Université de Paris

COURS 11

Programmation impérative

Implémentation de Types Abstraits de données

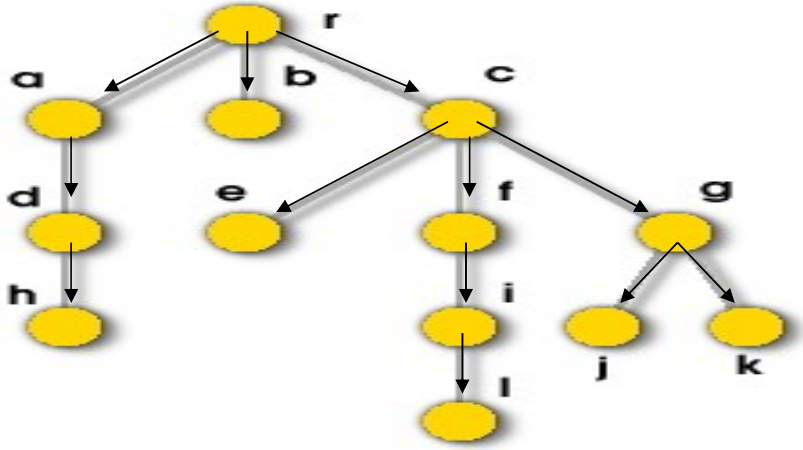
- Listes
  - Implémentation par liste chaînée
  - Cas particuliers : Piles - Files
- Arbres

SOMMAIRE

Implémentation de structures de données

- Graphe
  - Cycle
    - Chemin = liste de sommets telle qu'il existe dans le graphe une arête entre chaque paire de sommets successifs
    - Cycle = chemin finissant à son point de départ

- Arbre
  - Graphe orienté connexe et sans cycle

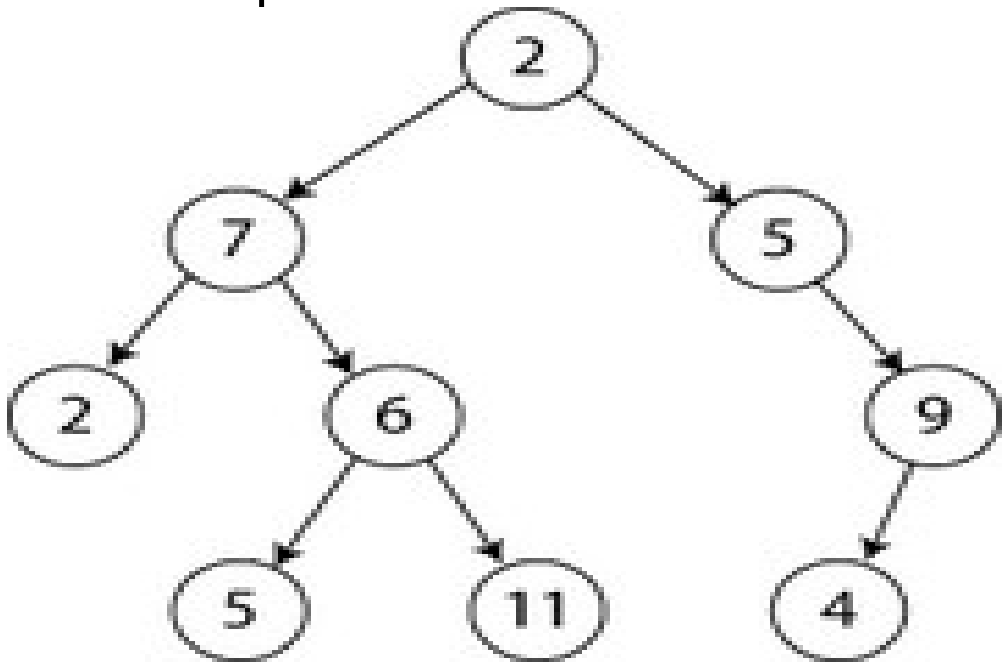


Arbres

- Vocabulaire
  - Arbre étiqueté
    - Chaque nœud possède une étiquette = contenu du noeud
  - Feuille – nœud externe
    - Éléments n'ayant pas de fils
  - Nœuds internes – nœud père
    - Éléments possédant des fils (sous branches)
  - Lien nœud père –fils
    - Reliés entre eux par des arêtes
  - Racine
    - unique nœud ne possédant pas de parents
  - Profondeur – hauteur d'un nœud
    - Distance (= nombre d'arêtes) de la racine au nœud
  - Hauteur d'un arbre
    - Plus grande profondeur d'une feuille de l'arbre

