

LAPORAN PRAKTIKUM

PERTEMUAN 9

Tree



Nama :

Andika Rifki Pratama (2311104011)

Dosen :

Yudha Islami Sulistya,
S.Kom.,M.Kom.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

B. Soal Unguided

1. Modifikasi guided tree diatas dengan program menu menggunakan input data tree dari user dan berikan fungsi tambahan untuk menampilkan node child dan descendant dari node yang diinputkan!
2. Buatlah fungsi rekursif `is_valid_bst(node, min_val, max_val)` untuk memeriksa apakah suatu pohon memenuhi properti Binary Search Tree. Uji fungsi ini pada berbagai pohon, baik yang valid maupun tidak valid sebagai BST.
3. Buatlah fungsi rekursif `cari_simpul_daun(node)` untuk menghitung jumlah simpul daun dalam Binary Tree. Simpul daun adalah node yang tidak memiliki anak kiri maupun kanan.

```

Unguided > @ Unguided3.cpp ...
1 #include <iostream>
2 using namespace std;
3
4 // BINARY TREE
5
6 struct Pohon
7 {
8     char data;
9     Pohon *left, *right, *parent;
10 };
11
12 Pohon *root, *baru;
13
14 void init()
15 {
16     root = NULL;
17 }
18
19 bool isEmpty()
20 {
21     return root == NULL;
22 }
23
24 void buatNode(char data)
25 {
26     if (isEmpty())
27     {
28         root = new Pohon(data, NULL, NULL, NULL);
29         cout << "\nNode " << data << " berhasil dibuat jadi root" << endl;
30     }
31     else
32     {
33         cout << "\nPohon sudah dibuat." << endl;
34     }
35 }
36
37 // Tambah kiri
38 Pohon *insertLeft(char data, Pohon *node)
39 {
40     if (isEmpty())
41     {
42         cout << "\nBuat tree dulu." << endl;
43         return NULL;
44     }
45
46     if (node->left != NULL)
47     {
48         cout << "\nNode " << node->data << " sudah ada child kiri." << endl;
49         return NULL;
50     }
51
52     baru = new Pohon(data, NULL, NULL, node);
53     node->left = baru;
54     cout << "\nNode " << data << " berhasil ditambahkan ke child kiri " << node->data << endl;
55     return baru;
56 }
57
58 // Tambah kanan
59 Pohon *insertRight(char data, Pohon *node)
60 {
61     if (isEmpty())
62     {
63         cout << "\nBuat tree dulu!" << endl;
64         return NULL;
65     }
66
67     if (node->right != NULL)
68     {
69         cout << "\nNode " << node->data << " sudah ada child kanan." << endl;
70         return NULL;
71     }
72
73     baru = new Pohon(data, NULL, NULL, node);
74     node->right = baru;
75     cout << "\nNode " << data << " berhasil ditambahkan ke child kanan " << node->data << endl;
76     return baru;
77 }
78
79 // Update Tree
80 void update(char data, Pohon *node)
81 {
82     if (isEmpty())
83     {
84         cout << "\nBuat tree dulu!" << endl;
85         return;
86     }
87
88     if (!node)
89     {
90         cout << "\nNode yang ingin diganti tidak ada." << endl;
91         return;
92     }
93
94     char temp = node->data;
95     node->data = data;
96     cout << "\nNode " << temp << " berhasil diubah jadi " << data << endl;

