

Ex. No. 10	IMPLEMENTATION OF BINARY SEARCH TREE
01-10-2017	

Question:

To implement the working of implementation of binary search tree.

Algorithm:

1. Start.
2. Create struct treenode and declare attributes treenode *left, int data, treenode *right and initialize as *head as null.
3. Create function insert2(), set, if(temp→data<monk→data) and if (monk→left==null), store the temp at left.
4. Else insert2(temp, monk→left)
5. Set, else if (temp→data<monk→data)
6. set, if (monk→right=null), store the temp at right.
7. Else insert2(temp, monk->right)
8. create another insert() function. In that, create a temp pointer of treenode using new
9. set left & right pointer null
10. insert data at info.
11. Set if head==null, then head =temp
12. else insert2(temp,monk).
13. Create show function, preorder(), print in the order data, left & right and for inorder(), print in the order left, data & right and postorder(), print in the order left , right & data.

14. In the main() function, create switch case to invoke the required functions.
15. Perform the necessary operations.
16. Display the result.
17. End.

Program:

//IMPLEMENTATION OF BINARY SEARCH TREE

```
#include <iostream>

using namespace std;

struct treenode
{
    treenode *left;
    int data;
    treenode *right;
}*head=NULL;

void Insert2(treenode *temp, treenode *monk)
{
    if(temp->data<monk->data)
    {
        if(monk->left==NULL)
        {
            monk->left=temp;
```

```
        }
        else
        {
            Insert2(temp,monk->left);
        }
    else if(temp->data>monk->data)
    {
        if(monk->right==NULL)
        {
            monk->right=temp;
        }
        else
        {
            Insert2(temp,monk->right);
        }
    }
}
```

```
void insert(int item)
{
    treenode *temp;
    temp=new treenode;
```

```
temp->left=NULL;
temp->right=NULL;
temp->data=item;
if(head==NULL)
{
    head=temp;
}
else
{
    treenode *monk=head;
    Insert2(temp,monk);
}
}

void inorder(treenode *monk)
{
    if(monk!=NULL)
    {
        inorder(monk->left);
        cout<<" "<<monk->data<<" ";
        inorder(monk->right);
    }
    return;
}
```

```
}  
  
void preorder(treenode *monk)  
{  
    if(monk!=NULL)  
    {  
        cout<<" "<<monk->data<<" ";  
        preorder(monk->left);  
        preorder(monk->right);  
    }  
    return;  
}  
  
void postorder(treenode *monk)  
{  
    if(monk!=NULL)  
    {  
        postorder(monk->left);  
        postorder(monk->right);  
        cout<<" "<<monk->data<<" ";  
    }  
    return;  
}  
  
int main()
```

```
{
    int x,n,d;
    while(1)
    {
prev:
        cout<<"\n To do.....";
        cout<<"\n 1.Insert\n 2.Show\n 3.Exit    ";
        cout<<"\n\n Enter your choice: ";
        cin>>n;
        switch(n)
        {
case 1:      cout<<"\n Enter the data: ";
                cin>>d;
                insert(d);
                break;
case 2:      while(1)
                {
                    cout<<"\n\n To do.....";
                    cout<<"\n 1.Pre order\n 2.In order\n 3.Post order \n 4.Return    ";
                    cout<<"\n\n Enter your  choice: ";
                    cin>>x;
                    switch(x)
```

```
{
case 1:
cout<<"\n Preorder Traversal:\n";
preorder(head);
    break;
case 2:
    cout<<"\n Inorder Traversal:\n";
    inorder(head);
    break;
case 3:
    cout<<"\n Postorder Traversal:\n";
    postorder(head);
    break;
case 4:
    goto prev;
default:
    cout<<"\n Invalid order"<<endl;
        }
    }

    break;
case 3:
return 0;
```

