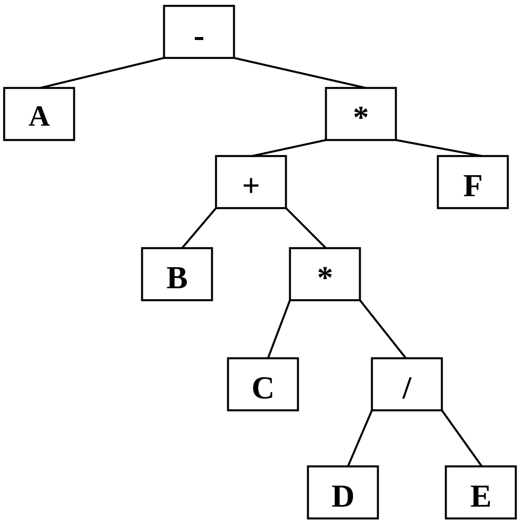
1. Introduction

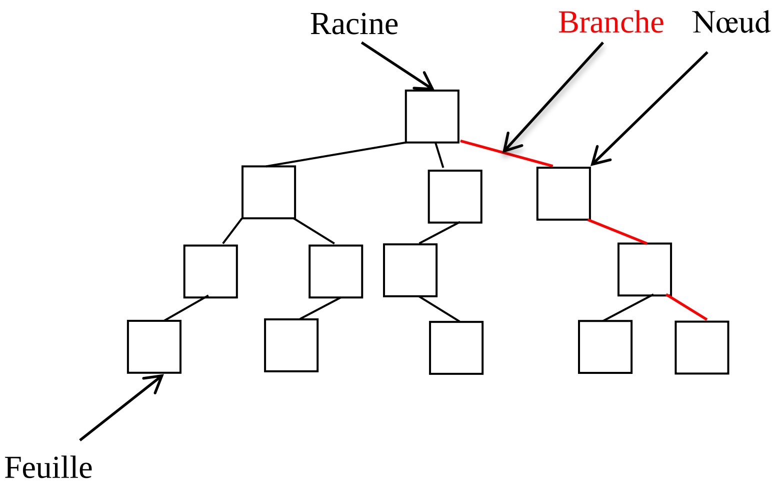
Un arbre impose une structure hiérarchique à un ensemble d'objets. Un arbre généalogique ou l'organigramme d'une entreprise en sont des exemples qui nous sont familiers. L'arbre est une structure de données fondamentale en informatique, se caractérise d'une grande commodité et d'une rapidité de manipulation.

Exemple : L'expression A - (B + C * (D / E)) * F se représente facilement par l'arbre suivant:



2. Vocabulaire employé sur les arbres

- **Nœud, racine, feuille et branche**



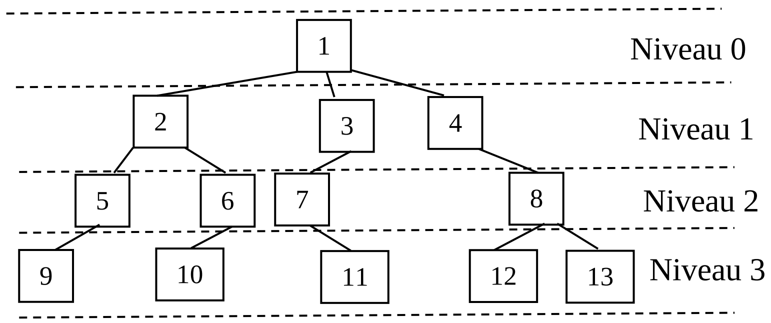
Un arbre est une structure de données qui peut soit être **vide,** soit comporter un nombre fini de **nœuds**. Chaque nœud possède un nombre fini de successeurs (fils). Un (et un seul) nœud n'est le successeur d'aucun autre, c'est la **racine**, tout autre nœud est le successeur d'un seul nœud, son père.
Un nœud qui ne possède aucun fils est appelé nœud externe ou **feuille**. Les autres nœuds sont appelés nœuds internes.

Une **branche** est une suite finie X_0, X_1, . . ., X_p de nœuds telle que X_0 est la racine et X_p est une feuille. La longueur d'une branche est le nombre de nœuds qui la composent moins 1.

