Modul Praktikum 4 : Binary Search Tree

Kompetensi:
Mahasiswa mampu mengimplementasikan Binary Search Tree.

Tujuan Instruksional Khusus:
Mahasiswa mampu mengimplementasikan binary search tree, dengan operasi meliputi:

1. Penambahan Node
2. Binary Tree Traversal
   a. Level Order
   b. Preorder
   c. Inorder
   d. Postorder
3. Pencarian data pada Node
4. Penghapusan Node

Praktikum:

1. Peralatan
   a. Perangkat komputer
   b. Sistem Operasi Windows
   c. C++ Compiler (MinGW)
   d. Text Editor (Geany, Notepad++, Visual Studio Code, SublimeText, dll)

2. Prosedur
   a. Baca dan pahami semua tahapan praktikum dengan cermat.
   b. Gunakan fasilitas yang disediakan dengan penuh rasa tanggung jawab.
   c. Rapihkan kembali setelah menggunakan komputer (mouse, keyboard, kursi, dll)
   d. Perhatikan sikap anda untuk tidak mengganggu rekan praktikan lain.
   e. Pastikan diri anda tidak menyentuh sumber listrik.

3. Teori Binary Search Tree
   a. Stack
      Struktur Node :

```
struct Node{
      int data;
      Node *left;
      Node *right;
};

Node *root=NULL, *n=NULL, *temp=NULL;
```

      Pembuatan Node Baru:

```
Node *createNode(int data){
      n = new Node;
      n->data = data;
      n->left = NULL;
      n->right = NULL;
      return n;
}
```

Proses Insert node pada BST:

```
void insert(Node *&root, int data){
	if (root==NULL)
		root = createNode(data);
	else if (data <= root->data)
		insert(root->left, data);
	else if (data > root->data)
		insert(root->right, data);
}
```

Binary Tree Traversal menggunakan Level Order:

```
void levelOrder(){
	if (root==NULL) return;

	queue<Node*> q;
	q.push(root);
	while(!q.empty()){
		cout << q.front()->data << " ";
		if(q.front()->left != NULL)
			q.push(q.front()->left);
		if(q.front()->right != NULL)
			q.push(q.front()->right);
		q.pop();
	}
}
```

Binary Tree Traversal menggunakan Preorder:

```
void preOrder(Node *root){
	if (root==NULL) return;

	cout << root->data << " ";
	preOrder(root->left);
	preOrder(root->right);
}
```

Binary Tree Traversal menggunakan Inorder:

```
void inOrder(Node *root){
	if (root==NULL) return;

	inOrder(root->left);
	cout << root->data << " ";
	inOrder(root->right);
}
```

Binary Tree Traversal menggunakan Postorder:

```
void postOrder(Node *root){
	if (root==NULL) return;

	postOrder(root->left);
	postOrder(root->right);
	cout << root->data << " ";
}
```

Pencarian data pada BST:

```
bool cari(Node *root, int data){
	if (root==NULL)
		return false;
	else if(data < root->data)
		return cari(root->left, data);
	else if(data > root->data)
		return cari(root->right, data);
	else
		return true;
}
```

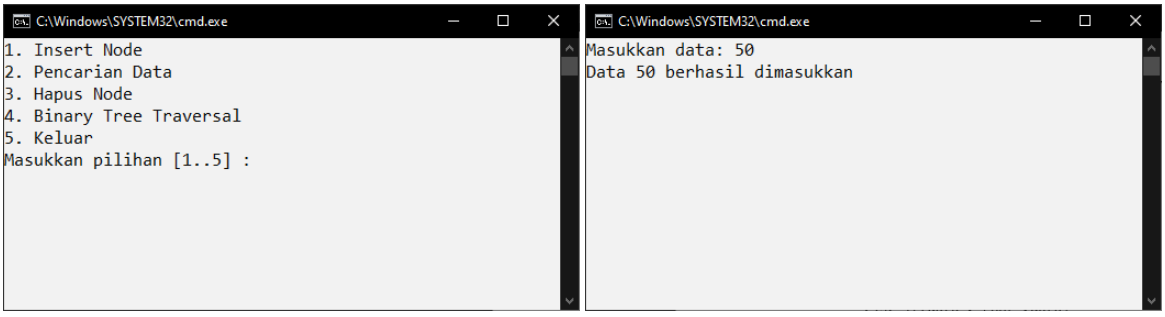
Mencari alamat memori node yang memiliki nilai minimum:

