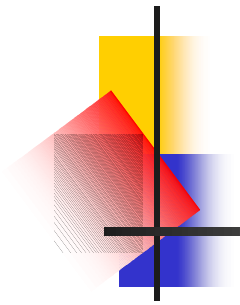
Chapitre 5

Structures de Données arborescentes :

Les Arbres

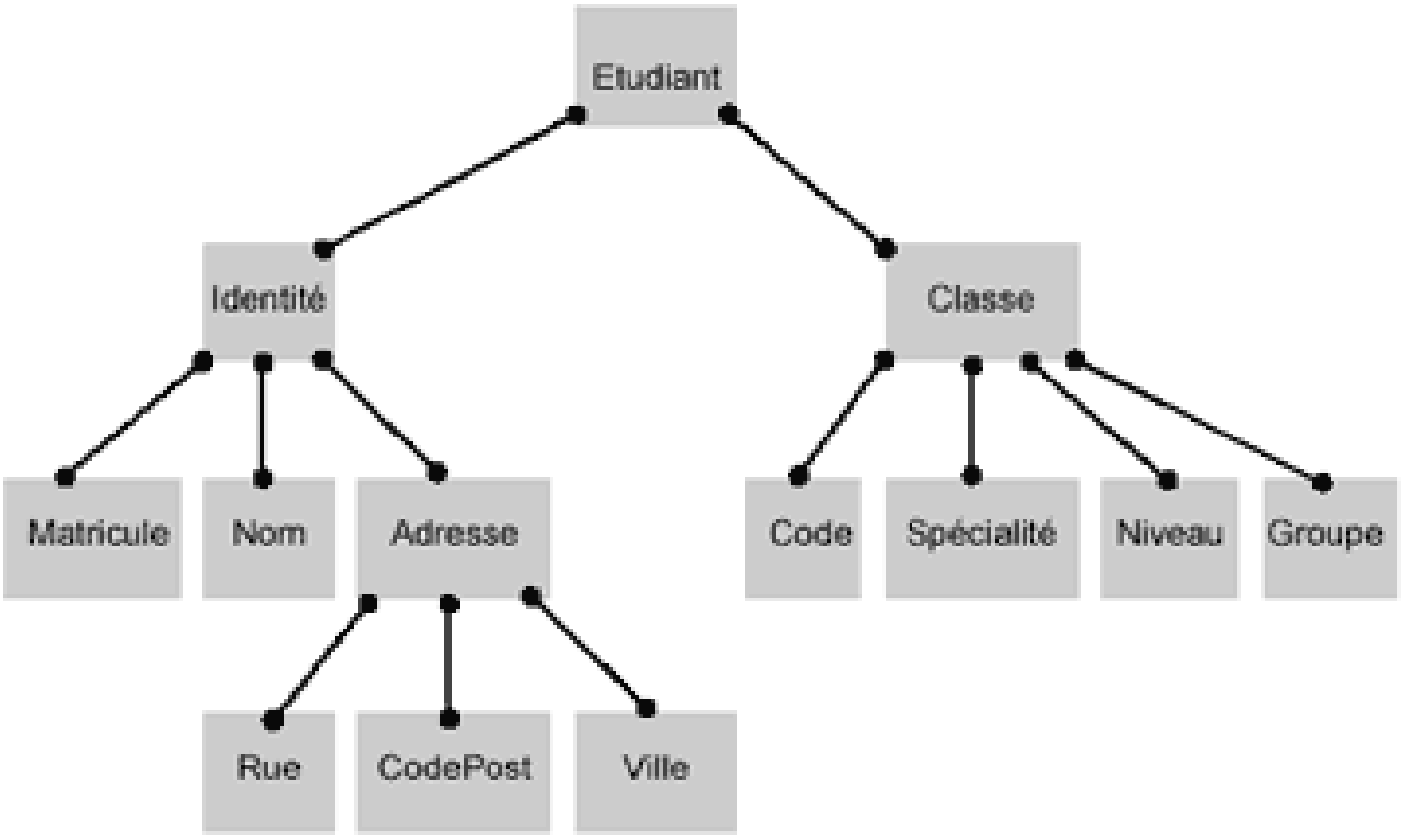


Plan

- Définition
- Types de parcours
- Primitives des Arbres binaires
- Implémentation chaînée
- Implémentation contiguë

Exemples d'utilisation

Une variable structurée peut être représentée sous forme d'un arbre. Par exemple la variable structurée Etudiant (de type TypeEtudiant) suivante peut être représentée par l'arborescence suivante :

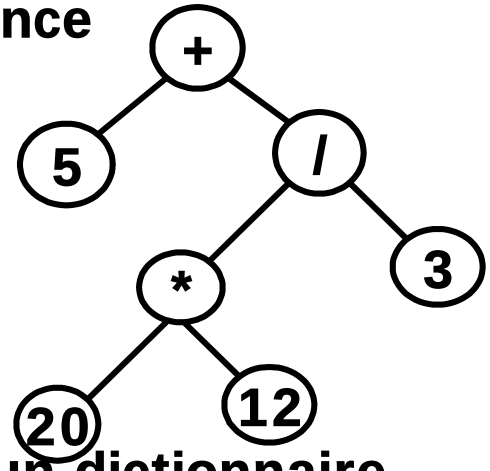


Exemples d'utilisation

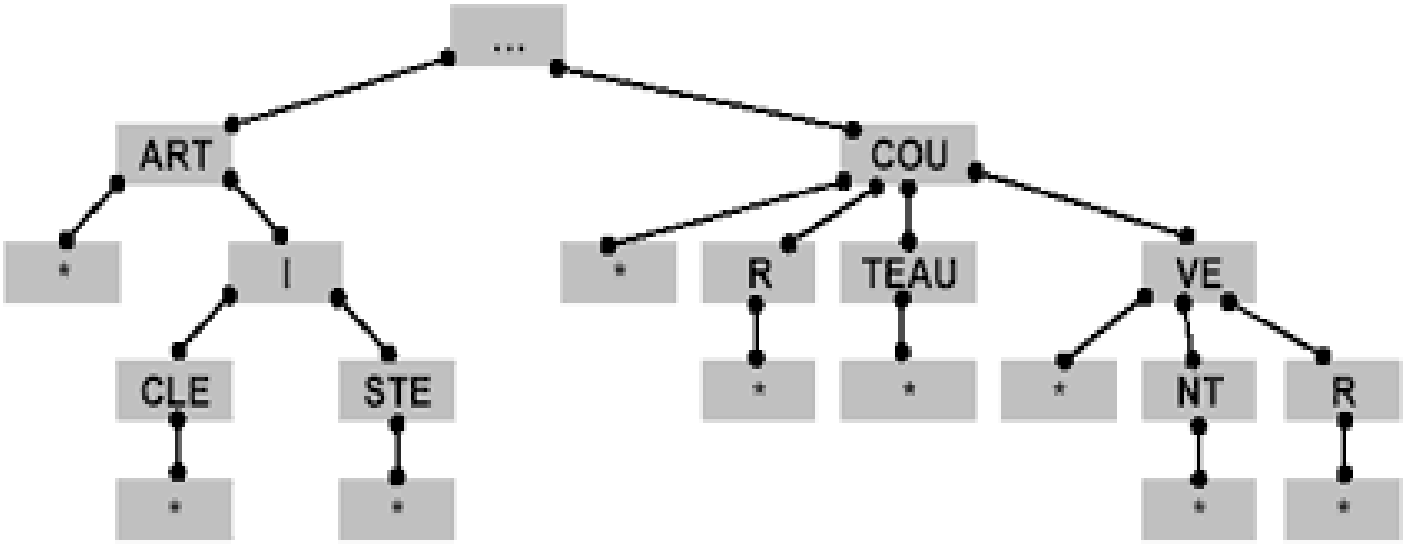
2. Une expression arithmétique peut se représenter sous forme d'une arborescence

