LAPORAN PRAKTIKUM STRUKTUR DATA

MODUL X
TREE

Disusun Oleh :
NAMA : IVAN RAMADHAN
NIM : 103112400186

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 tree.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
```

tree.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;
}
```

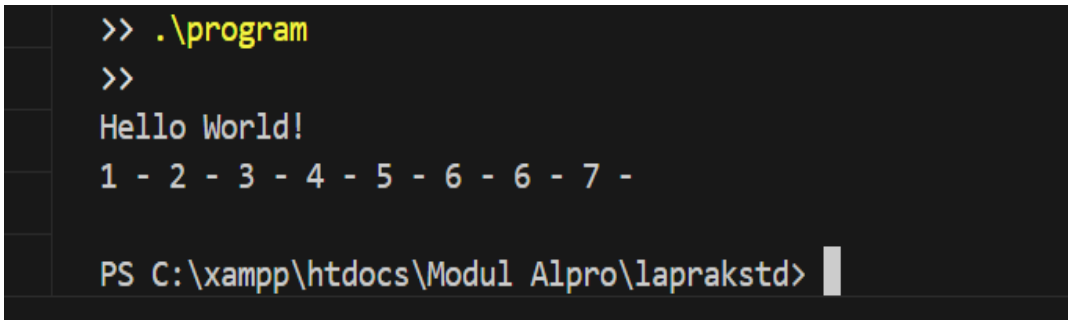
Main.cpp

```
#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);
   }
}
```

Screenshots Output:

1.



Guided 2 tree.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
```

tree.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:

1.

```
>>
Hello World!
1 - 2 - 3 - 4 - 5 - 6 - 6 - 7 -

Kedalaman : 5
Jumlah node : 8
Total : 34
PS C:\xampp\htdocs\Modul Alpro\laprakstd>
```

Guided 3 tree.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);
void preOrder(address root);
void postOrder(address root);
int hitungJumlahNode(address
root);
int hitungTotalInfo(address
root);
int hitungKedalaman(address
root);

#endif
```

tree.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);
   }
}

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 << " ";
    }
}

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);

    cout << "In-Order   : ";
    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:

1.

```
Hello World!
In-Order   : 1 - 2 - 3 - 4 - 5 - 6 - 6 - 7 -

Kedalaman  : 5
Jumlah node: 8
Total      : 34

Pre-Order  : 1 2 6 4 3 5 6 7
Post-Order : 3 5 4 7 6 6 2 1
PS C:\xampp\htdocs\Modul Alpro\laprakstd>
```

Deskripsi:

Program ini mengimplementasikan Binary Search Tree (BST) menggunakan pointer dan struktur Node yang berisi info, left, dan right. BST adalah struktur data berbasis pohon di mana nilai node di sebelah kiri selalu lebih kecil dari node induk, dan nilai node di sebelah kanan selalu lebih besar. Program ini memiliki fungsi untuk membuat node (alokasi), memasukkan node ke dalam BST (insertNode), mencari node tertentu (findNode), dan menampilkan isi BST menggunakan tiga metode traversal: In-Order, Pre-Order, dan Post-

Order. Selain itu, program juga menyediakan fungsi untuk menghitung jumlah node (hitungJumlahNode), total nilai semua node (hitungTotalInfo), dan kedalaman pohon (hitungKedalaman). Dalam main.cpp, program menambahkan beberapa elemen ke dalam BST, menampilkan traversal pohon, serta menghitung jumlah node, total nilai, dan kedalaman pohon, sehingga memudahkan pengguna memahami struktur dan sifat BST.

C. Kesimpulan

Dari program ini dapat disimpulkan bahwa BST merupakan struktur data yang efisien untuk menyimpan dan mencari data secara terurut. Penggunaan traversal In-Order menghasilkan output data secara terurut naik, sedangkan Pre-Order dan Post-Order berguna untuk berbagai keperluan seperti pencetakan struktur pohon atau penghapusan node. Fungsi tambahan seperti menghitung jumlah node, total nilai node, dan kedalaman pohon membantu mengevaluasi ukuran dan keseimbangan BST. Secara keseluruhan, program ini menunjukkan bagaimana BST dapat digunakan untuk operasi penyimpanan, pencarian, dan analisis data dengan efisien, serta memperkuat pemahaman tentang konsep pohon biner dan struktur data dinamis menggunakan pointer.

D. Referensi

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