- **Etiquette**

Un arbre dont tous les nœuds sont nommés est dit **étiqueté**. L'étiquette (ou nom du sommet) représente la "**valeur**" du nœud ou bien **l'information** associée au nœud.

- **Chemin, hauteur, profondeur et niveau d'un noeud**



- Le **chemin** d'un nœud X est la **suite des nœuds** par lesquels il faut passer pour aller de la racine vers le nœud X.

  <u>Exemple:</u> Chemin du nœud 12 = (1, 4, 8,12)

- La **hauteur** d'un noeud est la longueur du **plus long chemin** de ce noeud aux feuilles qui en ***dépendent plus 1*** : ***c'est le nombre de nœuds du chemin***. Les feuilles d'un arbre ont une hauteur 1.

  <u>Exemple:</u> Hauteur(9)=1 Hauteur(2) =3 Hauteur(1) = 4

  La hauteur d'un arbre est la hauteur de sa racine. L'arbre vide a une hauteur 0.

- La **profondeur** d'un nœud dans un arbre est **le nombre de noeuds** du chemin qui va de la racine à ce noeud. La racine d'un arbre est à une profondeur 1.

  <u>Exemple:</u> Profondeur(6)=3 Profondeur (3) =2 Profondeur(12)=4

Tous les nœuds d'un arbre de même profondeur appartiennent au **même Niveau**. Le nœud **racine** se trouve dans le **Niveau 0**.

La profondeur d'un arbre c'est le nombre de noeuds du chemin le plus long dans l'arbre.

- **Degré d'un nœud**

Le **degré** d'un nœud est égal au **nombre de ses fils**.

  <u>Exemple:</u> Degré(1)=3 degré (7) =1 degré (8) =2

Le degré d'un arbre est égal au **plus grand des degrés de ses nœuds.**

<u>Remarque :</u> Lorsqu'un arbre a **tous ses nœuds de degré 1**, on le nomme **arbre dégénéré.**

- **Taille d'un arbre**

La **taille** d'un arbre est le nombre total de nœuds de cet arbre.

- **arbres binaires**

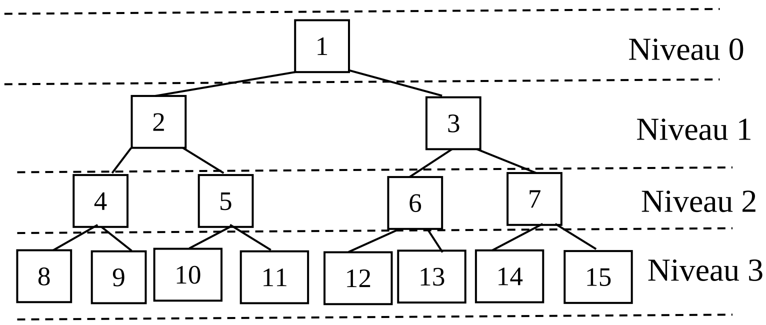
Un arbre **binaire** est un arbre de degré 2 (tous les noeuds sont de degré 2 au plus). Les fils d'un nœud sont lus de gauche à droite et sont appelés respectivement ***fils gauche*** (descendant gauche) et ***fils droit*** (descendant droit) de ce noeud.

- **Arbre n-aires *:*** c'est un arbre dont chaque nœud peut accepter de 0 à n nœuds.

- **Forêt *:*** c'est un ensemble d'arbres.

- **Arbre strictement binaire :** c'est un arbre binaire dont chaque nœud qui n'est pas une feuille a exactement deux fils. Si un arbre strictement binaire a ***n*** feuilles, le nombre total de ces nœuds est ***2n-1.***

- **Arbre binaire complet :** c'est un arbre strictement binaire où tous les nœuds feuilles sont au même niveau.