5+(20 * 12 / 3)

=+ 5 / * 20 12 3



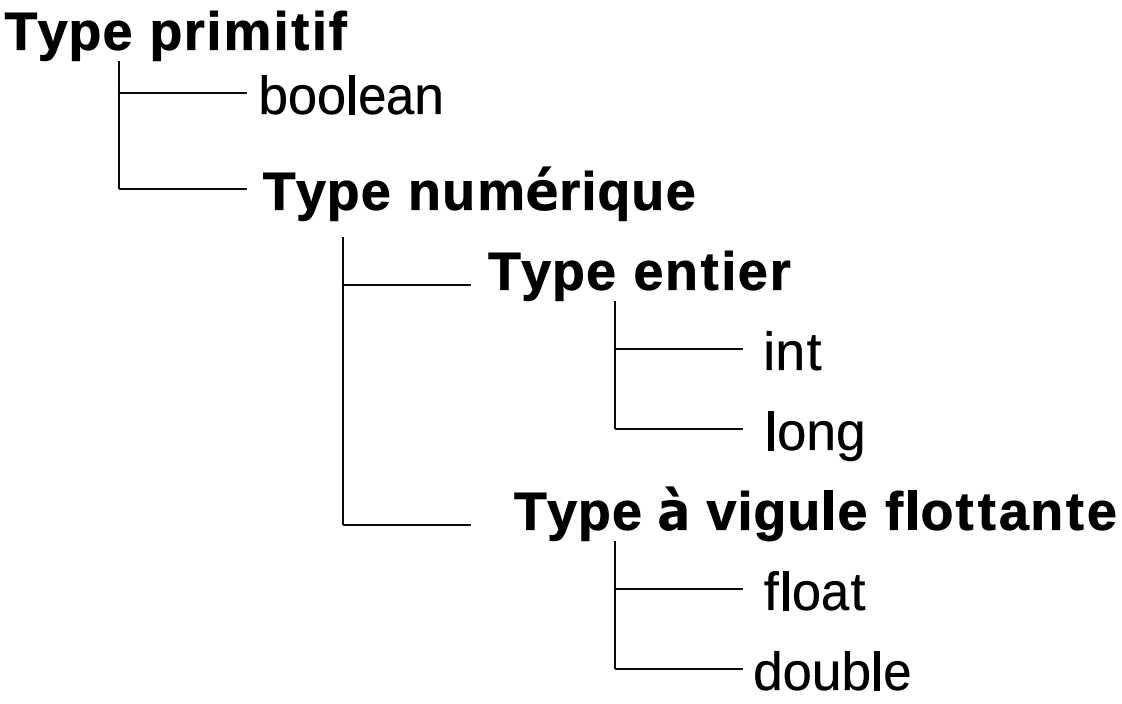
3. On peut construire ainsi un dictionnaire «arborescent»

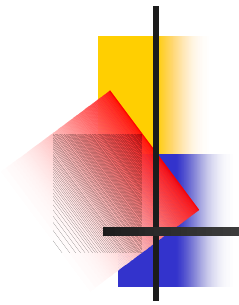


Rq: * marque la fin d'un mot, l'ordre alphabétique est respecté de gauche à droite):

Exemples d'utilisation

- Quelques types de données en Java





Introduction

- **Les arbres sont des structures de données non linéaires : structures arborescentes**
- **Les arbres modélisent une relation hiérarchique dans laquelle tous les éléments peuvent avoir zéro ou plusieurs successeurs et un prédécesseur unique sauf la racine**

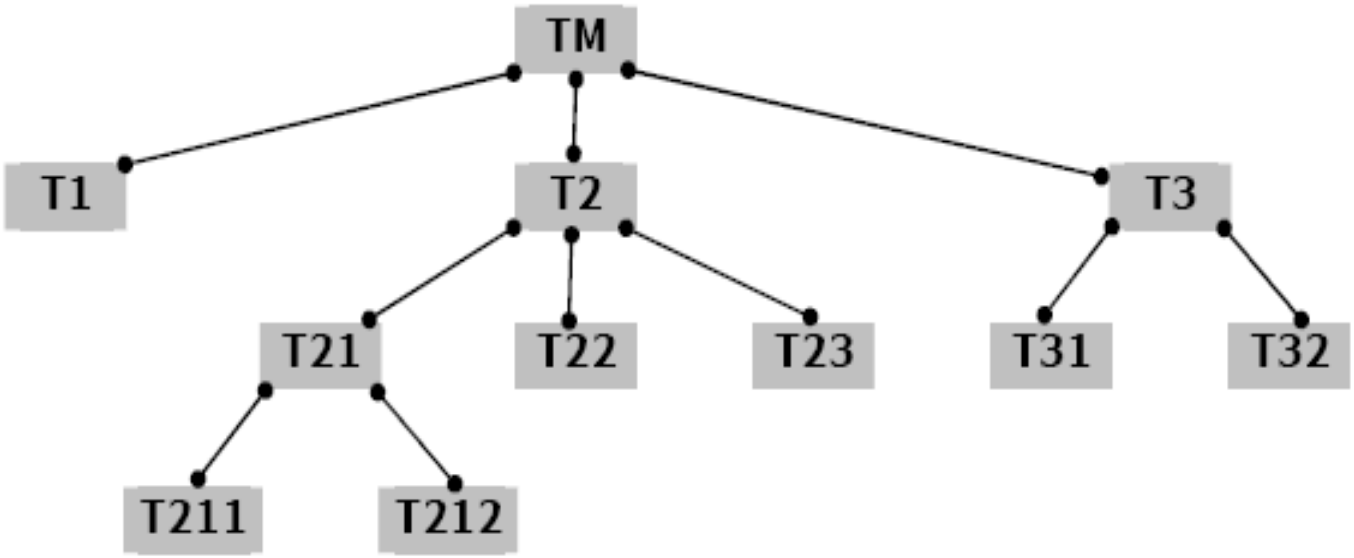
En d'autres termes …

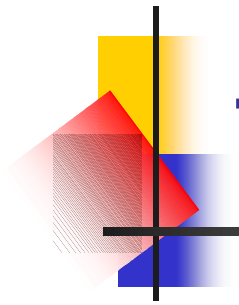
- **Un arbre est formé d'un ensemble de nœuds, reliés par des arêtes tq entre deux noeuds il existe au plus un et un seul chemin**
- **Un arbre organisé de façon hiérarchique: c'est un graphe connexe sans cycle (acyclique)**

