LAPORAN PRAKTIKUM

Matakuliah	Struktur Data
Pertemuan ke	10
Nama Praktikan	Wijayanto Agung Wibowo
NIM	22.11.4552
NILAI (diisi oleh dosen / asisten praktikum)	

A. Tujuan

Setelah praktikum ini, praktikan diharapkan dapat:

1. Memahami tipe data dasar Tree
2. Memahami tipe data bentukan Tree
3. Stuktur Program Tree menggunakan bahasa C++ Praktikum

B. Hasil Percobaan

1. Praktikum

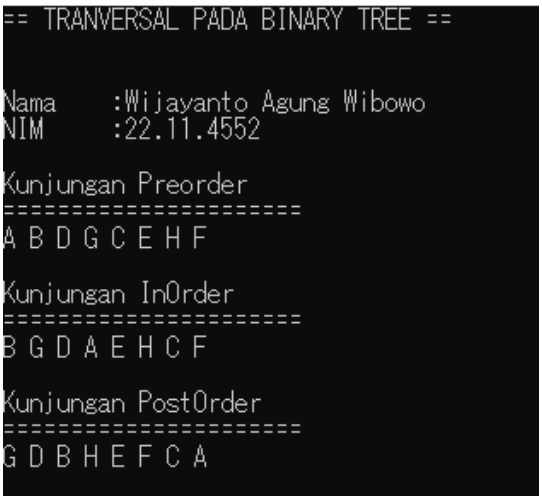
a) Tampilan Coding

```
1   //Program Binary Tree
2
3   #include <cstdio>
4   #include <cstdlib>
5
6   struct node
7   {
8       int data;
9       struct node* left;
10      struct node* right;
11
12  };
13
14  struct node* newNode(int data)
15  {
16      struct node* node = (struct node*)
17      malloc(sizeof(struct node));
18      node->data = data;
19      node->left = NULL;
20      node->right = NULL;
21
22      return(node);
23  }
24
25  void printPreorder(struct node* node)
26  {
27      if (node == NULL)
28          return;
29      printf("%C ", node->data);
30      printPreorder(node->left);
31      printPreorder(node->right);
32  }
```

```
33
34  void printInorder(struct node* node)
35  {
36      if (node == NULL)
37          return;
38      printInorder(node->left);
39      printf("%C ", node->data);
40      printInorder(node->right);
41  }
42
43  void printPostorder(struct node* node)
44  {
45      if (node == NULL)
46          return;
47      printPostorder(node->left);
48      printPostorder(node->right);
49      printf("%C ", node->data);
50  }
51
52  int main()
53  {
54      struct node* root = newNode('A');
55      root->left = newNode('B');
56      root->right = newNode('C');
57      root->left->right = newNode('D');
58      root->right->left = newNode('E');
59      root->right->right = newNode('F');
60      root->left->right->left = newNode('G');
61      root->right->left->right = newNode('H');
```

```
62
63      printf("== TRANVERSAL PADA BINARY TREE ==\n\n\n");
64
65      printf("Nama\t:Wijayanto Agung Wibowo\n");
66      printf("NIM\t:22.11.4552");
67
68
69      printf("\n\nKunjungan Preorder");
70      printf("\n======================\n");
71      printPreorder(root);
72
73      printf("\n\nKunjungan InOrder");
74      printf("\n======================\n");
75      printInorder(root);
76
77      printf("\n\nKunjungan PostOrder");
78      printf("\n======================\n");
79      printPostorder(root);
80
81      getchar();
82      return 0;
83  }
```

b) Hasil Running



c) Penjelasan

Pada program ini dideklarasikan sebuah struct yang didalamnya terdapat tiga field,

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

- Field pertama: variabel bertipe data integer, digunakan untuk menyimpan data pada node.

- Field kedua: variable pointer yang bertipe data abstrak (NODE), digunakan untuk menunjuk node anak(child) sebelah kiri.

- Field ketiga: variabel pointer yang bertipe data abtrak (NODE), digunakan untuk menunjuk node anak(child) sebelah kanan.

