LAPORAN PRAKTIKUM
ALGORITMA & STRUKTUR DATA
MODUL 7

TREE (POHON)

Oleh:

Rika Fauliana Rahmi NIM. 2410817120017

PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS LAMBUNG MANGKURAT

JUNI 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM ALGORITMA & STRUKTUR DATA
MODUL 7

Laporan Praktikum Algoritma & Struktur Data Modul 7: Tree (Pohon) ini disusun sebagai syarat lulus mata kuliah Praktikum Algoritma & Struktur Data. Laporan Prakitkum ini dikerjakan oleh:

Nama Praktikan : Rika Fauliana Rahmi
NIM : 2410817120017

Menyetujui,	Mengetahui,
Asisten Praktikum	Dosen Penanggung Jawab Praktikum
Muhammad Fauzan Ahsani	Muti'a Maulida, S.Kom., M.TI.
NIM. 2310817310009	NIP. 198810272019032013

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

SOAL PRAKTIKUM

Cobalah program berikut, perbaiki output, lengkapi fungsi inOrder dan postOrder pada coding, running, simpan program !

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <iostream>

using namespace std;
struct Node
{
    int data;
    Node *kiri;
    Node *kanan;
};

void tambah(Node **root, int databaru)
{
    if (*root == NULL)
    {
        Node *baru;
        baru = new Node;
        baru->data = databaru;
        baru->kiri = NULL;
        baru->kanan = NULL;
        (*root) = baru;
        (*root)->kiri = NULL;
        (*root)->kanan = NULL;
        cout << "Data bertambah";
    }
    else if (databaru < (*root)->data)
        tambah(&(*root)->kiri, databaru);
    else if (databaru > (*root)->data)
        tambah(&(*root)->kanan, databaru);
    else if (databaru == (*root)->data)
        cout << "Data sudah ada";
}

void preOrder(Node *root)
{
    if (root != NULL)
    {
        cout << root->data;
        preOrder(root->kiri);
        preOrder(root->kanan);
    }
}

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

```
49
50
51
52
53
54  }
55
56  void postOrder(Node *root)
57  {
58
59
60
61
62
63
64  }
65
66  int main()
67  {
68      int pil, data;
69      Node *pohon;
70      pohon = NULL;
71      do
72      {
73          system("cls");
74          cout << "1. Tambah\n";
75          cout << "2. PreOrder\n";
76          cout << "3. inOrder\n";
77          cout << "4. PostOrder\n";
78          cout << "5. Exit\n";
79          cout << "\nPilihan : ";
80          cin >> pil;
81          switch (pil)
82          {
83          case 1:
84              cout << "\n INPUT : ";
85              cout << "\n ---------";
86              cout << "\n Data baru : ";
87              cin >> data;
88              tambah(&pohon, data);
89              break;
90          case 2:
91              cout << "PreOrder";
92              cout << "\n------------------------\n";
93              if (pohon != NULL)
94              {
95                  preOrder(pohon);
96              }
```

```
            else
                cout << "Masih Kosong";
            break;
        case 3:
            cout << "InOrder";
            cout << "\n------------------------\n";
            if (pohon != NULL)
            {
                inOrder(pohon);
            }
            else
                cout << "Masih Kosong";
            break;
        case 4:
            cout << "PostOrder";
            cout << "\n------------------------\n";
            if (pohon != NULL)
            {
                postOrder(pohon);
            }
            else
                cout << "Masih Kosong";
            break;
        case 5:
            return 0;
        }
        _getch();
    } while (pil != 5);
    return EXIT_FAILURE;
}
```

