MIT World Peace University

Advanced Data Structures
Second Year B. Tech, Semester 4

Creation and Traversal of Binary Trees (Recursive and Non-Recursive)

Assignment No. 2

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

February 5, 2023

Contents

1 Objectives

1. To study data structure : Tree and Binary Tree
2. To study different traversals in Binary Tree
3. To study recursive and non-recursive approach of programming

2 Problem Statement

Implement binary tree using C++ and perform following operations: Creation of binary tree and traversal (recursive and non- recursive)

3 Theory

3.1 Tree

1. A tree is a non-linear data structure.
2. A tree is a collection of nodes connected by edges.
3. A tree is a hierarchical data structure.
4. A tree is a data structure that simulates a hierarchical tree structure, with a root value and subtrees of children with a parent node, represented as a set of linked nodes.
5. A tree data structure can be defined recursively (locally) as a collection of nodes (starting at a root node), where each node is a data structure consisting of a value, together with a list of references to nodes (the "children"), with the constraints that no reference is duplicated, and none points to the root.

3.2 Binary Tree

1. A binary tree is a tree data structure in which each node has at most two children, which are referred to as the left child and the right child.
2. A binary tree is a data structure for storing data such as numbers in an organized way.
3. Binary trees can be used to implement several types of abstract data types, such as sets, multisets, and associative arrays.
4. Binary trees are a useful data structure for implementing many common abstract data types, such as priority queues, binary search trees, and heaps.
5. Binary trees can be used to represent expressions involving binary operations.

3.2.1 Binary Tree Properties

1. The maximum number of nodes at level 'l' of a binary tree is 2l.
2. Maximum number of nodes in a binary tree of height 'h' is 2h - 1.

3. In a Binary Tree with N nodes, minimum possible height or minimum number of levels is

$$log_2(N+1)$$

4. A Binary Tree with L leaves has at least

$$\lceil log_2(L+1) \rceil$$

levels

3.2.2 Binary Tree Representation

1. A Binary Tree node contains following parts.
   (a) Data
   (b) Pointer to left child
   (c) Pointer to right child

3.2.3 Types of Binary Tree

1. Full Binary Tree
   (a) A Binary Tree is full if every node has 0 or 2 children.
   (b) Number of leaf nodes is always one more than nodes with two children.
2. Complete Binary Tree
   (a) A Binary Tree is complete Binary Tree if all levels are completely filled except possibly the last level and the last level has all keys as left as possible.
   (b) This property of Binary Tree makes them suitable to be stored in an array.
3. Perfect Binary Tree
   (a) A Binary tree is Perfect Binary Tree in which all internal nodes have two children and all leaves are at same level.

3.2.4 Different definitions related to binary tree.

1. *Height* of a node is the number of edges on the longest path from the node to a leaf.
2. *Height* of a tree is the height of its root node.
3. *Depth* of a node is the number of edges from the node to the tree's root node.
4. *Level* of a node is defined as 1 + (the number of connections between the node and the root).
5. *Degree* of a node is the number of sub trees of a node.
6. *Degree* of a tree is the maximum degree of the nodes in the tree.
7. *Leaf* is a node with no children.
8. *Internal* node is a node with at least one child.

9. *Sibling* is a group of nodes with the same parent.
10. *Ancestor* is a node reachable by repeated proceeding from parent to parent.
11. *Descendant* is a node reachable by repeated proceeding to a child.
12. *Subtree* is a set of nodes and edges comprised of a parent and all the descendants of that parent.
13. *Forest* is a set of n Greater than or Equal to 0 disjoint trees.

3.3 Different Traversals (Inorder, Preorder and Postorder)

3.3.1 Inorder Traversal

Inorder Traversal is a recursive algorithm for traversing or searching tree data structures. It starts at the tree's root node (or another designated node of a sub-tree), and explores the sub-tree's left branch until it reaches the left-most node (i.e., a node without a left child), which it then visits. It then explores the right branch in the same manner, i.e., it recursively visits the left-most node in that sub-tree, and so on, backtracking to the root node once all nodes have been visited.