Pada Tree terdapat operasi Traverse yaitu operasi kunjungan terhadap node-node dalam pohon dimana masing-masing node akan dikunjungi sekali. Dalam program ini digunakan tiga tranverse, yakni :

PreOrder: node yang dikunjungi (induk), kunjungi left, kunjungi right

InOrder: kunjungi left, node yang dikunjungi (induk), kunjungi right

PostOrder: kunjungi left, kunjungi right, node yang dikunjungi (induk)

2. Latihan 1
   a) Tampilan Coding

```
1   //Program Binary Tree
2
3   #include <cstdio>
4   #include <cstdlib>
5
6   struct node
7   {
8       int data;
9       struct node* left;
10      struct node* right;
11
12  };
13
14  struct node* newNode(int data)
15  {
16      struct node* node = (struct node*)
17      malloc(sizeof(struct node));
18      node->data = data;
19      node->left = NULL;
20      node->right = NULL;
21
22      return(node);
23  }
24
25  void printPreorder(struct node* node)
26  {
27      if (node == NULL)
28          return;
29      printf("%C ", node->data);
30      printPreorder(node->left);
31      printPreorder(node->right);
32  }
```

```
33
34  void printInorder(struct node* node)
35  {
36      if (node == NULL)
37          return;
38      printInorder(node->left);
39      printf("%C ", node->data);
40      printInorder(node->right);
41  }
42
43  void printPostorder(struct node* node)
44  {
45      if (node == NULL)
46          return;
47      printPostorder(node->left);
48      printPostorder(node->right);
49      printf("%C ", node->data);
50  }
51
```

```c
int main()
{
    struct node* root = newNode(20);
    root->left = newNode(10);
    root->right = newNode(28);
    root->left->left = newNode(6);
    root->left->right = newNode(15);
    root->right->right = newNode(35);
    root->left->left->right = newNode(8);
    root->left->left->left = newNode(2);
    root->left->left->right->left = newNode(7);
    root->right->right->right = newNode(40);
    root->left->right->left = newNode(12);
    root->left->left->right->right = newNode(9);
    root->right->left = newNode(25);
```

```c

    printf("Wijayanto Agung Wibowo\n");
    printf("22.11.4552\n");
    printf("\n== TRAVERSAL PADA BINARY TREE ==");

    printf("\n\nKunjungan Preoreder");
    printf("\n==================\n");
    printPreoder(root);

    printf("\n\nKunjungan Inoreder");
    printf("\n==================\n");
    printInorder(root);

    printf("\n\nKunjungan Postorder");
    printf("\n==================\n");
    printPostorder(root);

    getchar();
    return 0;
}

```

b) Hasil Running

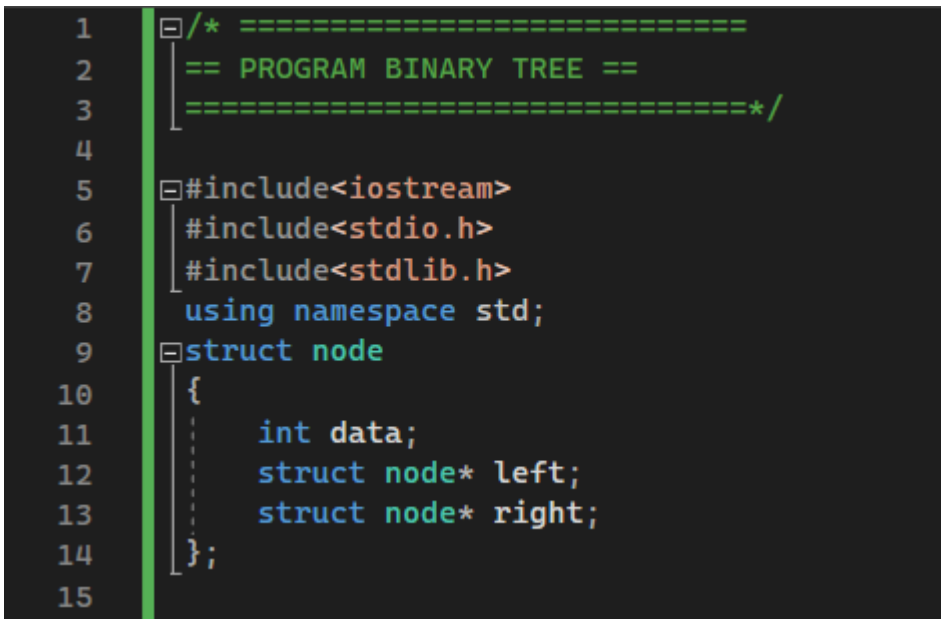
```
Wijayanto Agung Wibowo
22.11.4552

== TRAVERSAL PADA BINARY TREE ==

Kunjungan Preoreder
==================
20 10 6 2 8 7 9 15 12 28 25 35 40

Kunjungan Inoreder
==================
2 6 7 8 9 10 12 15 20 25 28 35 40

Kunjungan Postorder
==================
2 7 9 8 6 12 15 10 25 40 35 28 20
```