```

```

177 // Fungsi Retrieve
178 void retrieve(Pohon *node)
179 {
180     if (isEmpty())
181     {
182         cout << "\nBuat tree dulu." << endl;
183         return;
184     }
185     if (!node)
186     {
187         cout << "\nNode yang ditunjuk tidak ada." << endl;
188         return;
189     }
190     cout << "\nData node: " << node->data << endl;
191 }
192
193 // Search Data
194 void find(Pohon *node)
195 {
196     if (isEmpty())
197     {
198         cout << "\nBuat tree dulu." << endl;
199         return;
200     }
201     if (!node)
202     {
203         cout << "\nNode yang ditunjuk tidak ada." << endl;
204         return;
205     }
206
207     cout << "\nData Node : " << node->data << endl;
208     cout << "Root : " << root->data << endl;
209     cout << "Parent : " << (node->parent ? node->parent->data : "(Tidak ada parent)") << endl;
210
211     if (node->parent)
212     {
213         if (node->parent->left == node && node->parent->right)
214             cout << "Sibling : " << node->parent->right->data << endl;
215         else if (node->parent->right == node && node->parent->left)
216             cout << "Sibling : " << node->parent->left->data << endl;
217         else
218             cout << "Sibling : (Tidak ada sibling)" << endl;
219     }
220 }
221
222 bool is_valid_bst(Pohon *node, char min_val, char max_val)
223 {
224     if (node == NULL)
225         return true;
226
227     if (node->data <= min_val || node->data >= max_val)
228         return false;
229
230     return is_valid_bst(node->left, min_val, node->data) && is_valid_bst(node->right, node->data, max_val);
231 }
232
233 int cari_simpul_daun(Pohon *node)
234 {
235     if (node == NULL)
236         return 0;
237
238     if (node->left == NULL && node->right == NULL)
239         return 1;
240
241     return cari_simpul_daun(node->left) + cari_simpul_daun(node->right);
242 }
243
244 void inputNode()
245 {
246     char data, parentData;
247     int choice;
248     Pohon *current = root;
249
250     if (isEmpty())
251     {
252         cout << "Masukkan nama node untuk dijadikan root: ";
253         cin >> data;
254         buatNode(data);
255     }
256
257     do
258     {
259         cout << "\n== Pilihan Input Node == " << endl;
260         cout << "1. Tambah Child Kiri" << endl;
261         cout << "2. Tambah Child Kanan" << endl;
262         cout << "3. Update Node" << endl;
263         cout << "4. Retrieve Node" << endl;
264         cout << "5. Find Node" << endl;
265         cout << "6. Cek apakah BST?" << endl;
266         cout << "7. Jumlah simpul daun" << endl;
267         cout << "8. Keluar" << endl;
268         cout << "Pilih: ";
269         cin >> choice;
270
271         if (choice == 1 || choice == 2 || choice == 3 || choice == 4 || choice == 5)

```

```

196         cout << "Masukkan data node parent: ";
197         cin >> parentData;
198
199         if (parentData == root->data)
200             current = root;
201         else if (root->left && root->left->data == parentData)
202             current = root->left;
203         else if (root->right && root->right->data == parentData)
204             current = root->right;
205         else
206             cout << "Parent node tidak ditemukan!" << endl;
207
208         if (!current) continue;
209     }
210
211     switch (choice)
212     {
213     case 1:
214         cout << "Masukkan data untuk child kiri: ";
215         cin >> data;
216         insertLeft(data, current);
217         break;
218     case 2:
219         cout << "Masukkan data untuk child kanan: ";
220         cin >> data;
221         insertRight(data, current);
222         break;
223     case 3:
224         cout << "Masukkan data baru untuk update node: ";
225         cin >> data;
226         update(data, current);
227         break;
228     case 4:
229         retrieve(current);
230         break;
231     case 5:
232         find(current);
233         break;
234     case 6:
235         if (is_valid_bst(root, CHAR_MIN, CHAR_MAX))
236             cout << "\nPohon adalah Binary Search Tree yang valid." << endl;
237         else
238             cout << "\nPohon bukan Binary Search Tree yang valid." << endl;
239         break;
240     case 7:
241         cout << "\nJumlah simpul daun: " << cari_simpul_daun(root) << endl;
242         break;
243     case 8:
244         cout << "Keluar." << endl;
245         break;
246     default:
247         cout << "Pilihan tidak valid!" << endl;
248     }
249 } while (choice != 8);
250
251 }
252
253 int main()
254 {
255     init();
256     inputNode();
257     return 0;
258 }

```

Penjelasan :

Ditambahkan fungsi inputNode untuk memunculkan menu yang dapat dipilih oleh user. Pada method ini terdapat fungsi untuk memeriksa apakah tree kosong atau tidak, kemudian apabila tidak kosong maka akan memunculkan menu dari menambah child, mengupdate, memunculkan bahkan mencari node dan BST hingga jumlah simpul daun. Kemudian terdapat penggunaan if else untuk memeriksa apakah terdapat data dari parent node. Selanjutnya terdapat fungsi switch case yang digunakan untuk memunculkan pilihan dan juga untuk memilih menu. Kemudian terdapat method is_valid_bst untuk memeriksa apakah nodenya BST atau tidak dengan cara memeriksa apakah data yang lebih dari value minimum.