3.3.2 Algorithm for Inorder Recursive Travesal

1. Traverse the left subtree, i.e., call Inorder(left-subtree)
2. Visit the root.
3. Traverse the right subtree, i.e., call Inorder(right-subtree)

3.3.3 Algorithm for Inorder Iterative Travesal

1. Create an empty stack S.
2. Initialize current node as root
3. Push the current node to S and set current = current->left until current is NULL
4. If current is NULL and stack is not empty then
   (a) Pop the top item from stack.
   (b) Print the popped item, set current = poppedItem->right
   (c) Go to step 3.
5. If current is NULL and stack is empty then we are done.

In the case of binary search trees (BST), Inorder traversal gives nodes in non-decreasing order. To get nodes of BST in non-increasing order, a variation of Inorder traversal where Inorder traversal is reversed can be used.

3.3.4 Preorder Traversal

Preorder traversal is a recursive algorithm for traversing or searching tree data structures. It starts at the tree's root node (or another designated node of a sub-tree), visits the left subtree, then the right subtree. Printing the value of the root node after visiting the left and right subtrees. Preorder traversal is a depth-first traversal.

Preorder traversal is used to create a copy of the tree. Preorder traversal is also used to get prefix expressions on an expression tree.

3.3.5 Algorithm for PreOrder Recursive Travesal

1. Visit the root.
2. Traverse the left subtree, i.e., call Preorder(left-subtree)
3. Traverse the right subtree, i.e., call Preorder(right-subtree)

3.3.6 Algorithm for PreOrder Iterative Travesal

1. Create an empty stack S.
2. Initialize current node as root
3. Push the root node to the stack.
4. Pop the top item from stack and set current node = poppedItem.
5. Print the value of the node.
6. Push the right node if its not null, and then push the left node if its not null.
7. If stack isnt empty, go to step 4.
8. Exit algorithm when stack is empty.

3.3.7 Postorder Traversal

3.3.8 Algorithm for PostOrder Recursive Travesal

1. Traverse the left subtree, i.e., call Postorder(left-subtree)
2. Traverse the right subtree, i.e., call Postorder(right-subtree)
3. Visit the root.

3.3.9 Algorithm for PostOrder Iterative Travesal

1. Create empty stacks S1 and S2.
2. Add the root node to S1.
3. Pop S1, and push the popped node to S2.
4. Push the left and right node of the popped node to S1.
5. Repeat Step 3 and 4 until S1 is empty.
6. Iterate through S2 and print the value of the node as it is popped.

4 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers : g++ and gcc on linux for C++

5 Input

1. The Nodes of the Binary Tree

6 Output

1. The traversal of the binary tree in different ways.

7 Test Conditions

1. Input at least 10 nodes.
2. Display all traversals of binary tree with 10 nodes.(recursive and nonrecursive)

8 Pseudo Code

8.1 Inorder Traversal - Iterative Approach

```
    void inorder_iterative(Node){
        if(Node == NULL){
            return;
        }
        stack<Node> s;
        Node current = Node;
        while(current != NULL || s.empty() == false){
            while(current != NULL){
                s.push(current);
                current = current->left;
            }
            current = s.top();
            s.pop();
            cout << current->data << " ";
            current = current->right;
        }
    }
```

8.2 Preorder Traversal - Iterative Approach

```
    void preorder_iterative(Node){
        if(Node == NULL){
            return;
        }
        stack<Node> s;
        s.push(Node);
```

```
        while(s.empty() == false){
            Node current = s.top();
            cout << current->data << " ";
            s.pop();
            if(current->right){
                s.push(current->right);
            }
            if(current->left){
                s.push(current->left);
            }
        }
    }
```

8.3 Postorder Traversal - Iterative Approach

```
    void postorder_iterative(Node){
        if(Node == NULL){
            return;
        }
        stack<Node> s1, s2;
        s1.push(Node);
        while(s1.empty() == false){
            Node current = s1.top();
            s1.pop();
            s2.push(current);
            if(current->left){
                s1.push(current->left);
            }
            if(current->right){
                s1.push(current->right);
            }
        }
        while(s2.empty() == false){
            Node current = s2.top();
            cout << current->data << " ";
            s2.pop();
        }
    }
```

9 Time Complexity

Time Complexities of different Traversals in their Iterative Approaches are as follows:

- Inorder Traversal: O(n)
- Preorder Traversal: O(n)
- Postorder Traversal: O(n)

There is only a single loop involved in all of these traversals, so the time complexity is linear.

