

Tree Traversals (Inorder, Preorder and Postorder) In C++

Description: This packet contains all the three tree traversals like Inorder, Preorder and Postorder traversals in C++ programming language.

About: This packet is all about the tree traversals in C++ programming language. Unlike linear data structure which have one logical way to traverse them, tree can be traversed in different ways. Tree traversals like Inorder, Preorder and Postorder are mentioned in this packet. I used C++ programming language for coding.

Preorder:

```
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
struct node{
```

```
    int data;
```

```
    struct node* left;
```

```
    struct node* right;
```

```
};
```

```
struct node* createNode(int data){
```

```
    struct node *n; // creating a node pointer
```

```
    n = (struct node *) malloc(sizeof(struct node)); // Allocating  
memory in the heap
```

```
    n->data = data; // Setting the data
```

```
    n->left = NULL; // Setting the left and right children to
```

```
NULL n->right = NULL; // Setting the left and right children  
to NULL return n; // Finally returning the created node
```

```
}
```

```
void preOrder(struct node* root){
```

```
if(root!=NULL){  
    cout<<root->data<<" ";  
    preOrder(root->left);  
    preOrder(root->right);  
}  
}
```

```

int main(){

    //    Constructing the root node - Using Function
    (Recommended) struct node *root = createNode(4);

    struct node *p1 = createNode(1);
    struct node *p2 = createNode(6);
    struct node *p3 = createNode(5);
    struct node *p4 = createNode(2);
    // Finally The tree looks like this:

    //    4
    //   /\
    //  1 6
    // /\
    // 5 2

    //    Linking the root node with left and right
    children root->left = p1;
    root->right = p2;
    p1->left = p3;
    p1->right = p4;

    preOrder(root);
    return 0;
}

```

Inorder:

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
struct Node
```

```
{
```

```
    int key;
```

```
    struct Node *left;
```

```
    struct Node *right;
```

```
    Node(int k){
```

```
        key=k;
```

```

        left=right= NULL;
    }
}
;

void inorder(Node *root){
    if(root!=NULL){
        inorder(root->left);
        cout<<root->key<<" ";
        inorder(root->right);
    }
}

int main() {

    Node *root=new Node(10);
    root->left=new Node(20);
    root->right=new Node(30);
    root->right->left=new Node(40);
    root->right->right=new Node(50);

    inorder(root);
}

```

Postorder:

```

#include<bits/stdc++.h>

using namespace std;

struct node{
    int data;

```

```
    struct node* left;
    struct node* right;
};

struct node* createNode(int data){
    struct node *n; // creating a node pointer
    n = (struct node *) malloc(sizeof(struct node)); // Allocating
memory in the heap
    n->data = data; // Setting the data
```

```

n->left = NULL; // Setting the left and right children to
NULL n->right = NULL; // Setting the left and right children
to NULL return n; // Finally returning the created node
}

```

```

void postOrder(struct node* root){
    if(root!=NULL){
        postOrder(root->left);
        postOrder(root->right);
        cout<<root->data<<" ";
    }
}

```

```

int main(){

    //    Constructing the root node - Using Function
    (Recommended) struct node *root =
    createNode(4); struct node *p1 = createNode(1);

    struct node *p2 = createNode(6);
    struct node *p3 = createNode(5);
    struct node *p4 = createNode(2);

    // Finally The tree looks like this:

    //    4
    //   /\
    //  1 6
    // /\
    // 5 2

    // Linking the root node with left and right children

```



```
root->left = p1;  
root->right = p2;  
p1->left = p3;  
p1->right = p4;  
  
postOrder(root);  
return 0;  
}
```