Chapitre Les arbres n-aires et binaires

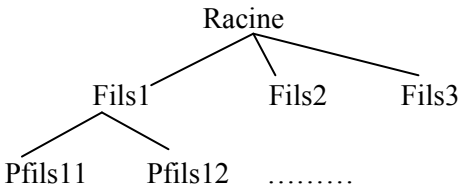
I. Introduction

Les structures **arborescentes,** appelées structures de données **hiérarchiques,** sont des structures de données arbre plus générales que les listes, elles servent à représenter des ensembles dont les éléments sont ordonnés par une relation d'ordre. En informatique les structures **arborescentes** sont très utilisées : l'organisation interne des fichiers, les modes de calcul d'une expression arithmétique, l'organisation des données triées.

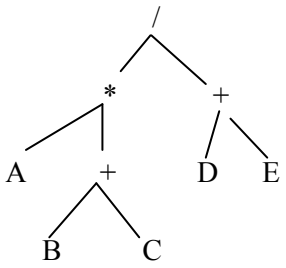
Le principal avantage des arbres par rapport aux listes est qu'ils permettent de ranger les données de telle sorte que les recherches sont de complexité plus intéressante.

Un **arbre** est une structure de données composée d'un ensemble de **nœuds** qui sont liés par une relation « Parent et Fils ». Un nœud père peut avoir plusieurs nœuds fils. Un fils n'a qu'un seul père et tous les nœuds ont un ancêtre commun appelé la **racine** de l'arbre (le seul nœud qui n'a pas de père). Chaque nœud contient l'information spécifique (qui peut être vide) vers d'autres nœuds d'arbres disjoints appelés sous-arbres.

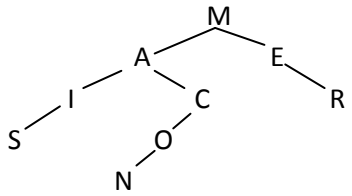
Représentation d'un arbre



Exemple d'expression arithmétique : (A*(B+C))/(D+E)



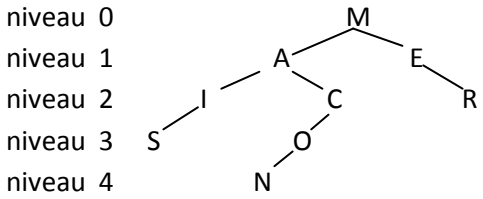
Exemple d'un arbre pour dictionnaire :



Terminologie : On associe à un arbre, généralement, les termes suivant :

- Les fils d'un nœud sont les racines de ses sous arbres, dans l'exemple précédent A et E sont les sous arbres de M ou M est leur père et aussi la racine de l'arbre.
- Le père d'un nœud (excepté la racine) est l'unique nœud dont il est fils.
- Un nœud interne est un nœud qui a au moins un fils, (A,E,I,C,O) un nœud interne.

- Une feuille d'un arbre est un nœud qui n'a pas de fils (S,N,R).
- Les ancêtres d'un nœud sont les nœuds qui le relient à la racine.
- Les descendants d'un nœud sont les nœuds qui appartiennent au sous-arbre ayant ce nœud comme racine.
- Arbre ordonné : un arbre ordonné est un arbre dans lequel l'ensemble des fils de chaque nœud est totalement ordonné.
- Forêt : une forêt est un ensemble fini d'arbres.
- La profondeur d'un nœud est la longueur du chemin reliant celui-ci à la racine.
  Le niveau d'un nœud est le nombre de nœuds entre celui-ci et la racine.
  Le niveau de la racine = 0
  Le niveau de chaque nœud est égale au niveau de son père plus 1
- La hauteur d'un arbre est la profondeur max. de ses nœuds.
- La taille d'un arbre est le nombre total de ses nœuds.
- Degré d'un nœud: Le degré d'un nœud est égal au nombre de ses fils.
  Degré de (M = 2, A =2, E= 1, R=0,...)
- Degré d'un arbre: C'est le degré maximum de ses nœuds.
  Degré de l'arbre précédent = 2.



la taille de l'arbre est 9
la hauteur est 4

Typologie des arbres

Arbre binaire : c'est un arbre ou le degré maximum d'un nœud est égal à 2.

