LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 10
TREE (BAGIAN PERTAMA)

Disusun Oleh :

NAMA : Muhammad Azzam Satria

NIM : 103112400112

Dosen

FAHRUDIN MUKTI WIBOWO S. Kom., M. Eng

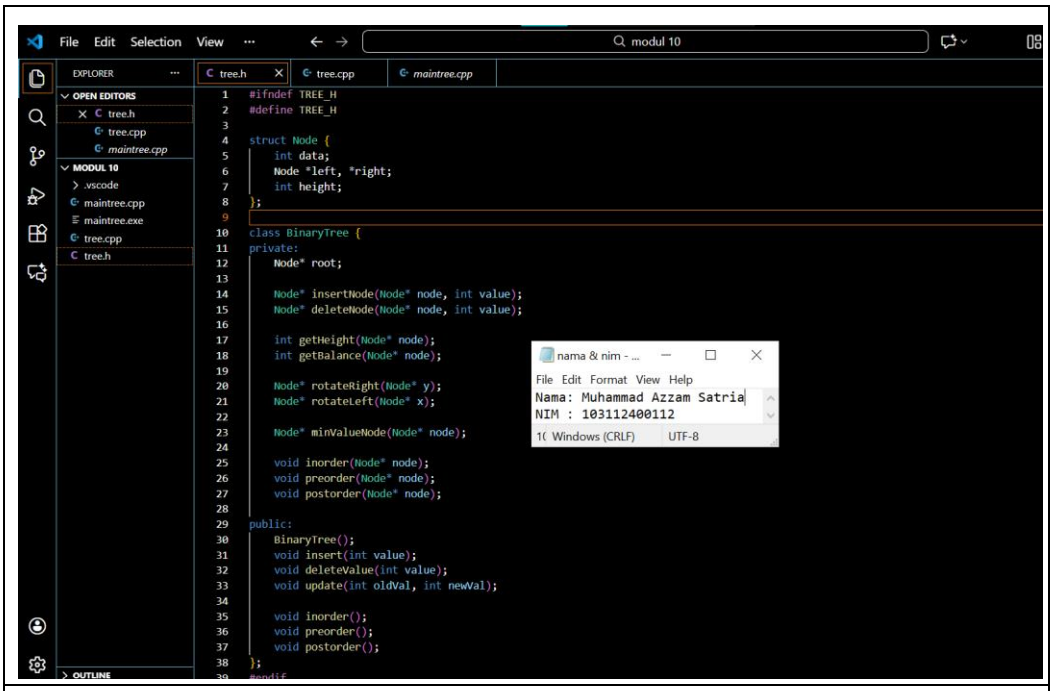
PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

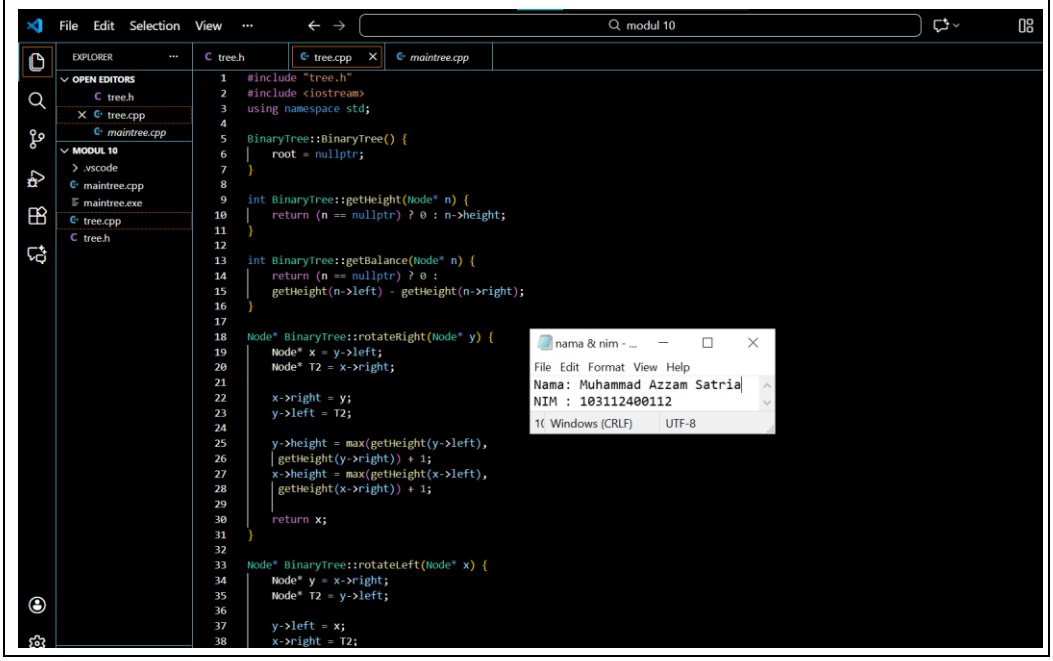
A. Dasar Teori

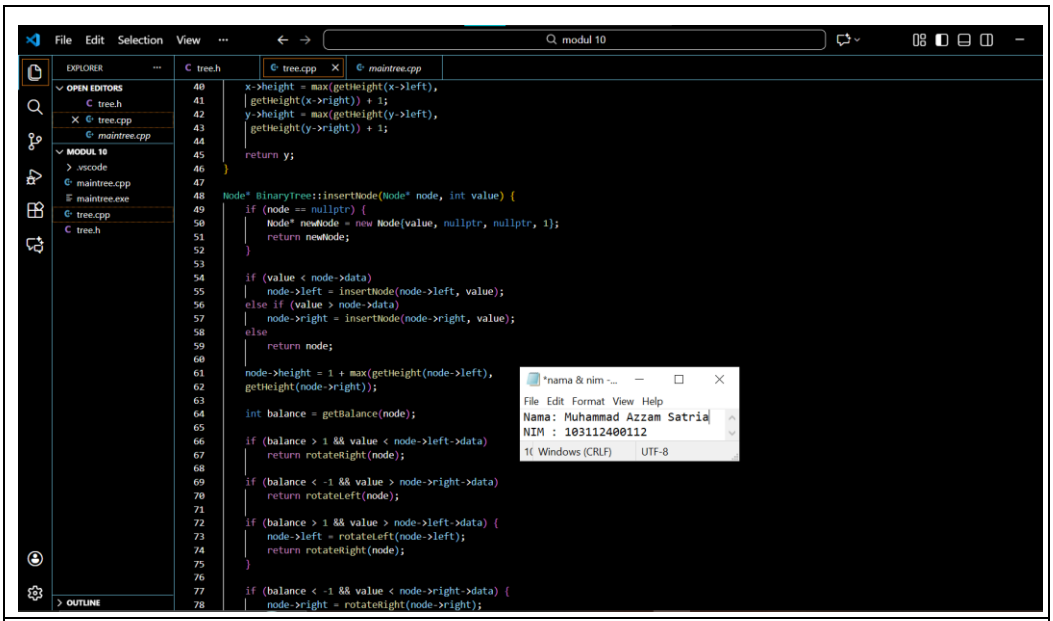
Struktur data tree adalah suatu bentuk penyimpanan data yang non-linier dan terdiri dari node yang saling terhubung dengan istilahnya yaitu parent dan child. Pada tree node pertama pada pohon akar disebut root sedangkan node yang tidak mempunyai akar disebut leaf. Setiap node tersebut bisa memiliki satu atau lebih child yang terhubung melalui cabang. Bentuk tree di modul ini ada 2 yaitu Binary Tree yang setiap nodenya maksimal hanya mempunyai dua child dan BinarySearchTree yaitu tree dengan aturan nilai yang lebih kecil akan ditempatkan di child sebelah kiri dan nilai yang lebih besar ditempatkan di child sebelah kanan.