PEMBAHASAN

A. Source Code

Tabel 1 Source Code Soal Praktikum

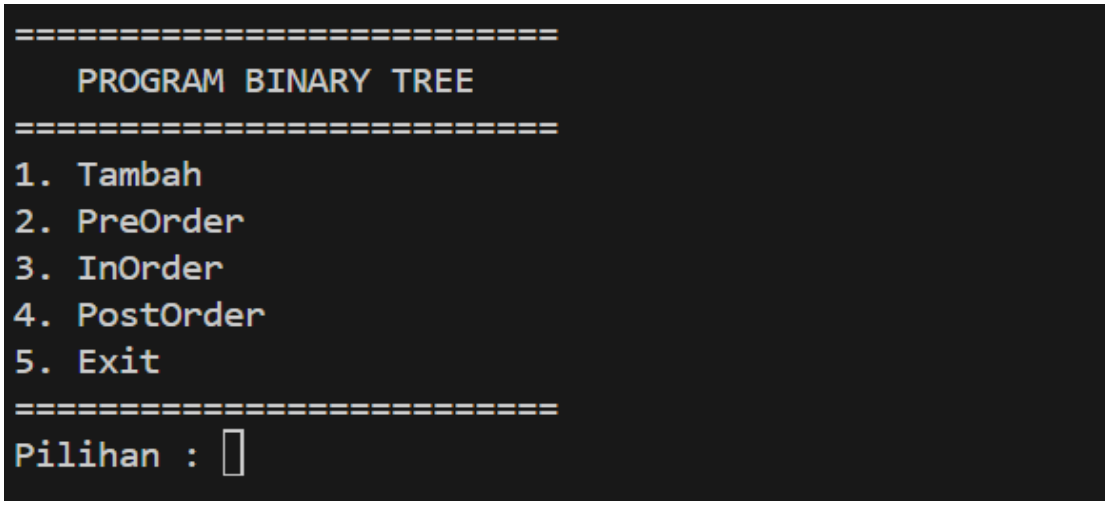
```
1   #include <stdio.h>
2   #include <conio.h>
3   #include <stdlib.h>
4   #include <iostream>
5
6   using namespace std;
7
8   struct Node {
9       int data;
10      Node *kiri;
11      Node *kanan;
12  };
13
14  void tambah(Node **root, int databaru) {
15      if (*root == NULL) {
16          Node *baru = new Node;
17          baru->data = databaru;
18          baru->kiri = NULL;
19          baru->kanan = NULL;
20          (*root) = baru;
21          cout << "Data bertambah";
22      } else if (databaru < (*root)->data) {
23          tambah(&(*root)->kiri, databaru);
24      } else if (databaru > (*root)->data) {
25          tambah(&(*root)->kanan, databaru);
26      } else {
27          cout << "Data sudah ada";
28      }
29  }
30
31  void preOrder(Node *root) {
32      if (root != NULL) {
33          cout << root->data << " ";
34          preOrder(root->kiri);
35          preOrder(root->kanan);
36      }
37  }
38
39  void inOrder(Node *root) {
40      if (root != NULL) {
```

```
41          inOrder(root->kiri);
42          cout << root->data << " ";
43          inOrder(root->kanan);
44      }
45  }
46
47  void postOrder(Node *root) {
48      if (root != NULL) {
49          postOrder(root->kiri);
50          postOrder(root->kanan);
51          cout << root->data << " ";
52      }
53  }
54
55  void freeTree(Node* root) {
56      if (root != NULL) {
57          freeTree(root->kiri);
58          freeTree(root->kanan);
59          delete root;
60      }
61  }
62
63  int main() {
64      int pil, data;
65      Node *pohon = NULL;
66
67      do {
68          system("cls");
69          cout << "=========================\n";
70          cout << "   PROGRAM BINARY TREE\n";
71          cout << "=========================\n";
72          cout << "1. Tambah\n";
73          cout << "2. PreOrder\n";
74          cout << "3. InOrder\n";
75          cout << "4. PostOrder\n";
76          cout << "5. Exit\n";
77          cout << "=========================\n";
78          cout << "Pilihan : ";
79          cin >> pil;
80
81          switch (pil) {
82              case 1:
83                  cout << "INPUT :";
84                  cout << "\n-----------------------
    \n";
```

