Tugas Pendahuluan 6

PRAKTIKUM STRUKTUR DATA

" Tree "

Asisten :

1. Muh. Azrial Mahesa
2. Niswa Ayu Lestari

Oleh

Nama : Firman Reski Ramadhan

Nim : 60900121062

Kelas : C

LABORATORIUM KOMPUTER TERPADU

JURUSAN SISTEM INFORMASI

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI ALAUDDIN MAKASSAR

2022

Soal

1. Jelaskan Apa itu tree
2. Sebutkan Operasi-operasi pada tree dan jelaskan fungsinya masing2
3. Buat satu program sederhana dari tree

Jawaban

1. Tree/pohon merupakan struktur data yang tidak linear/non linear yang digunakan terutama untuk merepresentasikan hubungan data yang bersifat hierarkis antara elemen- elemennya.

2. Operasi Dasar Pada Pohon Biner:

   - Menyisipkan elemen.
     Menambahkan elemen tree
   - Menghapus elemen.
     Menghapus elemen tree
   - Mencari elemen.
     mencari elemen tertentu tree
   - Melintasi sebuah elemen.
     tranversal elemen tree

   Operasi Bantu Pada Pohon Biner:

   - Mencari tinggi pohon
   - Temukan tingkat pohon
   - Menemukan ukuran seluruh pohon.

3. Program Tree

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

struct node {
  int data;
  struct node *left;
  struct node *right;
};

// New node creation
struct node *newNode(int data) {
  struct node *node = (struct node *)malloc(sizeof(struct node));

  node->data = data;

  node->left = NULL;
  node->right = NULL;
  return (node);
  }

// Traverse Preorder
void traversePreOrder(struct node *temp) {
  if (temp != NULL) {
    cout << " " << temp->data;
    traversePreOrder(temp->left);
    traversePreOrder(temp->right);
  }
}
```

```
// Traverse Inorder
void traverseInOrder(struct node *temp) {
  if (temp != NULL) {
  traverseInOrder(temp->left);
    cout << " " << temp->data;
    traverseInOrder(temp->right);
  }
}

// Traverse Postorder
void traversePostOrder(struct node *temp) {
  if (temp != NULL) {
  traversePostOrder(temp->left);
  traversePostOrder(temp->right);
    cout << " " << temp->data;
  }
}

int main() {
  struct node *root = newNode(1);
  root->left = newNode(2);
  root->right = newNode(3);
  root->left->left = newNode(4);

  cout << "preorder traversal: ";
  traversePreOrder(root);
  cout << "\nInorder traversal: ";
  traverseInOrder(root);
  cout<<"\nPostorder traversal:";
  traversePostOrder(root);
  }
```