10 Code

10.1 Program

```
#include <iostream>
#include <stack>

using namespace std;

class TreeNode
{
    char data[10];
    TreeNode *left;
    TreeNode *right;
    friend class BinaryTree;
};

class BinaryTree
{
public:
    TreeNode *root;
    BinaryTree()
    {
        root = NULL;
    }
    void create_root()
    {
        root = new TreeNode;
        cout << "Enter the data: " << endl;
        cin >> root->data;
        root->left = NULL;
        root->right = NULL;
        create_recursive(root);
    }
    void create_recursive(TreeNode *Node)
    {
        int choice = 0;
        TreeNode *new_node;
        cout << "Enter if you want to enter a left node (1/0): "
             << "for the node -- " << Node->data << "-- ";
        cin >> choice;
        if (choice == 1)
        {
            new_node = new TreeNode;
            cout << "Enter the data: ";
            cin >> new_node->data;
            Node->left = new_node;
            create_recursive(new_node);
        }
        cout << "Enter if you want to enter a right node (1/0): "
             << "for the node -- " << Node->data << "-- ";
        cin >> choice;
        if (choice == 1)
        {
            new_node = new TreeNode;
            cout << "Enter the data: ";
            cin >> new_node->data;
            Node->right = new_node;
            create_recursive(new_node);
        }
    }

    void inorder_recursive(TreeNode *temp)
```

```
60      {
61          if (temp == NULL)
62          {
63              return;
64          }
65          inorder_recursive(temp->left);
66          cout << temp->data << " ";
67          inorder_recursive(temp->right);
68      }
69      void inorder_iterative(TreeNode *temp)
70      {
71          if (!temp)
72          {
73              return;
74          }
75
76          stack<TreeNode *> s;
77          TreeNode *current = temp;
78
79          while (current != NULL || s.empty() == false)
80          {
81              while (current != NULL)
82              {
83                  s.push(current);
84                  current = current->left;
85              }
86              current = s.top();
87              s.pop();
88              cout << current->data << " ";
89              current = current->right;
90          }
91      }
92
93      void preorder_recursive(TreeNode *temp)
94      {
95          if (temp == NULL)
96          {
97              return;
98          }
99          cout << temp->data << " ";
100         preorder_recursive(temp->left);
101         preorder_recursive(temp->right);
102     }
103     void preorder_iterative(TreeNode *temp)
104     {
105         if (!temp)
106         {
107             return;
108         }
109
110         stack<TreeNode *> s;
111         s.push(temp);
112
113         while (s.empty() == false)
114         {
115             TreeNode *current = s.top();
116             cout << current->data << " ";
117             s.pop();
118
```

```
            if (current->right)
            {
                s.push(current->right);
            }
            if (current->left)
            {
                s.push(current->left);
            }
        }
    }

    void postorder_recursive(TreeNode *temp)
    {
        if (temp == NULL)
        {
            return;
        }
        postorder_recursive(temp->left);
        postorder_recursive(temp->right);
        cout << temp->data << " ";
    }
    void postorder_iterative(TreeNode *temp)
    {
        if (!temp)
        {
            return;
        }

        stack<TreeNode *> s1;
        stack<TreeNode *> s2;

        s1.push(temp);

        while (s1.empty() == false)
        {
            TreeNode *current = s1.top();
            s1.pop();
            s2.push(current);

            if (current->left)
            {
                s1.push(current->left);
            }
            if (current->right)
            {
                s1.push(current->right);
            }
        }
        while (s2.empty() == false)
        {
            TreeNode *current = s2.top();
            cout << current->data << " ";
            s2.pop();
        }
    }
};

int main()
{
```

```
178     int choice = 0;
179     BinaryTree main_tree;
180 
181     while (choice != 8)
182     {
183         cout << "\nWhat would like to do? " << endl;
184         cout << "\n\nWelcome to ADS Assignment 2 - Binary Tree Traversals\n\nWhat
    would you like to do? " << endl;
185         cout << "1. Create a Binary Tree"
186              << endl;
187         cout << "2. Traverse the Tree Inorder Recursively"
188              << endl;
189         cout << "3. Traverse the Tree Inorder Iteratively"
190              << endl;
191         cout << "4. Traverse the Tree PreOrder Recursively"
192              << endl;
193         cout << "5. Traverse the Tree PreOrder Iteratively"
194              << endl;
195         cout << "6. Traverse the Tree PostOrder Recursively"
196              << endl;
197         cout << "7. Traverse the Tree PostOrder Iteratively"
198              << endl;
199         cout << "8. Exit" << endl
200              << endl;
201 
202         cin >> choice;
203         switch (choice)
204         {
205         case 1:
206             main_tree.create_root();
207             break;
208         case 2:
209             cout << "Traversing through the binary tree inorder recursively: " <<
    endl;
210             main_tree.inorder_recursive(main_tree.root);
211             break;
212         case 3:
213             cout << "Traversing through the Binary Tree Inorder Iteratively: " <<
    endl;
214             main_tree.inorder_iterative(main_tree.root);
215             break;
216         case 4:
217             cout << "Traversing through the Binary Tree PreOrder Recursively: " <<
     endl;
218             main_tree.preorder_recursive(main_tree.root);
219             break;
220         case 5:
221             cout << "Traversing through the Binary Tree PreOrder Iteratively: " <<
     endl;
222             main_tree.preorder_iterative(main_tree.root);
223             break;
224         case 6:
225             cout << "Traversing through the Binary Tree PostOrder Recursively: "
    << endl;
226             main_tree.postorder_recursive(main_tree.root);
227             break;
228         case 7:
229             cout << "Traversing through the Binary Tree PostOrder Iteratively: "
    << endl;
```

```
            main_tree.postorder_iterative(main_tree.root);
            break;
        case 8:
            cout << "Exiting the program" << endl;
            exit(0);
            break;
        default:
            cout << "Invalid Choice" << endl;
            break;
        }
    }
}
```