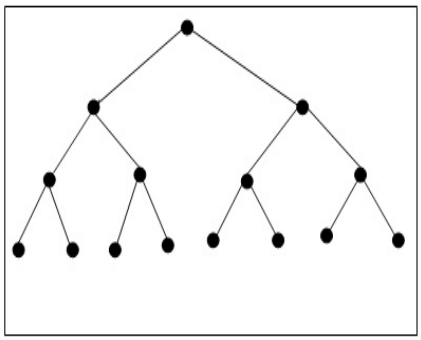
Arbre n-aire : un arbre n-aire d'ordre n est un arbre ou le degré maximum d'un nœud est égal à n.

Arbre binaire de recherche : c'est un arbre binaire où la clé de chaque nœud est supérieure à celles de ses descendants gauche, et inférieure à celles de ses descendants droits.

II. Arbre binaire

Un arbre binaire qui est soit un arbre vide, soit une racine avec deux sous-arbres binaires : fils gauche et fils droit.
Un arbre binaire est dit :
- **homogène** lorsque tous les nœuds ont deux ou zéro fils.
- **dégénéré** si tous ses nœuds n'ont qu'un seul fils.
- **complet** si chaque niveau de l'arbre est complètement rempli.
- **parfait** lorsque tous les niveaux sauf éventuellement le dernier sont remplis.

Exemple d'un arbre binaire complet.



II. 1 Implémentation

On peut réaliser l'implémentation d'un arbre, soit en statique, soit en dynamique.

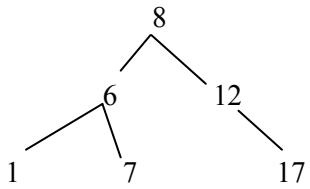
1 Statique : Par l'utilisation d'un tableau indexé où chaque élément du tableau possède trois champs :

– Un champ pour l'information effective du nœud la valeur clé,
– Un champ contenant l'indexe où se trouve le fils gauche,
– Un champ contenant l'indexe où se trouve le fils droit.

Exemple
A l'aide de tableaux (Statique)
Les arbres binaires d'entier peuvent être représentés par la structure suivante :

```
# define MaxNoeuds  100;
typedef struct Noeud{
          int val;
          int filsGauche;
          int filsDroit;
            };
Noeud Arbre[MaxNoeuds];
```

Représentation statique
Ci-dessous la représentation d'un arbre d'entiers par un tableau:



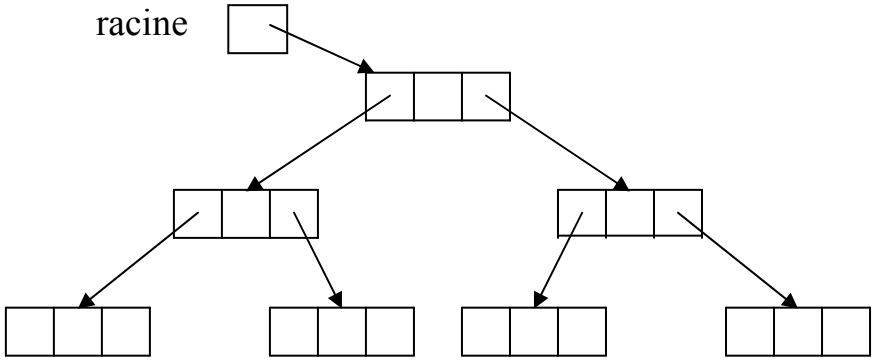
	Val	filsGauche	filsDroit
0	8	1	2
1	6	3	4
2	12	-1	**5**
3	1	**-1**	**-1**
4	7	**-1**	**-1**
5	17	**-1**	**-1**

2 Dynamique : avec une représentation chainée d'où une zone mémoire est allouée pour chaque noeud de l'arbre, cette zone contenant la **valeur clé** du noeud et deux pointeurs vers le fils gauche et le fils droit.

```
Typedef struct Noeud
        {  Element val;
           struct Noeud *FG, *FD;
        }  Noeud;
```

Un arbre binaire est déclaré et initialisé avec l'instruction :

Noeud *racine=NULL;



