

Assignment6.cpp

```
1  #include <iostream>
2  #include <cstdlib>
3  #define MAX_VALUE 65536
4  using namespace std;
5
6
7  struct N {    //node declaration
8      int k;
9      N *l, *r;
10     bool leftTh, rightTh;
11 };
12
13 //global root
14 N *root;
15
16 //creating the inorder threaded BT
17 void insert(int key)
18 {
19     N *ptr = NULL;
20     if(root==NULL)
21     {
22         root= new N();
23         root->r= root->l= root;
24         root->leftTh = true;
25         root->k = MAX_VALUE;
26     }
27     /*else
28     {
29         ptr= new N();
30         ptr->r= root->l= root;
31         ptr->leftTh = true;
32         ptr->k = MAX_VALUE;
33     }*/
34     N *p = root;
35     for (;;) {
36         if (p->k< key) { //move to right thread
37             if (p->rightTh)
38                 break;
39             p = p->r;
40         } else if (p->k > key) { // move to left thread
41             if (p->leftTh)
42                 break;
43             p = p->l;
44         } else {
45             return;
46         }
47     }
48     N *temp = new N();
49     temp->k = key;
50     temp->rightTh= temp->leftTh= true;
```

```
51     if (p->k < key) {
52         temp->r = p->r;
53         temp->l = p;
54         p->r = temp;
55         p->rightTh = false;
56     } else {
57         temp->r = p;
58         temp->l = p->l;
59         p->l = temp;
60         p->leftTh = false;
61     }
62 }
63
64
65 void displayTree() { //print the tree
66     N *temp = root, *p;
67     for (;;) {
68         p = temp;
69         temp = temp->r;
70         if (!p->rightTh) {
71             while (!temp->leftTh) {
72                 temp = temp->l;
73             }
74         }
75         if (temp == root)
76             break;
77         cout<<temp->k<<" ";
78     }
79     cout<<endl;
80 }
81
82 void preorderDisplayTree(){
83     N *temp = root, *p;
84     for (;;) {
85         p = temp;
86
87         // Visit the current node (pre-order)
88         if (temp != root) {
89             cout << temp->k << " ";
90         }
91
92         // If there is a left child, move to it
93         if (!temp->leftTh) {
94             temp = temp->l;
95         }
96         // Else move to the right threaded node
97         else {
98             while (temp->rightTh && temp->r != root) {
99                 temp = temp->r;
100             }
101             temp = temp->r;
102             if (temp == root) {
103                 break;
```

```
104     }
105 }
106 }
107 cout<<endl;
108 }
109
110
111 int main() {
112
113     cout<<"Inorder ThreadedBinaryTree\n";
114     char ch;
115     int c, v;
116     while(1) {
117         cout<<"1. Insert "<<endl;
118         cout<<"2. Inorder Traversal"<<endl;
119         cout<<"3. Preorder Traversal"<<endl;
120         cout<<"6. Exit"<<endl;
121         cout<<"Enter Your Choice: ";
122         cin>>c;
123         //perform switch operation
124         switch (c) {
125             case 1 :
126                 cout<<"Enter integer element to insert: ";
127                 cin>>v;
128                 insert(v);
129                 break;
130             case 2:
131                 cout<<"In-order Display tree: \n ";
132                 displayTree();
133                 break;
134             case 3:
135                 cout<<"Pre-order Display tree: \n";
136                 preorderDisplayTree();
137                 cout<<endl;
138                 break;
139             case 6:
140                 exit(1);
141             default:
142                 cout<<"\nInvalid type! \n";
143         }
144     }
145     cout<<"\n";
146     return 0;
147 }
148 /*
149 Sample output
150 Inorder ThreadedBinaryTree
151 1. Insert
152 2. Inorder Traversal
153 3. Preorder Traversal
154 6. Exit
155 Enter Your Choice: 1
156 Enter integer element to insert: 50
```

```
157 1. Insert
158 2. Inorder Traversal
159 3. Preorder Traversal
160 6. Exit
161 Enter Your Choice: 1
162 Enter integer element to insert: 25
163 1. Insert
164 2. Inorder Traversal
165 3. Preorder Traversal
166 6. Exit
167 Enter Your Choice: 1
168 Enter integer element to insert: 75
169 1. Insert
170 2. Inorder Traversal
171 3. Preorder Traversal
172 6. Exit
173 Enter Your Choice: 1
174 Enter integer element to insert: 20
175 1. Insert
176 2. Inorder Traversal
177 3. Preorder Traversal
178 6. Exit
179 Enter Your Choice: 1
180 Enter integer element to insert: 40
181 1. Insert
182 2. Inorder Traversal
183 3. Preorder Traversal
184 6. Exit
185 Enter Your Choice: 1
186 Enter integer element to insert: 60
187 1. Insert
188 2. Inorder Traversal
189 3. Preorder Traversal
190 6. Exit
191 Enter Your Choice: 1
192 Enter integer element to insert: 80
193 1. Insert
194 2. Inorder Traversal
195 3. Preorder Traversal
196 6. Exit
197 Enter Your Choice: 3
198 Pre-order Display tree:
199 50 25 20 40 75 60 80
200
201 1. Insert
202 2. Inorder Traversal
203 3. Preorder Traversal
204 6. Exit
205 Enter Your Choice: 1
206 Enter integer element to insert: 85
207 1. Insert
208 2. Inorder Traversal
209 3. Preorder Traversal
```

```
210 | 6. Exit
211 | Enter Your Choice: 2
212 | In-order Display tree:
213 | 20 25 40 50 60 75 80 85
214 | 1. Insert
215 | 2. Inorder Traversal
216 | 3. Preorder Traversal
217 | 6. Exit
218 | Enter Your Choice: 6
219 | */
```