

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

<b>1 Objectives</b>	<b>1</b>
<b>2 Problem Statement</b>	<b>1</b>
<b>3 Theory</b>	<b>1</b>
3.1 Tree . . . . .	1
3.2 Binary Tree . . . . .	1
3.2.1 Binary Tree Properties . . . . .	1
3.2.2 Binary Tree Representation . . . . .	2
3.2.3 Types of Binary Tree . . . . .	2
3.2.4 Different definitions related to binary tree. . . . .	2
3.3 Different Traversals (Inorder, Preorder and Postorder) . . . . .	3
3.3.1 Inorder Traversal . . . . .	3
3.3.2 Algorithm for Inorder Recursive Traversal . . . . .	3
3.3.3 Algorithm for Inorder Iterative Traversal . . . . .	3
3.3.4 Preorder Traversal . . . . .	4
3.3.5 Algorithm for PreOrder Recursive Traversal . . . . .	4
3.3.6 Algorithm for PreOrder Iterative Traversal . . . . .	4
3.3.7 Postorder Traversal . . . . .	4
3.3.8 Algorithm for PostOrder Recursive Traversal . . . . .	4
3.3.9 Algorithm for PostOrder Iterative Traversal . . . . .	4
<b>4 Platform</b>	<b>5</b>
<b>5 Input</b>	<b>5</b>
<b>6 Output</b>	<b>5</b>
<b>7 Test Conditions</b>	<b>5</b>
<b>8 Pseudo Code</b>	<b>5</b>
8.1 Inorder Traversal - Iterative Approach . . . . .	5
8.2 Preorder Traversal - Iterative Approach . . . . .	5
8.3 Postorder Traversal - Iterative Approach . . . . .	6
<b>9 Time Complexity</b>	<b>6</b>
<b>10 Code</b>	<b>6</b>
10.1 Program . . . . .	6
10.2 Input and Output . . . . .	11
<b>11 Conclusion</b>	<b>20</b>
<b>12 FAQ</b>	<b>21</b>

## **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  $2^l$ .
2. Maximum number of nodes in a binary tree of height 'h' is  $2^h - 1$ .

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 Traversal

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 Traversal

1. Create an empty stack  $S$ .
2. Initialize current node as root
3. Push the current node to  $S$  and set  $current = current \rightarrow left$  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  $current = poppedItem \rightarrow right$
  - (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 Traversal**

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 Traversal**

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 Traversal**

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 Traversal**

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

```
1 void inorder_iterative(Node){
2     if(Node == NULL){
3         return;
4     }
5     stack<Node> s;
6     Node current = Node;
7     while(current != NULL || s.empty() == false){
8         while(current != NULL){
9             s.push(current);
10            current = current->left;
11        }
12        current = s.top();
13        s.pop();
14        cout << current->data << " ";
15        current = current->right;
16    }
17 }
```

#### 8.2 Preorder Traversal - Iterative Approach

```
1 void preorder_iterative(Node){
2     if(Node == NULL){
3         return;
4     }
5     stack<Node> s;
6     s.push(Node);
```

```
7     while(s.empty() == false){
8         Node current = s.top();
9         cout << current->data << " ";
10        s.pop();
11        if(current->right){
12            s.push(current->right);
13        }
14        if(current->left){
15            s.push(current->left);
16        }
17    }
18 }
```

### 8.3 Postorder Traversal - Iterative Approach

```
1 void postorder_iterative(Node){
2     if(Node == NULL){
3         return;
4     }
5     stack<Node> s1, s2;
6     s1.push(Node);
7     while(s1.empty() == false){
8         Node current = s1.top();
9         s1.pop();
10        s2.push(current);
11        if(current->left){
12            s1.push(current->left);
13        }
14        if(current->right){
15            s1.push(current->right);
16        }
17    }
18    while(s2.empty() == false){
19        Node current = s2.top();
20        cout << current->data << " ";
21        s2.pop();
22    }
23 }
```