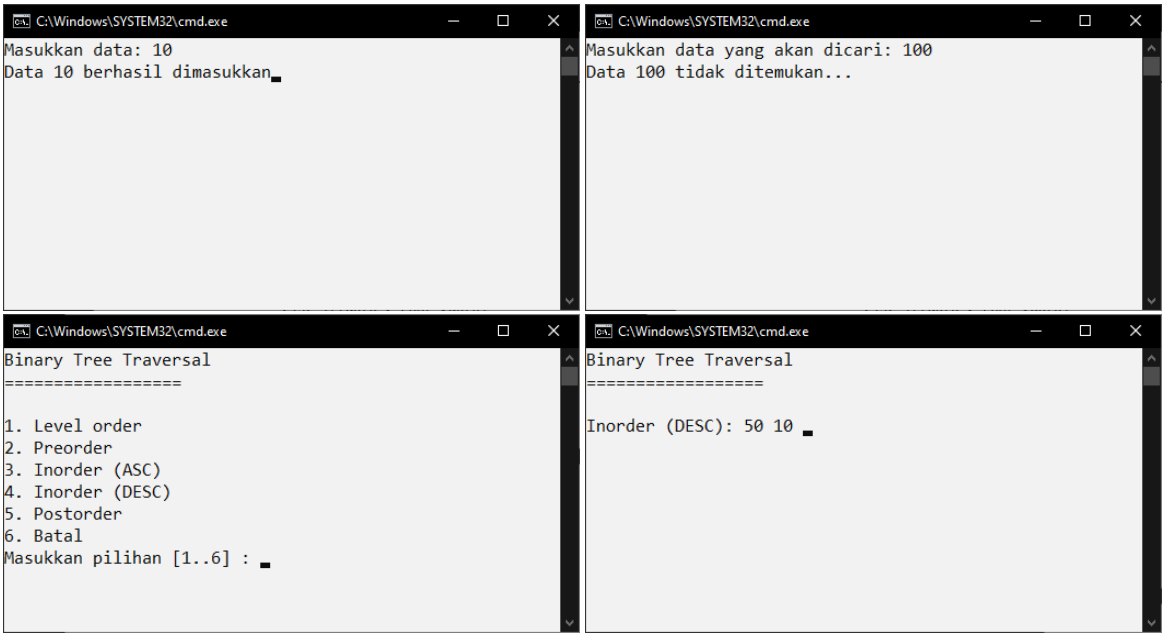
```
Node *cariMin(Node *root){
	if (root==NULL)
		return NULL;
	while(root->left != NULL)
		root = root->left;
	return root;
}
```

Penghapusan node pada BST:

```
void hapus(Node *&root, int data){
	if (root==NULL)
		return;
	else if(data < root->data)
		return hapus(root->left, data);
	else if(data > root->data)
		return hapus(root->right, data);
	else{
		// kasus I
		if (root->left == NULL && root->right == NULL){
			delete(root);
			root=NULL;
		}
		// kasus II
		else if (root->left == NULL){
			temp = root;
			root = root->right;
			delete(temp);
			temp=NULL;
		}
		else if (root->right == NULL){
			temp = root;
			root = root->left;
			delete(temp);
			temp=NULL;
		}
		// kasus III
		else{
			temp = cariMin(root->right);
			root->data = temp->data;
			hapus(root->right, root->data);
		}
	}
}
```

4. Kegiatan Praktikum:
   Buat program untuk implementasi Binary search tree meliputi operasi: insert, search, delete, dan binary tree traversal (level order, preorder, inorder dan postorder). Output dapat dilihat pada Gambar di bawah.





Untuk lebih jelasnya, silakan buka demo program yg telah disediakan.