DATA STRUCTURES LAB

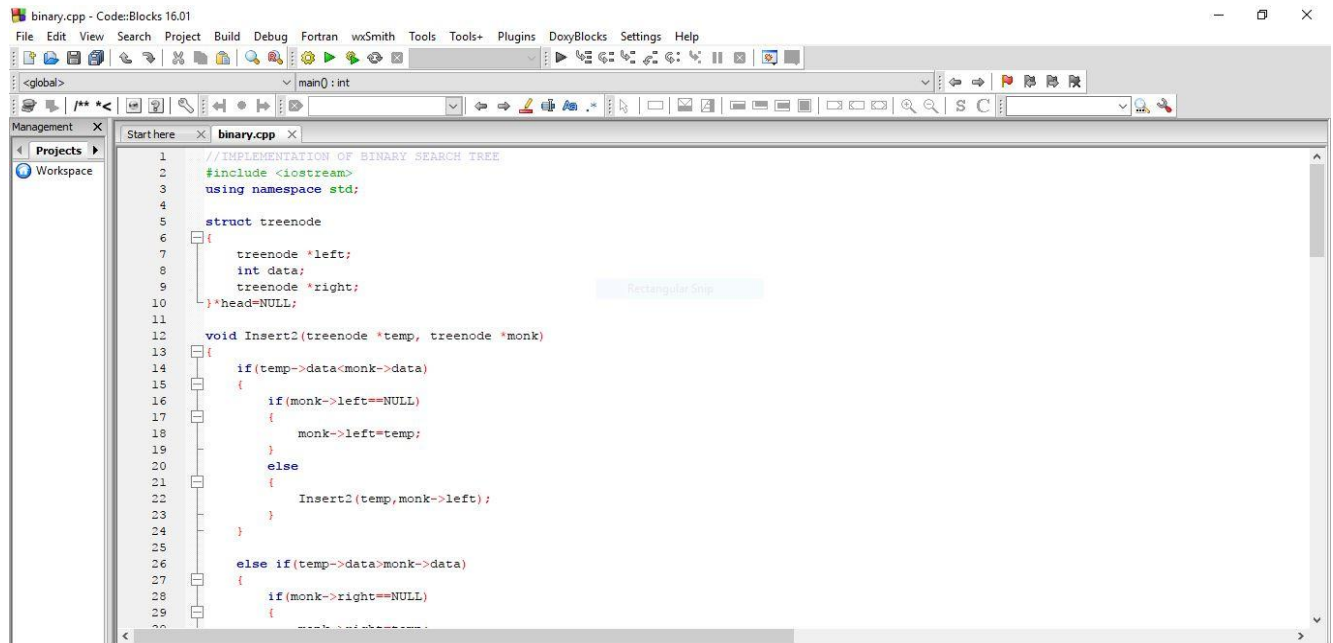
default: cout<<"Invalid order"<<endl;

