STRUKTUR DATA
Tugas BST

NAMA: Bagas Diatama Wardoyo
NPM: 140810230061

Dikumpulkan tanggal :
8 Juni 2024

Universitas Padjadjaran
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
Program Studi S-1 Teknik Informatika
2024

SOURCE CODE :

```
/* Nama program : BST
   Nama         : Bagas Diatama Wardoyo
   NPM          : 140810230061
   Tanggal buat : 08/06/2024
   Deskripsi    : BST dan semua operasinya
*********************************************************************/
#include <iostream>
#include <queue>

using namespace std;

//Membuat struct Node
struct Node
{
    int data;
    Node *L;
    Node *R;
};

typedef Node *pointer;
typedef pointer Tree;


void createTree(Tree &root); //Membuat fungsi untuk membuat Tree
void createNode(pointer &newNode); //Membuat fungsi untuk membuat Node
void insertBST(Tree &root, pointer newNode); //Memasukkan data ke Tree
void preOrder(Tree root); //Membuat fungsi untuk menampilkan Preorder
void inOrder(Tree root); //Membuat fungsi untuk menampilkan Inorder
void postOrder(Tree root); //Membuat fungsi untuk menampilkan Postorder
void levelOrder(Tree root); //Membuat fungsi untuk menampilkan Level-order

int main()
{
    Tree root;
    pointer newNode;
    int n, pilihan;

    createTree(root);

    do
    {
        cout << "\nMenu:\n";
        cout << "1. Masukkan data\n";
        cout << "2. Tampilkan Preorder\n";
        cout << "3. Tampilkan Inorder\n";
        cout << "4. Tampilkan Postorder\n";
        cout << "5. Tampilkan Level-order\n";
        cout << "6. Keluar\n";
        cout << "Pilih: ";
        cin >> pilihan;
```

```
        switch (pilihan)
        {
        case 1:
            cout << "Masukkan jumlah data: ";
            cin >> n;
            for (int i = 0; i < n; i++)
            {
                createNode(newNode);
                insertBST(root, newNode);
            }
            break;
        case 2:
            preOrder(root);
            cout << endl;
            break;
        case 3:
            inOrder(root);
            cout << endl;
            break;
        case 4:
            postOrder(root);
            cout << endl;
            break;
        case 5:
            levelOrder(root);
            cout << endl;
            break;
        case 6:
            cout << "Keluar dari program.\n";
            break;
        default:
            cout << "Pilihan tidak valid.\n";
        }
    } while (pilihan != 6);

    return 0;
}

void createTree(Tree &root)
{
    root = nullptr; //Membuat Tree kosong
}

void createNode(pointer &newNode)
{
    newNode = new Node;
    cout << "Masukkan data: ";
    cin >> newNode->data;
    newNode->L = nullptr;
    newNode->R = nullptr;
}
```

```
void insertBST(Tree &root, pointer newNode)
{
    if (root == nullptr) //Jika Tree kosong, maka root = newNode
    {
        root = newNode;
    }
    else if (newNode->data < root->data) //Jika data lebih kecil dari root,
maka masukkan ke node kiri
    {
        insertBST(root->L, newNode);
    }
    else if (newNode->data > root->data) //Jika data lebih besar dari root,
maka masukkan ke node kanan
    {
        insertBST(root->R, newNode);
    }
    else
    {
        cout << "Data sudah ada!" << endl; //Jika data sudah ada, maka tidak
masukkan data
    }
}

void preOrder(Tree root)
{
    if (root != nullptr)
        cout << root->data << " "; //Jika tidak kosong, maka tampilkan data
        preOrder(root->L); //Jalankan fungsi preOrder untuk node kiri
        preOrder(root->R); //Jalankan fungsi preOrder untuk node kanan
}

void inOrder(Tree root)
{
    if (root != nullptr)
        inOrder(root->L); //Jalankan fungsi inOrder untuk node kiri
        cout << root->data << " "; //Jika tidak kosong, maka tampilkan data
        inOrder(root->R); //Jalankan fungsi inOrder untuk node kanan
}

void postOrder(Tree root)
{
    if (root != nullptr)
        postOrder(root->L); //Jalankan fungsi postOrder untuk node kiri
        postOrder(root->R); //Jalankan fungsi postOrder untuk node kanan
        cout << root->data << " "; //Jika tidak kosong, maka tampilkan data
}

void levelOrder(Tree root)
{
    if (root == nullptr) //Jika Tree kosong, maka tidak ada data yang tersedia
        cout << "Tidak ada data yang tersedia" << endl;
```

```
    queue<Tree> q; //Membuat queue baru
    q.push(root); //Memasukkan root ke queue

    while (!q.empty()) //Jika queue tidak kosong
    {
        Tree head = q.front(); //Mengambil elemen pertama dari queue
        q.pop(); //Menghapus elemen pertama dari queue
        cout << head->data << " "; //Tampilkan data elemen head (elemen
pertama dari queue yang diambil)
        if (head->L != nullptr) //Jika node kiri tidak kosong, maka masukkan
ke queue
            q.push(head->L);
        if (head->R != nullptr) //Jika node kanan tidak kosong, maka masukkan
ke queue
            q.push(head->R);
    }
}
```

HASIL RUN :

```
Menu:
1. Masukkan data
2. Tampilkan Preorder
3. Tampilkan Inorder
4. Tampilkan Postorder
5. Tampilkan Level-order
6. Keluar
Pilih: 1
Masukkan jumlah data: 10
Masukkan data: 7
Masukkan data: 11
Masukkan data: 2
Masukkan data: 8
Masukkan data: 13
Masukkan data: 9
Masukkan data: 12
Masukkan data: 15
Masukkan data: 3
Masukkan data: 1

Menu:
1. Masukkan data
2. Tampilkan Preorder
3. Tampilkan Inorder
```

```
4. Tampilkan Postorder
5. Tampilkan Level-order
6. Keluar
Pilih: 2
7 2 1 3 11 8 9 13 12 15

Menu:
1. Masukkan data
2. Tampilkan Preorder
3. Tampilkan Inorder
4. Tampilkan Postorder
5. Tampilkan Level-order
6. Keluar
Pilih: 3
1 2 3 7 8 9 11 12 13 15

Menu:
1. Masukkan data
2. Tampilkan Preorder
3. Tampilkan Inorder
4. Tampilkan Postorder
5. Tampilkan Level-order
6. Keluar
Pilih: 4
1 3 2 9 8 12 15 13 11 7

Menu:
1. Masukkan data
2. Tampilkan Preorder
3. Tampilkan Inorder
4. Tampilkan Postorder
5. Tampilkan Level-order
6. Keluar
Pilih: 5
7 2 11 1 3 8 13 9 12 15

Menu:
1. Masukkan data
2. Tampilkan Preorder
```

```
3. Tampilkan Inorder
4. Tampilkan Postorder
5. Tampilkan Level-order
6. Keluar
Pilih: 6
Keluar dari program.
```