

Nama kelompok:

- Muhammad Zidan Alif Oktavian (045)
- Dika Wahyu Nuralixsyah (029)
- Fahreza Dzaky Rahmatullah (061)
- Ariel Fikri Ramadhani (067)
- Bariq Adyatma (019)

Coding:

```
1 #include<iostream>
2 #define SPACE 10
3
4 using namespace std;
5 //mendeklarasikan class
6 class TreeNode {
7 public:
8     int value;
9     TreeNode * left;
10    TreeNode * right;
11
12    TreeNode() {
13        value = 0;
14        left = NULL;
15        right = NULL;
16    }
17    TreeNode(int v) {
18        value = v;
19        left = NULL;
20        right = NULL;
21    }
22 };
23
24 class BST {
25 public:
26     TreeNode * root;
27     BST() {
28         root = NULL;
29     }
30     bool isTreeEmpty() {
31         if (root == NULL) {
32             return true;
33         } else {
34             return false;
35         }
36     }
37
38     // fungsi untuk menambahkan nilai ke tree
39     void insertNode(TreeNode * new_node) {
40         if (root == NULL) {
41             root = new_node;
42             cout << "Value Inserted as root node!" << endl;
43         } else {
44             TreeNode * temp = root;
45             while (temp != NULL) {
46                 if (new_node->value == temp->value) {
47                     cout << "Nilai telah ada," <<
48                         "masukkan nilai yang lain" << endl;
49                     return;
50                 } else if ((new_node->value < temp->value) && (temp->left == NULL)) {
51                     temp->left = new_node;
52                     cout << "nilai dimasukkan ke kiri" << endl;
53                     break;
54                 } else if (new_node->value < temp->value) {
55                     temp = temp->left;
56                 } else if ((new_node->value > temp->value) && (temp->right == NULL)) {
57                     temp->right = new_node;
58                     cout << "nilai dimasukkan ke kanan" << endl;
59                     break;
60                 } else {
61                     temp = temp->right;
62                 }
63             }
64         }
65     }
66     TreeNode* insertRecursive(TreeNode *r, TreeNode *new_node)
67     {
68         if(r==NULL)
69         {
70             r=new_node;
71             cout <<"sukses memasukkan nilai"<<endl;
72             return r;
73         }
74     }
75 }
```

```

73     if(new_node->value < r->value)
74     {
75         r->left = insertRecursive(r->left,new_node);
76     }
77     else if (new_node->value > r->value)
78     {
79         r->right = insertRecursive(r->right,new_node);
80     }
81     else
82     {
83         cout << "tidak boleh ada nilai yang sama" << endl;
84         return r;
85     }
86     return r;
87 }
88
89 void print2D(TreeNode * r, int space) {
90     if (r == NULL) // Base case
91     {
92         return;
93     }
94     space += SPACE; // meningkatkan jarak diantara angka
95     print2D(r->right, space); // memproses children atau subtree disebelah kanan terlebih dahulu
96     cout << endl;
97     for (int i = SPACE; i < space; i++)
98     {
99         cout << " ";
100     }
101     cout << r->value << "\n";
102     print2D(r->left, space); // memproses children disebelah kiri
103 }
104
105 void printPreorder(TreeNode * r) //mengurutkan dari(nilai sekarang, kiri, kanan)
106 {
107     if (r == NULL)
108     {
109         return;
110     }
111     /* print nilai pertama terlebih dahulu */
112     cout << r->value << " ";
113     /* rekursif dari kiri */
114     printPreorder(r->left);
115     /* Lalu dilanjutkan rekursif dari kanan */
116     printPreorder(r->right);
117 }
118
119 void printInorder(TreeNode * r) //mengurutkan dari (kiri, nilai sekarang, kanan)
120 {
121     if (r == NULL)
122     {
123         return;
124     }
125     /* rekursif dari kiri */
126     printInorder(r->left);
127     /* print data yang sekarang */
128     cout << r->value << " ";
129     /* rekursif dari kanan */
130     printInorder(r->right);
131 }
132
133 void printPostorder(TreeNode * r) //mengurutkan dari (kiri, kanan, Root)
134 {
135     if (r == NULL)
136     {
137         return;
138     }
139     // rekursif dari kiri
140     printPostorder(r->left);
141     // rekursif dari kanan
142     printPostorder(r->right);
143     // print data yang sekarang
144     cout << r->value << " ";
145 }
146
147 TreeNode * iterativeSearch(int v) {
148     if (root == NULL) {
149         return root;
150     } else {
151         TreeNode * temp = root;
152         while (temp != NULL) {
153             if (v == temp->value) {
154                 return temp;
155             } else if (v < temp->value) {
156                 temp = temp->left;
157             } else {
158                 temp = temp->right;
159             }
160         }
161         return NULL;
162     }
163 }
164
165 TreeNode * recursiveSearch(TreeNode * r, int val) {
166     if (r == NULL || r->value == val) {
167         return r;
168     }
169     else if (val < r->value)
170     {
171         return recursiveSearch(r->left, val);
172     }
173     else
174     {
175         return recursiveSearch(r->right, val);
176     }
177 }
178
179 int height(TreeNode * r) {
180     if (r == NULL)
181     {
182         return -1;
183     }
184     else {
185         /* menghitung tinggi setiap subtree */
186         int lheight = height(r->left);
187         int rheight = height(r->right);
188         /* gunakan yang terbesar */
189         if (lheight > rheight)
190             return (lheight + 1);
191         else return (rheight + 1);
192     }
193 }

