

SIDDHI SINGH

17BIT0028

ITE - 1004

DATA STRUCTURES AND ALGORITHM BINARY TREE TRAVERSALS

```
// C program for different tree traversals
#include <stdio.h>
#include <stdlib.h>

/* A binary tree node has data, pointer to left child
   and a pointer to right child */
struct node
{
    int data;
    struct node* left;
    struct node* right;
};

/* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */
struct node* newNode(int data)
{
    struct node* node = (struct node*)
                        malloc(sizeof(struct node));

    node->data = data;
    node->left = NULL;
    node->right = NULL;

    return(node);
}

/* Given a binary tree, print its nodes according to the
   "bottom-up" postorder traversal. */
void printPostorder(struct node* node)
{
    if (node == NULL)
        return;

    // first recur on left subtree
    printPostorder(node->left);

    // then recur on right subtree
    printPostorder(node->right);

    // now deal with the node
    printf("%d ", node->data);
}
```

```

}

/* Given a binary tree, print its nodes in inorder*/
void printInorder(struct node* node)
{
    if (node == NULL)
        return;

    /* first recur on left child */
    printInorder(node->left);

    /* then print the data of node */
    printf("%d ", node->data);

    /* now recur on right child */
    printInorder(node->right);
}

/* Given a binary tree, print its nodes in preorder*/
void printPreorder(struct node* node)
{
    if (node == NULL)
        return;

    /* first print data of node */
    printf("%d ", node->data);

    /* then recur on left subtree */
    printPreorder(node->left);

    /* now recur on right subtree */
    printPreorder(node->right);
}

/* Driver program to test above functions*/
int main()
{
    struct node *root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);

    printf("\nPreorder traversal of binary tree is \n");
    printPreorder(root);
}

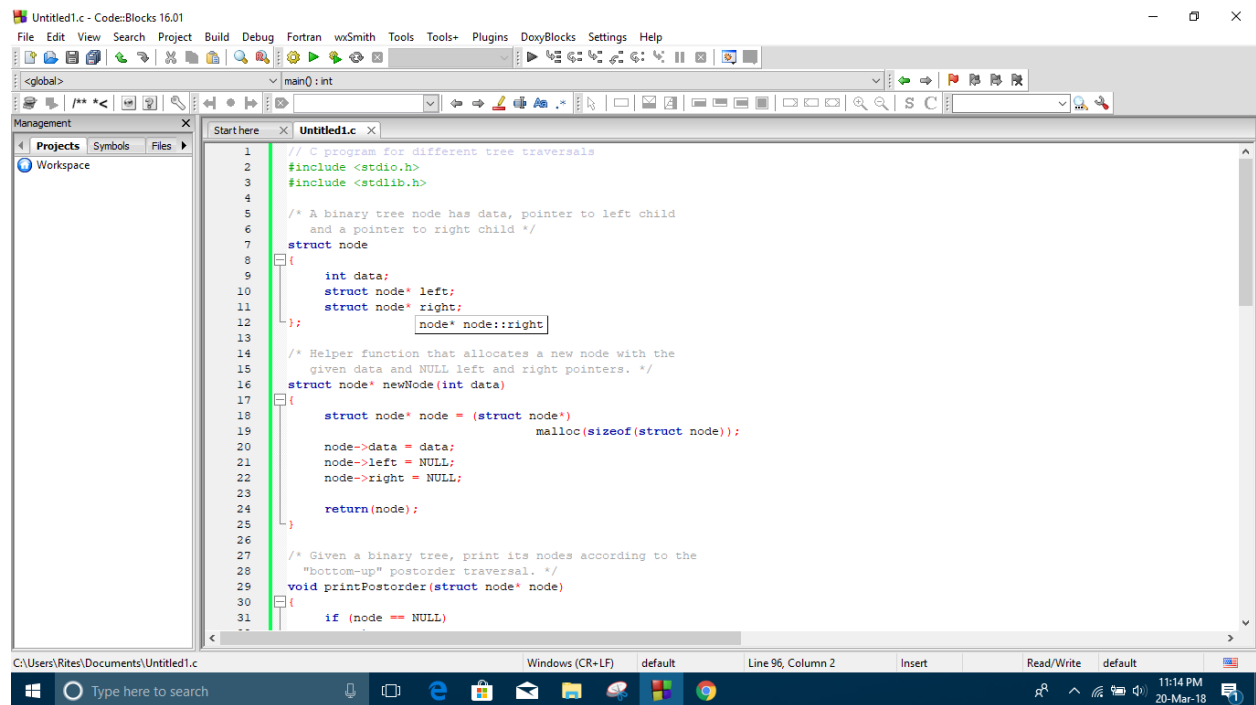
```

```
printf("\nInorder traversal of binary tree is \n");  
printInorder(root);
```

```
printf("\nPostorder traversal of binary tree is \n");  
printPostorder(root);
```

```
getchar();  
return 0;
```

```
}
```



The screenshot shows the Code::Blocks IDE with a C program for binary tree traversals. The program includes headers for stdio.h and stdlib.h, defines a struct node with data, left, and right pointers, and implements a newNode function to allocate and initialize nodes. It also includes a partial implementation of printPostorder.

```
1 // C program for different tree traversals  
2 #include <stdio.h>  
3 #include <stdlib.h>  
4  
5 /* A binary tree node has data, pointer to left child  
6    and a pointer to right child */  
7 struct node  
8 {  
9     int data;  
10    struct node* left;  
11    struct node* right;  
12 };  
13  
14 /* Helper function that allocates a new node with the  
15    given data and NULL left and right pointers. */  
16 struct node* newNode(int data)  
17 {  
18     struct node* node = (struct node*)  
19         malloc(sizeof(struct node));  
20     node->data = data;  
21     node->left = NULL;  
22     node->right = NULL;  
23     return(node);  
24 }  
25  
26  
27 /* Given a binary tree, print its nodes according to the  
28    "bottom-up" postorder traversal. */  
29 void printPostorder(struct node* node)  
30 {  
31     if (node == NULL)  
32     {
```


The screenshot shows the Code::Blocks IDE with a C program for binary tree operations. The code defines a `node` struct with `data`, `left`, and `right` pointers. It includes functions for pre-order, in-order, and post-order traversals, as well as a `main` function that constructs a binary tree with 5 nodes and prints the results of each traversal. The tree structure is: root (1) has left child (2) and right child (3); node (2) has left child (4) and right child (5); node (3) is a leaf.

```
67     printf("%d ", node->data);
68
69     /* then recur on left subtree */
70     printPreorder(node->left);
71
72     /* now recur on right subtree */
73     printPreorder(node->right);
74 }
75
76 /* Driver program to test above functions*/
77 int main()
78 {
79     struct node *root = newNode(1);
80     root->left = newNode(2);
81     root->right = newNode(3);
82     root->left->left = newNode(4);
83     root->left->right = newNode(5);
84
85     printf("\nPreorder traversal of binary tree is \n");
86     printPreorder(root);
87
88     printf("\ninorder traversal of binary tree is \n");
89     printInorder(root);
90
91     printf("\nPostorder traversal of binary tree is \n");
92     printPostorder(root);
93
94     getchar();
95     return 0;
96 }
97
```

OUTPUT

```
Preorder traversal of binary tree is
1 2 4 5 3
Inorder traversal of binary tree is
4 2 5 1 3
Postorder traversal of binary tree is
4 5 2 3 1
```