10.2 Input and Output

```
What would like to do?


Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

1
Enter the data:  1
Enter if you want to enter a left node (1/0): for the node -- 1-- 1
Enter the data:  2
Enter if you want to enter a left node (1/0): for the node -- 2-- 1
Enter the data:  3
Enter if you want to enter a left node (1/0): for the node -- 3-- 1
Enter the data:  4
Enter if you want to enter a left node (1/0): for the node -- 4-- 1
Enter the data:  5
Enter if you want to enter a left node (1/0): for the node -- 5-- 0
Enter if you want to enter a right node (1/0): for the node -- 5-- 0
Enter if you want to enter a right node (1/0): for the node -- 4-- 0
Enter if you want to enter a right node (1/0): for the node -- 3-- 0
Enter if you want to enter a right node (1/0): for the node -- 2-- 0
Enter if you want to enter a right node (1/0): for the node -- 1-- 0

What would like to do?


Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
```

```
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

5
Traversing through the binary tree inorder recursively:
5 4 3 2 1
What would like to do?


Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

7
Traversing through the Binary Tree PreOrder Recursively:
1 2 3 4 5
What would like to do?


Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

9
Traversing through the Binary Tree PostOrder Recursively:
5 4 3 2 1

What would like to do?


Welcome to ADS Assignment 2 - Binary Tree Traversals
```

```
104
105 What would you like to do?
106 1. Create a Binary Tree
107 2. Insert Elements to the Tree in Auto Ascending Order
108 3. Insert Elements to the Tree Level by Level
109 4. Insert Elements into the Tree Manually
110 5. Traverse the Tree Inorder Recursively
111 6. Traverse the Tree Inorder Iteratively
112 7. Traverse the Tree PreOrder Recursively
113 8. Traverse the Tree PreOrder Iteratively
114 9. Traverse the Tree PostOrder Recursively
115 10. Traverse the Tree PostOrder Iteratively
116 11. Exit
117
118 1
119 Enter the data:  1
120 Enter if you want to enter a left node (1/0): for the node -- 1-- 0
121 Enter if you want to enter a right node (1/0): for the node -- 1-- 1
122 Enter the data:  2
123 Enter if you want to enter a left node (1/0): for the node -- 2-- 0
124 Enter if you want to enter a right node (1/0): for the node -- 2-- 1
125 Enter the data:  3
126 Enter if you want to enter a left node (1/0): for the node -- 3-- 0
127 Enter if you want to enter a right node (1/0): for the node -- 3-- 1
128 Enter the data:  4
129 Enter if you want to enter a left node (1/0): for the node -- 4-- 0
130 Enter if you want to enter a right node (1/0): for the node -- 4-- 1
131 Enter the data:  5
132 Enter if you want to enter a left node (1/0): for the node -- 5-- 0
133 Enter if you want to enter a right node (1/0): for the node -- 5-- 0
134
135 What would like to do?
136
137
138 Welcome to ADS Assignment 2 - Binary Tree Traversals
139
140 What would you like to do?
141 1. Create a Binary Tree
142 2. Insert Elements to the Tree in Auto Ascending Order
143 3. Insert Elements to the Tree Level by Level
144 4. Insert Elements into the Tree Manually
145 5. Traverse the Tree Inorder Recursively
146 6. Traverse the Tree Inorder Iteratively
147 7. Traverse the Tree PreOrder Recursively
148 8. Traverse the Tree PreOrder Iteratively
149 9. Traverse the Tree PostOrder Recursively
150 10. Traverse the Tree PostOrder Iteratively
151 11. Exit
152
153 6
154 Traversing through the Binary Tree Inorder Iteratively:
155 1 2 3 4 5
156 What would like to do?
157
158
159 Welcome to ADS Assignment 2 - Binary Tree Traversals
160
161 What would you like to do?
162 1. Create a Binary Tree
```

```
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

8
Traversing through the Binary Tree PreOrder Iteratively:
1 2 3 4 5
What would like to do?


Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

10
Traversing through the Binary Tree PostOrder Iteratively:
5 4 3 2 1


Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

1
Enter the data:  1
Enter if you want to enter a left node (1/0): for the node -- 1-- 1
Enter the data:  2
Enter if you want to enter a left node (1/0): for the node -- 2-- 1
Enter the data:  4
Enter if you want to enter a left node (1/0): for the node -- 4-- 0
```

```
Enter if you want to enter a right node (1/0): for the node -- 4-- 0
Enter if you want to enter a right node (1/0): for the node -- 2-- 1
Enter the data:  5
Enter if you want to enter a left node (1/0): for the node -- 5-- 0