c) Penjelasan

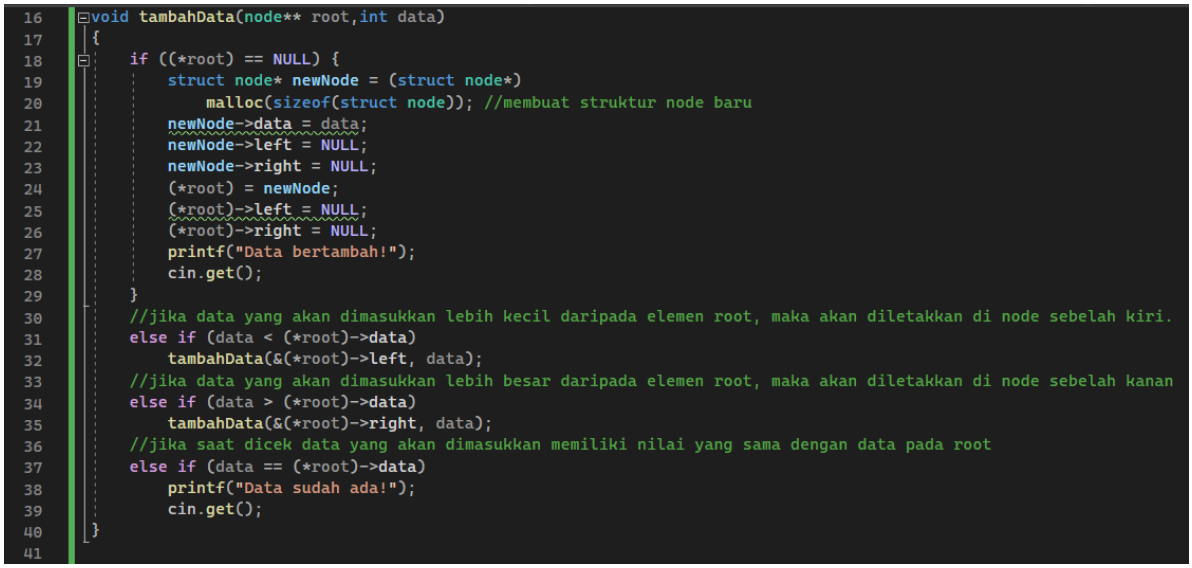
Penginputan data tree bisa dengan manual. Tetapi terdapat kaidah tertentu, yang mengharuskan pengisian data jika data yang dimasukan itu kurang dari root, maka data harus dimasukan ke child sebelah kiri. Namun apabila data lebih besar dari root, maka data harus dimasukan ke dalam child sebelah kanan.

Untuk melihat data yang di sort, bisa menggunakan kunjungan inorder.

3. Challenge
   a) Tampilan Coding



```
/* ================================
== PROGRAM BINARY TREE ==
================================*/

#include<iostream>
#include<stdio.h>
#include<stdlib.h>
 using namespace std;
struct node
{
    int data;
    struct node* left;
    struct node* right;
};

```



```
void tambahData(node** root,int data)
{
    if ((*root) == NULL) {
        struct node* newNode = (struct node*)
            malloc(sizeof(struct node)); //membuat struktur node baru
        newNode->data = data;
        newNode->left = NULL;
        newNode->right = NULL;
        (*root) = newNode;
        (*root)->left = NULL;
        (*root)->right = NULL;
        printf("Data bertambah!");
        cin.get();
    }
    //jika data yang akan dimasukkan lebih kecil daripada elemen root, maka akan diletakkan di node sebelah kiri.
    else if (data < (*root)->data)
        tambahData(&(*root)->left, data);
    //jika data yang akan dimasukkan lebih besar daripada elemen root, maka akan diletakkan di node sebelah kanan
    else if (data > (*root)->data)
        tambahData(&(*root)->right, data);
    //jika saat dicek data yang akan dimasukkan memiliki nilai yang sama dengan data pada root
    else if (data == (*root)->data)
        printf("Data sudah ada!");
        cin.get();
}