## 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



```
1 #include <iostream>
2 #include <stack>
3
4 using namespace std;
5
6 class TreeNode
7 {
8     char data[10];
9     TreeNode *left;
10    TreeNode *right;
11    friend class BinaryTree;
12 };
13
14 class BinaryTree
15 {
16 public:
17     TreeNode *root;
18     BinaryTree()
19     {
20         root = NULL;
21     }
22     void create_root()
23     {
24         root = new TreeNode;
25         cout << "Enter the data: " << endl;
26         cin >> root->data;
27         root->left = NULL;
28         root->right = NULL;
29         create_recursive(root);
30     }
31     void create_recursive(TreeNode *Node)
32     {
33         int choice = 0;
34         TreeNode *new_node;
35         cout << "Enter if you want to enter a left node (1/0): "
36              << "for the node -- " << Node->data << "-- ";
37         cin >> choice;
38         if (choice == 1)
39         {
40             new_node = new TreeNode;
41             cout << "Enter the data: ";
42             cin >> new_node->data;
43             Node->left = new_node;
44             create_recursive(new_node);
45         }
46         cout << "Enter if you want to enter a right node (1/0): "
47              << "for the node -- " << Node->data << "-- ";
48         cin >> choice;
49         if (choice == 1)
50         {
51             new_node = new TreeNode;
52             cout << "Enter the data: ";
53             cin >> new_node->data;
54             Node->right = new_node;
55             create_recursive(new_node);
56         }
57     }
58
59     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     }
```

```
119         if (current->right)
120         {
121             s.push(current->right);
122         }
123         if (current->left)
124         {
125             s.push(current->left);
126         }
127     }
128 }
129
130 void postorder_recursive(TreeNode *temp)
131 {
132     if (temp == NULL)
133     {
134         return;
135     }
136     postorder_recursive(temp->left);
137     postorder_recursive(temp->right);
138     cout << temp->data << " ";
139 }
140 void postorder_iterative(TreeNode *temp)
141 {
142     if (!temp)
143     {
144         return;
145     }
146
147     stack<TreeNode *> s1;
148     stack<TreeNode *> s2;
149
150     s1.push(temp);
151
152     while (s1.empty() == false)
153     {
154         TreeNode *current = s1.top();
155         s1.pop();
156         s2.push(current);
157
158         if (current->left)
159         {
160             s1.push(current->left);
161         }
162         if (current->right)
163         {
164             s1.push(current->right);
165         }
166     }
167     while (s2.empty() == false)
168     {
169         TreeNode *current = s2.top();
170         cout << current->data << " ";
171         s2.pop();
172     }
173 }
174 };
175
176 int main()
177 {
```

```
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;
```

```
230         main_tree.postorder_iterative(main_tree.root);
231         break;
232     case 8:
233         cout << "Exiting the program" << endl;
234         exit(0);
235         break;
236     default:
237         cout << "Invalid Choice" << endl;
238         break;
239     }
240 }
241 }
```

### 10.2 Input and Output

```
1 What would like to do?
2
3
4 Welcome to ADS Assignment 2 - Binary Tree Traversals
5
6 What would you like to do?
7 1. Create a Binary Tree
8 2. Insert Elements to the Tree in Auto Ascending Order
9 3. Insert Elements to the Tree Level by Level
10 4. Insert Elements into the Tree Manually
11 5. Traverse the Tree Inorder Recursively
12 6. Traverse the Tree Inorder Iteratively
13 7. Traverse the Tree PreOrder Recursively
14 8. Traverse the Tree PreOrder Iteratively
15 9. Traverse the Tree PostOrder Recursively
16 10. Traverse the Tree PostOrder Iteratively
17 11. Exit
18
19 1
20 Enter the data: 1
21 Enter if you want to enter a left node (1/0): for the node -- 1-- 1
22 Enter the data: 2
23 Enter if you want to enter a left node (1/0): for the node -- 2-- 1
24 Enter the data: 3
25 Enter if you want to enter a left node (1/0): for the node -- 3-- 1
26 Enter the data: 4
27 Enter if you want to enter a left node (1/0): for the node -- 4-- 1
28 Enter the data: 5
29 Enter if you want to enter a left node (1/0): for the node -- 5-- 0
30 Enter if you want to enter a right node (1/0): for the node -- 5-- 0
31 Enter if you want to enter a right node (1/0): for the node -- 4-- 0
32 Enter if you want to enter a right node (1/0): for the node -- 3-- 0
33 Enter if you want to enter a right node (1/0): for the node -- 2-- 0
34 Enter if you want to enter a right node (1/0): for the node -- 1-- 0
35
36 What would like to do?
37
38
39 Welcome to ADS Assignment 2 - Binary Tree Traversals
40
41 What would you like to do?
42 1. Create a Binary Tree
43 2. Insert Elements to the Tree in Auto Ascending Order
44 3. Insert Elements to the Tree Level by Level
```