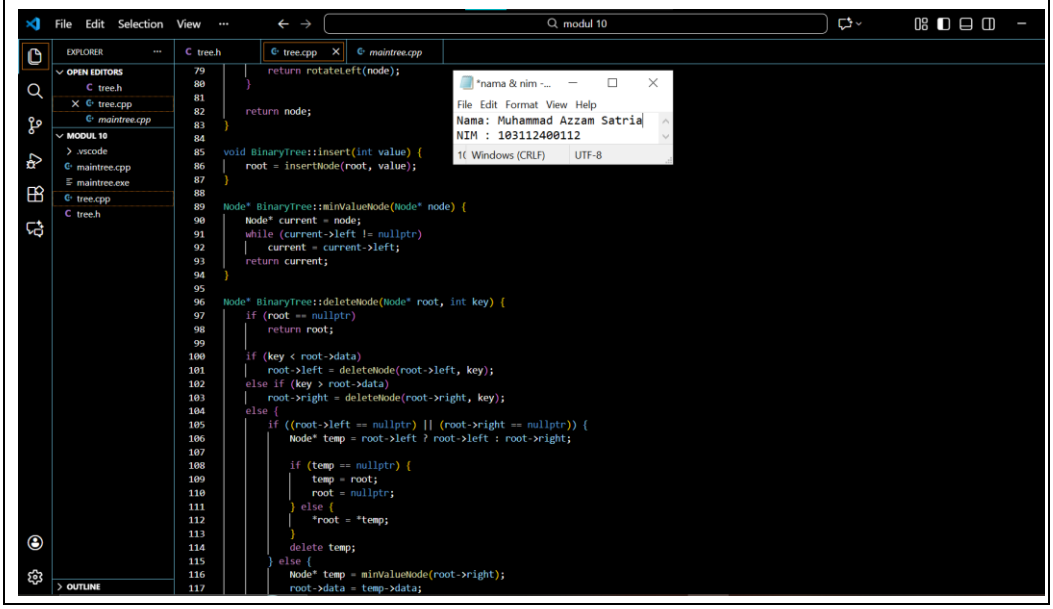
Di dalam modul terdapat beberapa operasi dasar diantaranya yaitu insert, search dan traversal untuk membantu dalam menyimpan dan menelusuri data di dalam struktur tree. Cara kerja penambahan dan pencarian data yaitu dengan membandingkan nilai pada setiap node. Traversal seperti inoorder, preorder dan postorder memiliki fungsi untuk menampilkan isi tree sesuai yang dijalankan. Penggunaan algoritma rekusrsif yang terdapat pada modul membuat traversal di suatu program dapat dibuat menjadi lebih sederhana. Akan teteapi, teknik ini memiliki kelemahan yaitu jika tree terlalu dalam, maka akan terjadi pemanggilan fungsi yang berlebihan. Sehingga kondisi tersebut dapat memicu terjadinya overflow pada program.

B. Guided

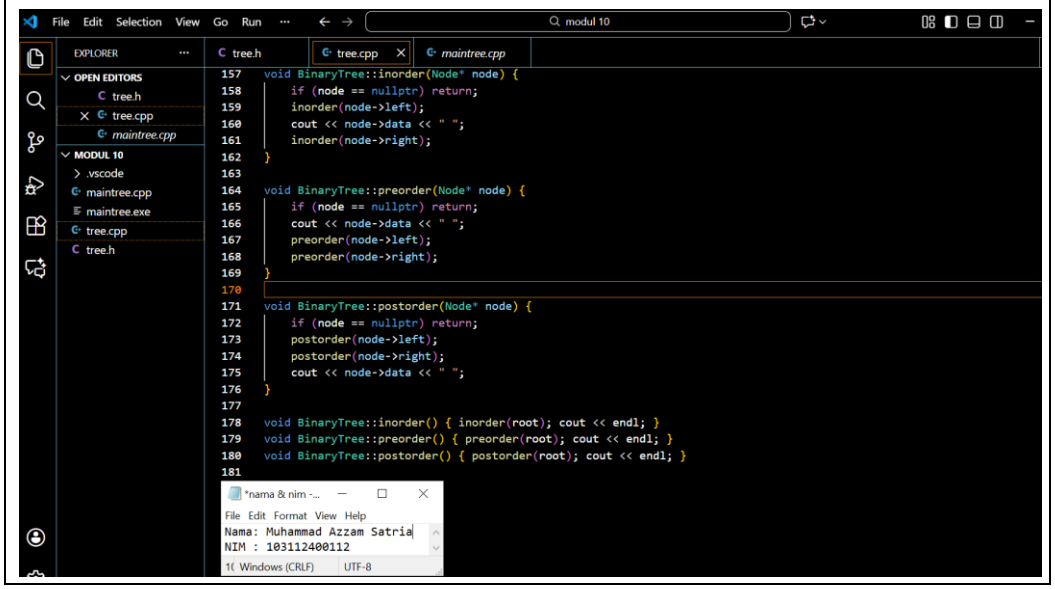

File Edit Selection View
modul 10
EXPLORER
OPEN EDITORS
tree.h
tree.cpp
maintree.cpp
MODUL 10
.vscode
maintree.cpp
maintree.exe
tree.cpp
tree.h
OUTLINE
#ifndef TREE_H
#define TREE_H
struct Node {
int data;
Node *left, *right;
int height;
};
class BinaryTree {
private:
Node* root;
Node* insertNode(Node* node, int value);
Node* deleteNode(Node* node, int value);
int getHeight(Node* node);
int getBalance(Node* node);
Node* rotateRight(Node* y);
Node* rotateLeft(Node* x);
Node* minValueNode(Node* node);
void inorder(Node* node);
void preorder(Node* node);
void postorder(Node* node);
public:
BinaryTree();
void insert(int value);
void deleteValue(int value);
void update(int oldVal, int newVal);
void inorder();
void preorder();
void postorder();
};
nama & nim - ...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Ln 1, Col 1 Windows (CRLF) UTF-8

