Ce TP est une introduction aux arbres et à leur manipulation.

1 Arbres binaires

1.1 Construction des arbres

Vous allez écrire une structure **noeud** qui aura 3 champs : un entier, et 2 pointeurs sur des variables de type "noeud".
Vous écrirez les fonctions suivantes :

- **createArbrePrefixe :** qui prend en paramètres la Racine de l'arbre **Racine**, et la profondeur **prof** de l'arbre. Cette fonction va créer un arbre binaire complet de profondeur "prof", en utilisant le parcours "préfixé". Les valeurs de l'arbre seront saisies par l'utilisateur pour vérifier le bon déroulement de la construction de l'arbre ;
- **createArbreInfixe :** qui fait la même chose que la fonction précédente mais en utilisant le parcours infixé ;
- **createArbrePostfixe :** qui fait la même chose que la fonction précédente mais en utilisant le parcours postfixé ;

1.2 Affichage des arbres

Vous allez écrire les 3 fonctions vues en TD qui permettent d'afficher un arbre suivant un parcours préfixé, infixé et postfixé.

1.3 Recherche dans un arbre

1.3.1 Recherche d'une valeur

Vous écrirez les fonctions vues en TD :

- **afficheValeurPrefixe**, **afficheValeurInfixe** et **afficheValeurPostfixe** qui permettent, en parcourant respectivement l'arbre de manière préfixée, infixée et postfixée, d'afficher toutes les valeurs de l'arbre inférieures à une valeur "X" passée en paramètre. On passera aussi le pointeur sur la racine de l'arbre.
- **rechercheValeur** qui permet de chercher dans l'arbre pointé par un pointeur sur la racine passé en paramètre, si la valeur "X" passée en paramètre existe. La fonction retournera Vrai si elle existe et Faux sinon. Vous utiliserez le parcours que vous désirez, vous le nommerez en suffixe du nom de votre fonction.
- **compterFeuilles :** qui compte le nombre de feuilles d'un arbre binaire quelconque

1.3.2 Recherche des caractéristiques

Ecrire les fonctions suivantes :

- **compterNoeuds :** qui compte le nombre de noeuds d'un arbre binaire quelconque
- **compterNoeudsInternes :** qui compte le nombre de noeuds interne d'un arbre binaire quelconque
- **profondeur :** qui retourne la profondeur d'un arbre binaire quelconque
- **combienProf :** qui retourne le nombre de noeuds d'un arbre binaire qui sont exactement à la profondeur p (passée en paramètre)
- **noeudsProf :** qui affiche les valeurs des noeuds d'un arbre binaire qui sont exactement à la profondeur p (passée en paramètre)
- **estPlein :** qui prend en paramètre la racine d'un arbre binaire et qui teste si cet arbre est plein.

1.4 Programme principal

Vous écrirez un programme qui permet de tester les fonctions écrites précédemment.

2 Arbres n-aires

Vous allez écrire une structure **noeudMult** comme vue en TD et qui aura 3 champs : un réel, le nombre de fils du noeud et un tableau de "pointeurs sur noeudMult" fixé à 100 éléments (mais vous pouvez aussi le créer dynamiquement) pointeur sur des variables de type "pointeur sur noeudMult".

Vous écrirez les fonctions suivantes :

- **createArbreNAire** qui crée un arbre n-aire et qui demande à l'utilisateur de saisir la valeur du noeud et le nombre de fils du noeud, l'arbre sera terminé lorsque toutes les feuilles (0 fils) auront été saisies.
- **afficherEnProfondeur** qui affiche l'arbre passé en paramètre et qui l'affiche "en profondeur" ;
- **afficherEnLargeur** qui affiche l'arbre passé en paramètre et qui l'affiche "en largeur" comme vu en TD ;
- **rechercheHauteur** qui recherche la hauteur de l'arbre dont la "racine" est passée en paramètre ;
- **afficherNiveau** qui affiche les valeurs des noeuds situés à un niveau "niveau" passé en paramètre de l'arbre dont la "racine" est passée en paramètre ;
- **afficherEnLargeur** qui affiche l'arbre passé en paramètre et qui l'affiche "en largeur" en utilisant les fonctions précédentes.

Vous écrirez un programme qui permet de tester les fonctions écrites précédemment.