LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 10
TREE

Disusun Oleh :
NAMA : FANDIKA PRIMADANI
NIM : 103112400231

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Binary Search Tree (BST) adalah struktur data non-linear berbasis pointer yang mengorganisir data dengan aturan hierarkis (kiri < root < kanan) untuk efisiensi pencarian, di mana manipulasinya sangat bergantung pada algoritma rekursif untuk melakukan operasi statistik (seperti menghitung kedalaman dan total node) serta metode traversal (PreOrder, InOrder, PostOrder) untuk menelusuri data sesuai kebutuhan urutan proses.

B. SOAL 1

bstree.h

```h
#ifndef BSTREE_H
#define BSTREE_H

#include <iostream>

#define Nil NULL

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 printInorder(address root);

#endif
```

bstree.cpp

```cpp
#include "bstree.h"

using namespace std;

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

```
    P->right = Nil;
    return P;
}

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);
        }
    }
}

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

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

main.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);
    insertNode(root, 7);

    printInorder(root);

    return 0;
}
```

SOAL 2

bstree1.h

```
#ifndef BSTREE_H
#define BSTREE_H

#include <iostream>

#define Nil NULL

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 printInorder(address root);
```

```
int hitungNode(address root);
int hitungTotal(address root);
int hitungKedalaman(address root, int start);

#endif
```

bstree1.cpp

```
#include "bstree.h"

using namespace std;

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

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);
        }
    }
}

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

void printInorder(address root) {
    if (root != Nil) {
```

```
        printInorder(root->left);
        cout << root->info << " - ";
        printInorder(root->right);
    }
}
int hitungNode(address root) {
    if (root == Nil) {
        return 0;
    } else {
        return 1 + hitungNode(root->left) +
hitungNode(root->right);
    }
}
int hitungTotal(address root) {
    if (root == Nil) {
        return 0;
    } else {
        return root->info + hitungTotal(root->left) +
hitungTotal(root->right);
    }
}

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);

        if (kiri > kanan) {
            return kiri;
        } else {
            return kanan;
        }
    }
}
```

main1.cpp

```
#include <iostream>
#include "bstree1.h"
#include "bstree1.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);

    printInorder(root);

    cout << "\n";
    cout << "kedalaman : " << hitungKedalaman(root, 0) <<
endl;
    cout << "jumlah Node : " << hitungNode(root) << endl;
    cout << "total : " << hitungTotal(root) << endl;

    return 0;
}
```

SOAL 3

bstree2.h

```
#ifndef BSTREE_H
#define BSTREE_H

#include <iostream>

#define Nil NULL

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 printInorder(address root);
void printPreOrder(address root);
void printPostOrder(address root);

int hitungNode(address root);
int hitungTotal(address root);
int hitungKedalaman(address root, int start);

#endif
```

queue2.cpp

```
#include "bstree.h"

using namespace std;

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

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);
        }
    }
}

address findNode(infotype x, address root) {
```

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

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

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

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

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

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

main2.cpp

```
#include <iostream>
#include "bstree2.h"
#include "bstree2.cpp"

using namespace std;

int main() {
    cout << "TUGAS 3: PreOrder & PostOrder Traversal" <<
endl;

    address root = Nil;

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

    cout << "\nPreOrder  : ";
    printPreOrder(root);

    cout << "\nInOrder   : ";
    printInorder(root);

    cout << "\nPostOrder : ";
    printPostOrder(root);

    cout << endl << endl;
    cout << "Kedalaman   : " << hitungKedalaman(root, 0) <<
endl;
    cout << "Jumlah Node : " << hitungNode(root) << endl;
    cout << "Total Info  : " << hitungTotal(root) << endl;
```

```
    return 0;
}
```

Screenshots Output

1. Output 1

```
PS D:\SEMESTER 3\Pratikum Struktur Data\Pratikum Modul10\guided\Soal> cd "d
n } ; if ($?) { .\main }
Hello World
1 - 2 - 3 - 4 - 5 - 6 - 7 -
```

2. Output 2

```
PS D:\SEMESTER 3\Pratikum Struktur Data\Pratikum Modul10\guided\Soal> cd "d:\SEMESTER 3\Pr
 ; if ($?) { .\main1 }
Hello World
1 - 2 - 3 - 4 - 5 - 6 - 7 -
kedalaman : 5
jumlah Node : 7
total : 28
```

3. Output 3

```
PS D:\SEMESTER 3\Pratikum Struktur Data> cd "d
TUGAS 3: PreOrder & PostOrder Traversal

PreOrder  : 6 - 4 - 2 - 1 - 3 - 5 - 7 -
InOrder   : 1 - 2 - 3 - 4 - 5 - 6 - 7 -
PostOrder : 1 - 3 - 2 - 5 - 4 - 7 - 6 -

Kedalaman   : 4
Jumlah Node : 7
Total Info  : 28
```

Deskripsi:

Soal 1 : Implementasi Dasar BST Membangun struktur data *Binary Search Tree* menggunakan Linked List. Fokus pada pembuatan Node, logika Insert (data kecil ke kiri, besar ke kanan), dan cetak InOrder (hasil terurut).

Soal 2 :Operasi Statistik (Rekursif) Membuat fungsi hitungan untuk mengetahui properti pohon: Jumlah Node (banyak kotak), Total Info (jumlah seluruh angka), dan Kedalaman (tinggi pohon).

Soal 3 : **Traversal Lanjutan** Melengkapi metode pembacaan data dengan **PreOrder** (Akar → kiri → kanan) dan **PostOrder** (kiri →kanan →akar) untuk melihat urutan struktur pohon yang berbeda.

C. Kesimpulan

mengimplementasikan struktur data Binary Search Tree (BST) secara utuh menggunakan C++, yang mencakup pembangunan struktur dinamis berbasis pointer, penerapan logika rekursif untuk perhitungan statistik data, serta pemahaman mendalam mengenai berbagai pola traversal (PreOrder, InOrder, PostOrder).

D. Referensi

Drozdek, Adam. *Data Structures and Algorithms in C++.* Cengage Learning, 2012

Malik, D.S. *Data Structures Using C++.* Course Technology, 2010.