Introduction

- **Les <u>schémas récursifs</u> simplifient beaucoup l'écriture d'algorithme sur les arbres, où on peut donner la définition suivante :**
  Un arbre est :
  - **soit vide,**
  - **soit constitué d'un élément auquel sont chaînés un ou plusieurs arbres.**

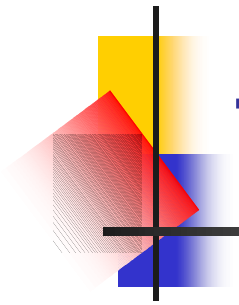
<u>Exemple</u>:table des matières d'un livre





Terminologie (1)

- **Un père (parent) est le prédécesseur direct d'un nœud. Exemple TM est le père de T1, T2 et T3**
- **Un fils (enfant) est un successeur direct d'un nœud. Exemple T1, T2 et T3 sont les trois fils de TM**
- **Un frère d'un nœud est le fils d'un même père. Exemple T2 est un frère d T1.**
- **Une feuille est un nœud sans fils. Exemple T1, T211, T212, T31 et T32 sont des feuilles.**
- **Le degré d'un nœud correspond à son nombre de fils. Exemple le degré de TM est 3.**
- **Le niveau de la racine est 0, si un nœud est au niveau n, son successeur est au niveau n+1. Exemple le niveau de T1 est 1.**
- **Une génération représente les nœuds d'un même niveau. Exemple** T21, T22, T23, T31, T32 sont de la même génération
- **Deux nœuds sont adjacents si l'un est le père de l'autre. Exemple TM et T1 sont adjacents.**



Terminologie (2)

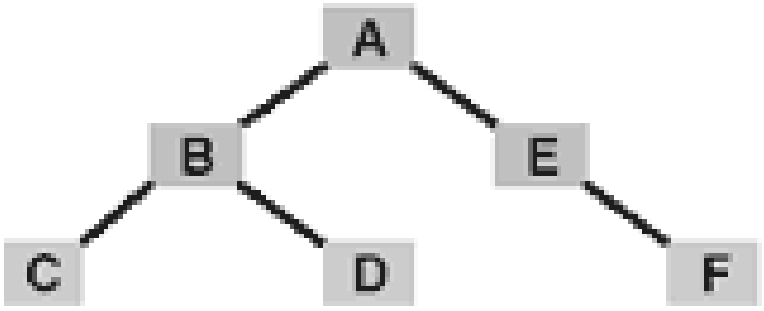
- **Si n_1, n_2, …, n_k est une suite de nœuds d'un arbre telle que n_i est le parent de n_{i+1} pour 1 <= i <= k, cette suite est appelée Chemin entre le nœud n_1 et le nœud n_k. Exemple (T2, T21, T212) est un chemin de T2 à T212**
- **La longueur d'un chemin est égale au nombre de nœuds qu'il contient moins 1. Exemple, le chemin (T2, T21, T212), de T2 à T212 est de longueur 2.**
- **Une branche c'est un chemin qui commence par la racine et qui se termine par une feuille. Exemple (TM, T2, T22).**
- **S'il existe un chemin entre les nœuds a et b, on dit que a est un ascendant ou un ancêtre de b et réciproquement que b est un descendant de a. Par exemple, les ascendants de T21 sont T2 et TM et ses descendants sont T211 et T212.**
- **Un sous-arbre (SA) d'un arbre est un nœud accompagné de toute sa descendance. Par exemple, le sous-arbre de nœud T2 a trois sous-arbres**

Terminologie (3)

- La **taille** d'un arbre est le nombre de nœuds de cet arbre. Exemple la taille de l'arbre est 11.
- La **profondeur d'un arbre (ou hauteur)** est le nombre de nœuds de la branche la plus longue=dernier niveau +1. Exemple la profondeur de l'arbre est 4.
- L'**ordre d'un arbre** est le degré maximum parmi tous ses nœuds. Exemple l'ordre de l'arbre est 3.
- Lorsque l'ordre d'un arbre est de n alors l'arbre est dit **n-aire**. Exemple l'ordre de l'arbre est ternaire.
- Si n est égal à 2, l'arbre est dit **binaire**. Un arbre binaire est soit un arbre vide, soit un arbre où chaque nœud (à part les feuilles) a un fils gauche, un fils droit ou les deux à la fois. On parlera aussi de **sous-arbre gauche (SAG)** et de **sous-arbre droit (SAD)**.

Terminologie (4)

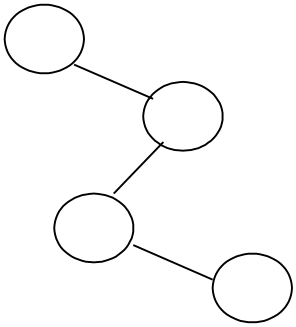
- **On peut noter un arbre binaire sous forme graphique ou sous forme parenthésée.**



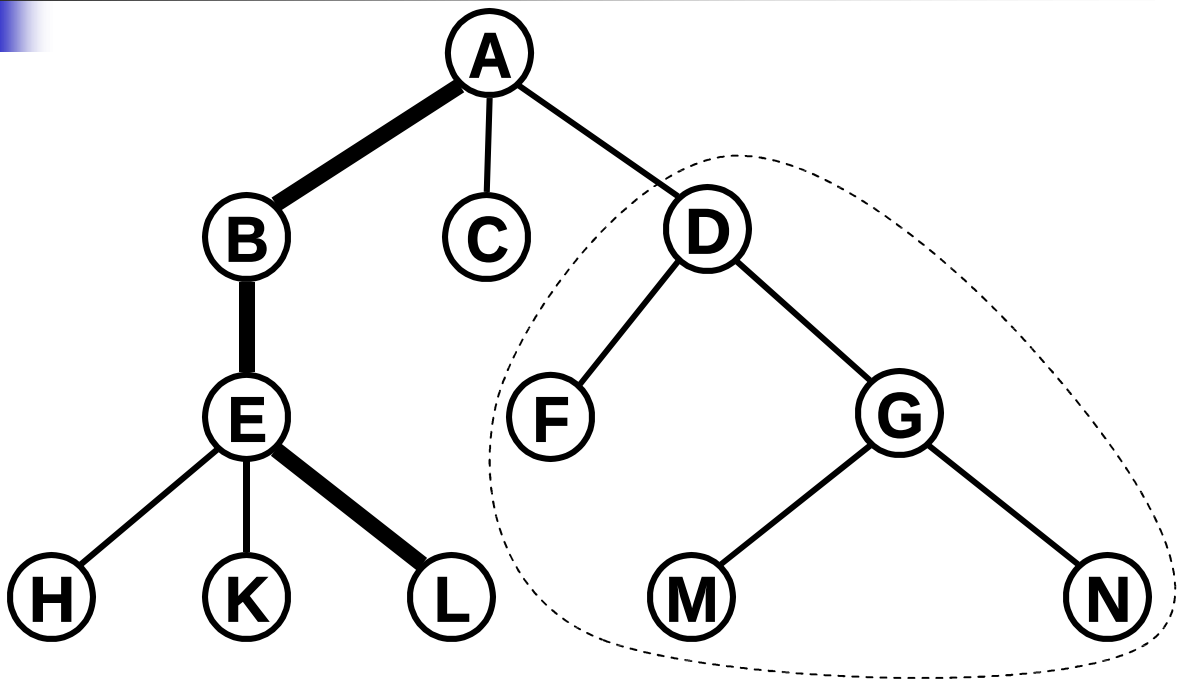
Par exemple cet arbre peut être aussi noter: A(B(C,D),E(,F)).