Dans un arbre binaire complet de profondeur P, le nombre de nœuds (taille) est :

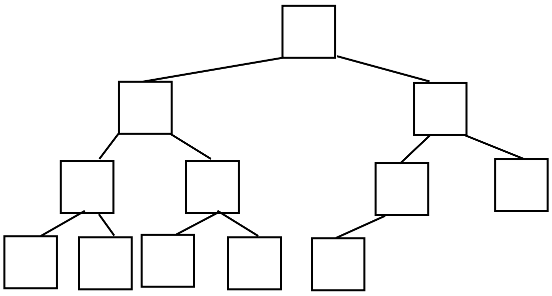
$$N = 2^0 + 2^1 + 2^2 + \ldots + 2^{P-1} = 2^P - 1$$

Donc 2^{P-1} noeuds feuilles et $2^{P-1}-1$ nœuds non feuilles.

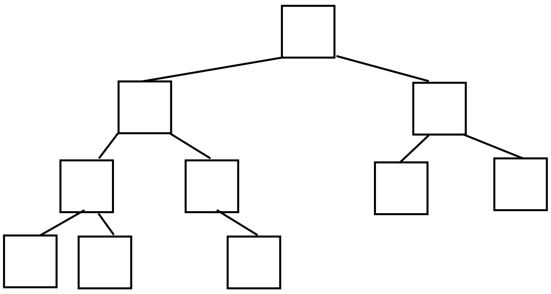
Inversement, connaissant le nombre de nœuds d'un arbre binaire complet on peut retrouver sa profondeur (hauteur): N = $2^P-1 \Rightarrow P = Lg_2(N+1)$

- **Arbre binaire parfait**

C'est un arbre binaire dont tous les noeuds de chaque niveau sont présents sauf éventuellement au dernier niveau où il peut manquer des noeuds (noeuds feuilles). Les feuilles du dernier niveau **doivent être regroupées à partir de la gauche** de l'arbre.



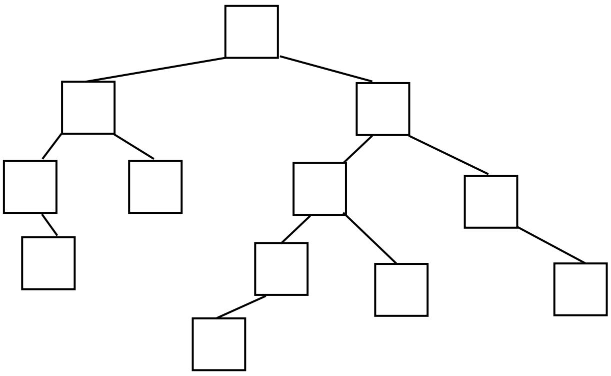
Arbre binaire parfait



Arbre binaire imparfait

- **Arbre binaire équilibré**

Un arbre binaire est dit équilibré si, pour tout nœud de l'arbre, les sous-arbres gauche et droit ont des hauteurs qui diffèrent au plus de 1. L'arbre ci-dessous est équilibré.

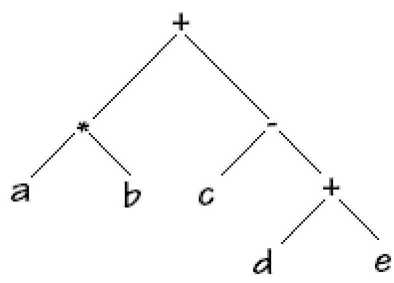
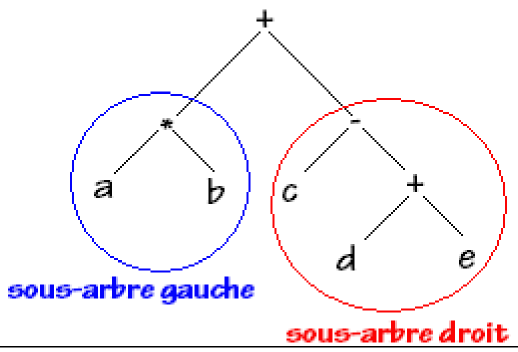


3. Arbres binaires

3.1. Définition

Un arbre binaire sur un ensemble fini de nœuds est soit vide, soit l'union disjointe d'un nœud appelé sa ***racine***, d'un arbre binaire appelé ***sous-arbre gauche***, et d'un arbre binaire appelé ***sous-arbre droit***. Il est utile de représenter un arbre binaire non vide sous la forme d'un triplet *A*= < R, SAG, SAD >.