## *Advanced Data Structures - Assignment 2*

---

```
45 4. Insert Elements into the Tree Manually
46 5. Traverse the Tree Inorder Recursively
47 6. Traverse the Tree Inorder Iteratively
48 7. Traverse the Tree PreOrder Recursively
49 8. Traverse the Tree PreOrder Iteratively
50 9. Traverse the Tree PostOrder Recursively
51 10. Traverse the Tree PostOrder Iteratively
52 11. Exit
53
54 5
55 Traversing through the binary tree inorder recursively:
56 5 4 3 2 1
57 What would like to do?
58
59
60 Welcome to ADS Assignment 2 - Binary Tree Traversals
61
62 What would you like to do?
63 1. Create a Binary Tree
64 2. Insert Elements to the Tree in Auto Ascending Order
65 3. Insert Elements to the Tree Level by Level
66 4. Insert Elements into the Tree Manually
67 5. Traverse the Tree Inorder Recursively
68 6. Traverse the Tree Inorder Iteratively
69 7. Traverse the Tree PreOrder Recursively
70 8. Traverse the Tree PreOrder Iteratively
71 9. Traverse the Tree PostOrder Recursively
72 10. Traverse the Tree PostOrder Iteratively
73 11. Exit
74
75 7
76 Traversing through the Binary Tree PreOrder Recursively:
77 1 2 3 4 5
78 What would like to do?
79
80
81 Welcome to ADS Assignment 2 - Binary Tree Traversals
82
83 What would you like to do?
84 1. Create a Binary Tree
85 2. Insert Elements to the Tree in Auto Ascending Order
86 3. Insert Elements to the Tree Level by Level
87 4. Insert Elements into the Tree Manually
88 5. Traverse the Tree Inorder Recursively
89 6. Traverse the Tree Inorder Iteratively
90 7. Traverse the Tree PreOrder Recursively
91 8. Traverse the Tree PreOrder Iteratively
92 9. Traverse the Tree PostOrder Recursively
93 10. Traverse the Tree PostOrder Iteratively
94 11. Exit
95
96 9
97 Traversing through the Binary Tree PostOrder Recursively:
98 5 4 3 2 1
99
100 What would like to do?
101
102
103 Welcome to ADS Assignment 2 - Binary Tree Traversals
```