Enter if you want to enter a right node (1/0): for the node -- 5-- 0
Enter if you want to enter a right node (1/0): for the node -- 1-- 1
Enter the data:  3
Enter if you want to enter a left node (1/0): for the node -- 3-- 1
Enter the data:  6
Enter if you want to enter a left node (1/0): for the node -- 6-- 0
Enter if you want to enter a right node (1/0): for the node -- 6-- 0
Enter if you want to enter a right node (1/0): for the node -- 3-- 1
Enter the data:  7
Enter if you want to enter a left node (1/0): for the node -- 7-- 0
Enter if you want to enter a right node (1/0): for the node -- 7-- 0


What would like to do?


Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

5
Traversing through the binary tree inorder recursively:
4 2 5 1 6 3 7
What would like to do?


Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

7
Traversing through the Binary Tree PreOrder Recursively:
1 2 4 5 3 6 7
```

```
What would like to do?


Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

9
Traversing through the Binary Tree PostOrder Recursively:
4 5 2 6 7 3 1


What would like to do?
Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

1
Enter the data:
1
Enter if you want to enter a left node (1/0): for the node -- 1-- 1
Enter the data: 2
Enter if you want to enter a left node (1/0): for the node -- 2-- 1
Enter the data: 4
Enter if you want to enter a left node (1/0): for the node -- 4-- 1
Enter the data: 8
Enter if you want to enter a left node (1/0): for the node -- 8-- 0
Enter if you want to enter a right node (1/0): for the node -- 8-- 0
Enter if you want to enter a right node (1/0): for the node -- 4-- 1
Enter the data: 9
Enter if you want to enter a left node (1/0): for the node -- 9-- 0
Enter if you want to enter a right node (1/0): for the node -- 9-- 0
Enter if you want to enter a right node (1/0): for the node -- 2-- 1
Enter the data: 5
Enter if you want to enter a left node (1/0): for the node -- 5-- 0
Enter if you want to enter a right node (1/0): for the node -- 5-- 0
Enter if you want to enter a right node (1/0): for the node -- 1-- 1
```

```
Enter the data: 3
Enter if you want to enter a left node (1/0): for the node -- 3-- 1
Enter the data: 6
Enter if you want to enter a left node (1/0): for the node -- 6-- 0
Enter if you want to enter a right node (1/0): for the node -- 6-- 0
Enter if you want to enter a right node (1/0): for the node -- 3-- 1
Enter the data: 7
Enter if you want to enter a left node (1/0): for the node -- 7-- 0
Enter if you want to enter a right node (1/0): for the node -- 7-- 0

What would like to do?


Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

5
Traversing through the binary tree inorder recursively:
8 4 9 2 5 1 6 3 7
What would like to do?


Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

7
Traversing through the Binary Tree PreOrder Recursively:
1 2 4 8 9 5 3 6 7
What would like to do?


Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
```

```
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

10
Traversing through the Binary Tree PostOrder Iteratively:
8 9 4 5 2 6 7 3 1
What would like to do?


Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

1
Enter the data:
1
Enter if you want to enter a left node (1/0): for the node -- 1-- 1
Enter the data: 2
Enter if you want to enter a left node (1/0): for the node -- 2-- 1
Enter the data: 5
Enter if you want to enter a left node (1/0): for the node -- 5-- 0
Enter if you want to enter a right node (1/0): for the node -- 5-- 0
Enter if you want to enter a right node (1/0): for the node -- 2-- 1
Enter the data: 3
Enter if you want to enter a left node (1/0): for the node -- 3-- 1
```

```
Enter the data: 4
Enter if you want to enter a left node (1/0): for the node -- 4-- 0
Enter if you want to enter a right node (1/0): for the node -- 4-- 0
Enter if you want to enter a right node (1/0): for the node -- 3-- 0
Enter if you want to enter a right node (1/0): for the node -- 1-- 1
Enter the data: 6
Enter if you want to enter a left node (1/0): for the node -- 6-- 0
Enter if you want to enter a right node (1/0): for the node -- 6-- 0

What would like to do?


Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

6
Traversing through the Binary Tree Inorder Iteratively:
5 2 4 3 1 6
What would like to do?


Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

8
Traversing through the Binary Tree PreOrder Iteratively:
1 2 5 3 4 6
What would like to do?


Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
```