Exemple : soit l'arbre binaire A :

A = 	Les sous-arbres gauche et droit de l'arbre A : SAG(A) = **<*,< a,φ,φ>,<b,φ,φ> >** SAD(A) = **<- , <c,φ,φ> , <+,<d,φ,φ>,<e,φ,φ> > >** 

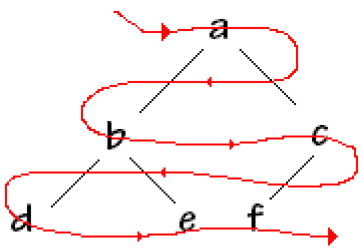
3.2. Parcours d'un arbre binaire

L'opération qui consiste à **retrouver** systématiquement tous les noeuds d'un arbre et d'y appliquer un **même traitement** se dénomme **parcours** de l'arbre.

3.2.1. Parcours en largeur ou hiérarchique :

Cet algorithme consiste à **explorer** chaque noeud d'un niveau donné de **gauche à droite (droite à gauche)**, puis de passer au niveau suivant.

Exemple:



Parcours en largeur : a b c d e f

3.2.2. Parcours en profondeur

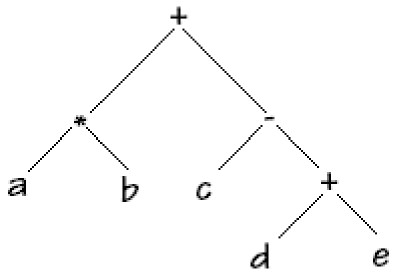
L'algorithme consiste à **descendre** le plus profondément **jusqu'aux feuilles** d'un nœud de l'arbre, puis lorsque toutes les feuilles du noeud ont été visitées, l'algorithme "**remonte**" au noeud plus haut dont les feuilles n'ont pas encore été visitées.

a. Parcours en pré-ordre (ordre préfixé)
On parcourt d'abord la racine, puis le fils gauche en ordre préfixé, puis le fils droit en ordre préfixé.

b. Parcours en ordre symétrique (ordre infixé)
On parcourt d'abord le fils gauche en ordre infixé, puis la racine, puis le fils droit en ordre infixé.

c. Parcours en post-ordre : (ordre postfixé)
On parcourt d'abord le fils gauche en ordre postfixé, puis le fils droit en ordre postfixé, puis la racine.

Exemple: donner les résultats du parcours de l'arbre binaire A ci-dessous.

A = 	**En largeur : + * - a b c + d e** **Pré-ordre (préfixé) :+ * a b – c + d e** **en-ordre (infixé) : a * b + c – d + e** **Post-ordre (postfixé) : a b * c d e + - +**

3.3. Les opérations de base sur les arbres binaires

Chaque nœud de l'arbre binaire contient une valeur de type TValeur.

Opération
CreerArbreVide : Création d'un nouvel arbre binaire vide
ArbreVide : Test si un arbre binaire est vide
ValeurNœud : retourne la valeur contenue dans un nœud d'un arbre binaire
CréerNœud : créer un nœud d'un arbre binaire
FilsGauche : retourne le fils gauche d'un nœud
FilsDroit : retourne le fils droit d'un nœud
Feuille, Degre, Hauteur, Profondeur, Taille,…