- **Si chaque nœud autre qu'une feuille admet deux descendants et si toutes les feuilles sont au même niveau, on dit que <u>l'arbre binaire est complet</u>.**

- **Un arbre binaire est dit <u>dégénéré</u>, si tous les nœuds de cet arbre ont au plus un descendant. Un arbre dégénéré est équivalent à une liste chaînée.**

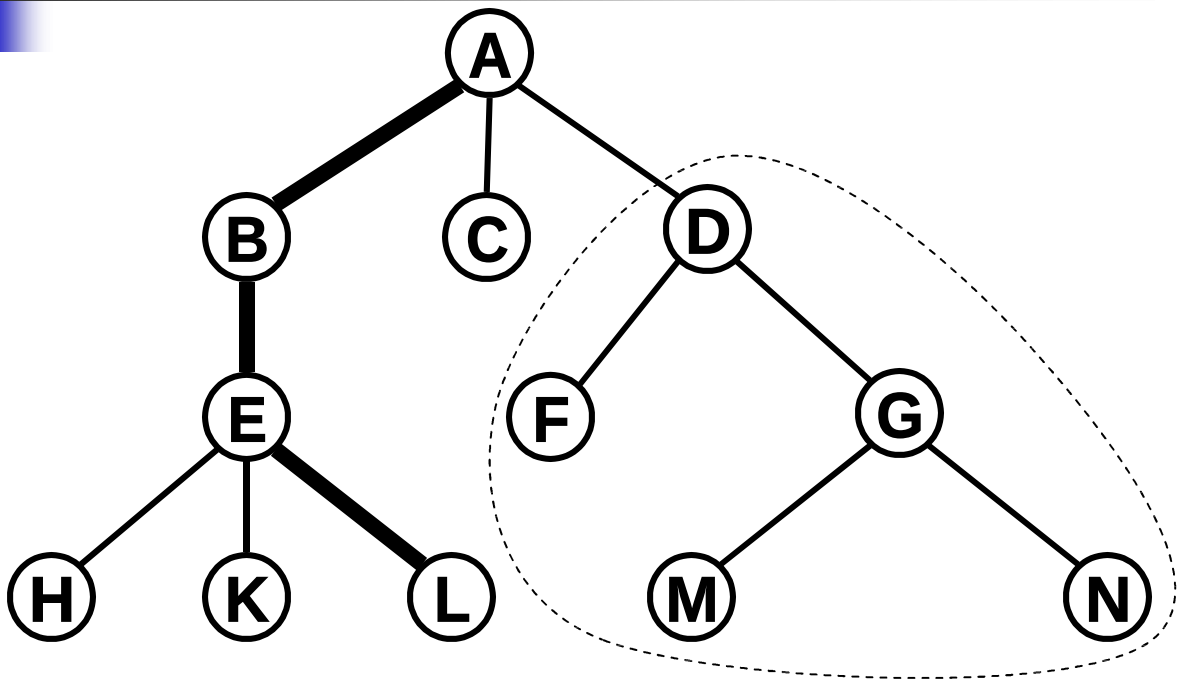


Illustration



- Le nœud A est la ….
- Les nœuds H, K, L, C, F, M et N sont des ….
- Le nœud D est le …. des nœuds F et G
- Le nœud K est …. des nœuds H et L
- L'ensemble des nœuds < D, F, G, M, N > forme un … de …. nœuds et avec D comme ….
- La …. de cet arbre est 4
- La taille de cet arbre est ….
- A-B-E-L est un ….
- Les nœuds E, F et G sont de la même …, ils appartiennent au niveau …

Illustration



- Le nœud A est la **racine**
- Les nœuds H, K, L, C, F, M et N sont des **feuilles**
- Le nœud D est le **père** des nœuds F et G
- Le nœud K est **frère** des nœuds H et L
- L'ensemble des nœuds < D, F, G, M, N > forme un **sous arbre** de **5** nœuds et avec D comme racine
- La **profondeur** de cet arbre est 4
- La **taille** de cet arbre est 12
- A-B-E-L est un **chemin et aussi une branche**
- Les nœuds E, F et G sont de la même **génération**, ils appartiennent au niveau **2**