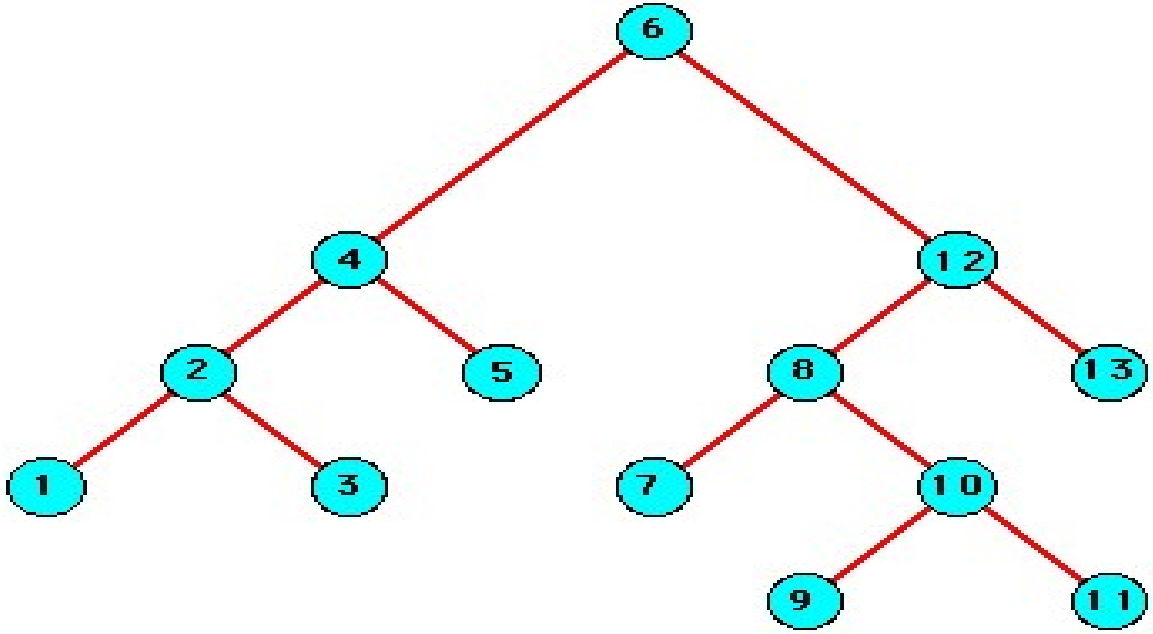
Arbres

- Arbre binaire
  - Arbre dont chaque nœud a au plus 2 fils
    - Fils gauche
    - Fils droit



Arbres

- Arbre binaire de recherche
  - Sous arbre gauche (resp. droit) de tout nœud ne contient que des valeurs strictement plus petites (resp. grandes)
  - Permettent des recherches rapides et efficaces



Arbres

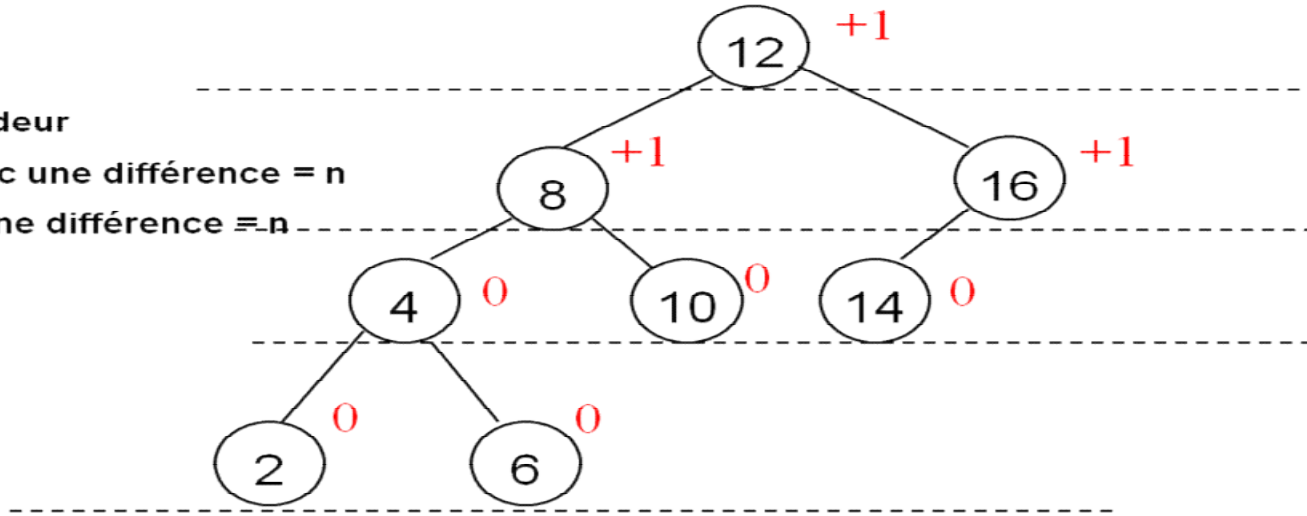
- Arbres équilibrés
  - AVL en 1962 par Adelson-Velskii et Landis
  - Différence entre les hauteurs des fils gauche et droit de tout nœud ne peut excéder 1

Pour chaque noeud on mettra

0 si ses deux sous-arbres ont la même profondeur

+n si le sous-arbre gauche est plus profond avec une différence = n

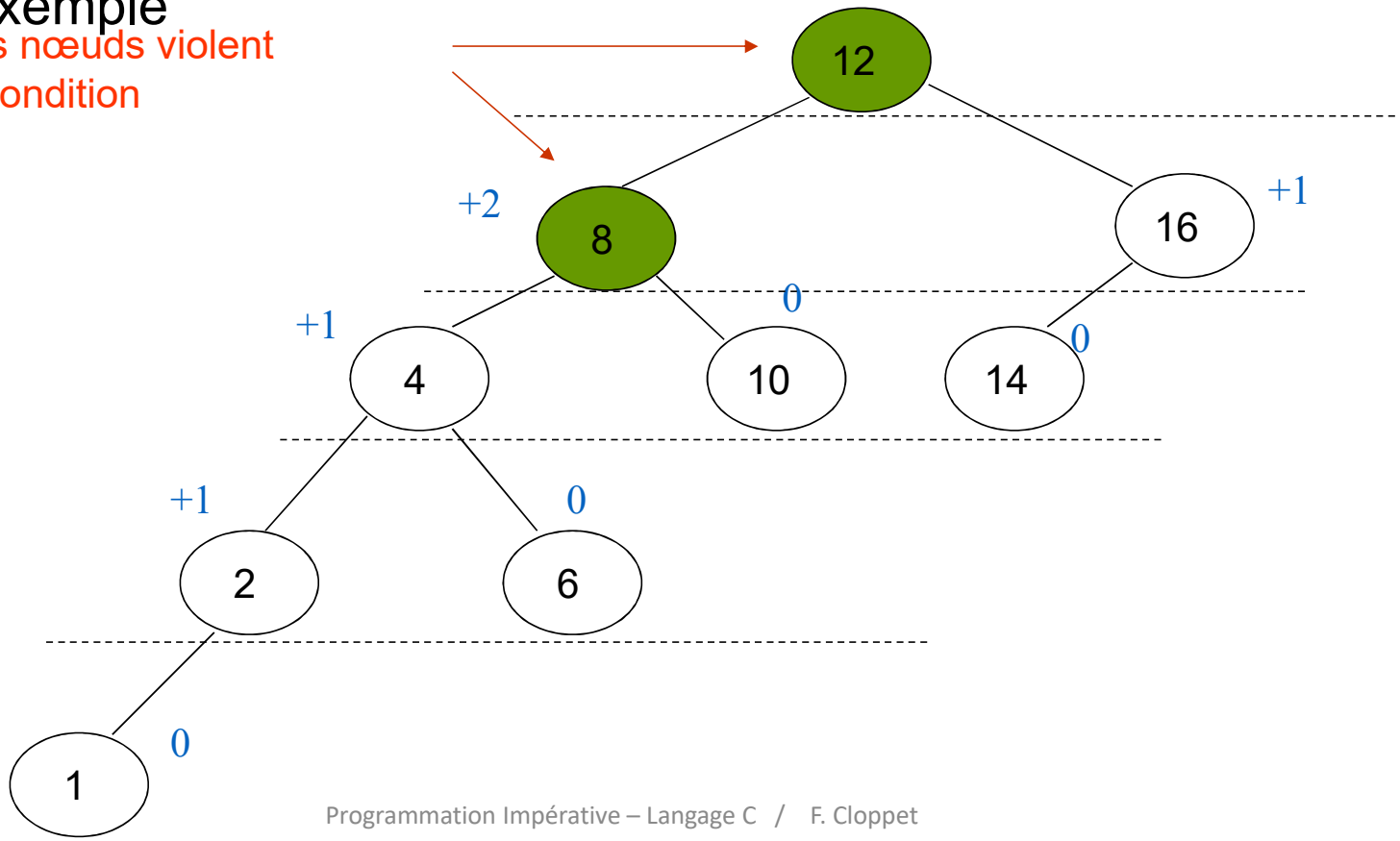
-n si le sous-arbre droit est plus profond avec une différence = n