}

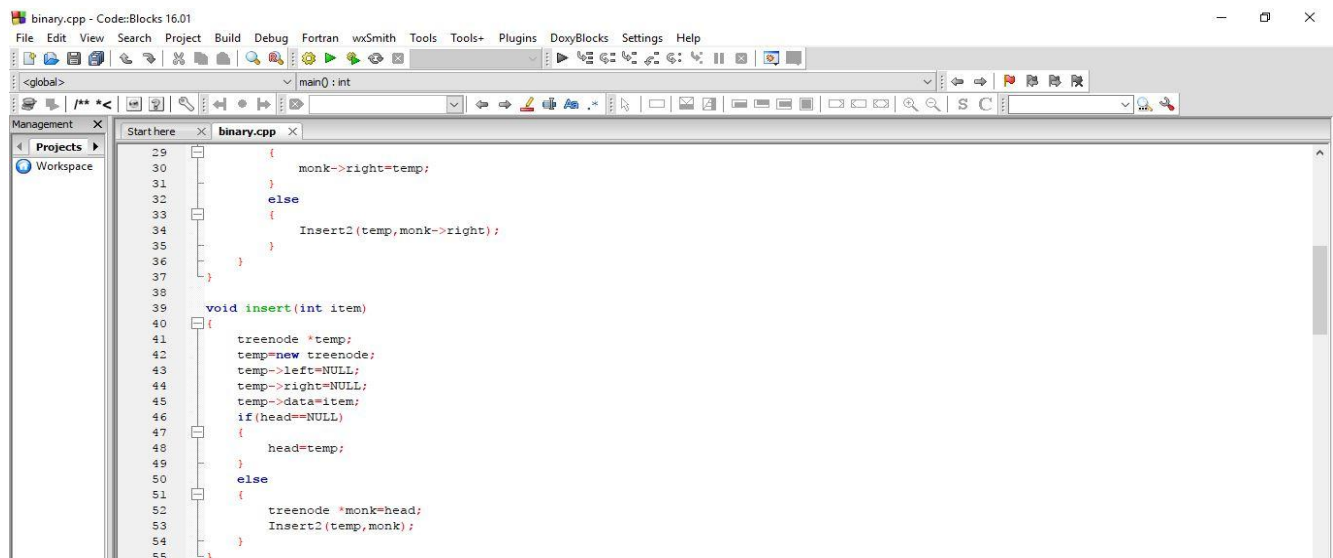
}

}

Output:



```
1 //IMPLEMENTATION OF BINARY SEARCH TREE
2 #include <iostream>
3 using namespace std;
4
5 struct treenode
6 {
7     treenode *left;
8     int data;
9     treenode *right;
10 }*head=NULL;
11
12 void Insert2(treenode *temp, treenode *monk)
13 {
14     if (temp->data<monk->data)
15     {
16         if (monk->left==NULL)
17         {
18             monk->left=temp;
19         }
20         else
21         {
22             Insert2(temp,monk->left);
23         }
24     }
25
26     else if (temp->data>monk->data)
27     {
28         if (monk->right==NULL)
29         {
30             monk->right=temp;
31         }
32         else
33         {
34             Insert2(temp,monk->right);
35         }
36     }
37 }
38
39 void insert(int item)
40 {
41     treenode *temp;
42     temp=new treenode;
43     temp->left=NULL;
44     temp->right=NULL;
45     temp->data=item;
46     if (head==NULL)
47     {
48         head=temp;
49     }
50     else
51     {
52         treenode *monk=head;
53         Insert2(temp,monk);
54     }
55 }
```



