Trees and their implementation in C++

**Definition 1:** A *tree* $t$ is a finite nonempty set of elements. One of these elements is called the root, and the remaining elements (if any) are partitioned into trees which are called the *subtrees* of $t$.

- By definition, the tree root is at level 1; its children (if any) are at level 2; their children (if any) are at level 3, and so on.
- The *depth* (height) of a tree is then defined in terms of the levels of its nodes.
- The *degree of an element* is the number of children it has.
- The *degree of a tree* is the maximum number of its element degrees.

**Binary Trees**

**Definition 2:** A *binary tree* $t$ is a finite (possibly empty) collection of elements. When the binary tree is not empty, it has a root element and the remaining elements (if any) are partitioned into two binary trees, which are called the left and right subtrees of $t$.

- A binary tree can be empty, whereas a tree cannot.
- Each element in a binary tree has exactly two subtrees (one or both of these subtrees may be empty). Each element in a tree can have any number of subtrees.
The subtrees of each element in a binary tree are ordered. That is, we distinguish between the left and the right subtrees. The subtrees in a tree are unordered.

**Definition 3:** A binary tree of height $h$ that contains exactly $2^h - 1$ elements is called a **full binary tree** (Fig. 5).

**Definition 4:** Suppose we number the elements in a full binary tree of height $h$ using the numbers 1 through $2^h - 1$. If we delete the $k$, $k \geq 0$, elements numbered $2^h - i$, $1 \leq i \leq k$ for any $k$. The resulting binary tree is called a **complete binary tree** (Fig. 6).

**Linked Representation of Binary Trees**

![Linked representation of binary tree](image_url)

Fig. 9. Linked representation of binary tree

/Trees and Their Implementation in C++
Common Binary Tree Operations

- Determine its height.
- Determine the number of elements in it.
- Make a copy.
- Display the binary tree on a screen or on paper.
- Determine whether two binary trees are identical.
- Delete the tree.
- If it is an expression tree, evaluate the expression.
- If it is an expression tree, obtain the parenthesized form of the expression.
Binary Tree Traversal

There are four common ways to traverse a binary tree:

- Preorder
- Inorder
- Postorder
- Level order

**Preorder (depth-first order) traversal method**

1. Visit the root.
2. Traverse the left subtree in preorder.
3. Traverse the right subtree in preorder.

**Inorder (symmetric order) traversal method**

1. Traverse the left subtree in inorder.
2. Visit the root.
3. Traverse the right subtree in inorder.

**Postorder traversal method**

1. Traverse the left subtree in postorder.
2. Traverse the right subtree in postorder.
3. Visit the root.
Preorder:  A B D G C E H F I
Inorder:  D G B A H E C F I
Postorder: G D B H E I F C A

Fig. 12. Binary tree and its traversal

Preorder:  A B C E I F J D G H K L
Inorder:  E I C F J B G D K H L A
Postorder: I E J F C G K L H D B A

Fig. 13. Binary tree and its traversal