

# Navlakhi®



# BST

# Non Recursive Traverse

**Methodology and Program**

**By Abhishek Navlakhi**  
**Semester 3: Data Structures**

This document is for private circulation for the students of Navlakhi's. More educational content can be found on [www.navlakhi.com](http://www.navlakhi.com) / [navlakhi.mobi](http://navlakhi.mobi)  
To enroll contact 9820246760/9769479368/9820009639

```
#include <stdio.h>
#include <alloc.h>
#include <stdlib.h>
#include <conio.h>

struct tree_node
{
int data;
struct tree_node *left;
struct tree_node *right;
};

struct headnode
{
int count;
struct tree_node *head;
}*tree;

void createTree( )
{
tree=(struct headnode*)malloc(sizeof(struct headnode));
tree->count=0;
tree->head=NULL;
}
```

```

void insertNode( int dataIn )
{
struct tree_node *pNew,*pLoc;
if (tree->count==0)
{
tree->head=(struct tree_node*)malloc(sizeof(struct tree_node));
tree->head->left=NULL;
tree->head->right=NULL;
tree->head->data=dataIn;
tree->count=1;
}
else
{
pNew=(struct tree_node*)malloc(sizeof(struct tree_node));
pNew->left=NULL;
pNew->right=NULL;
pNew->data=dataIn;
pLoc=tree->head;
while(1)
{
if (pNew->data>pLoc->data)
{
if (pLoc->right!=NULL) pLoc=pLoc->right;
else
{pLoc->right=pNew; tree->count+=1;break;}
}
if (pNew->data<pLoc->data)
{
if (pLoc->left!=NULL) pLoc=pLoc->left;
else
{pLoc->left=pNew; tree->count+=1;break;}
}
if (pNew->data==pLoc->data)
{
printf("Data already present....\n");
break;
}
}/*end of while*/
}/*end of count !=0*/
}

```

```
struct tree_node *stack[100] ;  
int top;
```

A red, brush-stroke style graphic with the word "NEW" written in white, slanted capital letters.

```
void Preorder (struct tree_node *p)  
{  
top=-1;  
do  
{  
    while (p!=NULL)  
    {  
        printf("%d\t",p->data);  
        stack[++top]=p;  
        p=p->left;  
    }  
    if (top!=-1)  
    {  
        p=(struct tree_node*)stack[top--];  
        p=p->right;  
    }  
}while(top!=-1 || p!=NULL);  
printf("\n");  
}
```

```
void Postorder(struct tree_node *p)
{
struct tree_node *prev=NULL;
top=-1;
do
{
while (p!=NULL)
{
stack[++top]=p;
p=p->left;
if (p==NULL) p=(stack[top])->right;
}
if (top!=-1)
{
p=stack[top--];
if (p->right!=NULL && p->right!=prev)
{
stack[++top]=p;
p=p->right;
}
else
{
printf("%d\t",p->data);
prev=p;
p=NULL;
}
}
}while(top!=-1);
printf("\n");
}
```



```
void Inorder (struct tree_node *p)
{
top=-1;
do
{
    while (p!=NULL)
    {
        stack[++top]=p;
        p=p->left;
    }
    if (top!=-1)
    {
        p=stack[top--];
        printf("%d\t",p->data);
        p=p->right;
    }
}while(top!=-1 || p!=NULL);
printf("\n");
}
```



```
void DeleteNode(struct tree_node **root,int dltKey)
{
if ((*root)==NULL) {printf("DATA NOT FOUND\n");}
if (dltKey < (*root)->data) DeleteNode(&(*root)->left,dltKey);
else if (dltKey>(*root)->data) DeleteNode(&(*root)->right,dltKey);
else if ((*root)->left==NULL)
    {
    struct tree_node *delPtr;
    delPtr=(*root);
    (*root)=(*root)->right;
    free(delPtr);
    tree->count-=1;
    }
else if ((*root)->right==NULL)
    {
    struct tree_node *delPtr;
    delPtr=(*root);
    (*root)=(*root)->left;
    free(delPtr);
    tree->count-=1;
    }
else {
    struct tree_node *delPtr;
    delPtr=(*root)->left;
    while(delPtr->right!=NULL)
        delPtr=delPtr->right;
    (*root)->data=delPtr->data;
    DeleteNode(&(*root)->left,delPtr->data);
    }
}
```

```
void main( )
{
int choice,delKey,dataIn;
createTree();
do
{
    printf("1.Add Node\n");
    printf("2. Preorder Traversal with Recursion\n");
    printf("3. Inorder Traversal with Recursion\n");
    printf("4. Postorder Traversal with Recursion\n");
    printf("5. Delete Node\n");
    printf("6.Exit\n");
    printf("Feed in your choice: ");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1:    printf("Feed in the data: ");
                   scanf("%d",&dataIn);
                   insertNode(dataIn);
                   break;
        case 2:    if (tree->count!=0) Preorder(tree->head);
                   break;
        case 3:    if (tree->count!=0) Inorder(tree->head);
                   break;
        case 4:    if (tree->count!=0) Postorder(tree->head);
                   break;
        case 5:    printf("Feed in data to delete: ");
                   scanf("%d",&delKey);
                   DeleteNode(&(tree->head),delKey);
                   break;
    }
}while(choice !=6 );
}
```