LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 10
PENGENALAN CODE BLOCKS

Disusun Oleh :
NAMA : Jauza Rasyiq Hernanta
NIM : **103112430033**

Dosen
WAHYU ANDI SAPUTRA

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

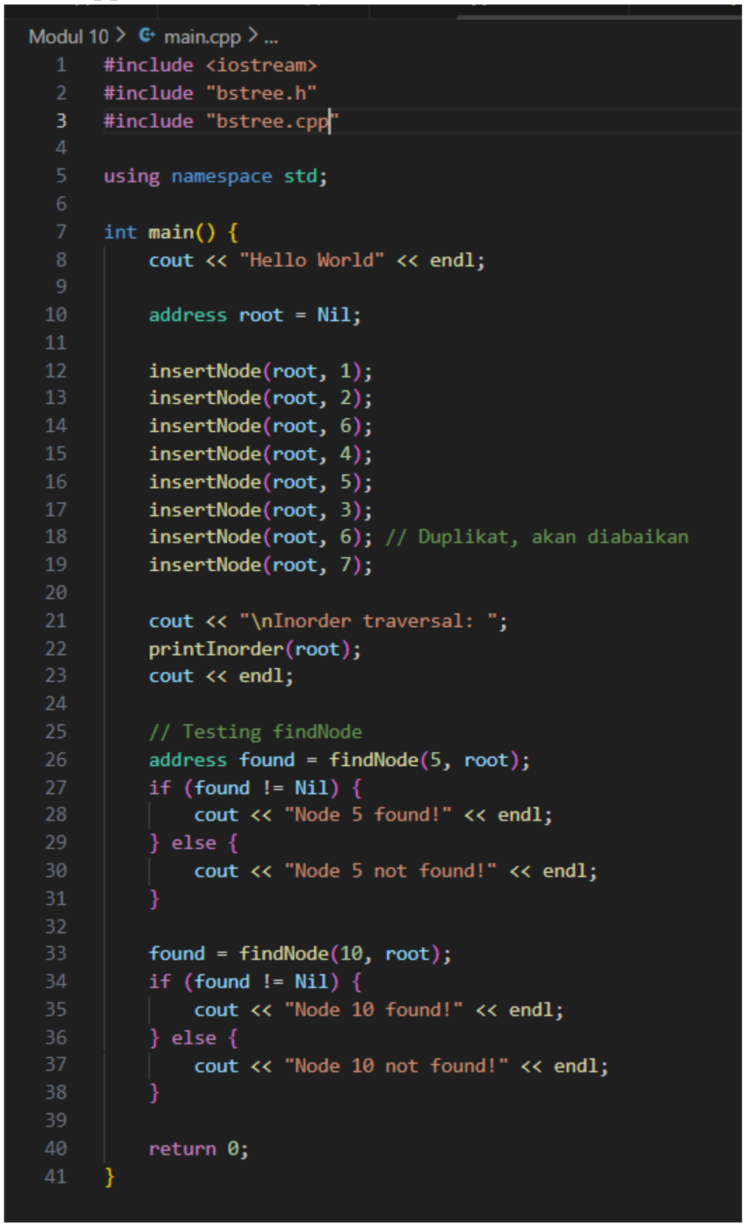
A. Dasar Teori

Stack merupakan salah satu bentuk struktur data dimana prinsip operasi yang digunakan seperti tumpukan. Seperti halnya tumpukan, elemen yang bisa diambil terlebih dahulu adalah elemen yang paling atas, atau elemen yang pertama kali masuk, prinsip ini biasa disebut LIFO (Last In First Out).

B. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1

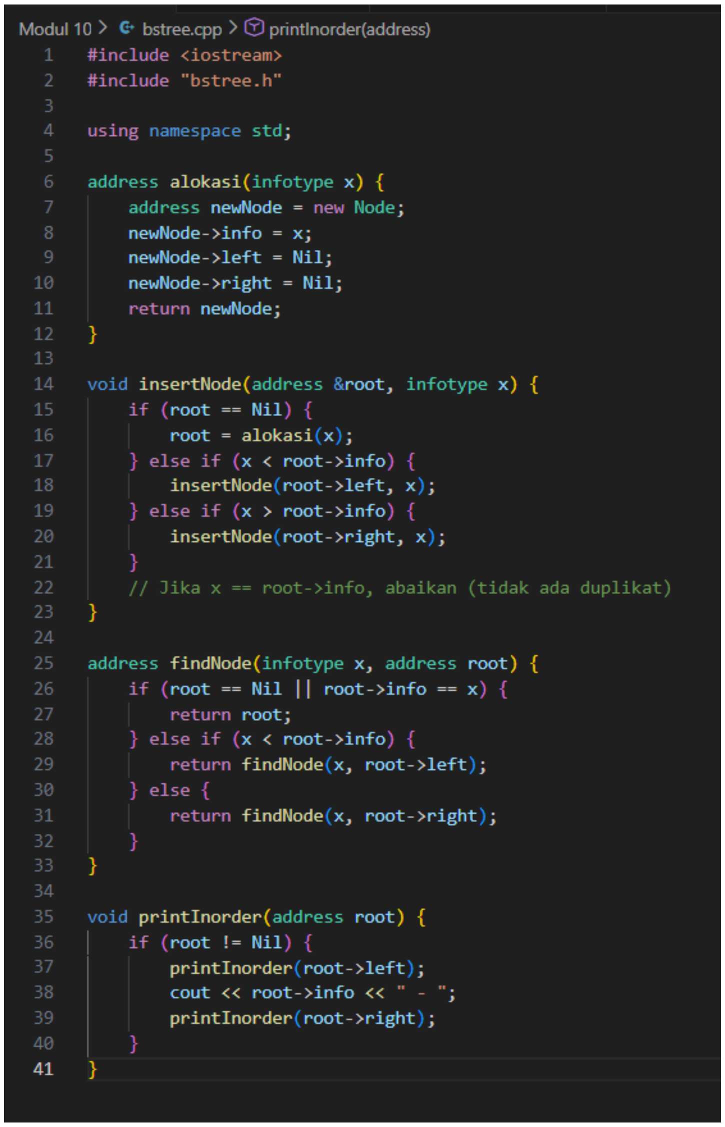
Main.cpp



Modul 10 > main.cpp > ...

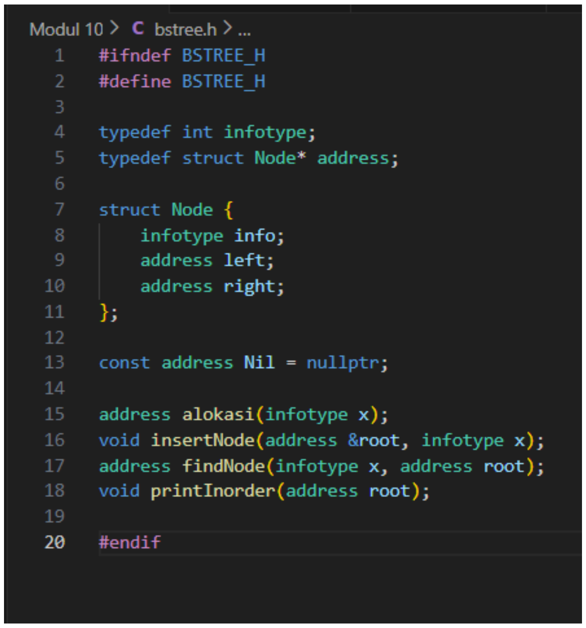
```cpp
#include <iostream>
#include "bstree.h"
#include "bstree.cpp"

using namespace std;

int main() {
    cout << "Hello World" << endl;

    address root = Nil;

    insertNode(root, 1);
    insertNode(root, 2);
    insertNode(root, 6);
    insertNode(root, 4);
    insertNode(root, 5);
    insertNode(root, 3);
    insertNode(root, 6); // Duplikat, akan diabaikan
    insertNode(root, 7);

    cout << "\nInorder traversal: ";
    printInorder(root);
    cout << endl;

    // Testing findNode
    address found = findNode(5, root);
    if (found != Nil) {
        cout << "Node 5 found!" << endl;
    } else {
        cout << "Node 5 not found!" << endl;
    }

    found = findNode(10, root);
    if (found != Nil) {
        cout << "Node 10 found!" << endl;
    } else {
        cout << "Node 10 not found!" << endl;
    }

    return 0;
}
```

Kendaraan,cpp



```cpp
#include <iostream>
#include "bstree.h"

using namespace std;

address alokasi(infotype x) {
    address newNode = new Node;
    newNode->info = x;
    newNode->left = Nil;
    newNode->right = Nil;
    return newNode;
}

void insertNode(address &root, infotype x) {
    if (root == Nil) {
        root = alokasi(x);
    } else if (x < root->info) {
        insertNode(root->left, x);
    } else if (x > root->info) {
        insertNode(root->right, x);
    }
    // Jika x == root->info, abaikan (tidak ada duplikat)
}

address findNode(infotype x, address root) {
    if (root == Nil || root->info == x) {
        return root;
    } else if (x < root->info) {
        return findNode(x, root->left);
    } else {
        return findNode(x, root->right);
    }
}

void printInorder(address root) {
    if (root != Nil) {
        printInorder(root->left);
        cout << root->info << " - ";
        printInorder(root->right);
    }
}
```