Arbres

- Arbres équilibrés
  - Contre exemple

Ces nœuds violent la condition



Arbres

- Arbres équilibrés
  - Il faut, après chaque insertion ou retrait, rétablir l'équilibre s'il a été rompu par l'opération
  - Observation importante: après une insertion, seuls les nœuds qui sont sur le chemin du point d'insertion à la racine sont susceptibles d'être déséquilibrés
  - Deux cas:

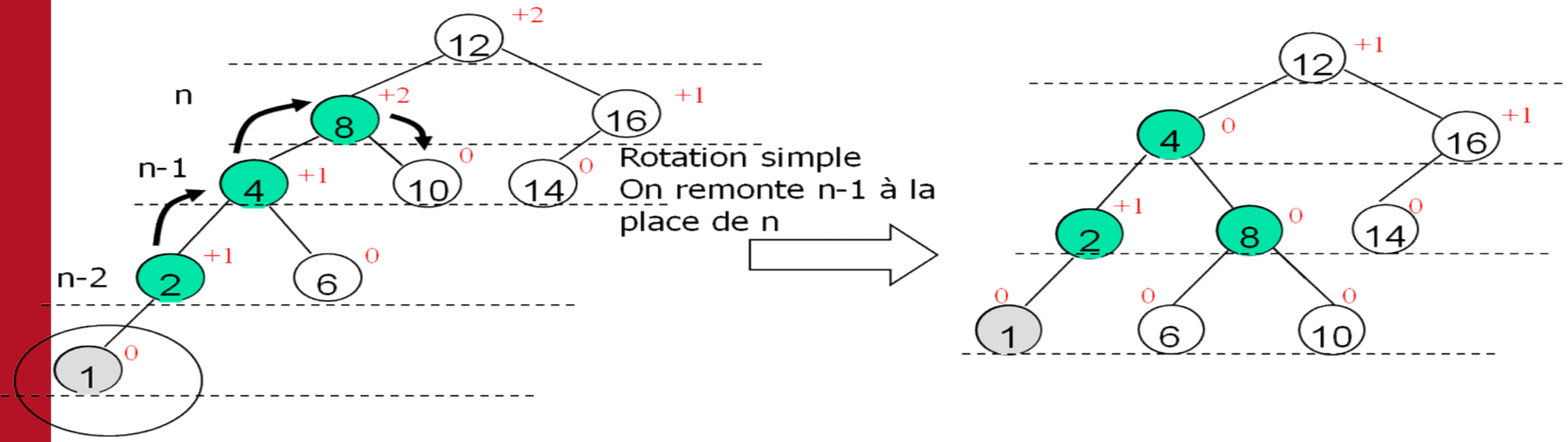
    déséquilibre du nœud n est dû à
    - Une insertion dans le sous-arbre de gauche du fils gauche du nœud n ou dans le sous-arbre de droite du fils droit de n:

    ⇒ Simple rotation
    - Une insertion dans le sous-arbre de droite du fils gauche de n ou dans le sous-arbre de gauche du fils droit de n:

