Structures de données et algorithmes – TP9

Iulia-Cristina Stanica

Objectifs pour aujourd'hui
Arbres

- Arbres binaires
  - Implémentation
  - Parcours
  - Exercices

Exercice 1

- Compte tenu d'un arbre binaire, calculer son **"maximumHeight"** (hauteur max) - le nombre de nœuds contenus dans le plus long chemin (qui lie la racine et la feuille la plus éloignée)

- Suggestion: l'hauteur maximale d'un arbre est le maximum entre les hauteurs de ses enfants.

- **Obs:** pour tous les exercices, vous devez dessiner l'arbre de l'exemple, pour vérifier vos solutions.

Exercice 2

- En utilisant la fonction de l'exercice précédant, écrivez une fonction qui retourne, pour le nœud racine, la différence entre l'hauteur du sous-arbre gauche et l'hauteur du sous-arbre droit.

Exercice 3

- Écrivez une fonction qui affiche les valeurs des nœuds situés à un niveau donné (envoyé en tant que paramètre).

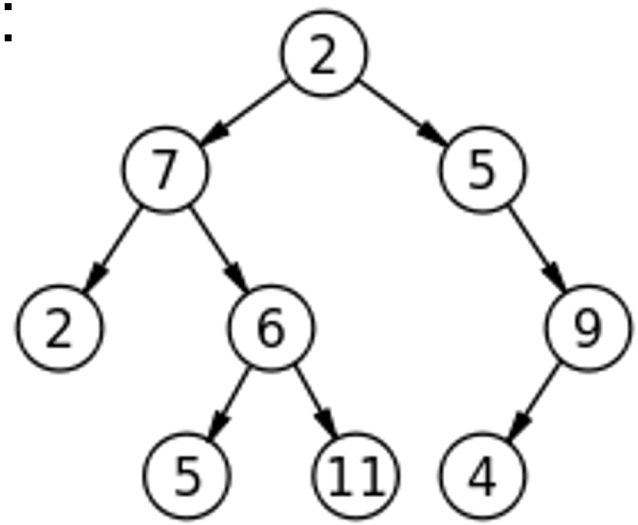
- Faites une fonction récursive dans laquelle vous décrémentez le niveau par 1 quand vous passez au niveau suivant. Lorsque le niveau est égal à 0, vous avez atteint le niveau donné.

- Pseudocode:
- Display (T, level):
  - if(level==0) then print(T.data)
  - else Display (T.left, level-1); Display (T.right, level-1);

Exercice 4

- Utiliser les fonctions antérieures pour implementer une nouvelle fonction qui affiche l'arbre entier sur les niveaux.

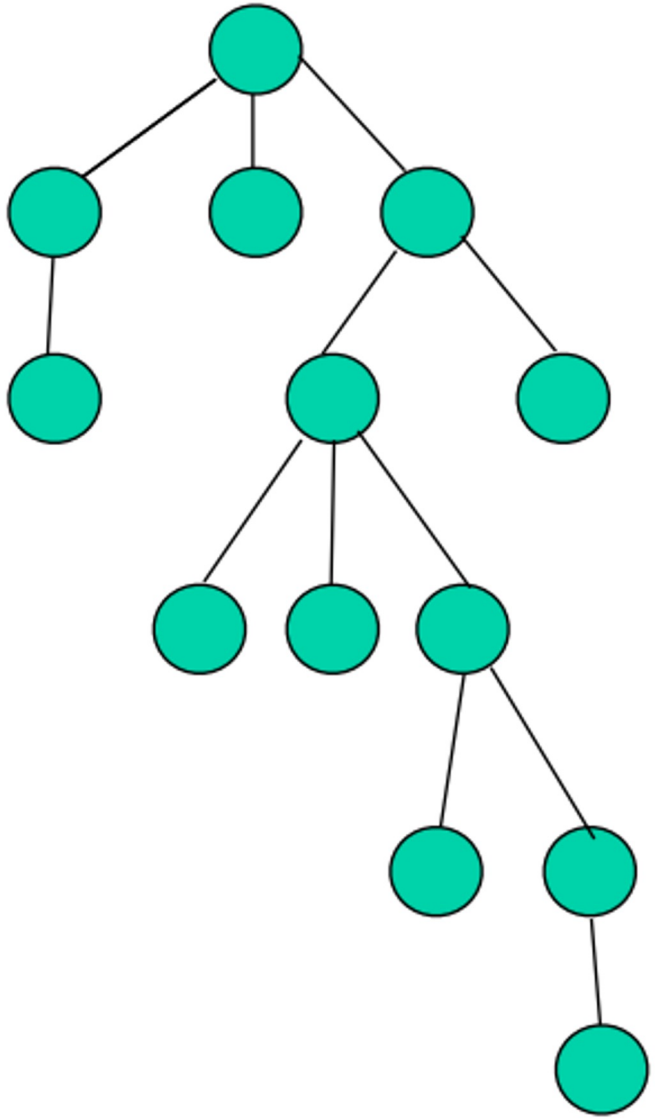
- Ex: pour l'arbre suivant on a:
  - Niveau 0: 2
  - Niveau 1: 7, 5
  - Niveau 2: 2, 6, 9
  - Niveau 3: 5, 11, 4