## Advanced Data Structures - Assignment 2

---

```
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
```

## Advanced Data Structures - Assignment 2

---

```
163 2. Insert Elements to the Tree in Auto Ascending Order
164 3. Insert Elements to the Tree Level by Level
165 4. Insert Elements into the Tree Manually
166 5. Traverse the Tree Inorder Recursively
167 6. Traverse the Tree Inorder Iteratively
168 7. Traverse the Tree PreOrder Recursively
169 8. Traverse the Tree PreOrder Iteratively
170 9. Traverse the Tree PostOrder Recursively
171 10. Traverse the Tree PostOrder Iteratively
172 11. Exit
173
174 8
175 Traversing through the Binary Tree PreOrder Iteratively:
176 1 2 3 4 5
177 What would like to do?
178
179
180 Welcome to ADS Assignment 2 - Binary Tree Traversals
181
182 What would you like to do?
183 1. Create a Binary Tree
184 2. Insert Elements to the Tree in Auto Ascending Order
185 3. Insert Elements to the Tree Level by Level
186 4. Insert Elements into the Tree Manually
187 5. Traverse the Tree Inorder Recursively
188 6. Traverse the Tree Inorder Iteratively
189 7. Traverse the Tree PreOrder Recursively
190 8. Traverse the Tree PreOrder Iteratively
191 9. Traverse the Tree PostOrder Recursively
192 10. Traverse the Tree PostOrder Iteratively
193 11. Exit
194
195 10
196 Traversing through the Binary Tree PostOrder Iteratively:
197 5 4 3 2 1
198
199
200 Welcome to ADS Assignment 2 - Binary Tree Traversals
201
202 What would you like to do?
203 1. Create a Binary Tree
204 2. Insert Elements to the Tree in Auto Ascending Order
205 3. Insert Elements to the Tree Level by Level
206 4. Insert Elements into the Tree Manually
207 5. Traverse the Tree Inorder Recursively
208 6. Traverse the Tree Inorder Iteratively
209 7. Traverse the Tree PreOrder Recursively
210 8. Traverse the Tree PreOrder Iteratively
211 9. Traverse the Tree PostOrder Recursively
212 10. Traverse the Tree PostOrder Iteratively
213 11. Exit
214
215 1
216 Enter the data: 1
217 Enter if you want to enter a left node (1/0): for the node -- 1-- 1
218 Enter the data: 2
219 Enter if you want to enter a left node (1/0): for the node -- 2-- 1
220 Enter the data: 4
221 Enter if you want to enter a left node (1/0): for the node -- 4-- 0
```



## Advanced Data Structures - Assignment 2

---

```
222 Enter if you want to enter a right node (1/0): for the node -- 4-- 0
223 Enter if you want to enter a right node (1/0): for the node -- 2-- 1
224 Enter the data: 5
225 Enter if you want to enter a left node (1/0): for the node -- 5-- 0
226 Enter if you want to enter a right node (1/0): for the node -- 5-- 0
227 Enter if you want to enter a right node (1/0): for the node -- 1-- 1
228 Enter the data: 3
229 Enter if you want to enter a left node (1/0): for the node -- 3-- 1
230 Enter the data: 6
231 Enter if you want to enter a left node (1/0): for the node -- 6-- 0
232 Enter if you want to enter a right node (1/0): for the node -- 6-- 0
233 Enter if you want to enter a right node (1/0): for the node -- 3-- 1
234 Enter the data: 7
235 Enter if you want to enter a left node (1/0): for the node -- 7-- 0
236 Enter if you want to enter a right node (1/0): for the node -- 7-- 0
237
238
239 What would like to do?
240
241
242 Welcome to ADS Assignment 2 - Binary Tree Traversals
243
244 What would you like to do?
245 1. Create a Binary Tree
246 2. Insert Elements to the Tree in Auto Ascending Order
247 3. Insert Elements to the Tree Level by Level
248 4. Insert Elements into the Tree Manually
249 5. Traverse the Tree Inorder Recursively
250 6. Traverse the Tree Inorder Iteratively
251 7. Traverse the Tree PreOrder Recursively
252 8. Traverse the Tree PreOrder Iteratively
253 9. Traverse the Tree PostOrder Recursively
254 10. Traverse the Tree PostOrder Iteratively
255 11. Exit
256
257 5
258 Traversing through the binary tree inorder recursively:
259 4 2 5 1 6 3 7
260 What would like to do?
261
262
263 Welcome to ADS Assignment 2 - Binary Tree Traversals
264
265 What would you like to do?
266 1. Create a Binary Tree
267 2. Insert Elements to the Tree in Auto Ascending Order
268 3. Insert Elements to the Tree Level by Level
269 4. Insert Elements into the Tree Manually
270 5. Traverse the Tree Inorder Recursively
271 6. Traverse the Tree Inorder Iteratively
272 7. Traverse the Tree PreOrder Recursively
273 8. Traverse the Tree PreOrder Iteratively
274 9. Traverse the Tree PostOrder Recursively
275 10. Traverse the Tree PostOrder Iteratively
276 11. Exit
277
278 7
279 Traversing through the Binary Tree PreOrder Recursively:
280 1 2 4 5 3 6 7
```