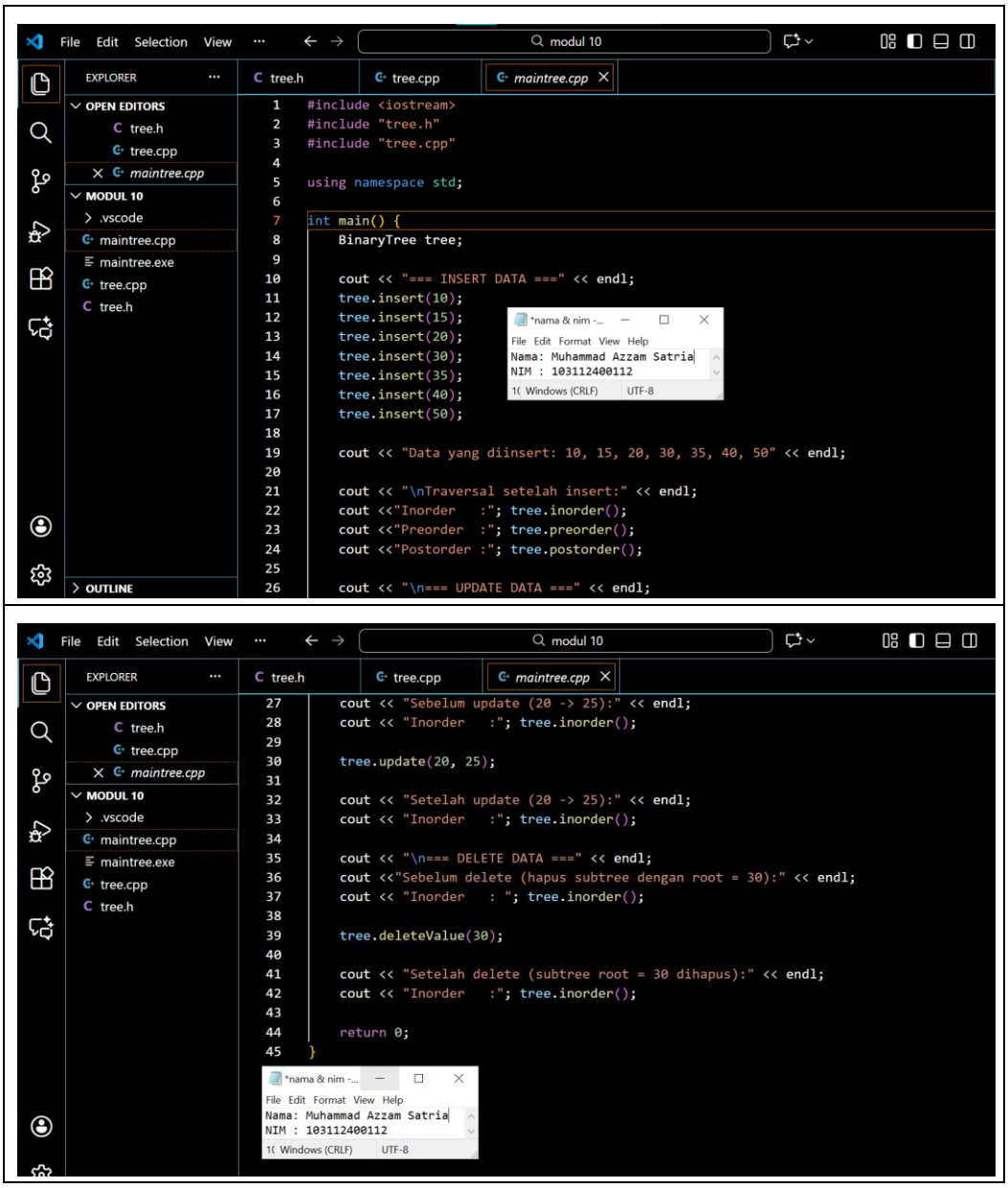

File Edit Selection View
modul 10
EXPLORER
OPEN EDITORS
tree.h
tree.cpp
maintree.cpp
MODUL 10
.vscode
maintree.cpp
maintree.exe
tree.cpp
tree.h
#include "tree.h"
#include <iostream>
using namespace std;
BinaryTree::BinaryTree() {
root = nullptr;
}
int BinaryTree::getHeight(Node* n) {
return (n == nullptr) ? 0 : n->height;
}
int BinaryTree::getBalance(Node* n) {
return (n == nullptr) ? 0 :
getHeight(n->left) - getHeight(n->right);
}
Node* BinaryTree::rotateRight(Node* y) {
Node* x = y->left;
Node* T2 = x->right;
x->right = y;
y->left = T2;
y->height = max(getHeight(y->left),
getHeight(y->right)) + 1;
x->height = max(getHeight(x->left),
getHeight(x->right)) + 1;
return x;
}
Node* BinaryTree::rotateLeft(Node* x) {
Node* y = x->right;
Node* T2 = y->left;
y->left = x;
x->right = T2;
nama & nim - ...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Ln 1, Col 1 Windows (CRLF) UTF-8


File Edit Selection View
modul 10
EXPLORER
OPEN EDITORS
tree.h
tree.cpp
maintree.cpp
MODUL 10
.vscode
maintree.cpp
maintree.exe
tree.cpp
tree.h
OUTLINE
x->height = max(getHeight(x->left),
getHeight(x->right)) + 1;
y->height = max(getHeight(y->left),
getHeight(y->right)) + 1;
return y;
}
Node* BinaryTree::insertNode(Node* node, int value) {
if (node == nullptr) {
Node* newNode = new Node{value, nullptr, nullptr, 1};
return newNode;
}
if (value < node->data)
node->left = insertNode(node->left, value);
else if (value > node->data)
node->right = insertNode(node->right, value);
else
return node;
node->height = 1 + max(getHeight(node->left),
getHeight(node->right));
int balance = getBalance(node);
if (balance > 1 && value < node->left->data)
return rotateRight(node);
if (balance < -1 && value > node->right->data)
return rotateLeft(node);
if (balance > 1 && value > node->left->data) {
node->left = rotateLeft(node->left);
return rotateRight(node);
}
if (balance < -1 && value < node->right->data) {
node->right = rotateRight(node->right);
*nama & nim -...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Windows (CRLF)
UTF-8