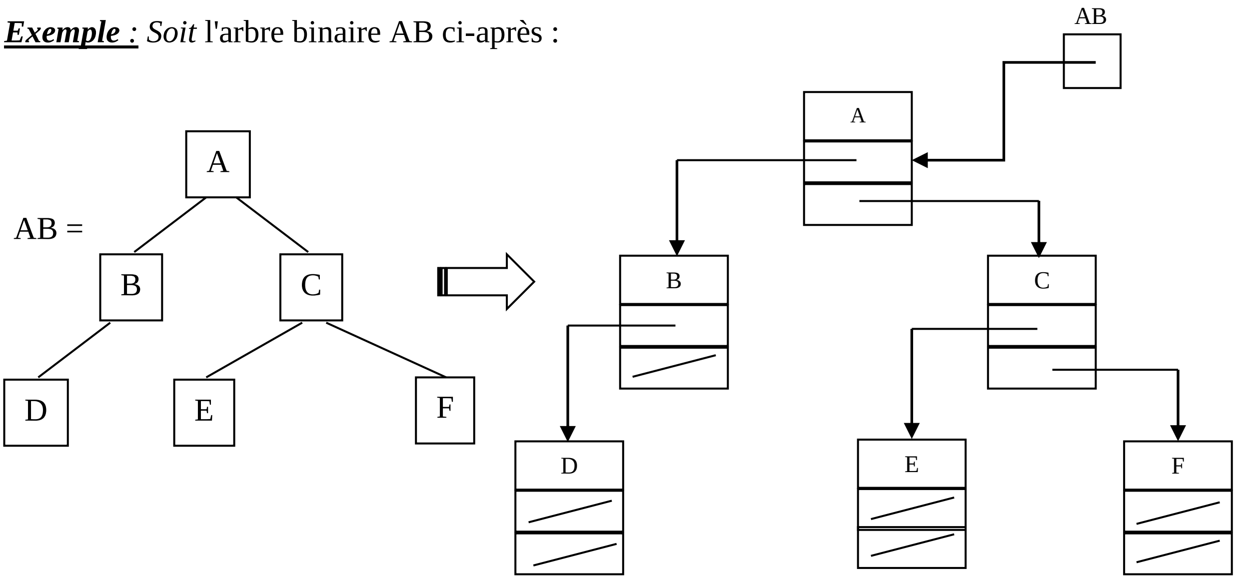
4. Implémentation d'arbre binaire étiqueté avec pointeurs *(listes chaînées)*

L'arbre binaire est donné par l'adresse de sa racine. Un noeud de l'arbre est une structure contenant les éléments suivants:

- l'information du nœud (la valeur du nœud)
- un pointeur vers le fils gauche
- un pointeur vers le fils droit

L'absence d'un fils est représentée par la valeur Nil.

Exemple : *Soit* l'arbre binaire AB ci-après :



- ***Déclaration***

Type

TValeur = …
ArbreBin = ↑ Nœud
Nœud = **Enregistrement**
Valeur : TValeur
FilsG, FilsD : ArbreBin
FinEnregistrement

- ***Description des opérations de base***

```
Procedure CreerArbreVide (Ref A : ArbreBin)
  Debut
        A ← Nil
  Fin
FinProcedure

Fonction  ArbreVide (Val A : ArbreBin) : booléen
  Debut
        ArbreVide ← (A = Nil)
  Fin
FinFonction
```

```
Fonction CreerNoeud (Val V : Tvaleur, FG : ArbreBin, FD : ArbreBin) : ArbreBin
  Variables
     P : ↑Noeud
  Debut
       Allouer(P)
       P ↑.Valeur ← V
       P ↑.FilsG ← FG
       P ↑.FilsD ← FD
       CreerNoeud  ← P
  Fin
FinFonction

Fonction  ValeurNoeud (Val A : ArbreBin) : TValeur
  Debut
            FilsGauche ← …………………….
  Fin
FinFonction

Fonction  FilsGauche (Val A : ArbreBin) : ArbreBin
  Debut
            FilsGauche ← ………………….
  Fin
FinFonction

Fonction  FilsDroit (Val A : ArbreBin) : ArbreBin
  Debut
            FilsDroit ← ………………
  Fin
FinFonction

Fonction  Taille (Val A : ArbreBin) : entier
  Debut
            Si A = Nil alors
                Taille  ←…..
            Sinon
                 Taille ← ………………………………………………………
            Finsi
  Fin
FinFonction

Fonction  Feuille (Val A : ↑Noeud) : booléen
  Debut
            Feuille ←  …………………………………………….
  Fin
FinFonction
```

```
Fonction  Hauteur (Val A : ↑Noeud) : entier
  Debut
        Si A = Nil alors
           Hauteur  ← ……
        Sinon
            Hauteur ← ……………………………………………………………………
        Finsi
  Fin
FinFonction

Fonction  Degre (Val A : ↑Noeud) : entier
  Debut
        Si Feuille(A) Alors
           Degre  ← ……
        Sinon
            Si (FilsGauche (A)  ≠  Nil  et  FilsDroit (A) ≠  Nil)  Alors
              Degre  ← …….
            Sinon
              Degre  ← ……
            Finsi
        Finsi
  Fin
FinFonction

Procédure ParcoursEnLargeur (Val A : ArbreBin)
     Variables
             F : File
      Début
           Si  A ≠ Nil Alors
               CreerFileVide(F)
               Enfiler (A,F)
               Tant que Non FileVide (F)Faire
                    A ← TeteFile(F)
                    Defiler (F)
                    Ecrire(ValeurNoeud (A))
                 Si ((FilsGauche (A)  ≠Nil) alors
                      Enfiler (FilsGauche (A),F)
                 Fsi
                 Si  (FilsDroit (A) ≠ Nil) alors
                     Enfiler (FilsDroit(A),F)
                Fsi
              FTantque
          Fsi
     Fin
 FinProcédure
```