## *Advanced Data Structures - Assignment 2*

---

```
281 What would like to do?
282
283
284 Welcome to ADS Assignment 2 - Binary Tree Traversals
285
286 What would you like to do?
287 1. Create a Binary Tree
288 2. Insert Elements to the Tree in Auto Ascending Order
289 3. Insert Elements to the Tree Level by Level
290 4. Insert Elements into the Tree Manually
291 5. Traverse the Tree Inorder Recursively
292 6. Traverse the Tree Inorder Iteratively
293 7. Traverse the Tree PreOrder Recursively
294 8. Traverse the Tree PreOrder Iteratively
295 9. Traverse the Tree PostOrder Recursively
296 10. Traverse the Tree PostOrder Iteratively
297 11. Exit
298
299 9
300 Traversing through the Binary Tree PostOrder Recursively:
301 4 5 2 6 7 3 1
302
303
304 What would like to do?
305 Welcome to ADS Assignment 2 - Binary Tree Traversals
306
307 What would you like to do?
308 1. Create a Binary Tree
309 2. Insert Elements to the Tree in Auto Ascending Order
310 3. Insert Elements to the Tree Level by Level
311 4. Insert Elements into the Tree Manually
312 5. Traverse the Tree Inorder Recursively
313 6. Traverse the Tree Inorder Iteratively
314 7. Traverse the Tree PreOrder Recursively
315 8. Traverse the Tree PreOrder Iteratively
316 9. Traverse the Tree PostOrder Recursively
317 10. Traverse the Tree PostOrder Iteratively
318 11. Exit
319
320 1
321 Enter the data:
322 1
323 Enter if you want to enter a left node (1/0): for the node -- 1-- 1
324 Enter the data: 2
325 Enter if you want to enter a left node (1/0): for the node -- 2-- 1
326 Enter the data: 4
327 Enter if you want to enter a left node (1/0): for the node -- 4-- 1
328 Enter the data: 8
329 Enter if you want to enter a left node (1/0): for the node -- 8-- 0
330 Enter if you want to enter a right node (1/0): for the node -- 8-- 0
331 Enter if you want to enter a right node (1/0): for the node -- 4-- 1
332 Enter the data: 9
333 Enter if you want to enter a left node (1/0): for the node -- 9-- 0
334 Enter if you want to enter a right node (1/0): for the node -- 9-- 0
335 Enter if you want to enter a right node (1/0): for the node -- 2-- 1
336 Enter the data: 5
337 Enter if you want to enter a left node (1/0): for the node -- 5-- 0
338 Enter if you want to enter a right node (1/0): for the node -- 5-- 0
339 Enter if you want to enter a right node (1/0): for the node -- 1-- 1
```