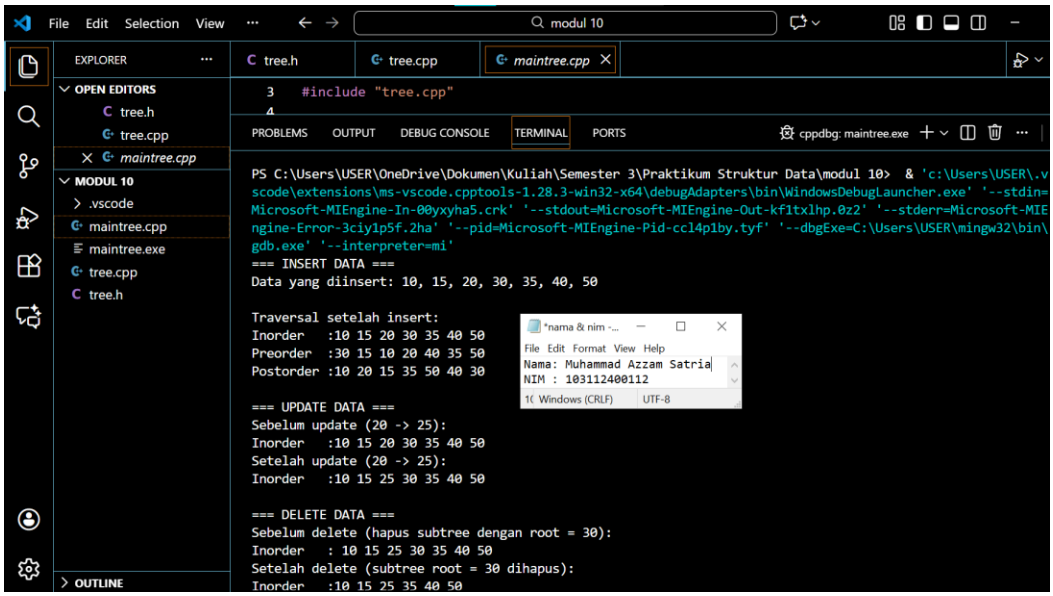

File Edit Selection View
modul 10
EXPLORER
OPEN EDITORS
tree.h
tree.cpp
maintree.cpp
MODUL 10
.vscode
maintree.cpp
maintree.exe
tree.cpp
tree.h
OUTLINE
return rotateLeft(node);
}
return node;
}
void BinaryTree::insert(int value) {
root = insertNode(root, value);
}
Node* BinaryTree::minValueNode(Node* node) {
Node* current = node;
while (current->left != nullptr)
current = current->left;
return current;
}
Node* BinaryTree::deleteNode(Node* root, int key) {
if (root == nullptr)
return root;
if (key < root->data)
root->left = deleteNode(root->left, key);
else if (key > root->data)
root->right = deleteNode(root->right, key);
else {
if ((root->left == nullptr) || (root->right == nullptr)) {
Node* temp = root->left ? root->left : root->right;
if (temp == nullptr) {
temp = root;
root = nullptr;
} else {
*root = *temp;
}
delete temp;
} else {
Node* temp = minValueNode(root->right);
root->data = temp->data;
*nama & nim -...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Windows (CRLF)
UTF-8


File Edit Selection View
modul 10
EXPLORER
OPEN EDITORS
tree.h
tree.cpp
maintree.cpp
MODUL 10
.vscode
maintree.cpp
maintree.exe
tree.cpp
tree.h
root->right = deleteNode(root->right, temp->data);
}
}
if (root == nullptr)
return root;
root->height = 1 + max(getHeight(root->left), getHeight(root->right));
int balance = getBalance(root);
if (balance > 1 && getBalance(root->left) >= 0)
return rotateRight(root);
if (balance > 1 && getBalance(root->left) < 0) {
root->left = rotateLeft(root->left);
return rotateRight(root);
}
if (balance < -1 && getBalance(root->right) <= 0)
return rotateLeft(root);
if (balance < -1 && getBalance(root->right) > 0) {
root->right = rotateRight(root->right);
return rotateLeft(root);
}
return root;
}
void BinaryTree::deleteValue(int value) {
root = deleteNode(root, value);
}
void BinaryTree::update(int oldVal, int newVal) {
deleteValue(oldVal);
insert(newVal);
}
*nama & nim -...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Ln 1, Col 1 Windows (CRLF) UTF-8


File Edit Selection View Go Run
modul 10
EXPLORER
OPEN EDITORS
tree.h
tree.cpp
maintree.cpp
MODUL 10
.vscode
maintree.cpp
maintree.exe
tree.cpp
tree.h
void BinaryTree::inorder(Node* node) {
if (node == nullptr) return;
inorder(node->left);
cout << node->data << " ";
inorder(node->right);
}
void BinaryTree::preorder(Node* node) {
if (node == nullptr) return;
cout << node->data << " ";
preorder(node->left);
preorder(node->right);
}
void BinaryTree::postorder(Node* node) {
if (node == nullptr) return;
postorder(node->left);
postorder(node->right);
cout << node->data << " ";
}
void BinaryTree::inorder() { inorder(root); cout << endl; }
void BinaryTree::preorder() { preorder(root); cout << endl; }
void BinaryTree::postorder() { postorder(root); cout << endl; }
*nama & nim -...
File Edit Format View Help
Nama: Muhammad Azzam Satria
NIM : 103112400112
Ln 1, Col 1 Windows (CRLF) UTF-8



