

LAPORAN BINARY SEARCH TREE KELOMPOK 4

Oleh:

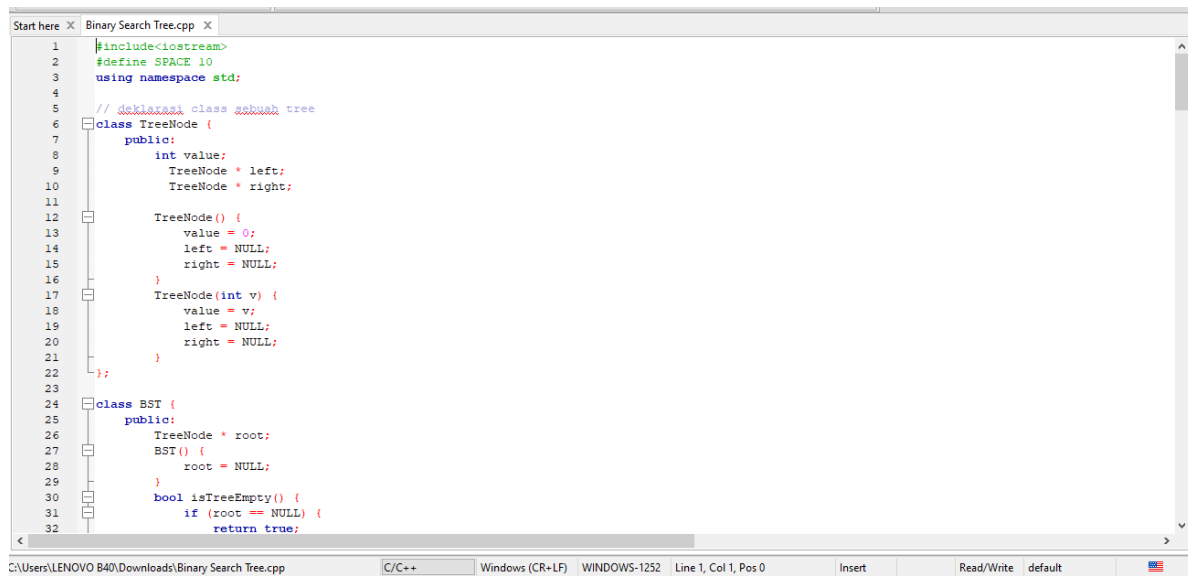
Salsabilla Octavianingrum (21091397005)

Muhammad Yuda Pratama (21091397025)

Azkia Kamila (21091397027)

Rafie Mirza Ramadhan (21091397037)

Refila Dyah Ghizanda Wardoyo (21091397041)



```
1  #include<iostream>
2  #define SPACE 10
3  using namespace std;
4
5  // class class abuhah tree
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         }
34     }
35 }
```

C:\Users\LENOVO B40\Downloads\Binary Search Tree.cpp C/C++ Windows (CR+LF) WINDOWS-1252 Line 1, Col 1, Pos 0 Insert Read/Write default

```
Start here X Binary Search Tree.cpp X
32         return true;
33     }
34     else {
35         return false;
36     }
37 }
38
39 // fungsi untuk menanamkan node baru
40 void insertNode(TreeNode * new_node) {
41     // jika root masih kosong
42     if (root == NULL) {
43         // mengalokasikan memori dari node yang telah dibuat
44         root = new_node;
45         cout << "Value Inserted as root node!" << endl;
46     } else {
47         TreeNode * temp = root;
48         while (temp != NULL) {
49             if (new_node->value == temp->value) {
50                 cout << "Value Already exist," <<
51                     "Insert another value!" << endl;
52                 return;
53             } else if ((new_node->value < temp->value) && (temp->left == NULL)) {
54                 temp->left = new_node;
55                 cout << "Value Inserted to the left!" << endl;
56                 break;
57             } else if (new_node->value < temp->value) {
58                 temp = temp->left;
59             } else if ((new_node->value > temp->value) && (temp->right == NULL)) {
60                 temp->right = new_node;
61                 cout << "Value Inserted to the right!" << endl;
62                 break;
63             } else {