## *Advanced Data Structures - Assignment 2*

---

```
340 Enter the data: 3
341 Enter if you want to enter a left node (1/0): for the node -- 3-- 1
342 Enter the data: 6
343 Enter if you want to enter a left node (1/0): for the node -- 6-- 0
344 Enter if you want to enter a right node (1/0): for the node -- 6-- 0
345 Enter if you want to enter a right node (1/0): for the node -- 3-- 1
346 Enter the data: 7
347 Enter if you want to enter a left node (1/0): for the node -- 7-- 0
348 Enter if you want to enter a right node (1/0): for the node -- 7-- 0
349
350 What would like to do?
351
352
353 Welcome to ADS Assignment 2 - Binary Tree Traversals
354
355 What would you like to do?
356 1. Create a Binary Tree
357 2. Insert Elements to the Tree in Auto Ascending Order
358 3. Insert Elements to the Tree Level by Level
359 4. Insert Elements into the Tree Manually
360 5. Traverse the Tree Inorder Recursively
361 6. Traverse the Tree Inorder Iteratively
362 7. Traverse the Tree PreOrder Recursively
363 8. Traverse the Tree PreOrder Iteratively
364 9. Traverse the Tree PostOrder Recursively
365 10. Traverse the Tree PostOrder Iteratively
366 11. Exit
367
368 5
369 Traversing through the binary tree inorder recursively:
370 8 4 9 2 5 1 6 3 7
371 What would like to do?
372
373
374 Welcome to ADS Assignment 2 - Binary Tree Traversals
375
376 What would you like to do?
377 1. Create a Binary Tree
378 2. Insert Elements to the Tree in Auto Ascending Order
379 3. Insert Elements to the Tree Level by Level
380 4. Insert Elements into the Tree Manually
381 5. Traverse the Tree Inorder Recursively
382 6. Traverse the Tree Inorder Iteratively
383 7. Traverse the Tree PreOrder Recursively
384 8. Traverse the Tree PreOrder Iteratively
385 9. Traverse the Tree PostOrder Recursively
386 10. Traverse the Tree PostOrder Iteratively
387 11. Exit
388
389 7
390 Traversing through the Binary Tree PreOrder Recursively:
391 1 2 4 8 9 5 3 6 7
392 What would like to do?
393
394
395 Welcome to ADS Assignment 2 - Binary Tree Traversals
396
397 What would you like to do?
398 1. Create a Binary Tree
```

## Advanced Data Structures - Assignment 2

---

```
399 2. Insert Elements to the Tree in Auto Ascending Order
400 3. Insert Elements to the Tree Level by Level
401 4. Insert Elements into the Tree Manually
402 5. Traverse the Tree Inorder Recursively
403 6. Traverse the Tree Inorder Iteratively
404 7. Traverse the Tree PreOrder Recursively
405 8. Traverse the Tree PreOrder Iteratively
406 9. Traverse the Tree PostOrder Recursively
407 10. Traverse the Tree PostOrder Iteratively
408 11. Exit
409
410 10
411 Traversing through the Binary Tree PostOrder Iteratively:
412 8 9 4 5 2 6 7 3 1
413 What would like to do?
414
415
416 Welcome to ADS Assignment 2 - Binary Tree Traversals
417
418 What would you like to do?
419 1. Create a Binary Tree
420 2. Insert Elements to the Tree in Auto Ascending Order
421 3. Insert Elements to the Tree Level by Level
422 4. Insert Elements into the Tree Manually
423 5. Traverse the Tree Inorder Recursively
424 6. Traverse the Tree Inorder Iteratively
425 7. Traverse the Tree PreOrder Recursively
426 8. Traverse the Tree PreOrder Iteratively
427 9. Traverse the Tree PostOrder Recursively
428 10. Traverse the Tree PostOrder Iteratively
429 11. Exit
430
431 Welcome to ADS Assignment 2 - Binary Tree Traversals
432
433 What would you like to do?
434 1. Create a Binary Tree
435 2. Insert Elements to the Tree in Auto Ascending Order
436 3. Insert Elements to the Tree Level by Level
437 4. Insert Elements into the Tree Manually
438 5. Traverse the Tree Inorder Recursively
439 6. Traverse the Tree Inorder Iteratively
440 7. Traverse the Tree PreOrder Recursively
441 8. Traverse the Tree PreOrder Iteratively
442 9. Traverse the Tree PostOrder Recursively
443 10. Traverse the Tree PostOrder Iteratively
444 11. Exit
445
446 1
447 Enter the data:
448 1
449 Enter if you want to enter a left node (1/0): for the node -- 1-- 1
450 Enter the data: 2
451 Enter if you want to enter a left node (1/0): for the node -- 2-- 1
452 Enter the data: 5
453 Enter if you want to enter a left node (1/0): for the node -- 5-- 0
454 Enter if you want to enter a right node (1/0): for the node -- 5-- 0
455 Enter if you want to enter a right node (1/0): for the node -- 2-- 1
456 Enter the data: 3
457 Enter if you want to enter a left node (1/0): for the node -- 3-- 1
```