    ⇒ Double rotation

Arbres

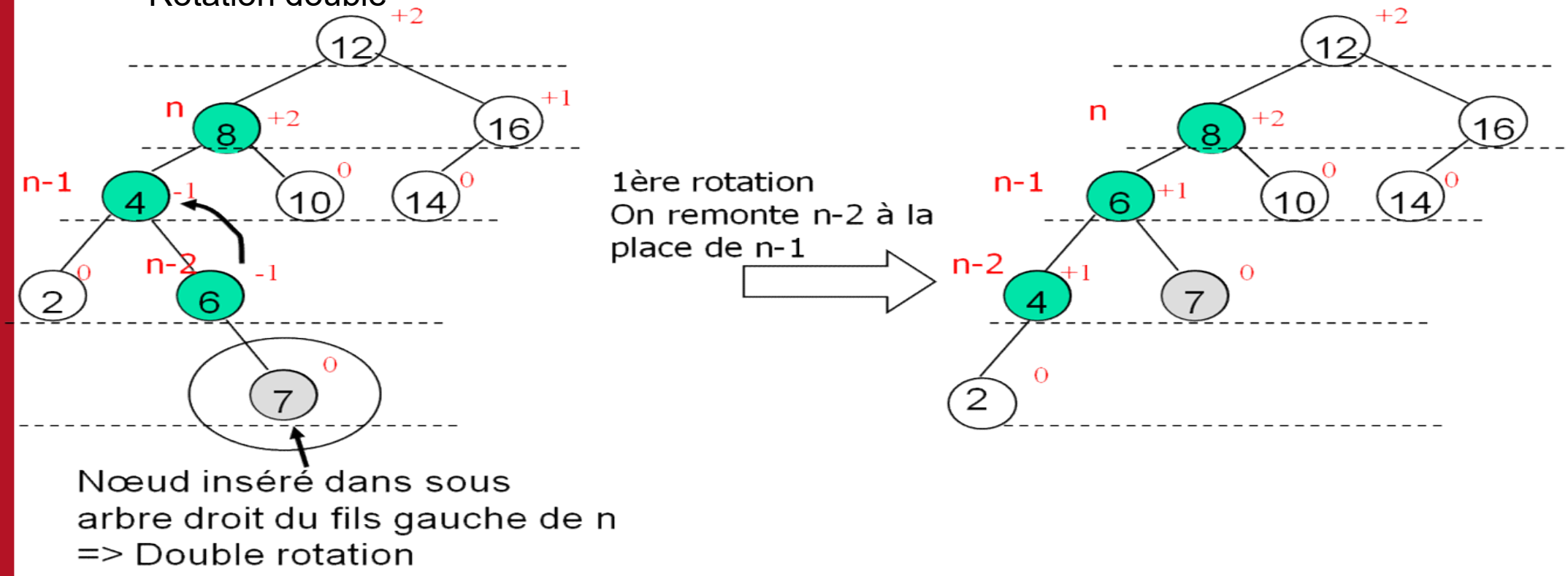
- Arbres équilibrés
  - Rotation simple



Nœud inséré dans sous arbre gauche du fils gauche de n
=> Déséquilibre corrigé par une rotation simple

Arbres

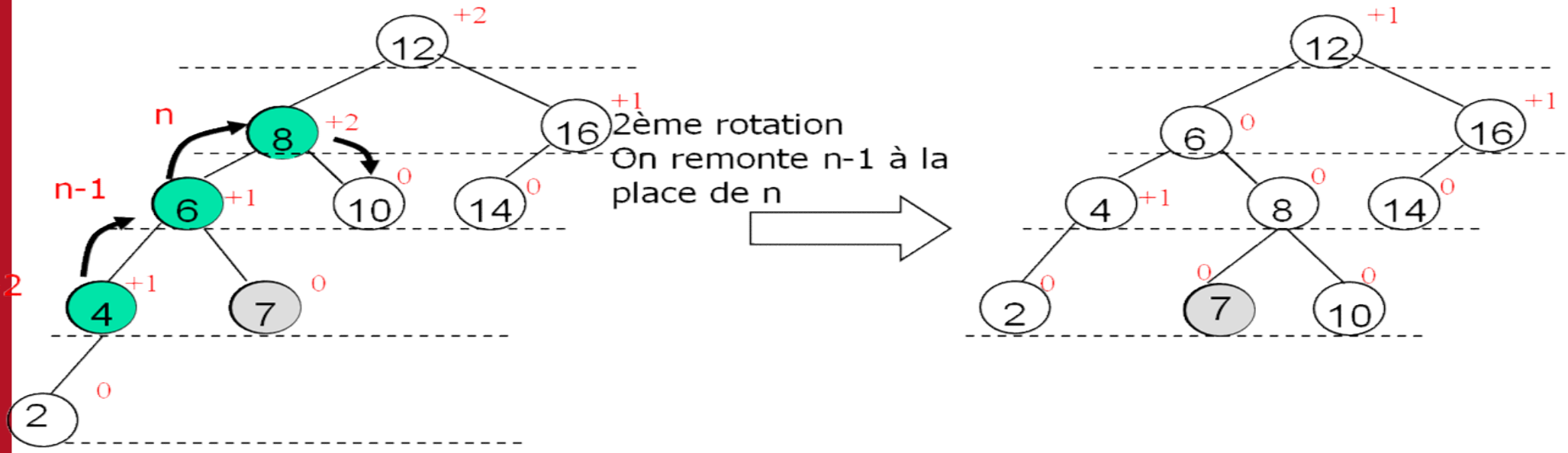
- Arbres équilibrés
  - Rotation double



Nœud inséré dans sous arbre droit du fils gauche de n
=> Double rotation

Arbres

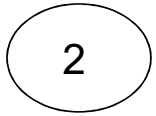
- Arbres équilibrés
  - Rotation double suite