```

```
Start here X Binary Search Tree.cpp X
63         } else {
64             temp = temp->right;
65         }
66     }
67 }
68
69 }
70
71 TreeNode* insertRecursive(TreeNode *r, TreeNode *new_node){
72     if(r==NULL){
73         r=new_node;
74         cout <<"Insertion successful"<<endl;
75         return r;
76     }if(new_node->value < r->value){
77         r->left = insertRecursive(r->left,new_node);
78     }else if (new_node->value > r->value){
79         r->right = insertRecursive(r->right,new_node);
80     }else{
81         cout << "No duplicate values allowed!" << endl;
82         return r;
83     }
84     return r;
85 }
86
87 void print2D(TreeNode * r, int space) {
88     // base case 1
89     if (r == NULL)
90         return;
91     // menambahkan jarak antara level 2
92     space += SPACE;
93     // proses anak kanan dulu 3
94     print2D(r->right, space);
95     cout << endl;
96     for (int i = SPACE; i < space; i++)

```

```
Start here x Binary Search Tree.cpp x
94     for (int i = SPACE; i < space; i++)
95         cout << " ";
96     cout << r -> value << "\n";
97     // proses anak kiri 7
98     print2D(r -> left, space);
99 }
100
101 // simbul saat ini, kiri, kanan
102 void printPreorder(TreeNode * r){
103     if (r == NULL)
104         return;
105     cout << r -> value << " "; // mencetak data simbul pertama
106     printPreorder(r -> left); // kemudian muncul kembali di submohon kiri
107     printPreorder(r -> right); // sekarang muncul kembali di submohon kanan
108 }
109
110 // kiri, simbul terakhir, kanan
111 void printInorder(TreeNode * r)
112 {
113     if (r == NULL)
114         return;
115     // pertama muncul kembali di anak kiri
116     printInorder(r -> left);
117     // kemudian cetak data di simbul
118     cout << r -> value << " ";
119     // sekarang muncul kembali di anak kanan
120     printInorder(r -> right);
121 }
122
123 // kiri, kanan, akar
124 void printPostorder(TreeNode * r)
125 {
```

C:\Users\LENOVO B40\Downloads\Binary Search Tree.cpp C/C++ Windows (CR+LF) WINDOWS-1252 Line 94, Col 44, Pos 2475 Insert Read/Write default

```
Start here x Binary Search Tree.cpp x
125 {
126     if (r == NULL)
127         return;
128     // pertama muncul kembali di submohon kiri
129     printPostorder(r -> left);
130     // kemudian muncul kembali di submohon kanan
131     printPostorder(r -> right);
132     // sekarang cetak dengan simbul
133     cout << r -> value << " ";
134 }
135
136 TreeNode * iterativeSearch(int v) {
137     if (root == NULL) {
138         return root;
139     } else {
140         TreeNode * temp = root;
141         while (temp != NULL) {
142             if (v == temp -> value) {
143                 return temp;
144             } else if (v < temp -> value) {
145                 temp = temp -> left;
146             } else {
147                 temp = temp -> right;
148             }
149         }
150         return NULL;
151     }
152 }
153
154 TreeNode * recursiveSearch(TreeNode * r, int val) {
155     if (r == NULL || r -> value == val)
156         return r;
```

