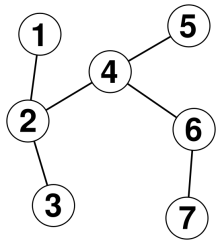
# Algorithmique avancée- Arbres binaires

## Arbres binaires

**Définition 1.** *On appelle* ***arbre binaire*** *une structure de données dans laquelle chaque élément a au plus deux éléments fils. Un élément particulier joue le rôle de* ***racine****. C'est l 'élément racine qui définit le sens de la relation père-fils.*

**Exercice 1.** On considère l'arbre suivant :




1. On choisit le sommet 6 comme racine. Représenter le graphe comme un arbre. Obtient-on un arbre binaire ?
2. Quels choix de racine font de ce graphe un arbre binaire ? On choisit le sommet 3 comme racine.
3. Représenter l'arbre. Si un noeud n'a qu'un fils, on considèrera que ce dernier est le fils gauche. Si un noeud a deux fils, on placera le fils contenant la plus petite valeur à gauche.
4. Quel est le fils droit du fils gauche de la racine ?
5. Quel est la génération du noeud 6 ? (la génération du noeud racine étant 0).

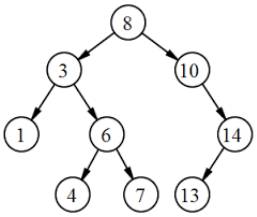
**Exercice 2.** On cherche à représenter un arbre.
1. Proposer une structure de données Noeud permettant de représenter un arbre.
2. Implémenter en langage C cette structure.

## Les algorithmes de parcours spécifiques aux arbres binaires

On donne le pseudo-code suivant :

ParcourPrefixe($Arbre, Noeud$)
1 Afficher($Noeud$)
2 ParcoursPrefixe($Noeud.FilsGauche$)
3 ParcoursPrefixe($Noeud.FilsDroite$)

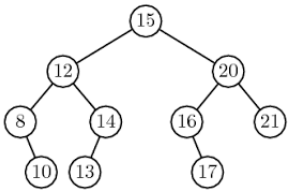
**Exercice 3.** On considère le graphe suivant :




Donner l'ordre dans lequel le parcours prefixe visite les noeuds de cet arbre.

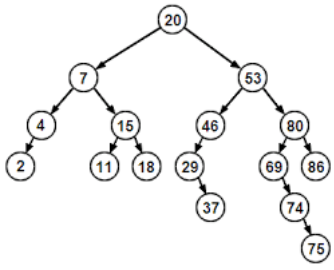
**Exercice 4.** On considère le graphe suivant :

Donner l'ordre dans lequel le parcours prefixe visite les noeuds l'arbre suivant :




**Exercice 5.** Proposer du pseudo-code pour un parcours postfixe.

**Exercice 6.** On considère le graphe suivant :

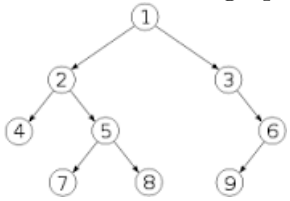



Donner l'ordre dans lequel le parcours postfixe visite les noeuds de cet arbre.

**Exercice 7.** Proposer du pseudo-code pour un parcours infixe.

**Exercice 8.** Proposer une implémentation en C pour chacun des trois types de parcours.

**Exercice 9.** On considère le graphe suivant :



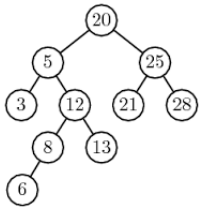

Donner l'ordre dans lequel le parcours infixe visite les noeuds de cet arbre.

## Le parcours en largeur

**Exercice 10.** On considère l'arbre binaire suivant :

```
PP(Arbre, Noeud)
1  File = ∅
2  Enfiler(File, Noeud)
3  while F
4        NoeudCourant = Defiler(F)
5        Afficher(NoeudCourant)
6        if Noeud.FilsGauche != Null
7              Enfiler(File, NoeudCourant.FilsGauche)
8        if Noeud.FilsDroit != Null
9              Enfiler(File, NoeudCourant.FilsDroit)
```

**Exercice 11.** On considère le graphe suivant :




Appliquer l'algorithme de parcours en largeur à cet arbre.

**Exercice 12.** On considère le graphe suivant :

Appliquer l'algorithme de parcours en largeur à cet arbre.

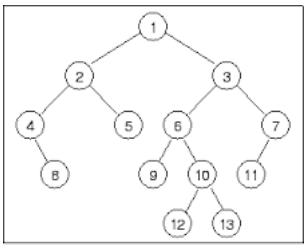



Fig. 18 – *Arbre binaire étiqueté*

**Exercice 13.** On considère le graphe de l'exercice précédent.
1. Appliquer l'algorithme de parcours en largeur au graphe de l'exercice précédent, sachant que la file a été remplacé par une pile.
2. Comment qualifier le parcours obtenu ?

**Exercice 14.** Proposer une implémentation en C pour le parcours ne largeur.