Theorie

Arbres

- **Def:** Un graphe simple connexe tel que, quels que soient les sommets distincts i et j, il existe une seule chaîne entre i à j.

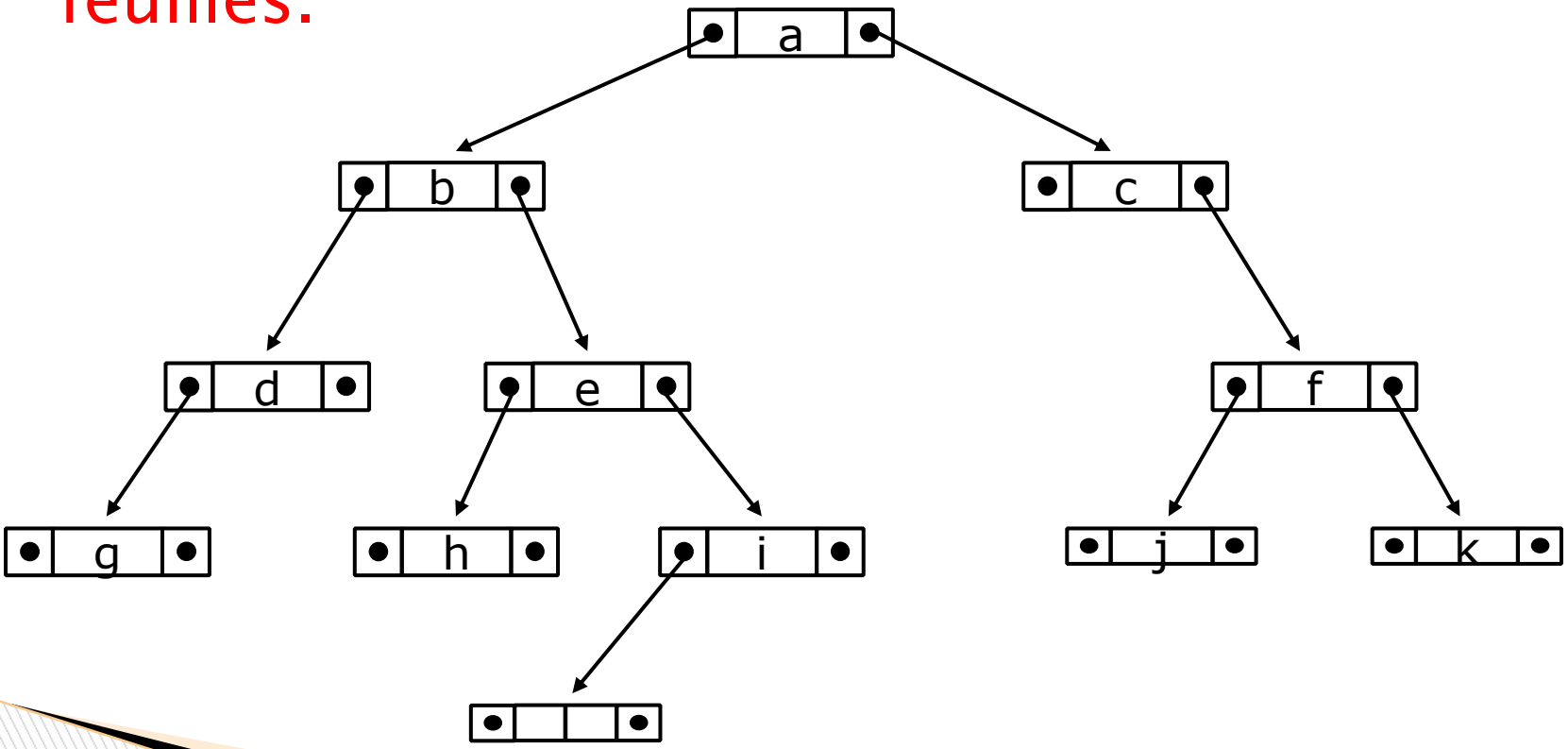


Arbres binaires

- **Def:** Un arbre binaire est composé de zéro ou plusieurs nœuds, mais chaque noued a au **maximum deux descendants**: un gauche et un droit.
- Chaque noeud contient:
  - Une valeur (donnée d'un certain type)
  - Une référence ou un pointeur vers un enfant gauche (peut être NULL)
  - Une référence ou un pointeur vers un enfant droit (peut être NULL)

Arbres binaires (suite)