Algorithmes de parcours d'arbres

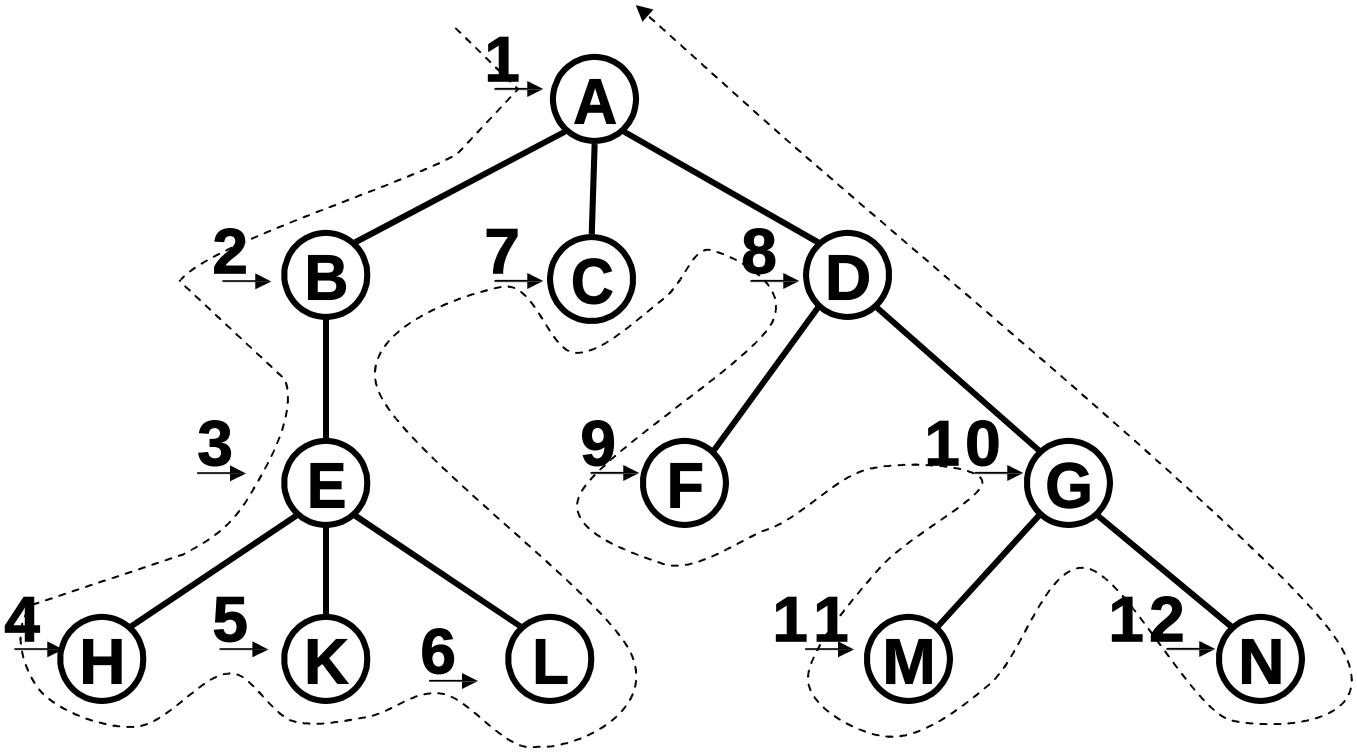
- Un algorithme de parcours est une méthode de traitement d'une SD qui applique une opération spécifique à chaque élément de cette structure

- **Il existe plusieurs méthodes pour parcourir les nœuds d'un arbre**
- **Parcours en profondeur:**
  - **parcours préfixé (preorder)**
  - **parcours infixé (inorder)**
  - **parcours postfixé (postorder).**
- **Parcours en largeur**

Parcours en profondeur

- **Si un arbre R est vide, la liste vide constitue le parcours préfixé, infixé et postfixé de R.**
- **Si R consiste en un seul nœud, la liste composée de ce nœud constitue le parcours préfixé, infixé et postfixé de R.**
- **Sinon, supposons que l'arbre à k sous-arbres A1, A2, …, Ak (de gauche à droite)**
  - **parcours préfixé**
    - **Racine**
    - **parcours préfixé des nœuds de A1, puis A2 et ainsi de suite jusqu'à Ak.**
  - **parcours infixé**
    - **parcours infixé des nœuds de A1**
    - **Racine**
    - **parcours infixé des nœuds de A2, …, Ak**
  - **parcours postfixé**
    - **parcours postfixé des nœuds de A1 puis de A2 et ainsi de suite jusqu'à Ak**
    - **Racine**

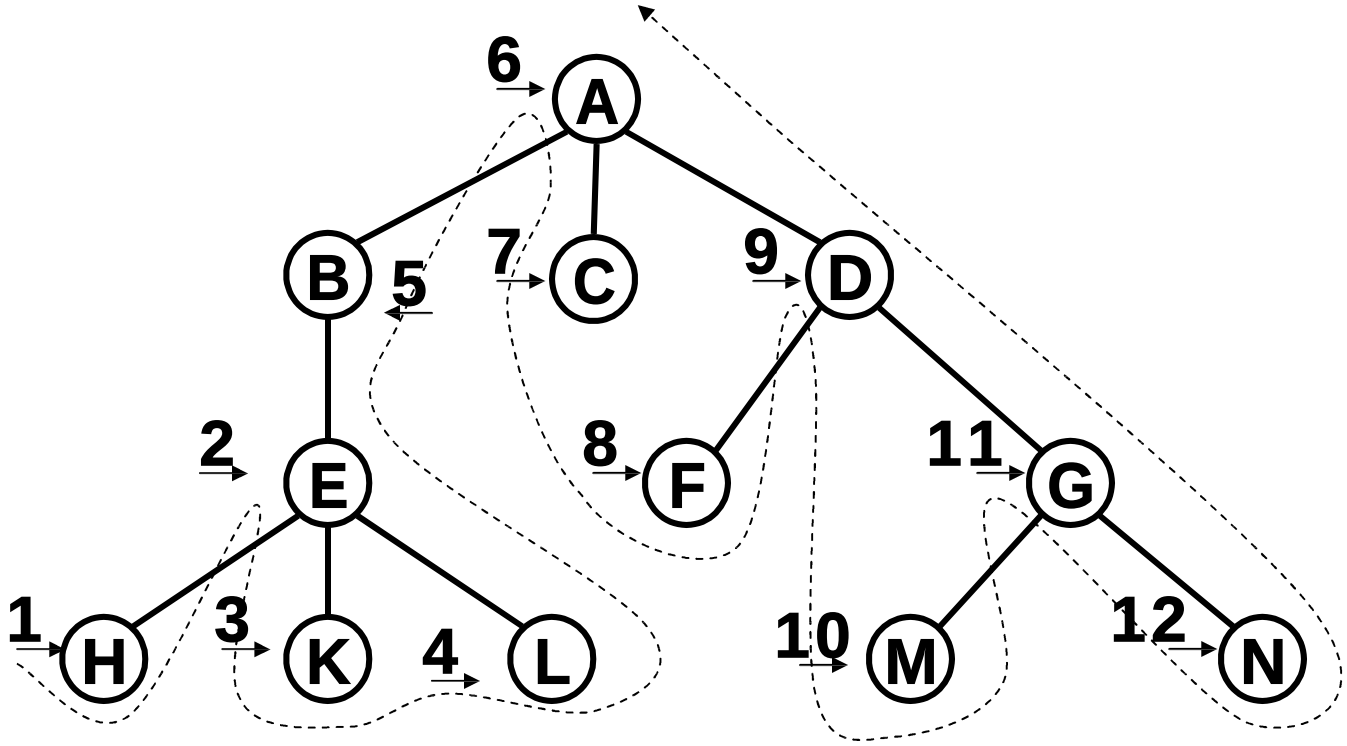
Algorithmes de parcours en profondeur (parcours préfixé)



è Parcours préfixé :

A, B, E, H, K, L, C, D, F, G, M, N.