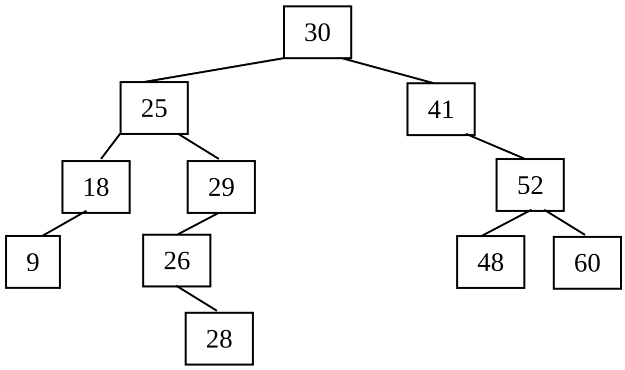
5. Arbres binaires de recherche

5.1. Définition

Un arbre binaire de recherche satisfait aux critères suivants :

- Une partie de l'étiquette (ou l'étiquette) est dénommée **clé**.
- Les **clés** de tous les noeuds du sous-arbre **gauche** d'un noeud X, sont ***strictement* inférieures *(*ou égales)** à la clé de X.
- Les **clés** de tous les noeuds du sous-arbre **droit** d'un noeud X, sont ***strictement* supérieures** à la clé de X.

Exemple:



Remarque: Un parcours infixe d'un arbre binaire de recherche permet de traiter les clés de la plus petite à la plus grande pour un parcours infixe gauche, ou de la plus grande à la plus petite pour un parcours infixe droite.

5.2. Implémentation

- ***Déclaration***

```
Type
  ABR = ↑ Nœud
  Tcle = …
  Tval = …
  TValeur =  Enregistrement
                  Cle : Tcle
                  Val : Tval
               FinEnregistrement
  Nœud = Enregistrement
              Valeur : TValeur
              FilsG, FilsD : ABR
           FinEnregistrement

Fonction  CleNoeud (Val A : ↑ Nœud) : Tcle
  Debut
        CleNoeud ← A ↑.Valeur.Cle
  Fin
FinFonction
```

- ***Opérations sur les arbres binaires de recherche***

1. Recherche d'un nœud ayant une clé donnée

```
Fonction RechercherABR (val Cl: Tcle, val A: ABR):ABR
    Début
      Si A = Nil Alors
        RechercherABR ← ……
      Sinon
            Si (Cl = CleNoeud(A)) Alors
                RechercherABR ← …….
            Sinon
                  Si (Cl < CleNoeud(A)) Alors
                        RechercherABR ← …………………………………….
                  Sinon
                        RechercherABR ← …………………………………….
                  Fsi
            Fsi
      Fsi
    Fin
 Finfonction
```