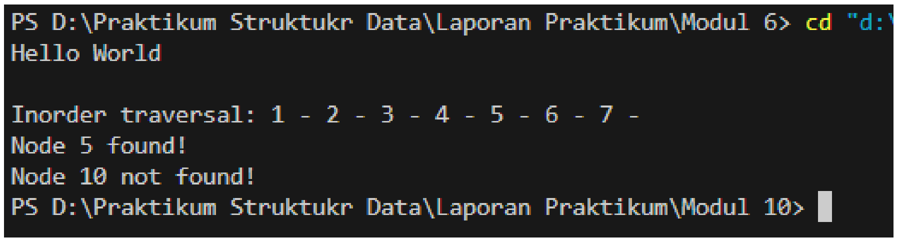
Kendaraan.h



Modul 10 > C bstree.h > ...

```cpp
#ifndef BSTREE_H
#define BSTREE_H

typedef int infotype;
typedef struct Node* address;

struct Node {
    infotype info;
    address left;
    address right;
};

const address Nil = nullptr;

address alokasi(infotype x);
void insertNode(address &root, infotype x);
address findNode(infotype x, address root);
void printInorder(address root);

#endif
```

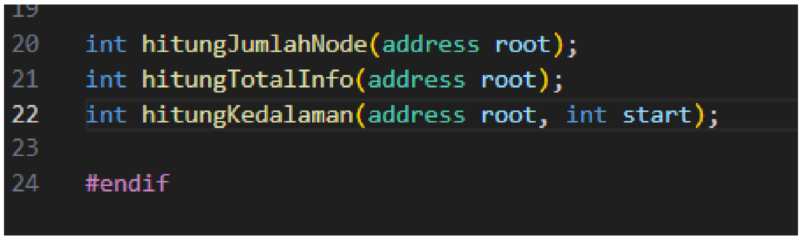
Screenshots Output



```
PS D:\Praktikum Struktukr Data\Laporan Praktikum\Modul 6> cd "d:\
Hello World

Inorder traversal: 1 - 2 - 3 - 4 - 5 - 6 - 7 -
Node 5 found!
Node 10 not found!
PS D:\Praktikum Struktukr Data\Laporan Praktikum\Modul 10> 
```

