PROLOG ACADEMY

- **DATA STRUCTURE**

By- MASEERA ALI

- Book followed - Data structures by Seymour Lipschutz (Schaum Series)

LET’S START!

Tree Traversal

Unlike linear data structures which have only one logical way to traverse them, trees can be traversed in different ways. Following are the generally used ways for traversing trees.

Depth First Traversals:

(a) Inorder

(b) Preorder

(c) Postorder

Breadth First or Level Order Traversal

Inorder Traversal

Algorithm Inorder(tree)

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)

```
void printInorder(struct node* node){
    if (node == NULL)
        return;
    printInorder(node->left);
    printf("%d ", node->data);
    printInorder(node->right);
}
```

Preorder Traversal

Algorithm Preorder(tree)

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)

```
void printPreorder(struct node* node){
     if (node == NULL)
          return;
      printf("%d ", node->data);
      printPreorder(node->left);
      printPreorder(node->right);
}
```

Post order

Algorithm Postorder(tree)

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.

```
void printPostorder(struct node* node){
     if (node == NULL)
        return;
      printPostorder(node->left);
      printPostorder(node->right);
      printf("%d ", node->data);
}
```

Inorder Tree Traversal without Recursion

1) Create an empty stack S.

2) Initialize current node as root

3) Push the current node to S and set current = current->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 = popped_item->right

    c) Go to step 3.

5) If current is NULL and stack is empty then we are done.

Code for BST

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

struct node
{
    int value;
    struct node *l;
    struct node *r;
}*root = NULL, *temp = NULL;

void inorder(struct node *t);
void preorder(struct node *t);
void postorder(struct node *t);
void search(struct node *t)
{
    if ((temp->value > t->value) && (t->r != NULL))
    search(t->r);
    else if ((temp->value > t->value) && (t->r == NULL))
    t->r = temp;
    else if ((temp->value < t->value) && (t->l != NULL))
    search(t->l);
    else if ((temp->value < t->value) && (t->l == NULL))
    t->l = temp;
}
```

```
void insert()
{

    int data,ch=1;
    while(ch==1){
    printf("Enter data: ");
    scanf("%d", &data);

    temp = (struct node *)malloc(sizeof(struct node));
    temp->value = data;
    temp->l = temp->r = NULL;
    if (root == NULL)
    root = temp;
    else
    search(root);

   printf("To continue press 1 else 0");
   scanf("%d",&ch);

        }
}
```

```
int main()
{   insert();
    int ch;
   printf("1 for Inorder Traversal\n");
   printf("2 for Preorder Traversal\n");
   printf("3 for Postorder Traversal\n");
   printf("4 for Exit\n");
    while(1)
    {
        printf("\nEnter your choice : ");
        scanf("%d", &ch);
        switch (ch)
        {
                case 1:
                inorder(root);
                break;
                case 2:
                preorder(root);
                break;
                case 3:
                 postorder(root);
                break;
                case 4:
                exit(0);
               }
    }
    return 0;
}
```

```
void inorder(struct node *root)
{
    if (root == NULL)
        return;
    inorder(root->l);
    printf("%d\t ", root->value);
    inorder(root->r);
}
```

```
void preorder(struct node *root)
{
    if (root == NULL)
        return;
    printf("%d\t ", root->value);
    preorder(root->l);
    preorder(root->r);
}
```

```
void postorder(struct node *root)
{
    if (root == NULL)
        return;
    postorder(root->l);
    postorder(root->r);
    printf("%d\t", root->value);
}
```

Search in BST

```
void searchElement(struct btnode *t)
{
  if(t->value==n)

  printf("Element Found\n");
  else if(n>t->value && t->r!=NULL)
  searchElement(t->r);
  else if(n>t->value && t->r==NULL)
  printf("Element Not Found\n");
  else if(n<t->value && t->l!=NULL)
  searchElement(t->l);
  else if(n<t->value && t->l==NULL)
  printf("ElementNot Found\n");

 }
```