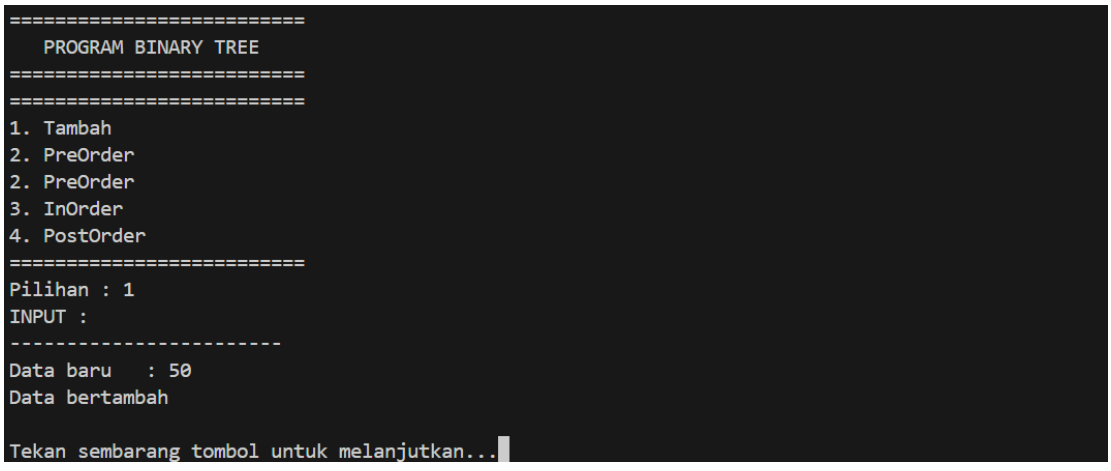
```
85                  cout << "Data baru   : ";
86                  cin >> data;
87                  tambah(&pohon, data);
88                  break;
89
90              case 2:
91                  cout << "PreOrder";
92                  cout << "\n-----------------------
    \n";
93                  if (pohon != NULL) {
94                      preOrder(pohon);
95                  } else {
96                      cout << "Masih Kosong";
97                  }
98                  break;
99
100             case 3:
101                 cout << "InOrder";
102                 cout << "\n-----------------------
    \n";
103                 if (pohon != NULL) {
104                     inOrder(pohon);
105                 } else {
106                     cout << "Masih Kosong";
107                 }
108                 break;
109
110             case 4:
111                 cout << "PostOrder";
112                 cout << "\n-----------------------
    \n";
113                 if (pohon != NULL) {
114                     postOrder(pohon);
115                 } else {
116                     cout << "Masih Kosong";
117                 }
118                 break;
119
120             case 5:
121                 freeTree(pohon);
122                 cout    <<    "\nTree    berhasil
    dibersihkan. Keluar program.\n";
123                 return 0;
124
125             default:
```

```
126                     cout << "Pilihan tidak valid!\n";
127                     break;
128             }
129
130             cout << "\n\nTekan sembarang tombol untuk
    melanjutkan...";
131             _getch();
132
133         } while (pil != 5);
134
135         return EXIT_FAILURE;
136 }
```

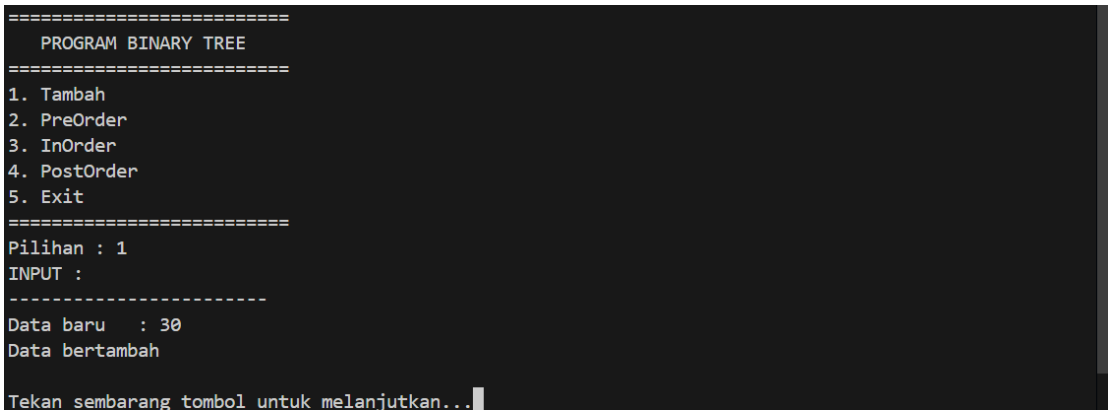
B. Output Program



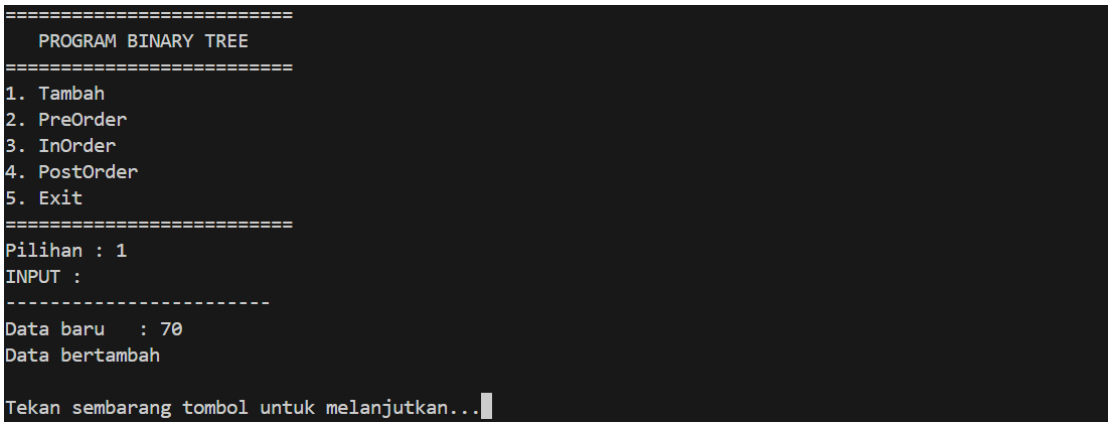
Gambar 1 Screenshot Tampilan Awal Program