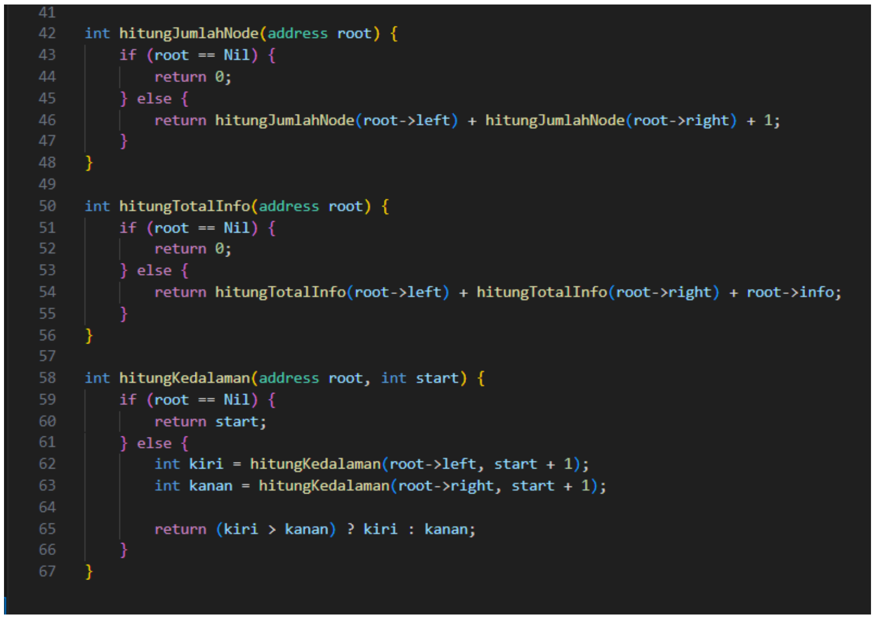
Guided 2

Menambahkan fungsi pada bstree.h



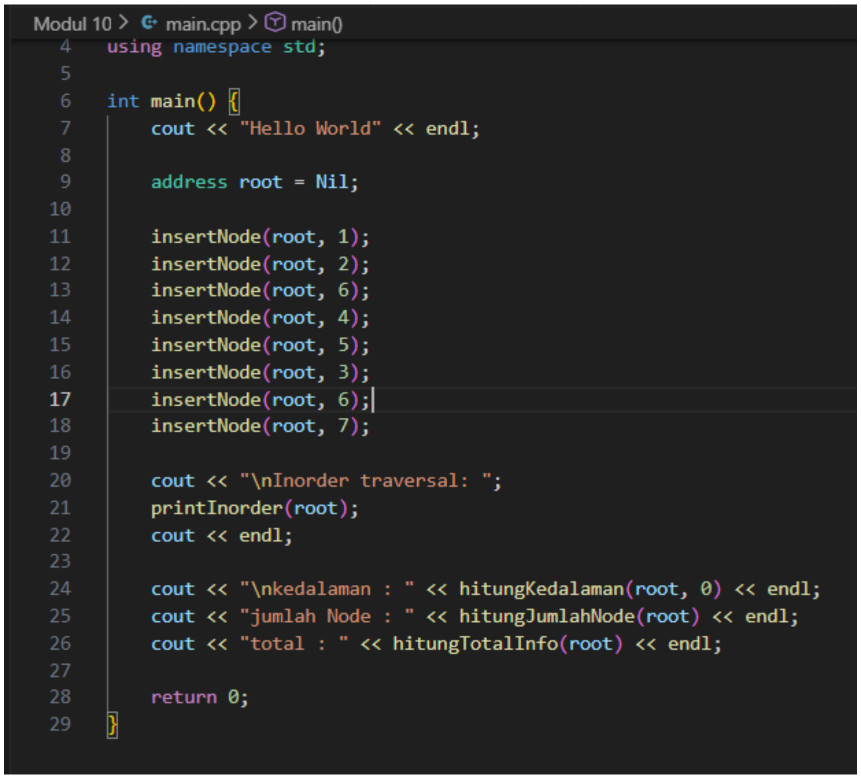
```cpp
int hitungJumlahNode(address root);
int hitungTotalInfo(address root);
int hitungKedalaman(address root, int start);

#endif
```

Bstree.cpp



```cpp

int hitungJumlahNode(address root) {
    if (root == Nil) {
        return 0;
    } else {
        return hitungJumlahNode(root->left) + hitungJumlahNode(root->right) + 1;
    }
}

int hitungTotalInfo(address root) {
    if (root == Nil) {
        return 0;
    } else {
        return hitungTotalInfo(root->left) + hitungTotalInfo(root->right) + root->info;
    }
}

int hitungKedalaman(address root, int start) {
    if (root == Nil) {
        return start;
    } else {
        int kiri = hitungKedalaman(root->left, start + 1);
        int kanan = hitungKedalaman(root->right, start + 1);

        return (kiri > kanan) ? kiri : kanan;
    }
}
```

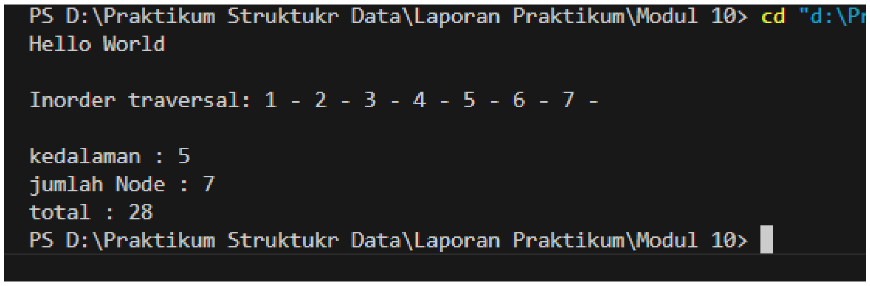
Main.cpp



Modul 10 > main.cpp > main()

```cpp
using namespace std;

int main() {
    cout << "Hello World" << endl;

    address root = Nil;

    insertNode(root, 1);
    insertNode(root, 2);
    insertNode(root, 6);
    insertNode(root, 4);
    insertNode(root, 5);
    insertNode(root, 3);
    insertNode(root, 6);
    insertNode(root, 7);

    cout << "\nInorder traversal: ";
    printInorder(root);
    cout << endl;

    cout << "\nkedalaman : " << hitungKedalaman(root, 0) << endl;
    cout << "jumlah Node : " << hitungJumlahNode(root) << endl;
    cout << "total : " << hitungTotalInfo(root) << endl;

    return 0;
}
```

Output



```
PS D:\Praktikum Struktukr Data\Laporan Praktikum\Modul 10> cd "d:\Pr
Hello World

Inorder traversal: 1 - 2 - 3 - 4 - 5 - 6 - 7 -

kedalaman : 5
jumlah Node : 7
total : 28
PS D:\Praktikum Struktukr Data\Laporan Praktikum\Modul 10>
```