Traversals is a process to visit all the nodes of a tree and may print their values too. Because, all nodes are connected via edges links we always start from the root head node. That is, we cannot random access a node in tree. There are three ways which we use to traverse a tree –

- In-order Traversal
- Pre-order Traversal
- Post-order Traversal

Generally we traverse a tree to search or locate given item or key in the tree or to print all the values it contains.

**Inorder Traversal**

In this traversal method, the left left-subtree is visited first, then root and then the right sub-tree.
We should always remember that every node may represent a subtree itself.

If a binary tree is traversed **in-order**, the output will produce sorted key values in ascending order.

We start from A, and following in-order traversal, we move to its left subtree B. B is also traversed in-ordered. And the process goes on until all the nodes are visited. The output of in-order traversal of this tree will be –

\[ D \rightarrow B \rightarrow E \rightarrow A \rightarrow F \rightarrow C \rightarrow G \]

**Algorithm**

Until all nodes are traversed –
Step 1 – Recursively traverse left subtree.
Step 2 – Visit root node.
Step 3 – Recursively traverse right subtree.

**Preorder Traversal**

In this traversal method, the root node is visited first, then left subtree and finally right sub-tree.
We start from A, and following pre-order traversal, we first visit A itself and then move to its left subtree B. B is also traversed pre-ordered. And the process goes on until all the nodes are visited. The output of pre-order traversal of this tree will be —

\[ A \rightarrow B \rightarrow D \rightarrow E \rightarrow C \rightarrow F \rightarrow G \]

**Algorithm**

Until all nodes are traversed —

**Step 1** — Visit root node.
**Step 2** — Recursively traverse left subtree.
**Step 3** — Recursively traverse right subtree.

**Postorder Traversal**

In this traversal method, the root node is visited last, hence the name. First we traverse left subtree, then right subtree and finally root.

We start from A, and following pre-order traversal, we first visit left subtree B. B is also traversed post-ordered. And the process goes on until all the nodes are visited. The output of post-order traversal of this tree will be —

\[ D \rightarrow E \rightarrow B \rightarrow F \rightarrow G \rightarrow C \rightarrow A \]

**Algorithm**

Until all nodes are traversed —

**Step 1** — Recursively traverse left subtree.
**Step 2** — Recursively traverse right subtree.
Step 3 - Visit root node.

To check the C implementation of tree traversing, please [click here](#).