Arbres

- AVL – exemple détaillé

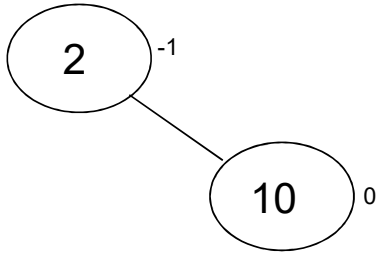
2 10 12 4 16 8 6 14



Arbres

- AVL – exemple détaillé

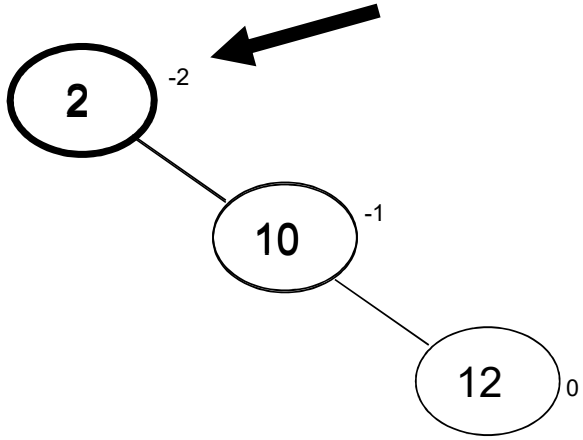
2 10 12 4 16 8 6 14



Arbres

- AVL – exemple détaillé

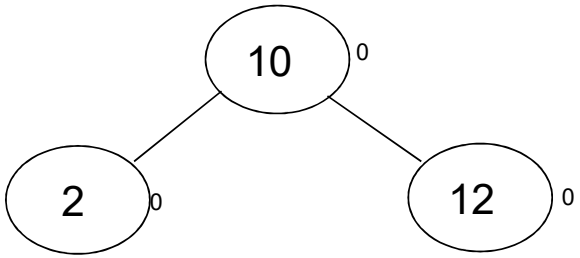
2 10 12 4 16 8 6 14



Arbres

- AVL – exemple détaillé

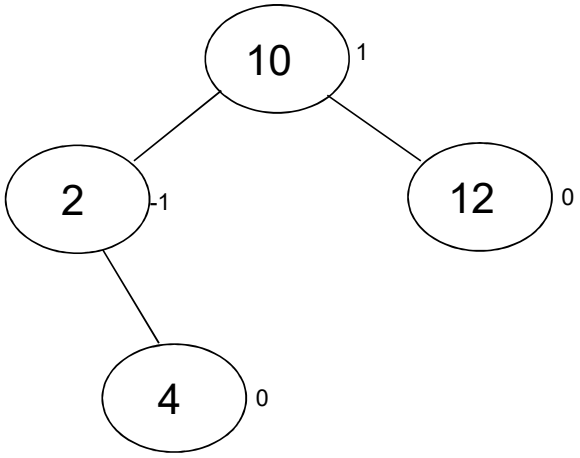
  2 10 12 4 16 8 6 14



Arbres

- AVL – exemple détaillé

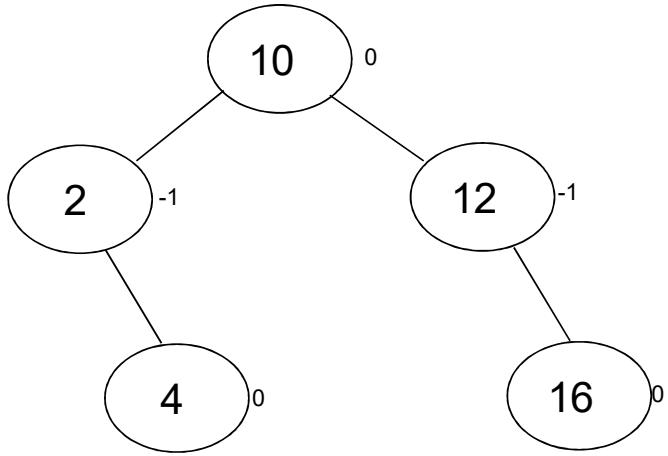
2 10 12 4 16 8 6 14



Arbres

- AVL – exemple détaillé

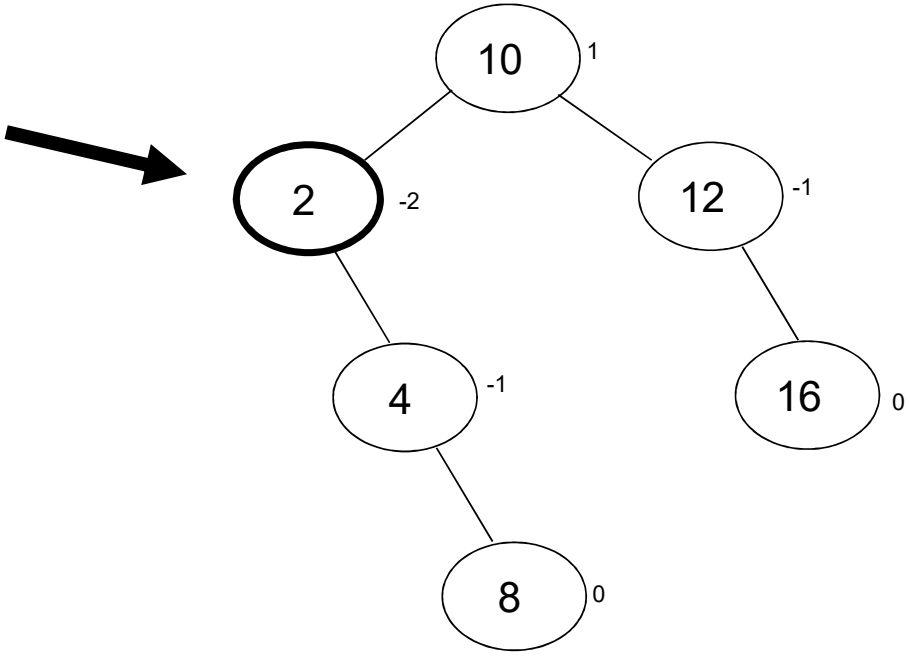