```

```
41
42  void printPreoder(struct node* node)
43  {
44      if (node == NULL)
45          return;
46      //cout<< node->data << " ";
47      printf("%d ", node->data);
48      printPreoder(node->left);
49      printPreoder(node->right);
50  }
51
52  void printInorder(struct node* node)
53  {
54      if (node == NULL)
55          return;
56      printInorder(node->left);
57      printf("%d ", node->data);
58      //cout<< node->data << " ";
59      printInorder(node->right);
60  }
61
62  void printPostorder(struct node* node)
63  {
64      if (node == NULL)
65          return;
66      printPostorder(node->left);
67      printPostorder(node->right);
68      printf("%d ", node->data);
69      //cout<< node->data << " ";
70  }
```

```
71  int main()
72  {
73      int data,pilihan = 0;
74      node* tree; //Membyat pointer tree;
75      tree = NULL; //inisialisasi node pohon
76      while (pilihan != 3) {
77          system("cls");
78          printf("\nMasukan Pilihan Menu : ");
79          printf("\n1. Input Data ");
80          printf("\n2. Menampilkan Kunjungan ");
81          printf("\n3. Exit");
82          printf("\nPilih: ");
83          scanf_s("%d", &pilihan);
84          switch (pilihan) {
85          case 1:
86              system("cls");
87              printf("\nINPUT : ");
88              printf("\n-------");
89              printf("\nData baru : ");
90              scanf_s("%d", &data);
91              //panggil fungsi untuk menambah node yang berisi data pada tree
92              tambahData(&tree, data);
93              break;
94          case 2:
95              system("cls");
96              printf("Wijayanto Agung Wibowo\n");
97              printf("22.11.4552\n");
98              printf("\n== TRAVERSAL PADA BINARY TREE ==");
99
100             printf("\n\nKunjungan Preoreder");
101             printf("\n==================\n");
102             printPreoder(tree);
```

```
103
104                 printf("\n\nKunjungan Inoreder");
105                 printf("\n====================\n");
106                 printInorder(tree);
107
108                 printf("\n\nKunjungan Postorder");
109                 printf("\n====================\n");
110                 printPostorder(tree);
111
112                 system("pause");
113
114                 break;
115             }
116     }
117     return 0;
118 }
119
```

b) Hasil Running

```
Masukan Pilihan Menu :
1. Input Data
2. Menampilkan Kunjungan
3. Exit
Pilih:
```

Hasil setelah Menginput data 12,90,86,1,34,23,40 Lalu di tampilkan Kunjungan:

```
Wijayanto Agung Wibowo
22.11.4552

== TRAVERSAL PADA BINARY TREE ==

Kunjungan Preoreder
====================
12 1 90 34 23 40 86

Kunjungan Inoreder
====================
1 12 23 34 40 86 90

Kunjungan Postorder
====================
1 23 86 40 34 90 12 Press any key to continue . .
```

c) Penjelasan

Fungsi tambahData menerima dua parameter: sebuah double pointer ke akar pohon pencarian biner (node** root) dan nilai data yang akan dimasukkan (int data).

Fungsi ini memulai dengan memeriksa apakah pointer ke root ((*root)) adalah NULL, yang menandakan bahwa pohon masih kosong. Jika pohon kosong, sebuah node baru dibuat menggunakan alokasi memori dinamis (malloc) dan nilai data yang diberikan ditetapkan pada node baru tersebut. Pointer kiri dan kanan dari node baru diatur menjadi NULL, dan pointer root

diperbarui agar menunjuk ke node baru tersebut. Pesan "Data bertambah!" dicetak, dan cin.get() digunakan untuk menjeda program hingga pengguna menekan tombol Enter.

- Jika pohon tidak kosong, fungsi membandingkan nilai data dengan nilai data dari node saat ini ((*root)->data) untuk menentukan posisi yang tepat untuk penyisipan.

-Jika data kurang dari (*root)->data, maka fungsi tambahData akan dipanggil secara rekursif dengan pointer ke anak kiri sebagai parameter.

- Jika data lebih besar dari (*root)->data, maka fungsi tambahData akan dipanggil secara rekursif dengan pointer ke anak kanan sebagai parameter.

- Jika data sama dengan (*root)->data, pesan "Data sudah ada!" akan dicetak, dan cin.get() digunakan untuk menjeda program.

Fungsi ini bertujuan untuk membangun sebuah pohon pencarian biner dengan mengatur data secara terurut. Node-node dengan nilai lebih kecil ditempatkan di sebelah kiri, sedangkan node-node dengan nilai lebih besar ditempatkan di sebelah kanan.

C. Kesimpulan
Setelah melakukan percobaan pada Latihan 1 sampai challenge, saya dapat memahami pengimplementasian binary tree.
binary tree hanya bisa tercipta oleh node.

D. Referensi (optional)