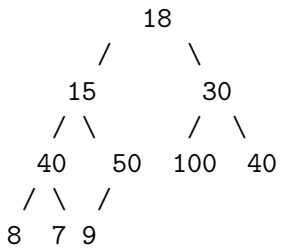
```
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit

9
Traversing through the Binary Tree PostOrder Recursively:
5 4 3 2 6 1
What would like to do?


Welcome to ADS Assignment 2 - Binary Tree Traversals

What would you like to do?
1. Create a Binary Tree
2. Insert Elements to the Tree in Auto Ascending Order
3. Insert Elements to the Tree Level by Level
4. Insert Elements into the Tree Manually
5. Traverse the Tree Inorder Recursively
6. Traverse the Tree Inorder Iteratively
7. Traverse the Tree PreOrder Recursively
8. Traverse the Tree PreOrder Iteratively
9. Traverse the Tree PostOrder Recursively
10. Traverse the Tree PostOrder Iteratively
11. Exit
```

11 Conclusion

Thus, learnt about the different kinds of traversals in binary tree and also learnt about the recursive and non-recursive approach of programming.

12 FAQ

1. Explain any one application of binary tree with suitable example.
2. Explain sequential representation of binary tree with example.
3. Write inorder, preorder and postorder for following tree.



Answer:

(a) Inorder: 8, 40, 7, 15, 9, 50, 18, 100, 30, 100, 40

(b) Preorder: 15, 40, 8, 7, 50, 9, 18, 30, 100, 40, 100

(c) Postorder: 8, 7, 40, 9, 50, 15, 100, 40, 100, 30, 18