Gambar 2 Screenshot Tampilan Ketika Menginput Data



Gambar 3 Screenshot Tampilan Ketika Menginput Data



Gambar 4 Screenshot Tampilan Ketika Menginput Data

Gambar 5 Screenshot Tampilan Ketika Menginput Data

```
==========================
   PROGRAM BINARY TREE
==========================
1. Tambah
2. PreOrder
3. InOrder
4. PostOrder
5. Exit
==========================
Pilihan : 1
INPUT :
-----------------------
Data baru   : 40
Data bertambah

Tekan sembarang tombol untuk melanjutkan...
```

Gambar 6 Screenshot Tampilan Ketika Menginput Data

```
==========================
   PROGRAM BINARY TREE
==========================
1. Tambah
2. PreOrder
3. InOrder
4. PostOrder
5. Exit
==========================
Pilihan : 1
INPUT :
-----------------------
Data baru   : 30
Data sudah ada

Tekan sembarang tombol untuk melanjutkan...
```

Gambar 7 Screenshot Tampilan Ketika Menginput Data

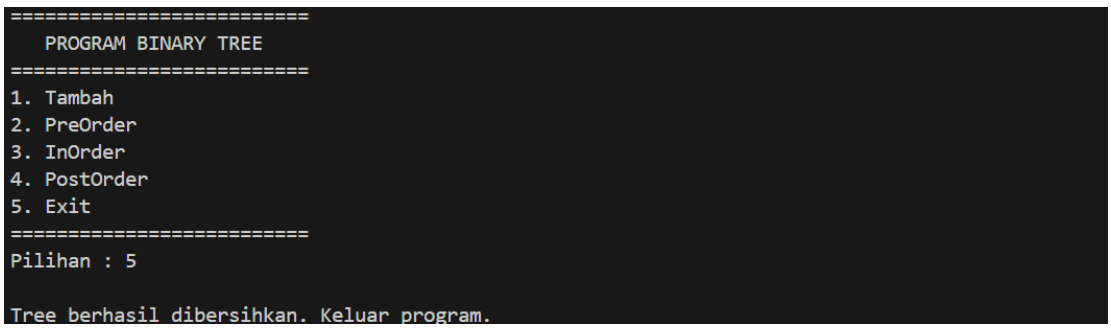
```
==========================
   PROGRAM BINARY TREE
==========================
1. Tambah
2. PreOrder
3. InOrder
4. PostOrder
5. Exit
==========================
Pilihan : 2
PreOrder
-----------------------
50 30 20 40 70

Tekan sembarang tombol untuk melanjutkan...
```

Gambar 8 Screenshot Tampilan Ketika Memilih Opsi PreOrder

Gambar 9 Screenshot Tampilan Ketika Memilih Opsi InOrder

Gambar 10 Screenshot Tampilan Ketika Memilih Opsi PostOrder



Gambar 11 Screenshot Tampilan Ketika Memilih Opsi Exit

C. Pembahasan

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <iostream>
```

- `#include <stdio.h>` dan `#include <stdlib.h>`: untuk fungsi C standar seperti `system("cls")`.
- `#include <conio.h>`: untuk `getch()` (khusus compiler berbasis DOS/Windows).
- `#include <iostream>`: untuk `cout` dan `cin` (C++).

```
using namespace std;
```

Fungsi ini untuk menghindari keharusan menulis `std::` sebelum `cin`, `cout`, dll.

```
struct Node {
    int data;
    Node *kiri;
    Node *kanan;
};
```

Mendefinisikan struktur `Node`, yang merepresentasikan simpul dalam binary tree.

- `data` = nilai simpul.
- `kiri` = pointer ke anak kiri.
- `kanan` = pointer ke anak kanan.

```
void tambah(Node **root, int databaru) {
```

Fungsi untuk menambahkan data baru ke binary tree. Parameter `Node **root`: pointer ke pointer untuk bisa mengubah pointer tree dari dalam fungsi.

```
    if (*root == NULL) {
```

Jika posisi ini kosong (NULL), berarti tempat untuk menambahkan data ditemukan.

```
        Node *baru = new Node;
        baru->data = databaru;
        baru->kiri = NULL;
        baru->kanan = NULL;
        *root = baru;
        cout << "Data bertambah";
```

Membuat node baru, mengisi nilainya, dan menghubungkannya ke tree.