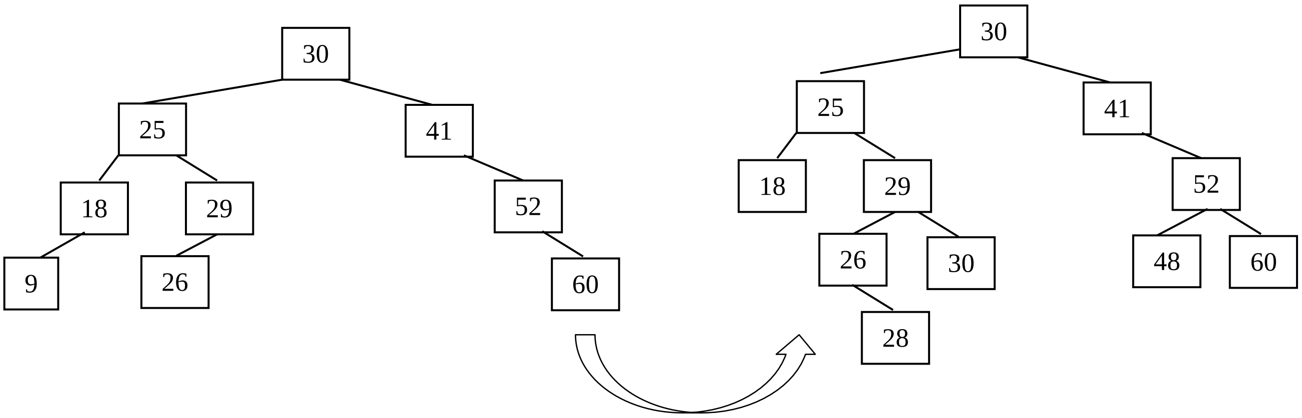
2. Recherche du nœud qui contient la plus petite clé (Minimum)

```
Fonction  MinABR (Val A : ABR) : ABR
  Début
    Si  A = Nil Alors
         MinABR ← Nil
    Sinon
         Tant que (……………………...) Faire
              ………………………
         Ftantque
         MinABR ← ……..
    Fsi
  Fin
Finfontion
```

3. Recherche du nœud qui contient la plus grande clé (Maximum)

```
Fonction MaxABR (Val A : ABR) : ABR
    Début
      Si  A = Nil Alors
          MaxABR ← Nil
     Sinon
         Si ……………………………  Alors
             MaxABR ← ………
         Sinon
             MaxABR ← ……………………….
         Finsi
    Fsi
  Fin
Finfontion
```

4. Insertion d'un nœud



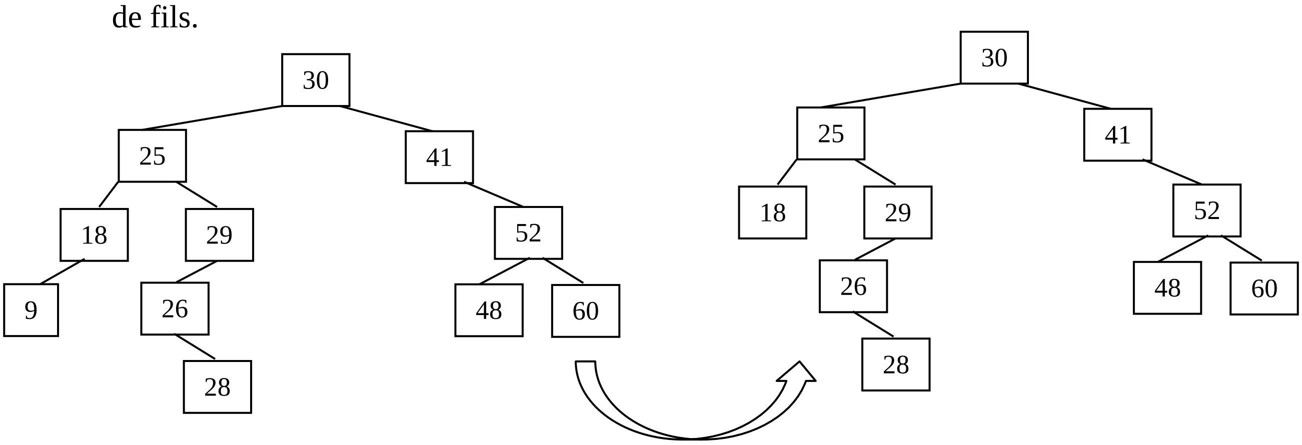
Insertion successive de trois nœuds ayant comme clés 28, 48 et 30

```
Procédure InsererABR (Val V: TValeur, Ref A: ABR)
     Début
         Si A = Nil Alors
              A ← CreerNoeud(V, Nil, Nil)
         Sinon
              Si V.Cle ≤ CleNoeud(A) Alors
                  Inserer (V, A↑.FilsG)
              Sinon
                 Inserer (V, A↑.FilsD)
               Fsi
         Fsi
     Fin
 FinProcédure
```