  2 10 12 4 16 8 6 14



Arbres

- AVL – exemple détaillé

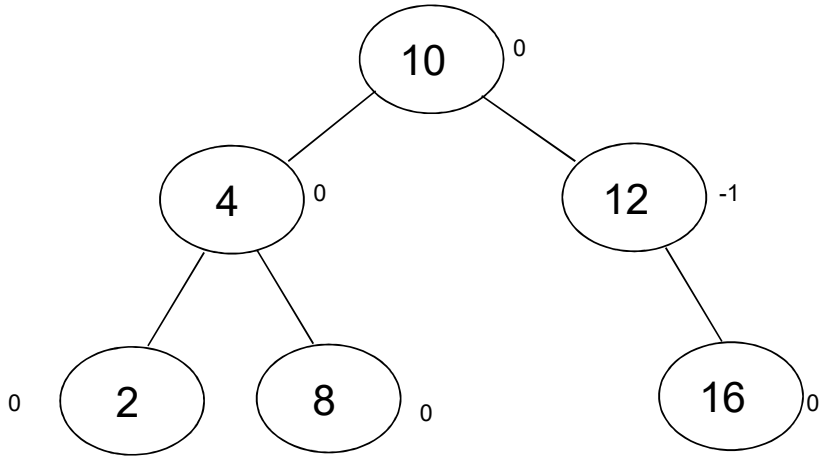
2 10 12 4 16 8 6 14



Arbres

- AVL – exemple détaillé

2 10 12 4 16 8 6 14

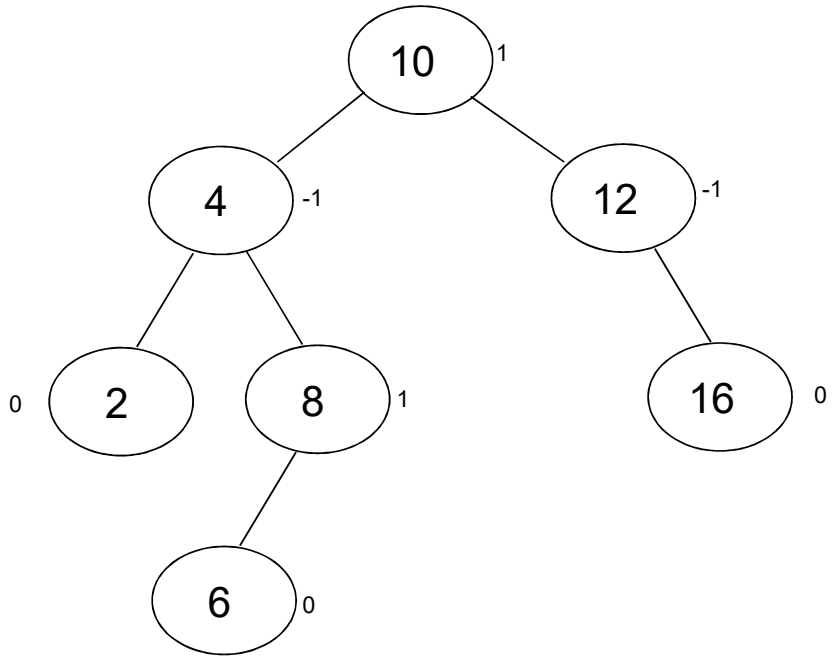


Rotation simple

Arbres

- AVL – exemple détaillé

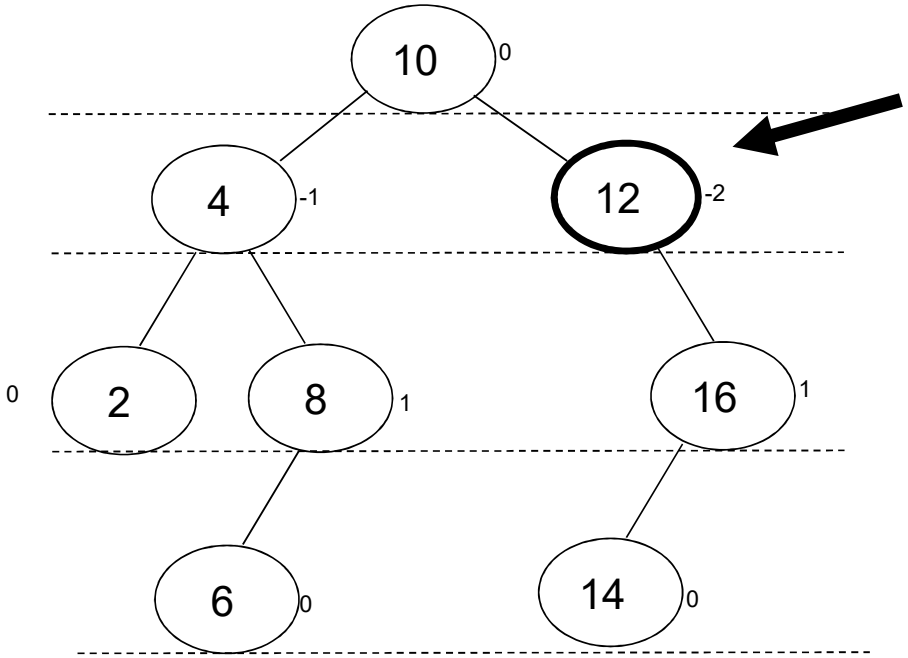
  2 10 12 4 16 8 6 14



Arbres

- AVL – exemple détaillé

2 10 12 4 16 8 6 14

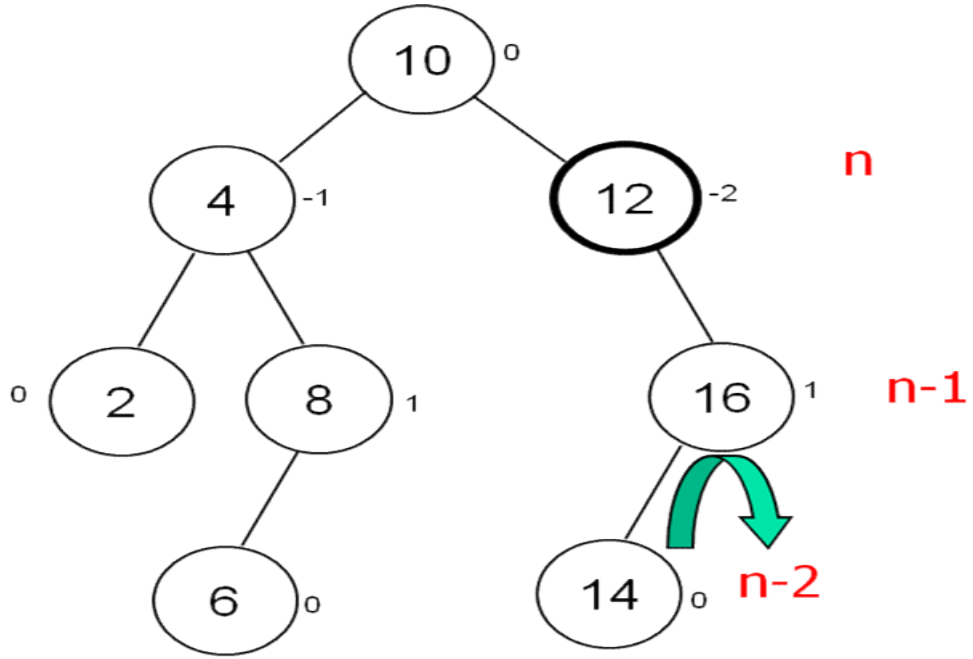


Rotation double

Arbres

- AVL – exemple détaillé

2 10 12 4 16 8 6 14

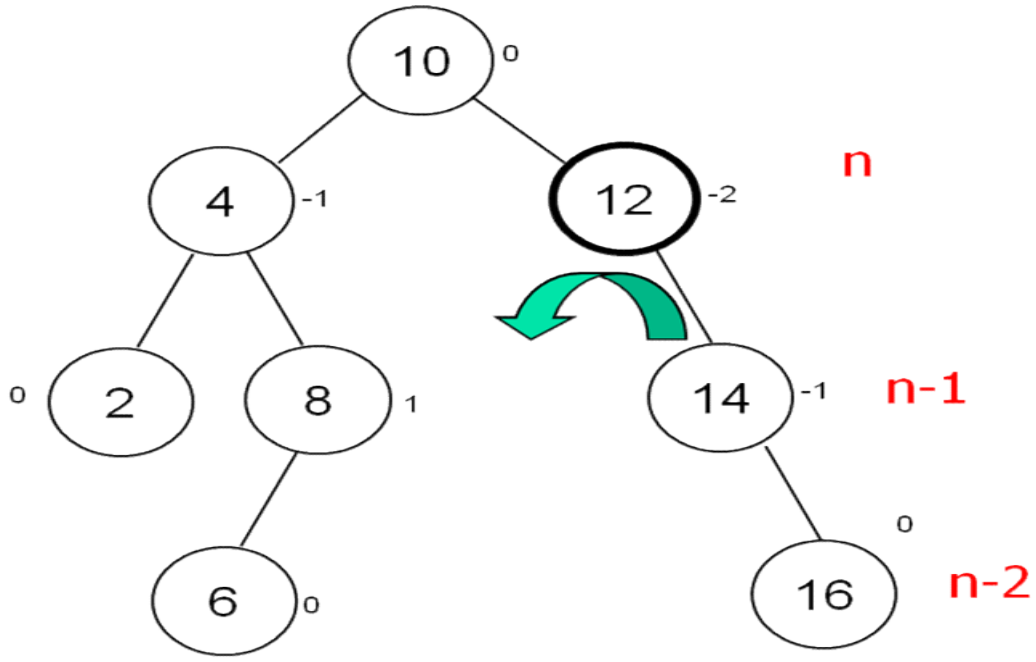