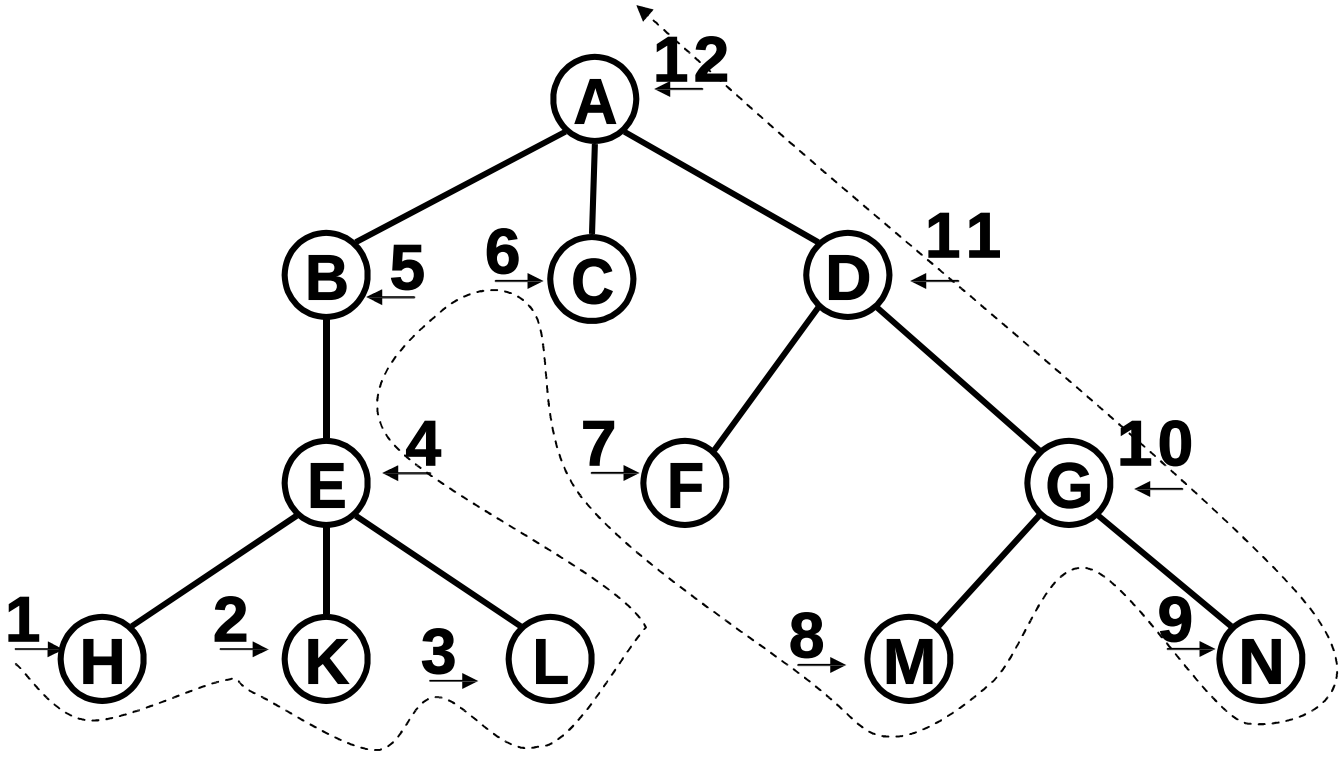
Algorithmes de parcours en profondeur (parcours infixé)



è Parcours infixé :

H, E, K, L, B, A, C, F, D, M, G, N.

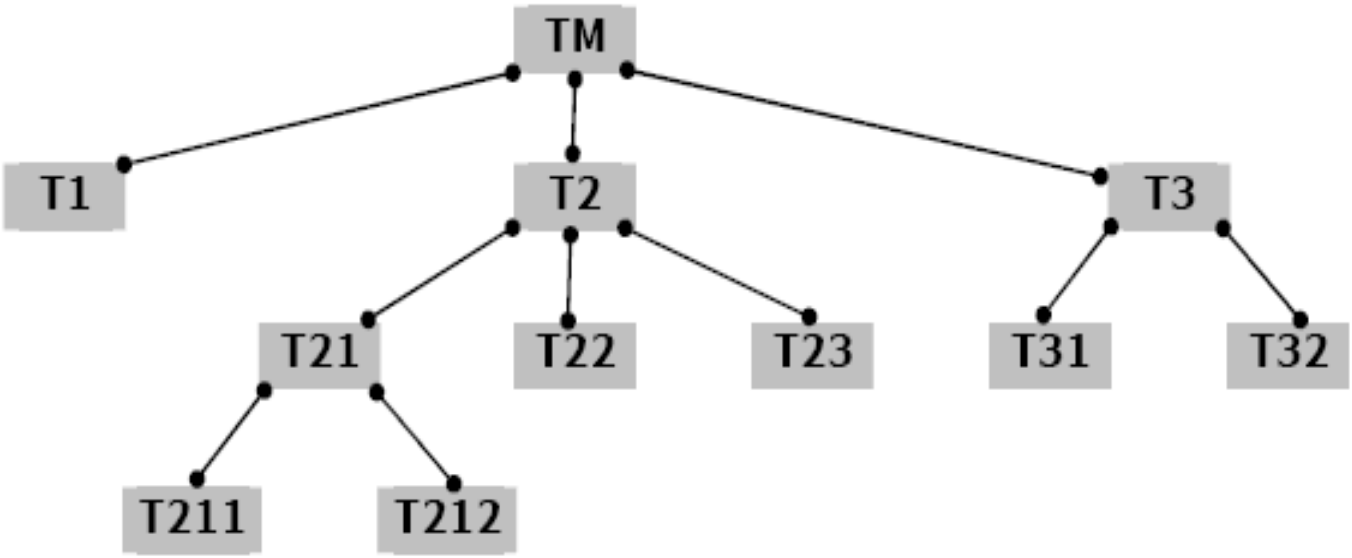
Algorithmes de parcours en profondeur (parcours postfixé)



è Parcours postfixé :

H, K, L, E, B, C, F, M, N, G, D, A.

