

# Data Structure and Algorithm Lab Experiment Tree Traversal

---

Name: Om Ashish Mishra

Registration No.: 16BCE0789

Slot: G2

### The Psuedo code:

- First we ask the user for characters or numbers.
- Then we insert the elements into the binary search tree.
- Then we ask user for inorder, postorder or preorder.
- Then we print the elements of the desired code.

### The Code:

```
# include <stdio.h>
# include <conio.h>
# include <stdlib.h>
typedef struct BST {
int data;
struct BST *lchild, *rchild;
} node;
void insert(node *, node *);
void inorder(node *);
void preorder(node *);
void postorder(node *);
node *search(node *, int, node **);
typedef struct BST1 {
char data[100];
struct BST1 *lchild1, *rchild1;
} node1;
void insert1(node1 *, node1 *);
void inorder1(node1 *);
void preorder1(node1 *);
void postorder1(node1 *);
node1 *search1(node1 *, char, node1 **);
void main() {
int choice,ch;
char ans = 'N';
int key;
```

```

node *new_node, *root, *tmp, *parent;
node *get_node();
root = NULL;
int choice1;
char ans1 = 'N';
char key1[100];
node1 *new_node1, *root1, *tmp1, *parent1;
node1 *get_node1();
root1 = NULL;
//clrscr();
printf("\n****Program For Binary Search Tree**** ");
printf("\nEnter your choice : ");
printf("\n1. for numbers and 2. for characters : ");
scanf("%d",&ch);
if(ch==1)
{
do {
printf("\n1.Create");
printf("\n2.Search");
printf("\n3.Recursive Traversals");
printf("\n4.Exit");
printf("\nEnter your choice :");
scanf("%d", &choice);
switch (choice) {
case 1:
do {
new_node = get_node();
printf("\nEnter The Element ");
scanf("%d", &new_node->data);
if (root == NULL) /* Tree is not Created */
root = new_node;
else
insert(root, new_node);
printf("\nWant To enter More Elements?(y/n)");
ans = getch();
} while (ans == 'y');

```

```

break;
case 2:
printf("\nEnter Element to be searched :");
scanf("%d", &key);
tmp = search(root, key, &parent);
printf("\nParent of node %d is %d", tmp->data, parent->data);
break;
case 3:
if (root == NULL)
printf("Tree Is Not Created");
else {
printf("\nThe Inorder display : ");
inorder(root);
printf("\nThe Preorder display : ");
preorder(root);
printf("\nThe Postorder display : ");
postorder(root);
}
break;
}
} while (choice != 4);
}
else if(ch==2)
{
do {
printf("\n1.Create");
printf("\n2.Search");
printf("\n3.Recursive Traversals");
printf("\n4.Exit");
printf("\nEnter your choice :");
scanf("%d", &choice1);
switch (choice1) {
case 1:
do {
new_node1 = get_node1();
printf("\nEnter The Element ");

```

```

scanf("%s", &new_node1->data);
if (root1 == NULL) /* Tree is not Created */
root1 = new_node1;
else
insert(root1, new_node1);
printf("\nWant To enter More Elements?(y/n)");
ans1 = getch();
} while (ans1 == 'y');
break;
case 2:
printf("\nEnter Element to be searched :");
scanf("%s", &key1);
tmp1 = search1(root1, key1, &parent1);
printf("\nParent of node %d is %d", tmp1->data, parent1->data);
break;
case 3:
if (root1 == NULL)
printf("Tree Is Not Created");
else {
printf("\nThe Inorder display : ");
inorder1(root1);
printf("\nThe Preorder display : ");
preorder1(root1);
printf("\nThe Postorder display : ");
postorder1(root1);
}
break;
}
} while (choice1 != 4);
}
else
{
printf("Wrong choice!!");
}
}
}
/*

```

Get new Node

```
*/  
node *get_node() {  
    node *temp;  
    temp = (node *) malloc(sizeof(node));  
    temp->lchild = NULL;  
    temp->rchild = NULL;  
    return temp;  
}  
/*
```

This function is for creating a binary search tree

```
*/  
void insert(node *root, node *new_node) {  
    if (new_node->data < root->data) {  
        if (root->lchild == NULL)  
            root->lchild = new_node;  
        else  
            insert(root->lchild, new_node);  
    }  
    if (new_node->data > root->data) {  
        if (root->rchild == NULL)  
            root->rchild = new_node;  
        else  
            insert(root->rchild, new_node);  
    }  
}  
/*
```

This function is for searching the node from  
binary Search Tree

```
*/  
node *search(node *root, int key, node **parent) {  
    node *temp;  
    temp = root;  
    while (temp != NULL) {  
        if (temp->data == key) {  
            printf("\nThe %d Element is Present", temp->data);  
        }  
    }  
}
```

```

return temp;
}
*parent = temp;
if (temp->data > key)
temp = temp->lchild;
else
temp = temp->rchild;
}
return NULL;
}
/*