Screenshots Output



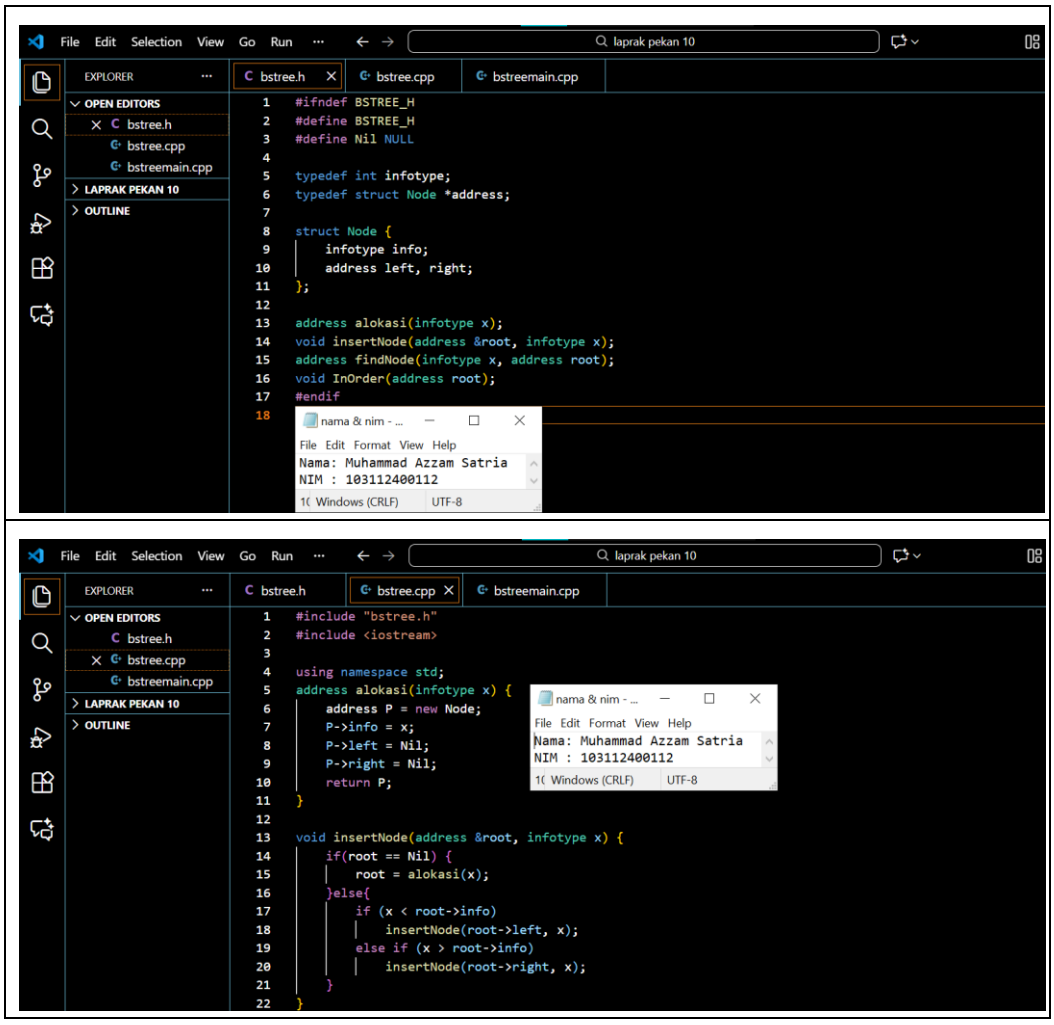
Deskripsi:

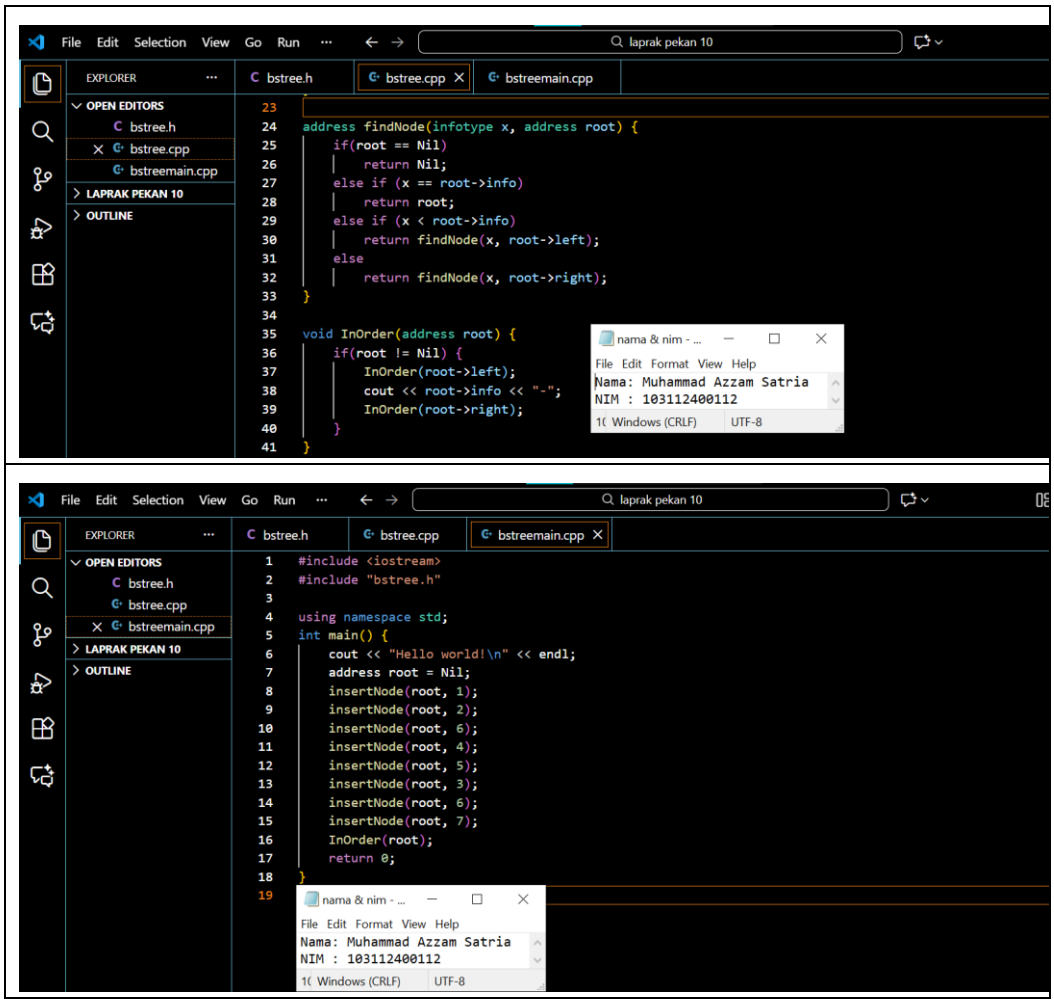
Program ini mengimplementasikan struktur data tree dengan menggunakan konsep ADT(Abstract Data Type). Setiap node pada tree mempunyai tiga bagian yaitu data bertipe integer, pointer ke anak sebelah kiri dan pointer ke anak sebelah kanan. Tinggi setiap node disimpan untuk membantu melakukan pengecekan keseimbangan tree. Fungsi insert() yaitu untuk menambahkan data baru ke dalam tree yang kemudian akan dilakukan pengecekan agar tree tetap seimbang. Jika struktur tree belum seimbang, maka akan dilakukan rotasi kiri, kanan atau kombinasi dari keduanya . Proses delete() cara kerjanya sama seperti fungsi insert, namun proses ini diawali dengan pencarian node pengganti seperti nilai terkecil di subtree kanan. Pada program juga mempunyai fungsi inorder(), preorder() dan postorder() untuk menampilkan isi tree berdasarkan jenis traversal yang dipilih.

