LAPORAN PRAKTIKUM STRUKTUR DATA

MODUL X
TREE (BAGIAN PERTAMA)

Disusun Oleh :

NAMA : ADIKA AUNURFIKRI NOVIYANTO

NIM : 103112400195

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

C++ adalah pengembangan dari dari bahasa c yang dibuat oleh Bjarne Stroustrup sekitar tahun 1980-an. C++ disebut bahasa multi-paradigma, artinya bisa dipakai dengan gaya prosedural (pakai fungsi biasa), beriorientasi objek (pakai class dan object), atau bahkan gabungan keduanya. C++ punya dasar-dasar seperti variabel, operator percabangan (if, switch), perulangan (for, while), dan bisa memakai class untuk membuat objek.

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

Guided 1

bstree.h

```
#ifndef BSTREE_H
#define BSTREE_H

#include <iostream>
using namespace std;

typedef int infotype;

typedef struct Node *address;

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

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

#endif
```

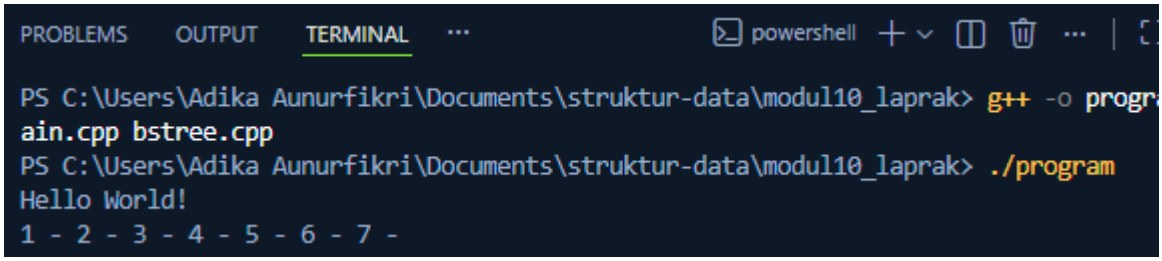
bstree..cpp

```
#include "bstree.h"

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

void insertNode(address &root, infotype x) {
    if (root == NULL) {
        root = alokasi(x);
    } else if (x < root->info) {
        insertNode(root->left, x);
    } else {
        insertNode(root->right, x);
    }
}

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

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

Main.cpp

```
#include <iostream>
#include "bstree.h"
using namespace std;

int main() {

    cout << "Hello World!" << endl;

    address root = NULL;

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

    InOrder(root);

    cout << "\n\n"

    return 0;
}
```

Screenshots Output:



Guided 2

bstree.h

```
#ifndef BSTREE_H
#define BSTREE_H

#include <iostream>
using namespace std;

typedef int infotype;

typedef struct Node *address;

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

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

int hitungJumlahNode(address root);
int hitungTotalInfo(address root);
int hitungKedalaman(address root);

#endif
```

bstree..cpp

```
#include "bstree.h"

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

void insertNode(address &root, infotype x) {
    if (root == NULL) {
        root = alokasi(x);
    } else if (x < root->info) {
        insertNode(root->left, x);
    } else {
        insertNode(root->right, x);
    }
}

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

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

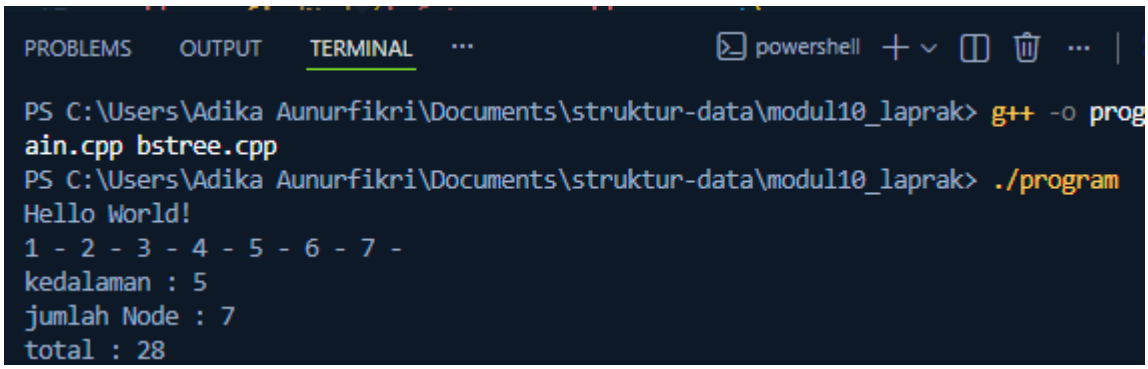
```
int hitungJumlahNode(address root) {
    if (root == NULL) return 0;
    return 1 + hitungJumlahNode(root->left) + hitungJumlahNode(root->right);
}

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

int hitungKedalaman(address root) {
    if (root == NULL) return 0;
    int L = hitungKedalaman(root->left);
    int R = hitungKedalaman(root->right);
    return 1 + max(L, R);
}
```