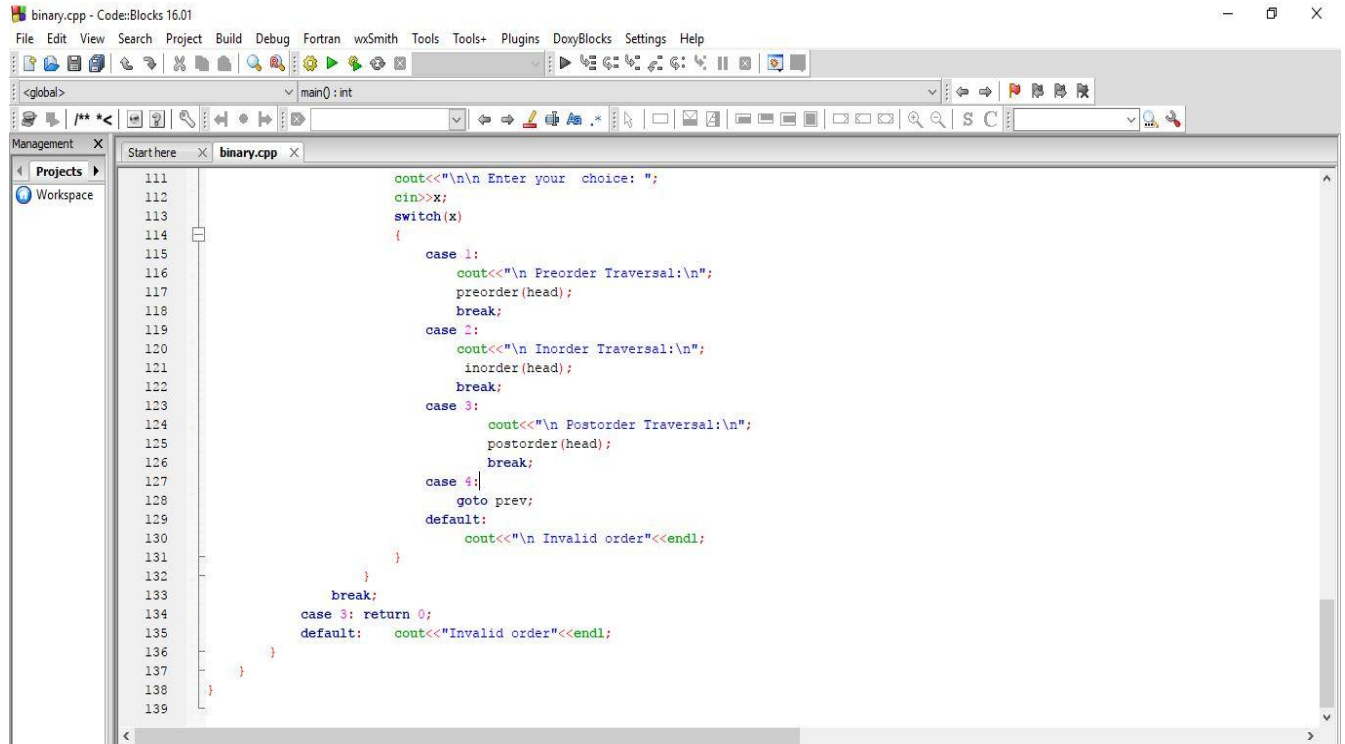
```
29     {
30         monk->right=temp;
31     }
32     else
33     {
34         Insert2(temp,monk->right);
35     }
36 }
37
38 void insert(int item)
39 {
40     treenode *temp;
41     temp=new treenode;
42     temp->left=NULL;
43     temp->right=NULL;
44     temp->data=item;
45     if (head==NULL)
46     {
47         head=temp;
48     }
49     else
50     {
51         treenode *monk=head;
52         Insert2(temp,monk);
53     }
54 }
55 }
```


DATA STRUCTURES LAB


```
binary.cpp - Code::Blocks 16.01
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global> main(): int
Start here binary.cpp
58 void inorder(treenode *monk)
59 {
60     if(monk!=NULL)
61     {
62         inorder(monk->left);
63         cout<<" "<<monk->data<<" ";
64         inorder(monk->right);
65     }
66     return;
67 }
68
69 void preorder(treenode *monk)
70 {
71     if(monk!=NULL)
72     {
73         cout<<" "<<monk->data<<" ";
74         preorder(monk->left);
75         preorder(monk->right);
76     }
77     return;
78 }
79
80 void postorder(treenode *monk)
81 {
82     if(monk!=NULL)
83     {
84         postorder(monk->left);
85         postorder(monk->right);
86         cout<<" "<<monk->data<<" ";
87     }
88     return;
89 }
90
91 int main()
92 {
93     int x,n,d;
94     while(1)
95     {
96         prev:
97         cout<<"\n To do.....";
98         cout<<"\n 1.Insert\n 2.Show\n 3.Exit  ";
99         cout<<"\n\n Enter your choice: ";
100         cin>>n;
101         switch(n)
102         {
103             case 1: cout<<"\n Enter the data: ";
104                     cin>>d;
105                     insert(d);
106                     break;
107             case 2: while(1)
108                     {
109                         cout<<"\n\n To do.....";
110                         cout<<"\n 1.Pre order\n 2.In order\n 3.Post order \n 4.Return  ";
111                         cout<<"\n\n Enter your choice: ";
112                         cin>>x;
113                         switch(x)
114                         {
115                             case 1:
116                                 preorder(monk->left);
117                                 preorder(monk->right);
118                                 cout<<" "<<monk->data<<" ";
119                                 return;
120                             case 2:
121                                 inorder(monk->left);
122                                 inorder(monk->right);
123                                 cout<<" "<<monk->data<<" ";
124                                 return;
125                             case 3:
126                                 postorder(monk->left);
127                                 postorder(monk->right);
128                                 cout<<" "<<monk->data<<" ";
129                                 return;
130                             case 4:
131                                 return;
132                             default:
133                                 continue;
134                         }
135                     }
136             default:
137                 return;
138         }
139     }
140 }
```

```
binary.cpp - Code::Blocks 16.01
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global> main(): int
Start here binary.cpp
87     }
88     return;
89 }
90
91 int main()
92 {
93     int x,n,d;
94     while(1)
95     {
96         prev:
97         cout<<"\n To do.....";
98         cout<<"\n 1.Insert\n 2.Show\n 3.Exit  ";
99         cout<<"\n\n Enter your choice: ";
100         cin>>n;
101         switch(n)
102         {
103             case 1: cout<<"\n Enter the data: ";
104                     cin>>d;
105                     insert(d);
106                     break;
107             case 2: while(1)
108                     {
109                         cout<<"\n\n To do.....";
110                         cout<<"\n 1.Pre order\n 2.In order\n 3.Post order \n 4.Return  ";
111                         cout<<"\n\n Enter your choice: ";
112                         cin>>x;
113                         switch(x)
114                         {
115                             case 1:
116                                 preorder(monk->left);
117                                 preorder(monk->right);
118                                 cout<<" "<<monk->data<<" ";
119                                 return;
120                             case 2:
121                                 inorder(monk->left);
122                                 inorder(monk->right);
123                                 cout<<" "<<monk->data<<" ";
124                                 return;
125                             case 3:
126                                 postorder(monk->left);
127                                 postorder(monk->right);
128                                 cout<<" "<<monk->data<<" ";
129                                 return;
130                             case 4:
131                                 return;
132                             default:
133                                 continue;
134                         }
135                     }
136             default:
137                 return;
138         }
139     }
140 }
```