```

```

181  /* Print angka/nilai pada tempat yang telah ditentukan*/
182  void printGivenLevel(TreeNode * r, int level) {
183      if (r == NULL)
184          return;
185      else if (level == 0)
186          cout << r -> value << " ";
187      else level > 0;
188      {
189          printGivenLevel(r -> left, level - 1);
190          printGivenLevel(r -> right, level - 1);
191      }
192  }
193  void printLevelOrderBFS(TreeNode * r) {
194      int h = height(r);
195      for (int i = 0; i <= h; i++)
196          printGivenLevel(r, i);
197  }
198  TreeNode * minValueNode(TreeNode * node) {
199      TreeNode * current = node;
200      /* Loop down to find the leftmost leaf */
201      while (current -> left != NULL) {
202          current = current -> left;
203      }
204      return current;
205  }
206  }
207
208
209
210 }
211
212 int main() {
213     BST obj;
214     int option, val;
215
216     do {
217         cout << "What operation do you want to perform? " <<
218             " Select Option number. Enter 0 to exit." << endl;
219         cout << "1. Insert Node" << endl;
220         cout << "2. Print/Traversal BST values" << endl;
221         cout << "0. Exit Program" << endl;
222
223         cin >> option;
224         TreeNode n1;
225         TreeNode * new_node = new TreeNode();
226
227         switch (option) {
228             case 0:
229                 break;
230             case 1:
231                 cout << "INSERT" << endl;
232                 cout << "Masukkan nilai satu persatu: ";
233                 cin >> val;
234                 new_node->value = val;
235                 obj.root = obj.insertRecursive(obj.root, new_node);
236                 obj.insertNode(new_node);
237                 cout << endl;
238                 break;
239
240             case 2:
241                 cout << "PRINT 2D: " << endl;
242                 obj.print2D(obj.root, 5);
243                 cout << endl;
244                 cout << "Print Level Order BFS: \n";
245                 obj.printLevelOrderBFS(obj.root);
246                 cout << endl;
247                 cout << "PRE-ORDER: ";
248                 obj.printPreorder(obj.root);
249                 cout << endl;
250                 cout << "IN-ORDER: ";
251                 obj.printInorder(obj.root);
252                 cout << endl;
253                 cout << "POST-ORDER: ";
254                 obj.printPostorder(obj.root);
255                 break;
256             }
257
258     } while (option != 0);
259
260     return 0;
261 }
262
263 }

```

input

```

INSERT
Masukkan nilai satu persatu: 1
sukses memasukkan nilai
Nilai telah ada, masukkan nilai yang lain

```

```
What operation do you want to perform? Select Option number. Enter 0 to exit.
1. Insert Node
2. Print/Traversal BST values
0. Exit Program
1
INSERT
Masukkan nilai satu persatu: 2
sukses memasukkan nilai
Nilai telah ada,masukkan nilai yang lain
```

```
What operation do you want to perform? Select Option number. Enter 0 to exit.
1. Insert Node
2. Print/Traversal BST values
0. Exit Program
1
INSERT
Masukkan nilai satu persatu: 6
sukses memasukkan nilai
Nilai telah ada,masukkan nilai yang lain
```

output

```
What operation do you want to perform? Select Option number. Enter 0 to exit.
1. Insert Node
2. Print/Traversal BST values
0. Exit Program
2
PRINT 2D:

          6
        2
      1

Print Level Order BFS:
1 2 6
PRE-ORDER: 1 2 6
IN-ORDER: 1 2 6
POST-ORDER: 6 2 1 What operation do you want to perform? Select Option number. Enter 0 to exit.
```