II.2 Opération fonctionnelle : arbre est de type Element

1. crée_nœud crée un nœud contenant la valeur x et sans fils (FG=NULL, FD=NULL) retourne un pointeur vers le nœud crée.

```
Nœud  * crée_noeud (Element x)
{    Nœud *nouv;
     nouv= malloc(sizeof(Nœud));
     nouv->val =x;
     nouv->FD=NULL ;
     nouv->FG=NULL;
     return nouv ;
}
```

2. La fonction valeur : retourne le contenu (la valeur) d'un nœud.

```
Element valeur (Noeud *N)
{
     if (N != NULL) return N -> val;
}
```

3. filsG (): retourne le fils gauche d'un noeud (l'adresse du sous arbre gauche).*

```
Noeud * filsG (Noeud *N)
{
  if (N!=NULL)return N->FG ;
}
```

4. filsD (): retourne le fils droit d'un noeud (l'adresse du sous arbre gauche).

```
Noeud * filsD (Noeud *N)
{
  if (N!=NULL)return N->FD ;
}
```

5. arbre_vide(): teste si un arbre est vide ou non.

```
int arbre_vide(Noeud *a)
{
     return a==NULL ;
}
```

6. feuille() : teste si un nœud est une feuille ou non.

```
int feuille(Noeud *N)
{
return (N!=NULL && N->FG ==NULL && N->FD ==NULL)
}
```

II.3 Parcours d'un arbre binaire

II.3.1 Parcours en profondeur :

En général pour les parcours en profondeur on distingue trois types de parcours:

1- Le parcours en pré-ordre (PREFIXE) : tout nœud est suivi des nœuds de son sous-arbre Gauche, puis des nœuds de son sous-arbre Droit.

«RGD = RACINE GAUCHE DROITE »

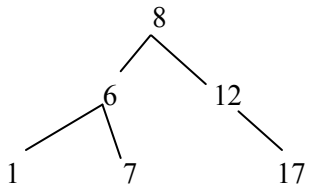
Il s'agit d'implémenter récursivement les étapes suivantes :

- Examiner (traiter l'élément de)la racine
- Parcourir en pré-ordre le sous-arbre gauche
- Parcourir en pré-ordre le sous-arbre droit

```
void prefixe (Noeud *a)

{    if  (a != NULL)    {  affiche(valeur (a)); /*
fonction qui permet d'afficher les champs de la
structure Element*/
                            prefixe ( filsG(a)) ;
                            prefixe ( filsD(a)) ;
                          }
}
```

Pour l'arbre suivant l'affichage préfixé est :8 6 1 7 12 17



2- Le parcours en ordre (INFIXE) : tout nœud est précédé des nœuds de son sous arbre Gauche, puis suivi des nœuds de son sous-arbre Droit.
« GRD = GAUCHE RACINE DROITE »
Il s'agit d'implémenter récursivement les étapes suivantes :
- Parcourir en ordre le sous-arbre gauche
- Examiner la racine
- Parcourir en ordre le sous-arbre droit

```
void infixe (Noeud *a)

{    if  (a != NULL)    {  infixe ( filsG(a)) ;
                            affiche(valeur (a));
                             infixe ( filsD(a)) ;
                          }
}
```

l'affichage infixé de l'arbre précédent est : 1 6 7 8 12 17

3- Parcours en post-ordre(POSTFIXE) : tout nœud est précédé des nœuds de son sous-arbre Gauche, et des nœuds de son sous-arbre Droit
« GDR = GAUCHE DROITE RACINE »
Il s'agit d'implémenter récursivement les étapes suivantes :

- Parcourir en post-ordre le sous-arbre gauche
- Parcourir en post-ordre le sous-arbre droit
- Examiner la racine

```
void postfixe (Noeud *a)

{    if  (a != NULL)    {  postfixe ( filsG(a)) ;
                           postfixe ( filsD(a)) ;
                           affiche(valeur (a));
                         }
}
```

l'affichage postfixé de l'arbre précédent est : 1 7 6 17 12 8

II.3.2 Parcours en largeur: Dans un parcours en largeur, tous les nœuds à la profondeur **i** doivent avoir été visités avant que le premier nœud à la profondeur **i + 1** ne soit visité. Pour y parvenir une structure de file FIFO est nécessaire pour organiser l'ensemble des branches qu'il faut visiter.