- S'il n'est pas vide, l'arbre binaire a un noeud racine.
- Les nœuds qui n'ont pas d'enfants s'appellent feuilles.



Representation

```
template <typename T> class BinaryTree
 {
    public:
     BinaryTree();
     ~BinaryTree();
    private:
     BinaryTree<T> *leftNode;
     BinaryTree<T> *rightNode;
     T *pData;
};
```

Pour les membres d'un noeud, vous devez allouer dynamiquement la memoire, mais pas dans le constructeur! Faitez cela où vous en avez besoin.

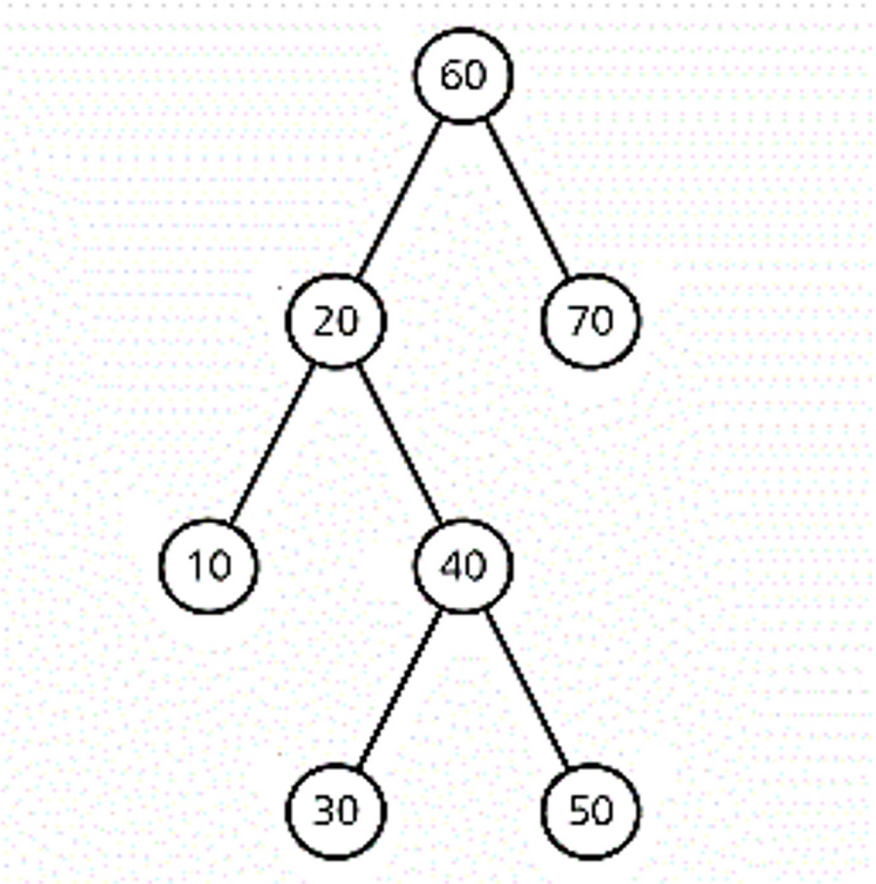
```
BinaryTree <T> *node = new BinaryTree <T>( );
delete node;
T *pData = new T;
delete pData;
```