```
    } else if (databaru < (*root)->data) {
        tambah(&(*root)->kiri, databaru);
```

Jika data baru lebih kecil, rekursi ke anak kiri.

```
    } else if (databaru > (*root)->data) {
        tambah(&(*root)->kanan, databaru);
```

Jika data baru lebih besar, rekursi ke anak kanan.

```
    } else {
        cout << "Data sudah ada";
    }
}
```

Jika datanya sama dengan yang sudah ada, tampilkan bahwa data duplikat dan tidak ditambahkan.

```
void preOrder(Node *root) {
    if (root != NULL) {
        cout << root->data << " ";
        preOrder(root->kiri);
        preOrder(root->kanan);
    }
}
```

`Traversal PreOrder (Root - Left - Right)`: cetak data root, lalu subtree kiri, lalu kanan.

```
void inOrder(Node *root) {
    if (root != NULL) {
        inOrder(root->kiri);
        cout << root->data << " ";
        inOrder(root->kanan);
    }
}
```

`Traversal InOrder (Left - Root - Right)` hasilnya urutan menaik jika tree adalah BST.

```
void postOrder(Node *root) {
    if (root != NULL) {
        postOrder(root->kiri);
        postOrder(root->kanan);
        cout << root->data << " ";
    }
}
```

`Traversal PostOrder (Left - Right - Root):` berguna untuk menghapus atau membongkar tree.

```
void destroyTree(Node *root) {
    if (root != NULL) {
        destroyTree(root->kiri);
        destroyTree(root->kanan);
        delete root;
    }
}
```

Fungsi untuk menghapus semua node dari tree. Harus postorder supaya anak dihapus dulu sebelum induk.

```
int main() {
    int pil, data;
    Node *pohon;
    pohon = NULL;
```

- Variabel `pil` = pilihan menu.
- `data` = data input dari user.
- `pohon` = root tree, awalnya NULL.

```
    do {
        system("cls");
```

Membersihkan layar.

```
        cout << "=========================\n";
        cout << "   PROGRAM BINARY TREE\n";
        cout << "=========================\n";
        cout << "1. Tambah\n";
        cout << "2. PreOrder\n";
        cout << "3. InOrder\n";
        cout << "4. PostOrder\n";
        cout << "5. Exit\n";
        cout << "=========================\n";
        cout << "Pilihan : ";
        cin >> pil;
```

Menampilkan menu interaktif untuk memilih operasi terhadap tree.

```
        switch (pil) {
```

Menentukan aksi berdasarkan pilihan user.

```
            case 1:
                cout << "INPUT :";
                cout << "\n----------------------\n";
                cout << "Data baru   : ";
```

```
                cin >> data;
                tambah(&pohon, data);
                break;
```

Case 1 : Tambah Data. Input data dari user, lalu panggil fungsi `tambah`.

```
 case 2:
                cout << "PreOrder";
                cout << "\n------------------------\n";
                if (pohon != NULL) {
                    preOrder(pohon);
                } else {
                    cout << "Masih Kosong";
                }
                break;
```

Jika tree tidak kosong, tampilkan pre-order traversal.

```
case 3:
                cout << "InOrder";
                cout << "\n------------------------\n";
                if (pohon != NULL) {
                    inOrder(pohon);
                } else {
                    cout << "Masih Kosong";
                }
                break;
```

Menampilkan traversal in-order.

```
case 4:
                cout << "PostOrder";
                cout << "\n------------------------\n";
                if (pohon != NULL) {
                    postOrder(pohon);
```

```
                } else {
                    cout << "Masih Kosong";
                }
                break;
```

Menampilkan traversal post-order.

```
case 5:
                freeTree(pohon);
                  cout << "\nTree berhasil dibersihkan.
Keluar program.\n";
                return 0;
```

Menghapus semua node dari tree, lalu keluar program.

```
            default:
                cout << "Pilihan tidak valid!\n";
                break;
```

Menangani input menu yang tidak sesuai.

```
        cout << "\n\nTekan sembarang tombol untuk
melanjutkan...";
        _getch();
```

Menunggu input sebelum kembali ke menu.

```
    } while (pil != 5);
```

Looping sampai user memilih Exit.

```
    return EXIT_FAILURE;
}
```

Jika keluar dari `do-while` tanpa `return 0`, kembalikan `EXIT_FAILURE` (tidak berhasil). Namun, ini tidak akan pernah terjadi karena `return 0` sudah ada di case 5.

GITHUB

https://github.com/DSA25-ULM/task-7-tree-traversal-rikafaulianarahmi