1ère rotation

Arbres

- AVL – exemple détaillé

2 10 12 4 16 8 6 14

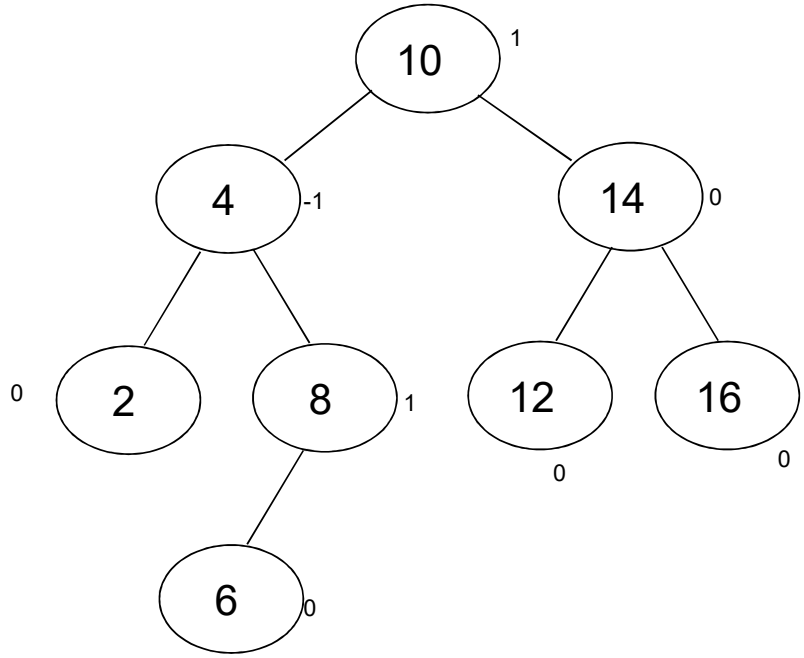


2ème rotation

Arbres

- AVL – exemple détaillé

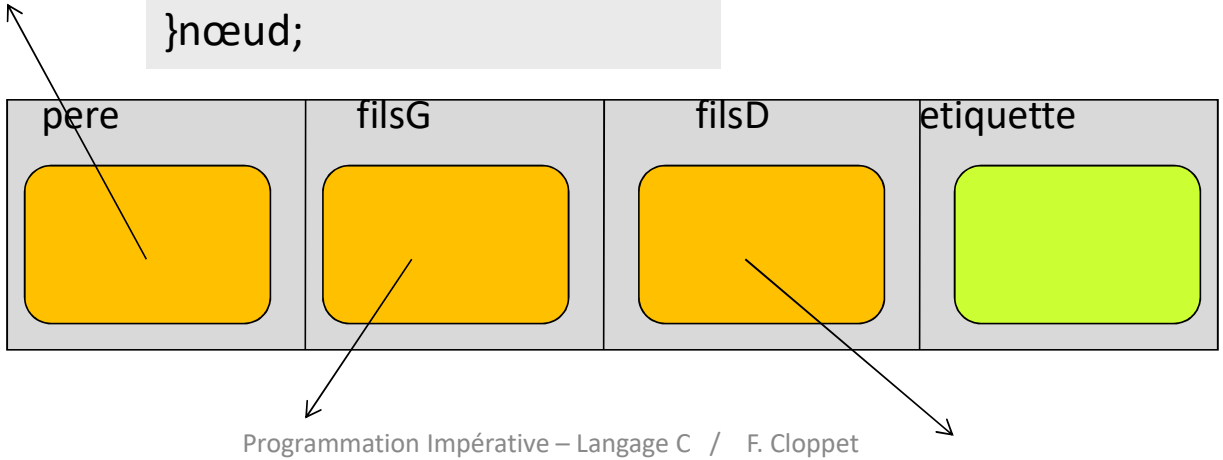
2 10 12 4 16 8 6 14



Implémentation Arbres

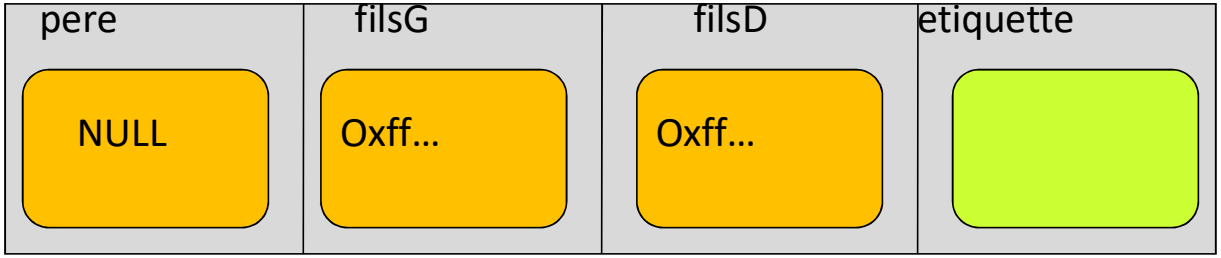
- Structures de données pour implémenter un arbre
  - Nœud
    - type ⇔ type prédéfini ou utilisateur

```
typedef struct noeud{
        type etiquette;
        struct nœud *pere,
                *filsG,
                *filsD;
}nœud;
```

