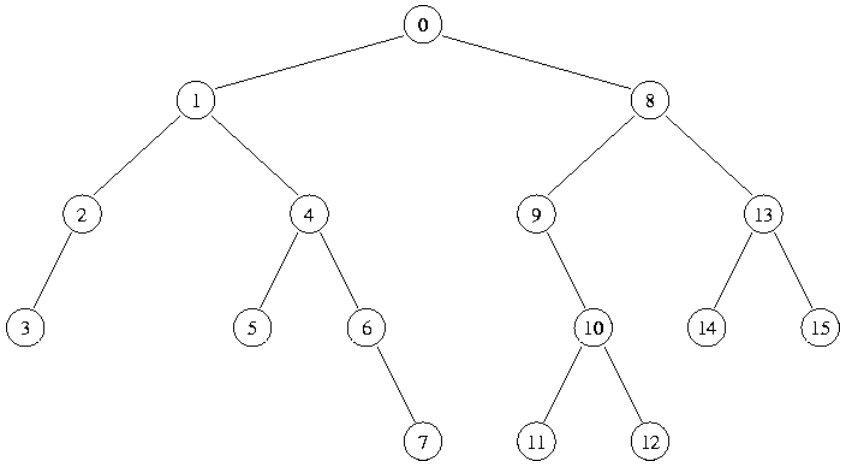
TD/TP Structure de données - Arbres binaires

M. Tellene

1 Appréhender les arbres binaires

Soit l'arbre binaire suivant :



Exercice 1
Donner la taille et la hauteur de cette arbre

Exercice 2
Donner l'ordre des noeuds visités avec :

- parcours infixe
- parcours préfixe
- parcours postfixe
- parcours en largeur : consiste à parcourir l'arbre niveau par niveau. Les noeuds de niveau 0 sont d'abord parcourus puis les noeuds de niveau 1 et ainsi de suite. Dans chaque niveau, les nœuds sont parcourus de la gauche vers la droite.

2 Manipuler un arbre binaire

Organisation : aller dans votre dossier personnel, puis créer un dossier « structure de données » (ou « SD »), dans ce dossier, créer un dossier « arbre binaire », c'est ce dossier qui contiendra le travail fait lors de ce TP.

Pour ce TP, nous construirons un arbre binaire sans classe noeud.

Vous créerez donc directement une classe `Arbre`. Cette classe est composée de trois attributs : `valeur`, `fg` (pour fils gauche) et `fd` (pour fils droit). L'arbre binaire de l'exercice 1 se crée de la manière suivante :

```
a = Arbre(0,
        Arbre(1,
            Arbre(2, Arbre(3), None),
            Arbre(4,
                Arbre(5),
                Arbre(6, None, Arbre(7))
            ),
        ),
        Arbre(8,
            Arbre(9,
                None,
                Arbre(10, Arbre(11), Arbre(12))
            ),
            Arbre(13, Arbre(14), Arbre(15))
        )
    )
```

Une fois fait, créer les méthodes associées à la structure de données d'arbre binaire. Si vous ne vous en souvenez pas, elles sont données dans la suite.

- `est_vide()` : renvoie `True` si l'arbre est vide, `False` sinon
- `insertion(x)` : insère `x` dans l'arbre, l'insertion dans un arbre binaire se déroule de la manière suivante :

```
SI l'arbre est vide ALORS
    on fixe la valeur de la racine a x
SINON
    on crée un nouveau noeud avec pour valeur x (n par exemple)
    on crée une file d'attente
    on récupère la racine (c par exemple)
    TANT QUE c n'est pas nul
        SI le fils gauche de c est nul ALORS
            on met n dans le fils gauche de c
            RETOURNE
        SINON SI le fils droit de c est nul ALORS
            on met n dans le fils droit de c
            RETOURNE
        on ajoute le fils gauche puis le fils droit dans la file
        on retire un noeud de la file et on l'attribue à c
```

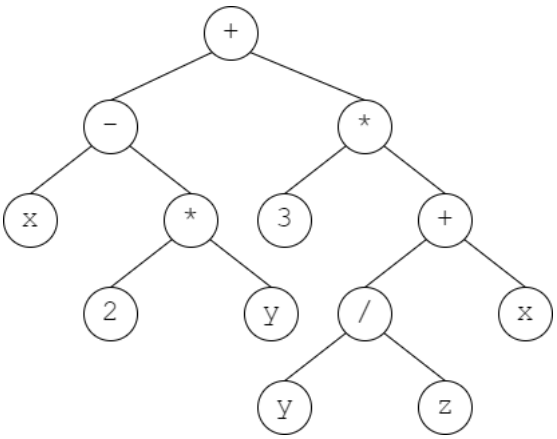
- `parcours_largeur()` : affiche les valeurs contenues dans l'arbre dans l'ordre du parcours en largeur
- `parcours_infixe()` : affiche les valeurs contenues dans l'arbre dans l'ordre infixe
- `parcours_prefixe()` : affiche les valeurs contenues dans l'arbre dans l'ordre préfixe
- `parcours_postfixe()` : affiche valeurs contenues dans l'arbre dans l'ordre postfixe
- `taille()` : renvoie la taille de l'arbre
- `hauteur()` : renvoie la hauteur de l'arbre
- `rechercher_element(x)` : renvoie `True` si la valeur `x` est dans l'arbre, `False` sinon

Exercice 3

On souhaite représenter par des arbres binaire des expressions arithmétiques sous différentes formes de notation. Considérons l'expression arithmétique :

$$((x - (2 * y)) + (3 * ((y/z) + x)))$$

L'arbre représentant l'expression est donné ci-dessous :



1. Quels seraient les affichages par un parcours infixe, préfixe, postfixe ?
2. Donner l'arbre représentant l'expression $((12 * (x + 2)) - ((y/3) * 6 + (x + z)))$
3. Quel est l'expression représentée par l'arbre suivant