C:\Users\LENOVO B40\Downloads\Binary Search Tree.cpp C/C++ Windows (CR+LF) WINDOWS-1252 Line 125, Col 4, Pos 3342 Insert Read/Write default

```
Start here x Binary Search Tree.cpp x
156         return r;
157
158     else if (val < r -> value)
159         return recursiveSearch(r -> left, val);
160
161     else
162         return recursiveSearch(r -> right, val);
163 }
164
165 int height(TreeNode * r) {
166     if (r == NULL)
167         return -1;
168     else {
169         // tinggi kiri dan kanan sama saja | maka akan tinggi simpul kanan | kiri
170         int lheight = height(r -> left);
171         int rheight = height(r -> right);
172
173         // menggunakan yang terbesar
174         if (lheight > rheight)
175             return (lheight + 1);
176         else return (rheight + 1);
177     }
178 }
179
180 // mencetak simpul
181 void printGivenLevel(TreeNode * r, int level) {
182     if (r == NULL)
183         return;
184     else if (level == 0)
185         cout << r -> value << " ";
186     // level > 0
187     else
```

```
Start here x Binary Search Tree.cpp x
187     else
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
199 TreeNode * minValueNode(TreeNode * node) {
200     TreeNode * current = node;
201     // perjalanan untuk menemukan daun paling kiri
202     while (current -> left != NULL) {
203         current = current -> left;
204     }
205     return current;
206 }
207
208 };
209
210 int main() {
211     //Membuat nama class menjadi objek
212     BST obj;
213     int pilihan, nilai;
214
215     do {
216         cout << "===== Menu =====< endl;
217         cout << "0. Keluar Program" << endl;
218         cout << "1. Masukkan Simpul" << endl;
```

```
Start here x Binary Search Tree.cpp x
218 cout << "1. Masukkan Simpul" << endl;
219 cout << "2. Mencetak Nilai BFS" << endl;
220 cout << endl;
221
222 cin >> pilihan;
223 // mengukhah variable;
224 TreeNode * new_node = new TreeNode();
225
226 switch (pilihan) {
227     case 0:
228         break;
229     case 1:
230         cout << "Masukkan Angka : ";
231         cin >> nilai;
232         new_node -> value = nilai;
233         obj.root = obj.insertRecursive(obj.root, new_node);
234         cout << endl;
235         break;
236     case 2:
237         cout << "Cetak" << endl;
238         obj.print2D(obj.root, 5);
239         cout << endl;
240         cout << "Mencetak Level Order BFS: \n";
241         obj.printLevelOrderBFS(obj.root);
242         cout << "\n\n";
243         break;
244     default:
245         cout << "Homor yang anda masukkan tidak sesuai";
246 }
247
248 while (pilihan != 0);
249
```

\\Users\\LENOVO B40\\Downloads\\Binary Search Tree.cpp C/C++ Windows (CR+LF) WINDOWS-1252 Line 218, Col 46, Pos 5576 Insert Read/Write default

```
246 }
247
248 while (pilihan != 0);
249
250 return 0;
251 }
252
```

\\Users\\LENOVO B40\\Downloads\\Binary Search Tree.cpp C/C++ Windows (CR+LF) WINDOWS-1252 Line 248, Col 26, Pos 6332 Insert Read/Write default

Hasil Ouput

```
"C:\Users\LENOVO B40\Downloads\Binary Search Tree.exe"
===== Menu =====
0. Keluar Program
1. Masukkan Simpul
2. Mencetak Nilai BFS

1
Masukkan Angka : 2
Insertion successful

===== Menu =====
0. Keluar Program
1. Masukkan Simpul
2. Mencetak Nilai BFS

1
Masukkan Angka : 5
Insertion successful

===== Menu =====
0. Keluar Program
1. Masukkan Simpul
2. Mencetak Nilai BFS

1
Masukkan Angka : 6
Insertion successful

===== Menu =====
0. Keluar Program
1. Masukkan Simpul
2. Mencetak Nilai BFS

1
Masukkan Angka : 8
Insertion successful

10:12 1
11/05/2022 Masukkan Angka : 9
Insertion successful

===== Menu =====

Select "C:\Users\LENOVO B40\Downloads\Binary Search Tree.exe"
===== Menu =====
0. Keluar Program
1. Masukkan Simpul
2. Mencetak Nilai BFS

1
Masukkan Angka : 10
Insertion successful

===== Menu =====
0. Keluar Program
1. Masukkan Simpul
2. Mencetak Nilai BFS

1
Masukkan Angka : 11
Insertion successful

===== Menu =====
0. Keluar Program
1. Masukkan Simpul
2. Mencetak Nilai BFS

2
Cetak

      11
     /  \
    5    10
   /  \  /  \
  2   6 9   8
 /  \
1   3
/  \
2  5

Mencetak Level Order BFS:
2 5 6 8 9 10 11

10:13 2 5 6 8 9 10 11
11/05/2022

===== Menu =====
0. Keluar Program
1. Masukkan Simpul
```