Travail à faire



- **Parcours préfixé :**

TM, T1, T2, T21, T211, T212, T22, T23, T3, T31, T32

- **Parcours Infixé :**

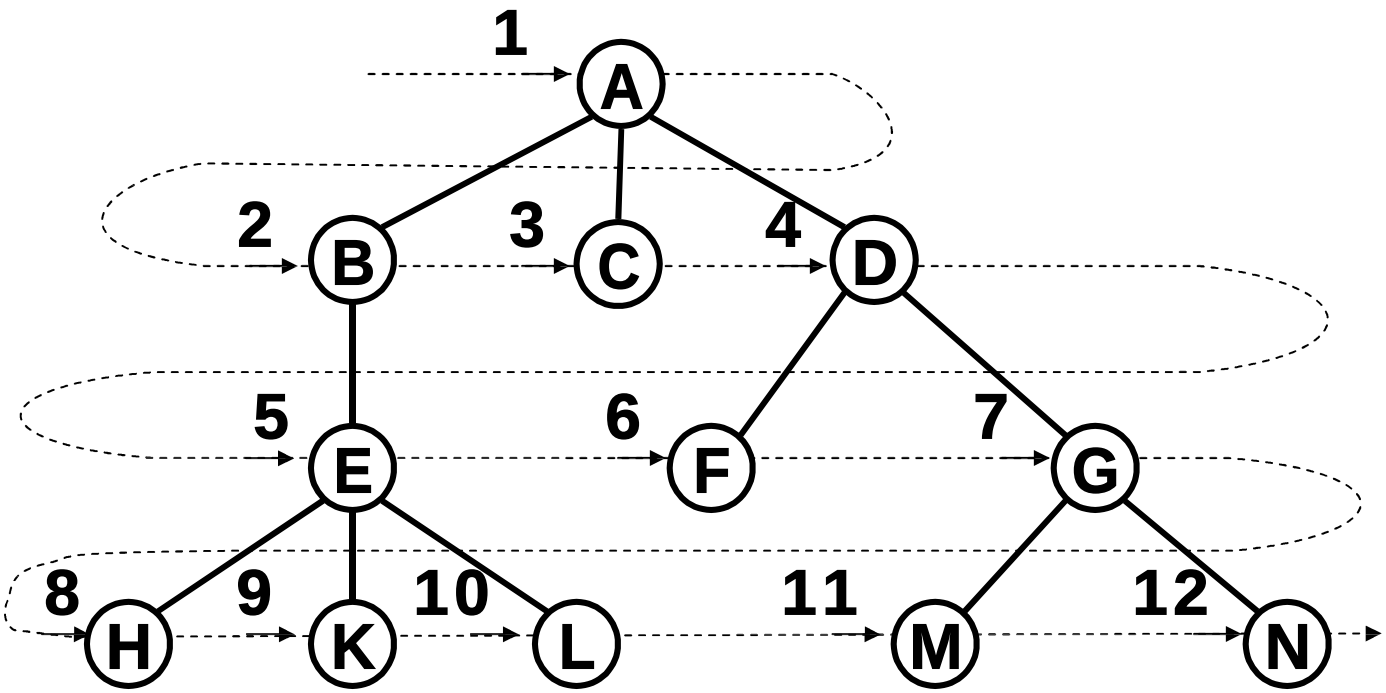
T1, TM, T211, T21, T212, T2, T22, T23, T31, T3, T32

- **Parcours postfixé :**

T1, T211, T212, T21, T22, T23, T2, T31, T32, T3, TM

Algorithme de parcours en largeur

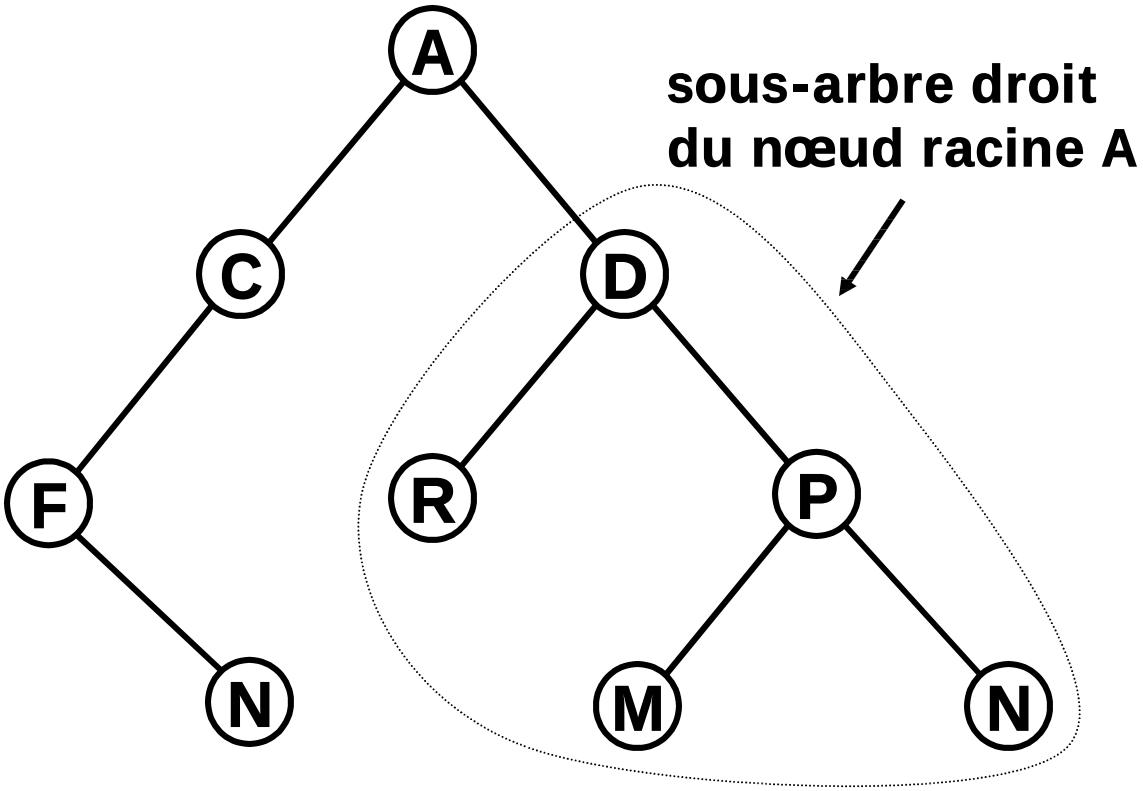
- L'algorithme de parcours en largeur visite la racine, puis chaque élément du premier niveau, avant de visiter chaque élément du deuxième niveau, etc., en visitant systématiquement chaque élément d'un niveau avant de passer au niveau suivant
- Illustration :



è Parcours en largeur : A, B, C, D, E, F, G, H, K, L, M, N.

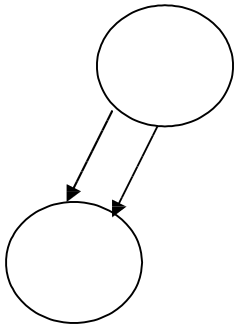
Arbres binaires

- Définitions :
  - Un arbre binaire est soit vide soit il contient une racine avec deux sous-arbres binaires
  - Un arbre binaire est un arbre où chaque nœud peut avoir au maximum deux fils : un fils gauche et un fils droit



Spécifications des primitives

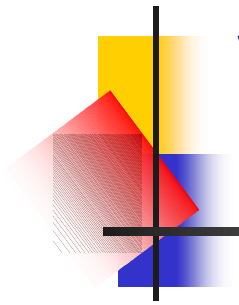
- **Procédure Construire (Don Racine: typeDesEléments, Don AG: Arb_Bin, Don AD : Arb_Bin) : Arb_Bin**



-- Précond :
-- Taille(AG) + Taille(AD) +1 < Max-size
-- *Racine* de même type de données que
-- les éléments de AG et AD
-- AG et AD ne sont pas identiques
-- Postcond : AB est un nouveau arbre binaire
-- dont la racine est *Racine*, le sous arbre droit
-- AD est et le sous arbre gauche est AG