```

This function displays the tree in inorder fashion

```

*/
void inorder(node *temp) {
if (temp != NULL) {
inorder(temp->lchild);
printf("%d ", temp->data);
inorder(temp->rchild);
}
}
/*

```

This function displays the tree in preorder fashion

```

*/
void preorder(node *temp) {
if (temp != NULL) {
printf("%d ", temp->data);
preorder(temp->lchild);
preorder(temp->rchild);
}
}
/*

```

This function displays the tree in postorder fashion

```

*/
void postorder(node *temp) {
if (temp != NULL) {
postorder(temp->lchild);

```

```

postorder(temp->rchild);
printf("%d ", temp->data);
}
}

```

```

node1 *get_node1() {
node1 *temp1;
temp1 = (node1 *) malloc(sizeof(node1));
temp1->lchild1 = NULL;
temp1->rchild1 = NULL;
return temp1;
}
/*

```

This function is for creating a binary search tree  
\*/

```

void insert1(node1 *root1, node1 *new_node1) {
if (strcmp(new_node1->data,root1->data)<0) {
if (root1->lchild1 == NULL)
strcpy(root1->lchild1,new_node1);
else
insert(root1->lchild1, new_node1);
}
if (strcmp(new_node1->data,root1->data)>0) {
if (root1->rchild1 == NULL)
strcpy(root1->rchild1,new_node1);
else
insert(root1->rchild1, new_node1);
}
}
}
/*

```

This function is for searching the node from  
binary Search Tree  
\*/

```

node1 *search1(node1 *root1, char key1, node1 **parent1) {
node1 *temp1;
strcpy(temp1,root1);
while (temp1 != NULL) {

```



```

if (strcmp(temp1->data,key1)==0) {
printf("\nThe %d Element is Present", temp1->data);
return temp1;
}
strcpy(*parent1,temp1);
if (strcmp(temp1->data,key1)>0)
strcpy(temp1,temp1->lchild1);
else
strcpy(temp1,temp1->rchild1);
}
return NULL;
}
/*

```

This function displays the tree in inorder fashion

```

*/
void inorder1(node1 *temp1) {
if (temp1 != NULL) {
inorder1(temp1->lchild1);
printf("%s ", temp1->data);
inorder1(temp1->rchild1);
}
}
/*

```

This function displays the tree in preorder fashion

```

*/
void preorder1(node1 *temp1) {
if (temp1 != NULL) {
printf("%s ", temp1->data);
preorder1(temp1->lchild1);
preorder1(temp1->rchild1);
}
}
/*

```

This function displays the tree in postorder fashion

```

*/
void postorder1(node1 *temp1) {

```

```

if (temp1 != NULL) {
postorder1(temp1->lchild1);
postorder1(temp1->rchild1);
printf("%s ", temp1->data);
}
}

```

The Output:

For numbers:

```

****Program For Binary Search Tree****
Enter your choice :
1. for numbers and 2. for characters : 1

1.Create
2.Search
3.Recursive Traversals
4.Exit
Enter your choice :1

Enter The Element 23

Want To enter More Elements?(y/n)
Enter The Element 56

Want To enter More Elements?(y/n)
Enter The Element 78

Want To enter More Elements?(y/n)
Enter The Element 90

Want To enter More Elements?(y/n)
Enter The Element 39

Want To enter More Elements?(y/n)
1.Create
2.Search
3.Recursive Traversals
4.Exit
Enter your choice :3

The Inorder display : 23 39 56 78 90
The Preorder display : 23 56 39 78 90
The Postorder display : 39 90 78 56 23
1.Create
2.Search
3.Recursive Traversals
4.Exit
Enter your choice :4

Process returned 4 (0x4)   execution time : 39.749 s
Press any key to continue.

```

For characters:

```
****Program For Binary Search Tree****
Enter your choice :
1. for numbers and 2. for characters : 2

1.Create
2.Search
3.Recursive Traversals
4.Exit
Enter your choice :1

Enter The Element A

Want to enter More Elements?(y/n)
Enter The Element B

Want to enter More Elements?(y/n)
Enter The Element C

Want to enter More Elements?(y/n)
Enter The Element D

Want to enter More Elements?(y/n)
Enter The Element E

Want to enter More Elements?(y/n)
1.Create
2.Search
3.Recursive Traversals
4.Exit
Enter your choice :3

The Inorder display : A E B C D
The Preorder display : A B D E C
The Postorder display : D E B C A
1.Create
2.Search
3.Recursive Traversals
4.Exit
Enter your choice :4
Process returned 69 (0x45)    execution time : 0.170 s
Press any key to continue.
```

**THANK YOU**