

IBM19CS090

RATHIL

# Lab Program 10: BINARY SEARCH TREE.

Code:

```
#include <stdio.h>

struct node
{
    int info;
    struct node *rlink;
    struct node *llink;
};

typedef struct node *NODE;

NODE getnode()
{
    NODE x;
    x = (NODE) malloc (sizeof (struct node));
    if (x == NULL)
    {
        printf("Memory Full\n");
        exit(0);
    }
    return x;
}

void freenode (NODE x)
{
    free(x);
}

NODE insert (NODE root, int item)
{
    NODE temp, cur, prev;
    temp = getnode();
    temp->rlink = NULL;
    temp->llink = NULL;
    temp->info = item;
    if (root == NULL) return temp;
    prev = NULL;
    cur = root;
    while (cur != NULL)
```

```

{
    prev = cur;
    cur = (item < cur->info) ? cur->llink : cur->rlink;
}
if (item < prev->info)
    prev->llink = temp;
else
    prev->rlink = temp;
return root;
}

void display (NODE root, int i)
{
    int j;
    if (root != NULL)
    {
        display (root->rlink, i+1);
        for (j=0; j<i; j++)
            printf(" ");
        printf("%d\n", root->info);
        display (root->llink, i+1);
    }
}

void preorder (NODE root)
{
    if (root != NULL)
    {
        printf("%d\n", root->info);
        preorder (root->llink);
        preorder (root->rlink);
    }
}

void postorder (NODE root)
{
    if (root != NULL)
    {
        postorder (root->llink);
        postorder (root->rlink);
        printf("%d\n", root->info);
    }
}

```

```
void inorder(NODE root)
```

```
{
```

```
    if (root -> llink);
```

```
    printf("%d\n", root -> info);
```

```
    inorder (root -> rlink);
```

```
}
```

```
}
```

```
void main()
```

```
{
```

```
    int item, choice;
```

```
    NODE root = NULL;
```

```
    for (;;) 
```

```
    {
```

```
        printf("1. Insert 2. Display 3. Pre-Order 4. Post-Order  
              5. In-Order 6. Exit\n");
```

```
        printf("Enter choice : ");
```

```
        scanf("%d", &choice);
```

```
        switch (choice)
```

```
        {
```

```
            case 1: printf("Enter the item : ");
```

```
                    scanf("%d", &item);
```

```
                    root = insert (root, item);
```

```
                    break;
```

```
            case 2: display (root, 0);
```

```
                    break;
```

```
            case 3: preorder (root);
```

```
                    break;
```

```
            case 4: inorder (root);
```

```
                    break;
```

```
            case 5: Postorder (root);
```

```
                    break;
```

```
            default: exit (0);
```

```
                    break;
```

```
        }
```

```
    }
```

```
}
```

```

BinarySearchTree.c
Saved

1 #include<stdio.h>
2 struct node
3 {
4     int info;
5     struct node *rlink;
6     struct node *llink;
7 };
8 typedef struct node *NODE;
9 NODE getnode()
10 {
11     NODE x;
12     x=(NODE)malloc(sizeof(struct node));
13     if(x==NULL)
14     {
15         printf("Memory Full\n");
16         exit(0);
17     }
18     return x;
19 }
20 void freenode(NODE x)
21 {
22     free(x);
23 }
24 NODE insert(NODE root,int item)
25 {
26     NODE temp,cur,prev;
27     temp=getnode();
28     temp->rlink=NULL;
29     temp->llink=NULL;
30     temp->info=item;
31     if(root==NULL)
32         return temp;
33     prev=NULL;
34     cur=root;
35     while(cur!=NULL)
36     {
37         prev=cur;
38         cur=(item<cur->info)?cur->llink:cur->rlink;
39     }
40     if(item<prev->info)
41         prev->llink=temp;
42     else
43         prev->rlink=temp;
44     return root;
45 }
46 void display(NODE root,int i)
47 {
48     int j;
49     if(root!=NULL)
50     {
51         display(root->rlink,i+1);
52         for(j=0;j<i;j++)
53             printf(" ");
54         printf("%d\n",root->info);
55         display(root->llink,i+1);
56     }
57 }
58 void preorder(NODE root)
59 {
60     if(root!=NULL)
61     {
62         printf("%d\n",root->info);
63         preorder(root->llink);
64         preorder(root->rlink);
65     }
66 }
67 void postorder(NODE root)
68 {
69     if(root!=NULL)
70     {
71         postorder(root->llink);
72         postorder(root->rlink);
73         printf("%d\n",root->info);
74     }
75 }
76 void inorder(NODE root)
77 {
78     if(root!=NULL)
79     {
80         inorder(root->llink);
81         printf("%d\n",root->info);
82         inorder(root->rlink);
83     }
84 }
85 void main()
86 {
87     int item,choice;
88     NODE root=NULL;
89     for(;;)
90     {
91         printf("\n1.Insert\n2.Display\n3.Pre-Order\n4.In-Order\n5.Post-Order\n6.Exit\n");
92         printf("Enter Choice:");
93         scanf("%d",&choice);
94         switch(choice)
95         {
96             case 1:
97                 printf("Enter Item:");
98                 scanf("%d",&item);
99                 root=insert(root,item);
100                break;
101                case 2:
102                    display(root,0);
103                    break;
104                    case 3:
105                        preorder(root);
106                        break;
107                        case 4:
108                            inorder(root);
109                            break;
110                            case 5:
111                                postorder(root);
112                                break;
113                                default:
114                                    exit(0);
115                                    break;
116                            }
117            }
118 }

```

```
Terminal
1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:1
Enter Item:100

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:1
Enter Item:20

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:1
Enter Item:10

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:1
Enter Item:30

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:1
Enter Item:200

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:1
Enter Item:150

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:1
Enter Item:300

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:3
100
20
10
30
200
150
300

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:4
10
20
30
100
150
200
300

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:5
10
30
20
150
300
200
100

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:6
Process finished.
```