Programmation et Algorithmique II

Devoir 3-1 : Les arbres : Partie I

Ce devoir se compose de trois séances. Vous devez soumettre tous les trois à la fin du semestre à yiru.zhang1@cyu.fr.

### Exercice 1 :

Avec une représentation chaînée d’un arbre binaire, écrire en langage C les algorithmes qui affichent les parcours Préfixe, Infixe, Post-Fixe.

struct node **{**

struct node **\***left**;**

char data**;**

struct node **\***right**;**

**};**

### Exercice 2 :

1. Proposer un type *tree* basé sur une structure pour représenter un arbre binaire avec un type générique *TElement*.
2. Ecrire le code en C qui permet de réaliser les opérations suivantes (privilégier un algorithme **récursif** quand c’est possible) :
3. *bool IsEmpty( tree T);* // cette fonction définie si l’arbre T est vide ou pas
4. *tree Left( tree T);* // cette fonction retourne le fils gauche de T
5. *tree Right( tree T);* // cette fonction retourne le fils droit de T
6. *bool IsLeave(tree T);* // cette fonction détermine si T est une feuille
7. *bool IsInternalNode(tree T);* // cette fonction détermine si T est un nœud interne (un nœud interne est un nœud qui n’est pas une feuille)
8. *unsigned Height (tree T);* // cette fonction retourne la profondeur de T
9. *unsigned NbNode(tree T);* // cette fonction retourne le nombre de nœud de T
10. *unsigned NbLeaves( tree T);* // cette fonction détermine le nombre de feuilles de T
11. *unsigned NbInternalNode(tree T);* // cette fonction détermine le nombre de nœuds internes de T
12. void DFS\_prefix(tree T); // cette fonction effectue un parcours préfixe de T
13. void DFS\_infix(tree T); // cette fonction effectue un parcours infixe de T
14. void DFS\_postfix(tree T); // cette fonction effectue un parcours postfixe de T
15. void **BFS**(tree T); // cette fonction effectue un parcours en largeur (BFS) de T
16. tree Create(TElement val,tree ls, tree rs); // cette fonction crée un arbre binaire (nœud).
17. void AddElt(tree src, TElement elt); //cette fonction permet d’ajouter un élément dans l’arbre binaire selon les règles suivantes :
    1. Les éléments sont ajoutés de gauche à droite dans un même niveau
    2. Un élément ne peut être rajouté dans un niveau sauf si le niveau précédant est rempli
18. bool Exist(tree src , TElement elt); // cette fonction détermine si *elt* existe dans T
19. void Erase(tree \* src); // cette fonction supprime src et libère l’espace mémoire occupé
20. **Réécrire les fonctions récursives en forme non-récursive (itérative), pour le parcours en largeur (BFS) il faut utiliser une file comme structure de données intermédiaire.**