**USN:- 1NT21IS039**

## Aim:- BINARY TREE & ITS TRAVERSALS

## [GitHub Link](https://github.com/logan-14/BinaryTree)

### **Algorithm:-**

1. A new binary tree is created and values are assigned  
   2. Write a function insert() in such a way that node and key will be two parameters and check for below conditions,  
   a. If rootNode == NULL, then return new node to calling function.  
   b. If rootNode => data < keyValue, then call insert() with rootNode => rightNode and assign return value in rootNode => rightNode.  
   c. If rootNode => data > keyValue, then call insert() with rootNode => leftNode and assign return value in rootNode => leftNode  
   3. Then finally, we can return original rootNode pointer to calling function.

**Code:-**

// Tree traversal in C

#include <stdio.h>

#include <stdlib.h>

struct node

{

    int item;

    struct node \*left;

    struct node \*right;

};

// Inorder traversal

void inorderTraversal(struct node \*root)

{

    if (root == NULL)

        return;

    inorderTraversal(root->left);

    printf("%d ", root->item);

    inorderTraversal(root->right);

}

// Preorder traversal

void preorderTraversal(struct node \*root)

{

    if (root == NULL)

        return;

    printf("%d ", root->item);

    preorderTraversal(root->left);

    preorderTraversal(root->right);

}

// Postorder traversal

void postorderTraversal(struct node \*root)

{

    if (root == NULL)

        return;

    postorderTraversal(root->left);

    postorderTraversal(root->right);

    printf("%d ", root->item);

}

// Create a new Node

struct node \*create(int value)

{

    struct node \*newNode = malloc(sizeof(struct node));

    newNode->item = value;

    newNode->left = NULL;

    newNode->right = NULL;

    return newNode;

}

// Insert on the left of the node

struct node \*insertLeft(struct node \*root, int value)

{

    root->left = create(value);

    return root->left;

}

// Insert on the right of the node

struct node \*insertRight(struct node \*root, int value)

{

    root->right = create(value);

    return root->right;

}

int main()

{

    struct node \*root = create(1);

    insertLeft(root, 4);

    insertRight(root, 6);

    insertLeft(root->left, 42);

    insertRight(root->left, 3);

    insertLeft(root->right, 2);

    insertRight(root->right, 33);

    printf("Traversal of the inserted binary tree \n");

    printf("Inorder traversal \n");

    inorderTraversal(root);

    printf("\nPreorder traversal \n");

    preorderTraversal(root);

    printf("\nPostorder traversal \n");

    postorderTraversal(root);

}

**Output:-**

\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*

Traversal of the inserted binary tree

Inorder traversal

42 4 3 1 2 6 33

Preorder traversal

1 4 42 3 6 2 33

Postorder traversal

42 3 4 2 33 6 1