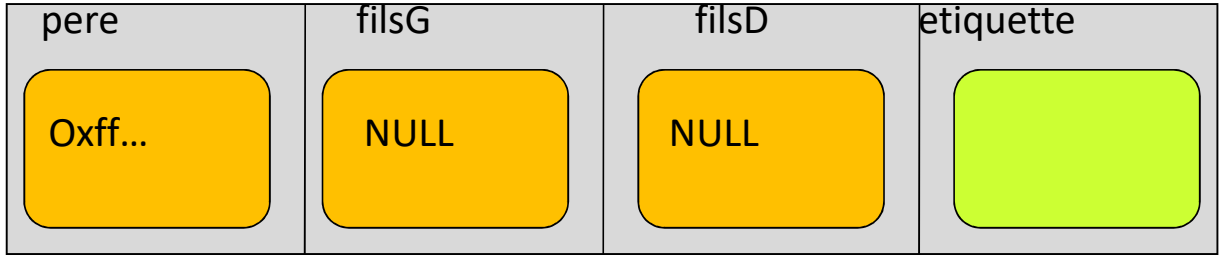


Implémentation Arbres

- Nœud Racine

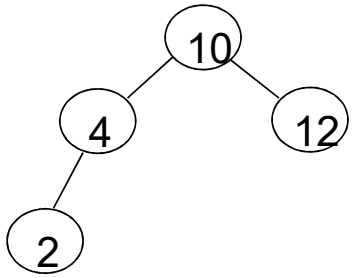


- Feuille



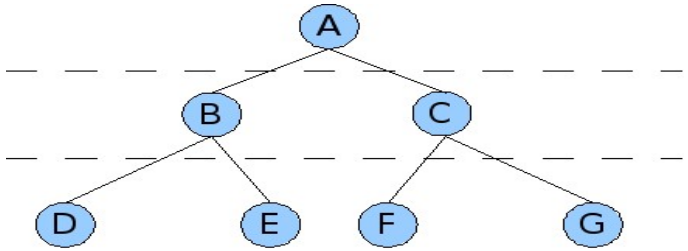
Implémentation Arbres

- Arbre construit – représentation mémoire



Implémentation Arbres

- Parcours
  - Parcours en largeur
    - Parcours par niveau et nœud par nœud
      - On utilise une file

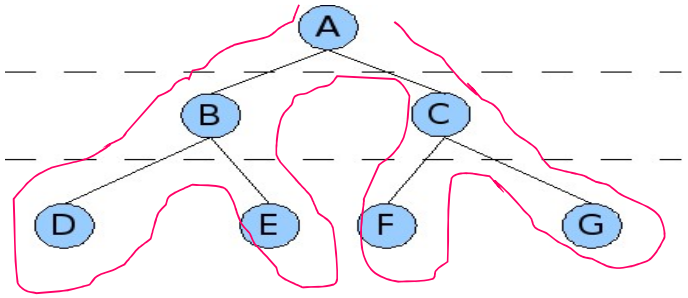


A B C D E F G

* Marquer(*n*)
marque un nœud n comme exploré, de manière à ne pas le considérer plusieurs fois.

```
ParcoursLargeur(noeud *arbre) {
   noeud *nCourant= arbre, *n, *fils;
   file *f = CreerFile();
   Marquer(nCourant);
   AjoutFile(f, nCourant);
   while !FileVide(f){
      n = SupprimerFile(f);
      Afficher(n);
      while (ExiteFils(n)){
         fils=filsSuivant(n);
          if(!estMarque(fils)){
                Marquer(fils);
                AjoutFile(f,fils);
         }
      }
   }
}
```

Implémentation Arbres



- Parcours
  - Parcours en profondeur
    - Préfixe
      - Racine traitée avant l'appel récursif de parcours des sous arbres

        A B D E C F G
    - Infixe
      - Racine traitée entre l'appel récursif de parcours des sous arbres

        D B E A F C G
    - Suffixe
      - Racine traitée après l'appel récursif de parcours des sous arbres

        D E B F G C A

Implémentation Arbres

- Parcours en profondeur
  - Préfixe
  - Infixe
  - Suffixe

```
ParcoursProfPrefixe(noeud *n) {
   if(n ==NULL)
      return;
   Afficher(n->etiquette);
   ParcoursProfPrefixe(n->filsG);
   ParcoursProfPrefixe(n->filsD);
}
```

```
ParcoursProfInfixe(noeud *n) {
   if(n ==NULL)
      return;
   ParcoursProfInfixe(n->filsG);
   Afficher(n->etiquette);
   ParcoursProfInfixe(n->filsD);
}
```

```
ParcoursProfSuffixe(noeud *n) {
   if(n ==NULL)
      return;
   ParcoursProfSuffixe(n->filsG);
   ParcoursProfSuffixe(n->filsD);
   Afficher(n->etiquette);
}
```

Implémentation Arbres

- Parcours en profondeur infixe
  - Version itérative simple
    - Utilisation d'une pile

```
void parcoursProfInfixe(noeud *n) {
   noeud *nCourant;
   if (n == null)
      return;
   Pile p = CreerPile();
   ajouterPile(p,n);
   while (!pileVide()) {
      nCourant = depilerPile();
      Afficher(nCourant->etiquette);
      if (nCourant->filsG != null)
         ajouterPile(p, nCourant->filsG);
      if (nCourant->filsD != null)
         ajouterPile(p, nCourant->filsD);
   }
}
```