LAPORAN PRAKTIKUM
Modul 9
“TREE”

Disusun Oleh:
Benedictus Qsota Noventino Baru - 2311104029
S1SE07A

Dosen :
Yudha Islami Sulistya, S.Kom., M.Cs

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM
PURWOKERTO

2024

Unguided

```cpp
#include <iostream>
#include <climits> // Untuk nilai batas integer
using namespace std;

// Deklarasi Pohon
struct Pohon {
    char data;
    Pohon *left, *right, *parent;
};

Pohon *root, *baru;

// Inisialisasi
void init() {
    root = NULL;
}

// Cek Node
bool isEmpty() {
    return root == NULL;
}

// Buat Node Baru
void buatNode(char data) {
    if (isEmpty()) {
        root = new Pohon{data, NULL, NULL, NULL};
        cout << "\nNode " << data << " berhasil dibuat menjadi root." << endl;
    } else {
        cout << "\nPohon sudah dibuat." << endl;
    }
}

// Tambah Kiri
Pohon *insertLeft(char data, Pohon *node) {
    if (isEmpty()) {
        cout << "\nBuat tree terlebih dahulu!" << endl;
        return NULL;
    }
    if (node->left != NULL) {
        cout << "\nNode " << node->data << " sudah ada child kiri!" << endl;
        return NULL;
    }
    baru = new Pohon{data, NULL, NULL, node};
    node->left = baru;
    cout << "\nNode " << data << " berhasil ditambahkan ke child kiri " << node->data << endl;
    return baru;
}

// Tambah Kanan
Pohon *insertRight(char data, Pohon *node) {
    if (isEmpty()) {
        cout << "\nBuat tree terlebih dahulu!" << endl;
        return NULL;
    }
    if (node->right != NULL) {
        cout << "\nNode " << node->data << " sudah ada child kanan!" << endl;
        return NULL;
    }
    baru = new Pohon{data, NULL, NULL, node};
    node->right = baru;
    cout << "\nNode " << data << " berhasil ditambahkan ke child kanan " << node->data << endl;
    return baru;
}


```

```cpp
// Menampilkan Child
void showChild(Pohon *node) {
    if (!node) {
        cout << "Node tidak ada." << endl;
        return;
    }
    cout << "Node " << node->data << " memiliki:" << endl;
    cout << "- Child kiri: " << (node->left ? node->left->data : '-') << endl;
    cout << "- Child kanan: " << (node->right ? node->right->data : '-') << endl;
}

// Menampilkan Descendant
void showDescendant(Pohon *node) {
    if (!node) return;
    cout << node->data << " ";
    showDescendant(node->left);
    showDescendant(node->right);
}

// Fungsi Validasi BST
bool is_valid_bst(Pohon *node, int min_val, int max_val) {
    if (!node) return true;
    if (node->data <= min_val || node->data >= max_val) return false;
    return is_valid_bst(node->left, min_val, node->data) &&
           is_valid_bst(node->right, node->data, max_val);
}

// Fungsi Cari Simpul Daun
int cari_simpul_daun(Pohon *node) {
    if (!node) return 0;
    if (!node->left && !node->right) return 1;
    return cari_simpul_daun(node->left) + cari_simpul_daun(node->right);
}

// Fungsi Menu
void menu() {
    int pilihan;
    char data, parent_data;
    Pohon *parent_node = NULL;

    do {
        cout << "\nMENU:\n";
        cout << "1. Buat Root\n";
        cout << "2. Tambah Kiri\n";
        cout << "3. Tambah Kanan\n";
        cout << "4. Tampilkan Child\n";
        cout << "5. Tampilkan Descendant\n";
        cout << "6. Periksa BST\n";
        cout << "7. Hitung Simpul Daun\n";
        cout << "0. Keluar\n";
        cout << "Pilihan: ";
        cin >> pilihan;

        switch (pilihan) {
            case 1:
                cout << "Masukkan data root: ";
                cin >> data;
                buatNode(data);
                break;
            case 2:
                cout << "Masukkan data parent: ";
                cin >> parent_data;
                cout << "Masukkan data child kiri: ";
                cin >> data;
                parent_node = root; // Anda dapat menambahkan pencarian node berdasarkan parent_data
                insertLeft(data, parent_node);
                break;
            case 3:
                cout << "Masukkan data parent: ";
                cin >> parent_data;
                cout << "Masukkan data child kanan: ";
                cin >> data;
                parent_node = root; // Anda dapat menambahkan pencarian node berdasarkan parent_data
                insertRight(data, parent_node);
                break;
            case 4:
                cout << "Masukkan node untuk melihat child: ";
                cin >> parent_data;
                parent_node = root; // Anda dapat menambahkan pencarian node berdasarkan parent_data
                showChild(parent_node);
                break;
            case 5:
                cout << "Masukkan node untuk melihat descendant: ";
                cin >> parent_data;
                parent_node = root; // Anda dapat menambahkan pencarian node berdasarkan parent_data
                cout << "Descendant dari " << parent_data << ": ";
                showDescendant(parent_node);
                cout << endl;
                break;
            case 6:
                cout << (is_valid_bst(root, INT_MIN, INT_MAX) ? "Pohon adalah BST." : "Pohon bukan BST.") << endl;
                break;
            case 7:
                cout << "Jumlah simpul daun: " << cari_simpul_daun(root) << endl;
                break;
            case 0:
                cout << "Keluar dari program." << endl;
                break;
            default:
                cout << "Pilihan tidak valid." << endl;
        }
    } while (pilihan != 0);
}

// Fungsi Utama
int main() {
    init();
    menu();
    return 0;
}

```

Penjelasan:

1. **Menu interaktif**: Memudahkan pengguna untuk memasukkan data secara dinamis.
2. **Fungsi `is_valid_bst`**: Memeriksa properti BST menggunakan rekursi dengan batas `min_val` dan `max_val`.
3. **Fungsi `cari_simpul_daun`**: Menghitung jumlah simpul daun secara rekursif.
4. **Fungsi tambahan `showChild` dan `showDescendant`**: Menampilkan child dan descendant dari node tertentu.