Selain operasi dasar insert dan delete, program ini mempunyai fungsi update() yang cara kerjanya menghapus nilai lama dan menambahkan nilai baru. Kemudian program mempunyai fungsi tambahan yaitu getHeight() dan getBalance() untuk memudahkan dalam melakukan pengecekan kondisi tree setelah menjalankan fungsi update(). Pada file maintree.cpp program menjalankan berbagai operasi seperti menambha data, mengubah nilai data, menghapus data dan menampilkan isi tree. Output program ini mencetak kondisi

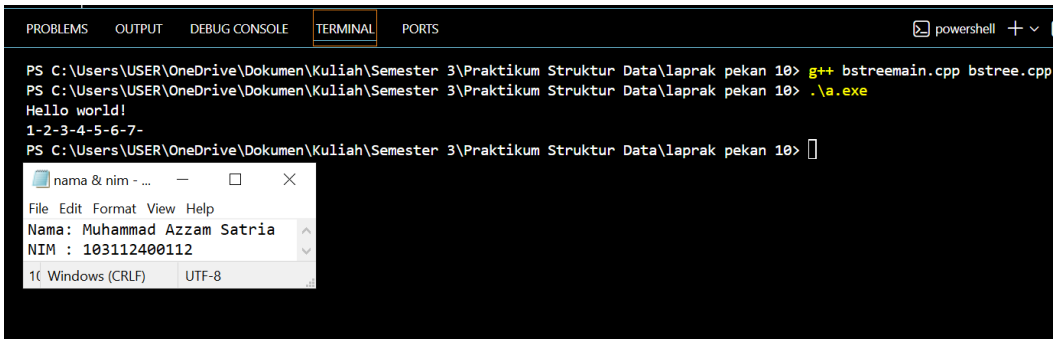
tree yaitu berupa data yang diinsert, traversal setelah insert, update data dan delete data.

C. Unguided 1





Screenshots Output



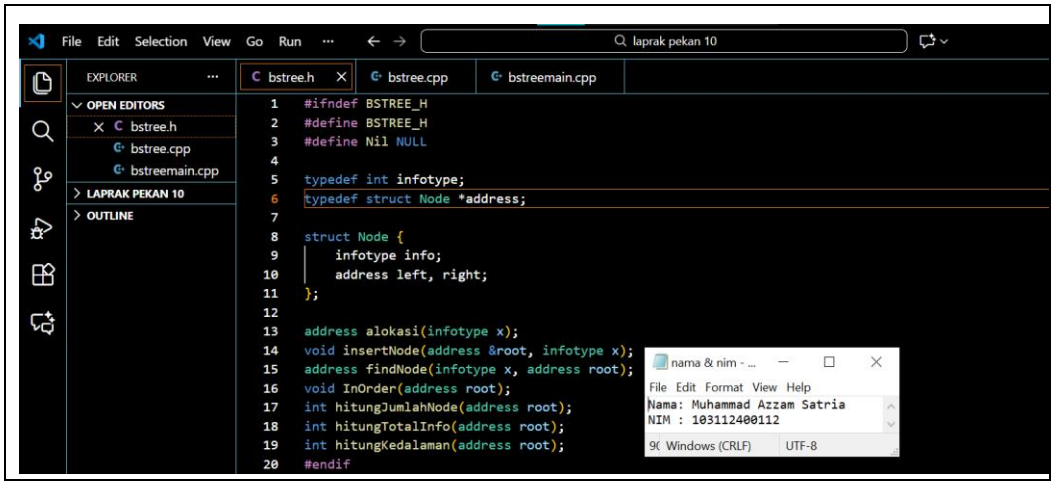
Deskripsi:

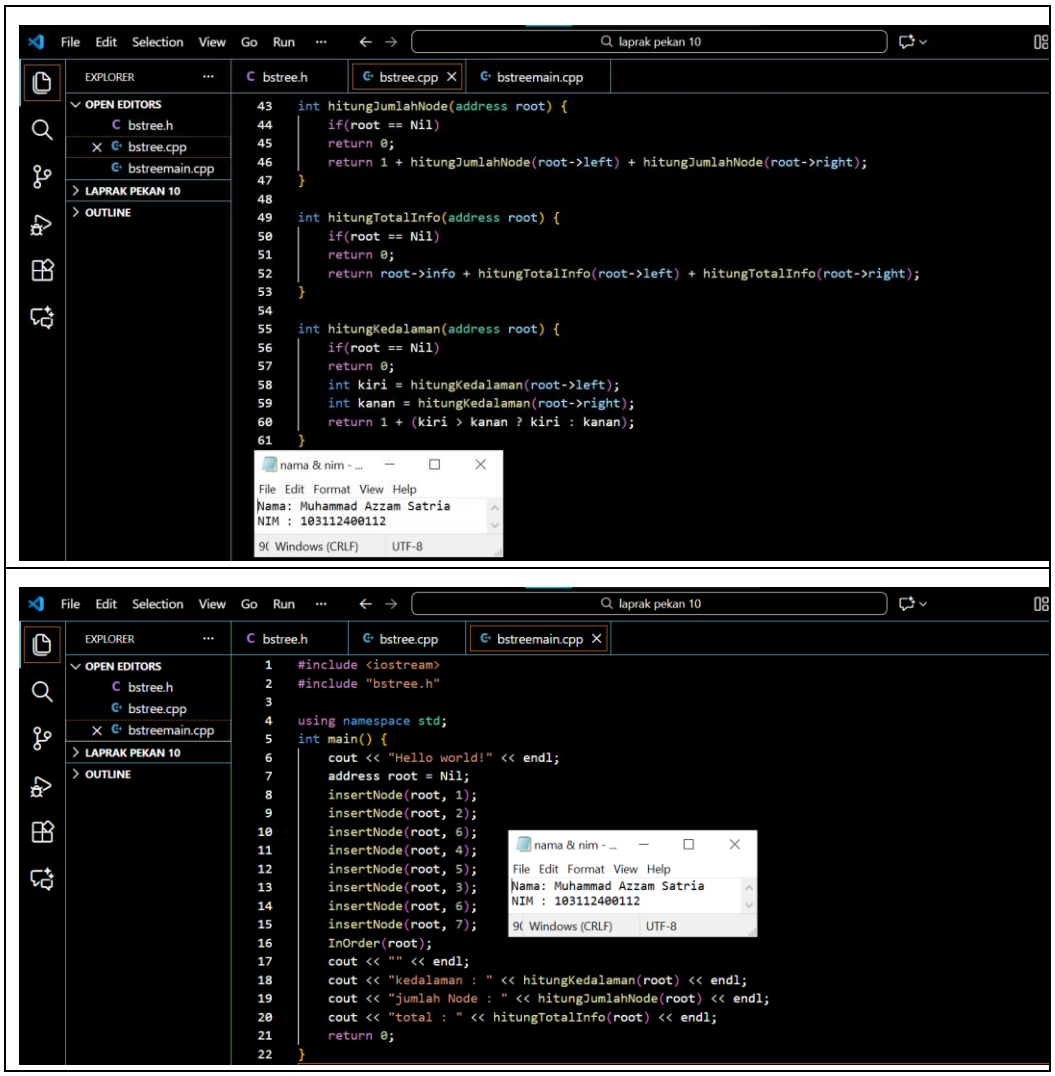
Pada soal nomor 1 di bagian header, program mendefinisikan tipe node yang menyimpan data serta pointer ke anak sebelah kiri dan kanan. Beberapa fungsi