## *Advanced Data Structures - Assignment 2*

---

```
458 Enter the data: 4
459 Enter if you want to enter a left node (1/0): for the node -- 4-- 0
460 Enter if you want to enter a right node (1/0): for the node -- 4-- 0
461 Enter if you want to enter a right node (1/0): for the node -- 3-- 0
462 Enter if you want to enter a right node (1/0): for the node -- 1-- 1
463 Enter the data: 6
464 Enter if you want to enter a left node (1/0): for the node -- 6-- 0
465 Enter if you want to enter a right node (1/0): for the node -- 6-- 0
466
467 What would like to do?
468
469
470 Welcome to ADS Assignment 2 - Binary Tree Traversals
471
472 What would you like to do?
473 1. Create a Binary Tree
474 2. Insert Elements to the Tree in Auto Ascending Order
475 3. Insert Elements to the Tree Level by Level
476 4. Insert Elements into the Tree Manually
477 5. Traverse the Tree Inorder Recursively
478 6. Traverse the Tree Inorder Iteratively
479 7. Traverse the Tree PreOrder Recursively
480 8. Traverse the Tree PreOrder Iteratively
481 9. Traverse the Tree PostOrder Recursively
482 10. Traverse the Tree PostOrder Iteratively
483 11. Exit
484
485 6
486 Traversing through the Binary Tree Inorder Iteratively:
487 5 2 4 3 1 6
488 What would like to do?
489
490
491 Welcome to ADS Assignment 2 - Binary Tree Traversals
492
493 What would you like to do?
494 1. Create a Binary Tree
495 2. Insert Elements to the Tree in Auto Ascending Order
496 3. Insert Elements to the Tree Level by Level
497 4. Insert Elements into the Tree Manually
498 5. Traverse the Tree Inorder Recursively
499 6. Traverse the Tree Inorder Iteratively
500 7. Traverse the Tree PreOrder Recursively
501 8. Traverse the Tree PreOrder Iteratively
502 9. Traverse the Tree PostOrder Recursively
503 10. Traverse the Tree PostOrder Iteratively
504 11. Exit
505
506 8
507 Traversing through the Binary Tree PreOrder Iteratively:
508 1 2 5 3 4 6
509 What would like to do?
510
511
512 Welcome to ADS Assignment 2 - Binary Tree Traversals
513
514 What would you like to do?
515 1. Create a Binary Tree
516 2. Insert Elements to the Tree in Auto Ascending Order
```

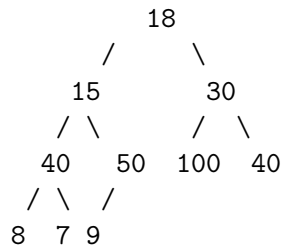
```
517 3. Insert Elements to the Tree Level by Level
518 4. Insert Elements into the Tree Manually
519 5. Traverse the Tree Inorder Recursively
520 6. Traverse the Tree Inorder Iteratively
521 7. Traverse the Tree PreOrder Recursively
522 8. Traverse the Tree PreOrder Iteratively
523 9. Traverse the Tree PostOrder Recursively
524 10. Traverse the Tree PostOrder Iteratively
525 11. Exit
526
527 9
528 Traversing through the Binary Tree PostOrder Recursively:
529 5 4 3 2 6 1
530 What would like to do?
531
532
533 Welcome to ADS Assignment 2 - Binary Tree Traversals
534
535 What would you like to do?
536 1. Create a Binary Tree
537 2. Insert Elements to the Tree in Auto Ascending Order
538 3. Insert Elements to the Tree Level by Level
539 4. Insert Elements into the Tree Manually
540 5. Traverse the Tree Inorder Recursively
541 6. Traverse the Tree Inorder Iteratively
542 7. Traverse the Tree PreOrder Recursively
543 8. Traverse the Tree PreOrder Iteratively
544 9. Traverse the Tree PostOrder Recursively
545 10. Traverse the Tree PostOrder Iteratively
546 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