- **Fonction Vide (Don AB : Arb_Bin) : Booléen**
  -- Précond :
  -- Postcond : retourne vrai si l'arbre binaire AB
  -- est vide et faux sinon

- **Fonction Racine (Don AB : Arb_Bin) : typeDesEléments**
  -- Précond : Non Vide (AB)
  -- Postcond : retourne le contenu de la racine de



Spécifications des primitives

- **Fonction SAGauche (Don AB : Arb_Bin) : Arb_Bin**
  -- Précond : Non Vide (AB)
  -- Postcond : retourne le sous-arbre gauche de
  -- AB

- **Fonction SADroit (Don AB : Arb_Bin) : Arb_Bin**
  -- Précond : Non Vide (AB)
  -- Postcond : retourne le sous-arbre droit de AB

- **Procédure Taille(Don AB : Arb_Bin) : Entier**
  -- Précond :
  -- Postcond : retourne la taille de AB

- **Fonction Hauteur (Don AB : Arb_Bin) : Entier**
  -- Précond :
  -- Postcond : retourne l'hauteur de AB

Spécifications des primitives

- **Procédure Afficher (Don AB : Arb_Bin, Don Ordre_parcours: Entier)**
  -- Précond : Ordre_parcours ∈ [1,3];
  -- 1: préordre, 2: inordre, 3: postordre
  -- Postcond : affiche les éléments de AB selon
  -- l'ordre spécifié par Ordre_parcours

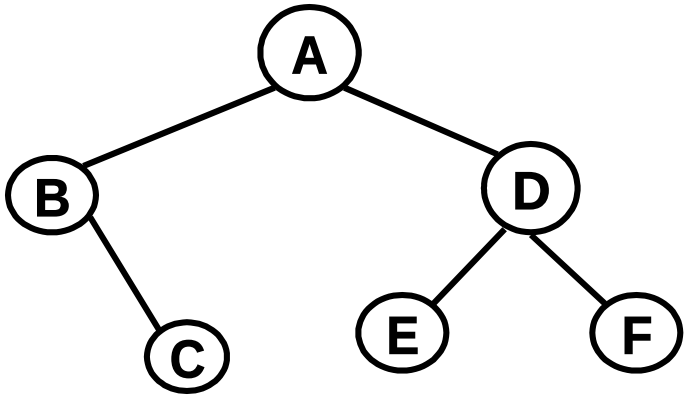
- **Fonction Copier (Don AB : Arb_Bin) : Arb_Bin**
  -- Précond :
  -- Postcond : permet d'avoir une copie de tous
  –- les éléments de AB

- **Fonction Egal(Don AB1: Arb_Bin, Don AB2: Arb_Bin): Booléen**
  -- Précond :
  -- Postcond : retourne true si AB1 et AB2 ont
  -- les mêmes éléments dans le même ordre
  -- sinon retourne false

Parcours en profondeur

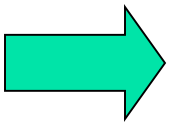
- **Si un arbre R est vide, la liste vide constitue le parcours préfixé, infixé et postfixé de R.**
- **Si R consiste en un seul nœud, la liste composée de ce nœud constitue le parcours préfixé, infixé et postfixé de R.**
- **Sinon:**
  - **parcours préfixé**
    - **Racine**
    - **Parcours préfixé du SAG**
    - **Parcours préfixé du SAD**
  - **parcours infixé**
    - **Parcours infixé du SAG**
    - **Racine**
    - **Parcours infixé du SAD**
  - **parcours postfixé**
    - **Parcours postfixé du SAG**
    - **Parcours postfixé du SAD**
    - **Racine**

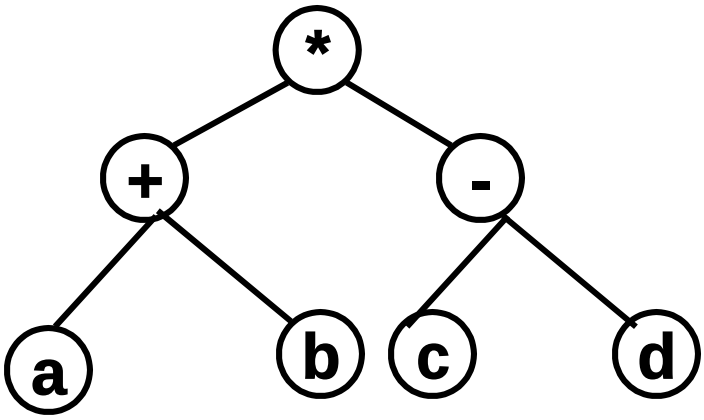
Parcours en profondeur: Exemples



- **Parcours préfixé : ABCDEF**
- **Parcours Infixé : BCAEDF**
- **Parcours postfixé : CBEFDA**

(a+b)*(c-d)





- **Parcours préfixé : *+ab-cd**
- **Parcours Infixé : a+b*c-d**
- **Parcours postfixé : ab+cd-***

Parcours préfixé dans les arbres binaires

- **Algorithme de parcours <u>préfixé</u> «RGD»: Ce parcours consiste à effectuer dans l'ordre :**
  1. **le traitement de la racine,**
  2. **le parcours du sous-arbre gauche,**
  3. **le parcours du sous-arbre droit.**

```
Procédure Préfixé (Don R : Arbre)
-- PréCond    : R est un arbre binaire
-- PostCond   : parcours préfixé de l'arbre binaire R
-- Schéma récursif
Début
    Si Non Vide (R) Alors
        Afficher (Racine (R))
        Préfixé (SAGauche (R))
        Préfixé (SADroit (R))
    Fin Si
Fin
```

Parcours infixé dans les arbres binaires

- **Algorithme de parcours <u>infixé</u> «GRD»: Ce parcours consiste à effectuer dans l'ordre :**
  1. **le parcours du sous-arbre gauche,**
  2. **le traitement de la racine,**
  3. **le parcours du sous-arbre droit.**

```
Procédure Infixé (Don R : Arbre)
-- PréCond      : R est un arbre binaire
-- PostCond     : parcours infixé de l'arbre binaire R
-- Schéma récursif
Début
    Si Non Vide (R) Alors
        Infixé (SAGauche (R))
        Afficher (Racine (R))
        Infixé (SADroit (R))
    Fin Si
Fin
```

Parcours postfixé dans les arbres binaires

- **Algorithme de parcours <u>postfixé</u> «GDR»: Ce parcours consiste** à **effectuer dans l'ordre :**
  1. **le parcours du sous-arbre gauche,**
  2. **le parcours du sous-arbre droit,**
  3. **le traitement de la racine.**

```
Procédure Postfixé (Don R : Arbre)
-- PréCond    : R est un arbre binaire
-- PostCond   : parcours postfixé de l'arbre binaire R
-- Schéma récursif
Début
    Si Non Vide (R) Alors
        Postfixé (SAGauche (R))
        Postfixé (SADroit (R))
        Afficher (Racine (R))
    Fin Si
Fin
```