dasar yang terdapat didalamnya yaitu fungsi alokasi node baru, insertNode, findNode dan InOrder. Mekanisme insert dilakukan dengan cara rekursif mengikuti aturan BST (Binary Search Tree), yaitu nilai yang lebih kecil akan masuk ke kiri dan nilai yang lebih besar akan masuk ke kanan.

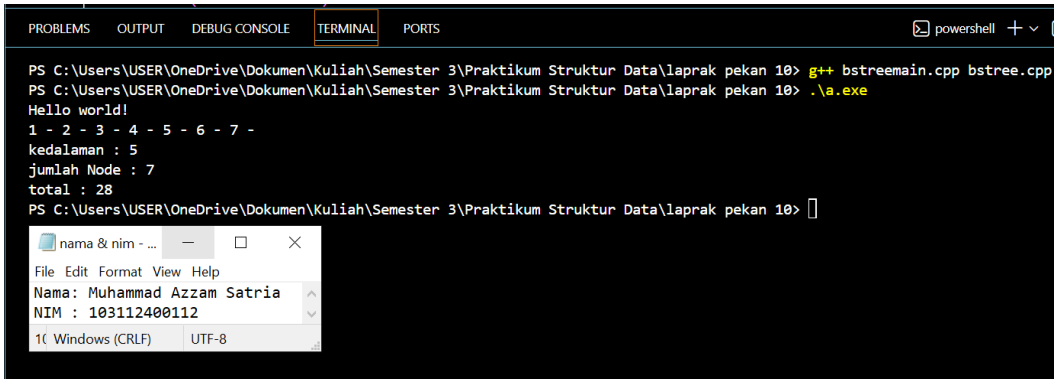
Di bagian file implementasi (bstree.cpp) terdapat Semua fungsi yang bekerja secara rekursif. Fungsi findNode bekerja dengan cara membandingkan nilai yang sedang dicari kemudian bergerak turun ke kiri atau kanan hingga terdapat node yang kosong. Proses InOrder memiliki fungsi yaitu untuk menampilkan tree dengan urutan naik pada output. Pada bagian file bstreemain terdapat sejumlah angka yang dimsukkan sebagai inputan program. Kemudian output akan menampilkan daftar urutan angka yang tersusun dengan aturan inorder.