DATA STRUCTURES LAB



```
binary.cpp - Code::Blocks 16.01
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global> main0: int
Management Projects Workspace
111 cout<<"\n\n Enter your choice: ";
112 cin>>x;
113 switch(x)
114 {
115     case 1:
116         cout<<"\n Preorder Traversal:\n";
117         preorder(head);
118         break;
119     case 2:
120         cout<<"\n Inorder Traversal:\n";
121         inorder(head);
122         break;
123     case 3:
124         cout<<"\n Postorder Traversal:\n";
125         postorder(head);
126         break;
127     case 4:
128         goto prev;
129     default:
130         cout<<"\n Invalid order"<<endl;
131 }
132 }
133 break;
134 case 3: return 0;
135 default: cout<<"Invalid order"<<endl;
136 }
137 }
138 }
139 }
```



```
"C:\Users\JUSTIN PAUL\Desktop\binary.exe"
To do.....
1.Insert
2.Show
3.Exit
Enter your choice: 1
Enter the data: 10
To do.....
1.Insert
2.Show
3.Exit
Enter your choice: 1
Enter the data: 20
To do.....
1.Insert
2.Show
3.Exit
Enter your choice: 1
Enter the data: 33
To do.....
1.Insert
2.Show
3.Exit
Enter your choice: 1
Enter the data: 43
```

DATA STRUCTURES LAB

```
"C:\Users\JUSTIN PAUL\Desktop\binary.exe"
To do.....
1.Insert
2.Show
3.Exit

Enter your choice: 2

To do.....
1.Pre order
2.In order
3.Post order
4.Return

Enter your choice: 1

Preorder Traversal:
10 20 33 43

To do.....
1.Pre order
2.In order
3.Post order
4.Return

Enter your choice:
2

Inorder Traversal:
10 20 33 43

To do.....
1.Pre order
2.In order
3.Post order
4.Return

Enter your choice: 3

Postorder Traversal:
43 33 20 10
```

VIDEO URL:

<https://youtu.be/AftsqAUyA2Q>

RESULT:

The program of implementation of binary search tree is implemented successfully and the output is verified.