Main.cpp

```
#include <iostream>
#include "bstree.h"
using namespace std;

int main() {

    cout << "Hello World!" << endl;

    address root = NULL;

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

```
    insertNode(root, 7);

    InOrder(root);

    cout << "\n\n"

    cout << "kedalaman : " << hitungKedalaman(root) << endl;
    cout << "jumlah node : " << hitungJumlahNode(root) << endl;
    cout << "total : " << hitungTotalInfo(root) << endl;

    return 0;
}
```

Screenshots Output:



Guided 3

bstree.h

```
#ifndef BSTREE_H
#define BSTREE_H

#include <iostream>
using namespace std;

typedef int infotype;

typedef struct Node *address;

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

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

int hitungJumlahNode(address root);
int hitungTotalInfo(address root);
int hitungKedalaman(address root);

void preOrder(address root);
void postOrder(address root);

#endif
```

bstree..cpp

```
#include "bstree.h"


address alokasi(infotype x) {
   address P = new Node;
   P->info = x;
   P->left = NULL;
   P->right = NULL;
   return P;
```

```
}

void insertNode(address &root, infotype x) {
   if (root == NULL) {
      root = alokasi(x);
   } else if (x < root->info) {
      insertNode(root->left, x);
   } else {
      insertNode(root->right, x);
   }
}

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

void InOrder(address root) {
   if (root != NULL) {
      InOrder(root->left);
      cout << root->info << " - ";
      InOrder(root->right);
   }
}
int hitungJumlahNode(address root) {
   if (root == NULL) return 0;
   return 1 + hitungJumlahNode(root->left) + hitungJumlahNode(root->right);
}

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

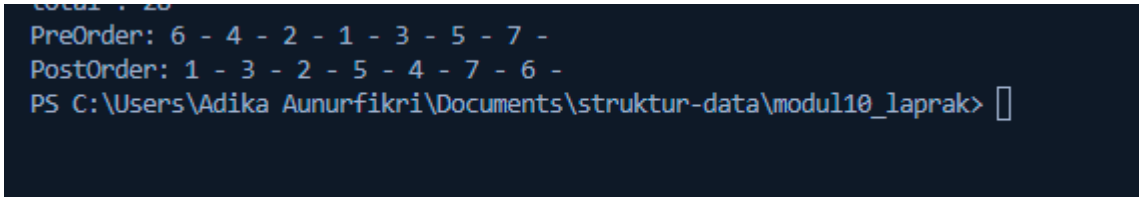
```
int hitungKedalaman(address root) {
    if (root == NULL) return 0;
    int L = hitungKedalaman(root->left);
    int R = hitungKedalaman(root->right);
    return 1 + max(L, R);
}

void preOrder(address root) {
    if (root != NULL) {
        cout << root->info << " ";
        preOrder(root->left);
        preOrder(root->right);
    }
}

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

Main.cpp

```
#include <iostream>
#include "bstree.h"
using namespace std;

int main() {

    cout << "Hello World!" << endl;

    address root = NULL;
```

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

    InOrder(root);

    cout << "\n\n"

    cout << "kedalaman : " << hitungKedalaman(root) << endl;
    cout << "jumlah node : " << hitungJumlahNode(root) << endl;
    cout << "total : " << hitungTotalInfo(root) << endl;

   cout << "\nPre-Order : ";
    preOrder(root);

    cout << "\nPost-Order: ";
    postOrder(root);

    cout << endl;

    return 0;
}
```

Screenshots Output:



Deskripsi:

Program ini mengimplementasikan struktur data *Binary Search Tree* (BST) menggunakan bahasa C++, lengkap dengan operasi dasar seperti alokasi node, penyisipan data, pencarian node, serta penelusuran *In-order*, *Pre-order*, dan *Post-order*. Selain itu, program juga menambahkan fungsi rekursif untuk menghitung jumlah node, total nilai dari seluruh node, serta kedalaman maksimal tree. Implementasi ini mengikuti ilustrasi dan aturan BST, di mana setiap nilai yang lebih kecil diletakkan di kiri dan nilai lebih besar di kanan, sehingga memungkinkan proses pencarian dan penelusuran berlangsung secara efisien.

C. Kesimpulan

Dari hasil implementasi, dapat disimpulkan bahwa BST mampu menyimpan data secara terstruktur dan memberikan kemudahan dalam operasi seperti penyisipan, pencarian, dan penelusuran. Fungsi tambahan seperti perhitungan jumlah node, total nilai, dan kedalaman tree menunjukkan bahwa BST tidak hanya efektif untuk penyimpanan data, tetapi juga mendukung analisis struktur dengan cepat melalui rekursi. Program ini berhasil menampilkan output sesuai modul dan menggambarkan bagaimana BST bekerja secara menyeluruh.

D. Referensi

- W3Resource. (2020). *C++ String Exercises: Convert digit/number to words*.
- GeeksforGeeks. (2020). *Loops in C++ (for, while, do-while).*