Unguided 2





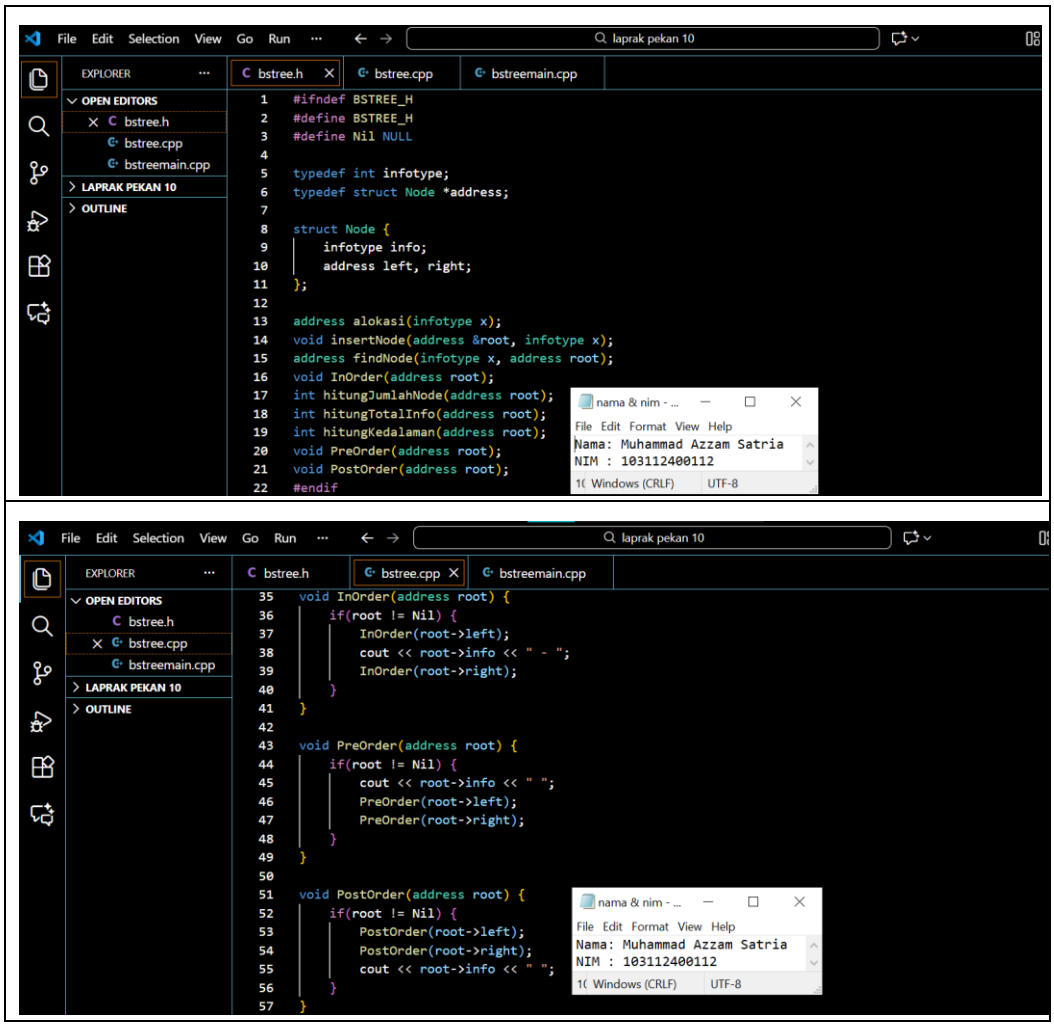
Screenshots Output

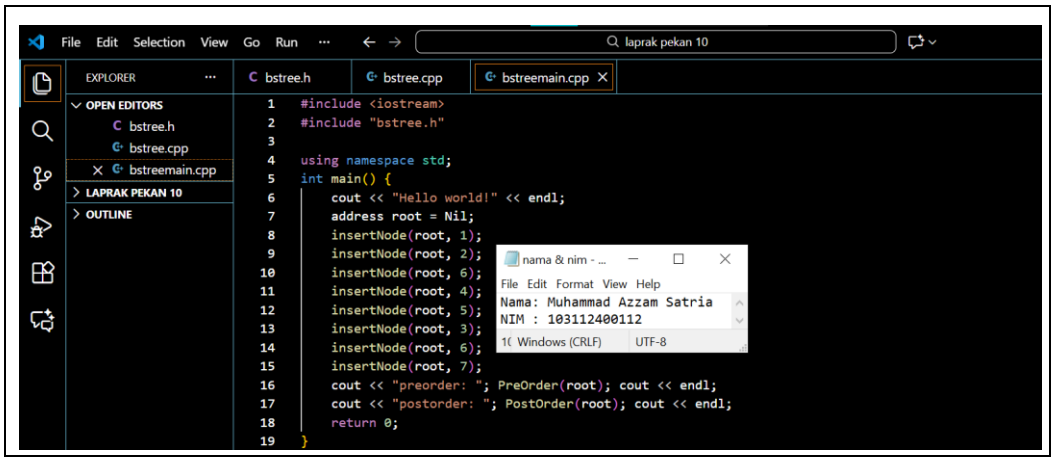


Deskripsi:

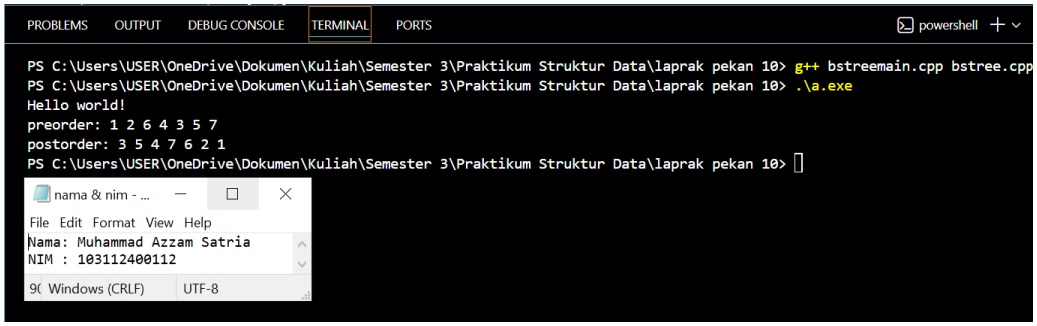
Pada soal nomor 2, program memiliki 3 fungsi tambahan yaitu fungsi hitungJumlahNode() untuk mengetahui berapa banyak node di dalam tree, fungsi hitungTotalInfo() untuk menjumlahkan semua nilai info pada node, fungsi hitungKedalaman() untuk mencari dalam dari sebuah tree dengan membandingkan kedalaman subtree kiri dan kanan.

Unguided 3





Screenshots Output



Deskripsi:

Pada soal nomor 3, program memiliki 2 fungsi tambahan traversal yaitu PreOrder dan PostOrder. PreOrder fungsinya untuk menampilkan data dengan urutan root → anak sebelah kiri → anak sebelah kanan, sedangkan PostOrder fungsinya untuk menampilkan data dengan urutan anak sebelah kiri → anak sebelah kanan → root.

D. Kesimpulan

Struktur data tree merupakan salah satu bentuk cara penyimpanan data yang cara penyusunannya memiliki aturan bahwa nilai yang lebih kecil ditempatkan di bagian anak sebelah kiri, sedangkan nilai yang lebih besar ditempatkan di bagian anak sebelah kanan. Melalui contoh dan latihan pada modul saya belajar secara langsung bagaimana membangun struktur data tree dengan konsep ADT (Abstract Data Type). Pada soal latihan, saya mempelajari cara Membuat node baru, menambah data ke dalam tree, mencari suatu nilai dalam tree dan

menampilkan isi tree dengan menggunakan traversal InOrder.

Pada soal latihan selanjutnya, terdapat operasi tambahan untuk menganalisis isi tree seperti menghitung jumlah node, total nilai dan menentukan kedalaman suatu tree. Lalu di soal latihan terakhir terdapat penambahan fungsi PreOrder dan PostOrder agar output tree bisa menampilkan pola urutan yang berbeda. Dengan adanya laporan praktikum modul 10 ini, memberikan gambaran kepada saya mengenai pengelolaan dan analisis pada suatu struktur data tree.

E. Referensi

Ginting, S. H. N., Effendi, H., Kumar, S., Marsisno, W., Sitanggang, Y. R. U., Anwar, K., … & Smrti, N. N. E. (2024). *Pengantar struktur data*. Penerbit Mifandi Mandiri Digital.

Febriansyah, M. F., Rhamadani, M., & Sutabri, T. (2025). Perbandingan pemanfaatan algoritma rekursif dan iteratif dalam penyelesaian struktur data pohon. *Jurnal Manajemen Informatika & Teknologi*, 5(1), 46–56.