5. Suppression d'un nœud

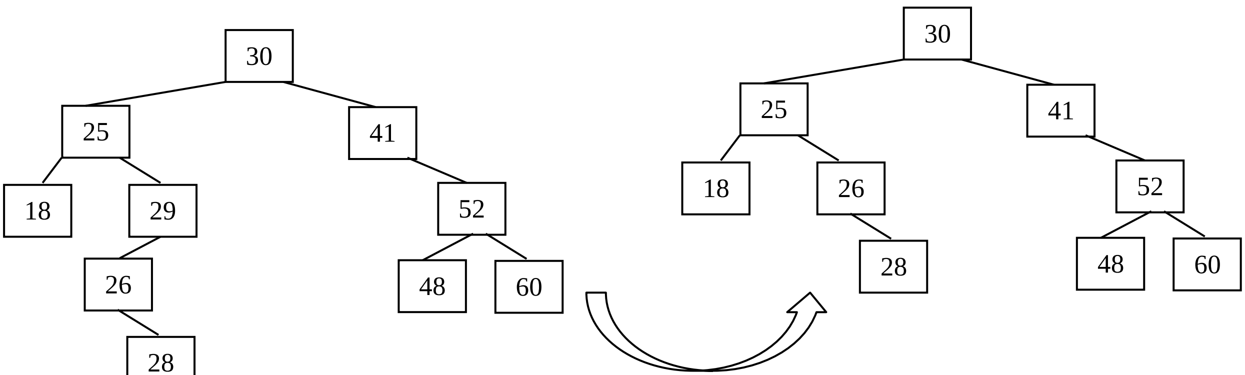
La suppression d'un noeud nécessite une précaution : il faut garder la propriété d'arbre binaire de recherche.
Une fois que le nœud à supprimer a été trouvé à partir de sa clé, on se trouve dans une des situations suivantes:

- ***Le nœud en question est une feuille*** : Il suffit de l'enlever de l'arbre vu qu'elle n'a pas de fils.



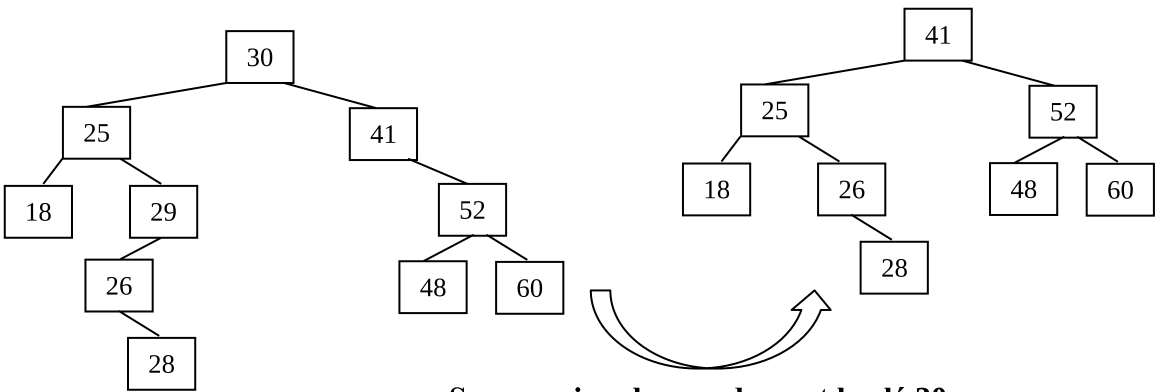
Suppression du nœud ayant la clé 9

- ***Le nœud en question est*** **un nœud avec un enfant** : Il faut l'enlever de l'arbre en le remplaçant par son fils.



Suppression du nœud ayant la clé 29

- ***Le nœud en question est*** **un nœud avec deux enfants** : remplacer la valeur du nœud à supprimer par la plus grande valeur des descendants de son fils gauche ou par la plus petite valeur des descendants de son fils droit puis supprimer le nœud qui a cette valeur.



Suppression du nœud ayant la clé 30