sachiez toujours basculer dans un autre langage de programmation. Pour cela, il est important que vous établissiez votre raisonnement au niveau du pseudo-langage algorithmique, en utilisant les spécificités du langage utilisé seulement quand vous effectuez la mise en œuvre.

1 Arbre Binaire de Recherche

Mettre en place le module `Arbre Binaire de Recherche` que vous avez déjà étudié dans des UEs précédentes. Les nœuds de vos arbres contiendront des `Eléments` pour lesquels il est nécessaire de disposer d'une opération de comparaison (surcharge de l'opérateur <). Les `Eléments` seront décrits dans le module `Element` ainsi que les opérations associées.

Votre module `Arbre Binaire de Recherche` devra comprendre :

— un constructeur par défaut (1),
— un constructeur par copie (réalisant une copie profonde) (6),
— une surcharge de l'opérateur d'affectation (avec copie profonde) (7),
— recherche (2) et insertion (3) d'un élément,
— un parcours avec affichage (4) préfixé, infixé ou postfixé des éléments de l'arbre (ainsi qu'un affichage permettant de voir l'état interne d'un arbre),
— un destructeur (5)
— ...

Les numéros entre parenthèses suggèrent l'ordre dans lequel coder les différentes fonctionnalités.

Pour coder ces fonctionnalités, vous pourrez recourir à des fonctionnalités internes récursives (par exemple une fonctionnalité d'affichage d'un sous-arbre d'un arbre, le sous-arbre étant désigné par l'adresse de son nœud racine, ainsi qu'une fonctionnalité de copie profonde du sous-arbre d'un arbre, etc...). Concernant l'affichage de l'état interne d'un arbre, le plus simple est de procéder à un affichage récursif infixé de l'arbre, chaque nœud étant précédé d'un nombre d'espace correspondant à la profondeur de récursion.

Concernant la présentation de vos modules, il est important que la partie publique de vos structures de données ne fasse apparaître que les fonctionnalités réellement utiles et utilisables sans trop de risque par l'utilisateur, en accompagnant la déclaration de ces fonctions de leurs conditions d'utilisation (préconditions et postconditions).

Vous aurez besoin de ce module au prochain TP, il sera donc important que vous le terminiez chez vous.

2 Tables de Hachage

Finir le travail sur les tables de hachage entrepris au TP précédent.

3 Comparaison de performance

Comparer expérimentalement les performances des tables de hachage (profil du temps d'insertion et de recherche en fonction du taux de remplissage de la table) et des arbres binaires

de recherche (profil du temps d'insertion et de recherche en fonction du nombre d'éléments dans l'ABR) après avoir enregistré et visualisé les profils de performance.