Parcours

- On définit un arbre binaire d'une manière recursive: il contient la racine, un sous-arbre droit et un sous-arbre gauche
- Pour parcourir un arbre, on doit visiter chaque noeud une fois seule

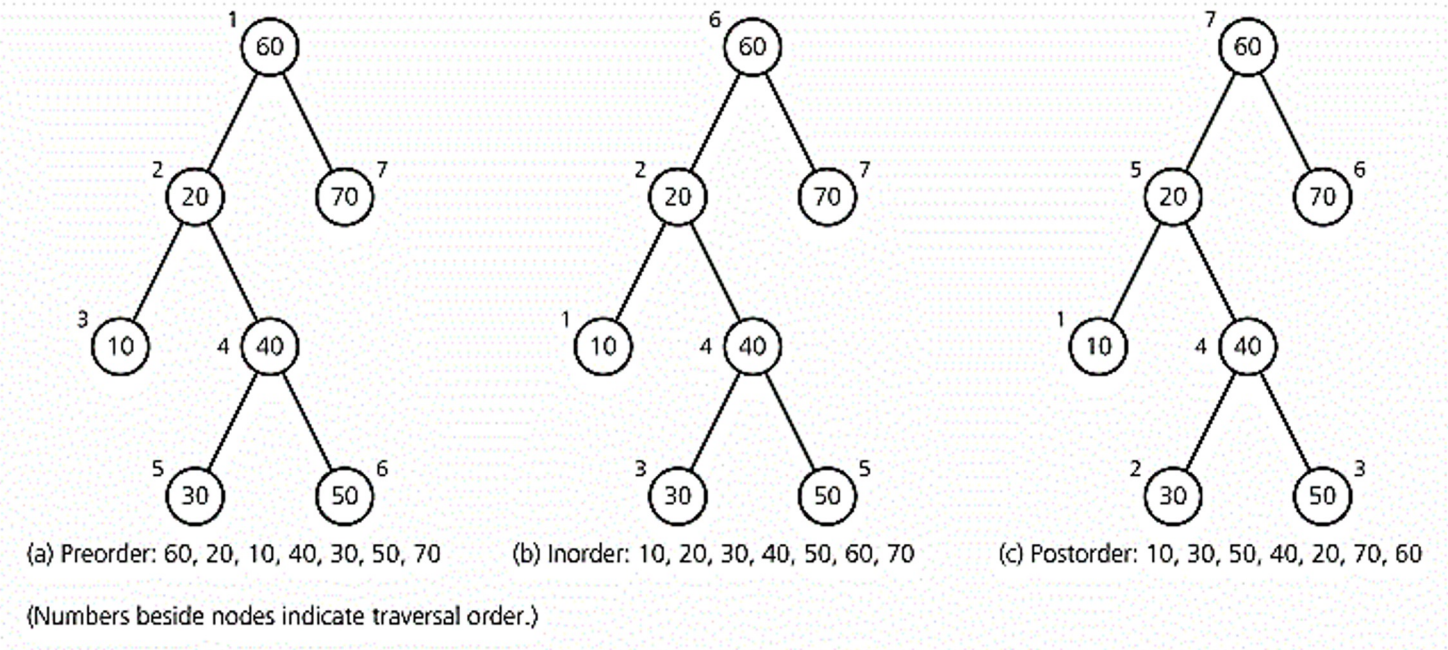
- 3 méthodes de parcours:
  - Pre-ordre (RGD: racine-gauche-droite)
  - En-ordre (GRD)
  - Post-ordre (GDR)

Parcours



Idée: appliquer l'algorithme de parcours au sous-arbre correspondant

Parcours



RGD

GRD

GDR