

PRAKTIKUM 12

TREE

A. Tujuan

Setelah praktikum ini, praktikan diharapkan dapat:

1. Memahami tipe data dasar
2. Memahami tipe data bentukan
3. Struktur Program menggunakan bahasa C++

B. Peralatan

1. PC Desktop
2. Windows 7
3. Notepad++ dan MinGW atau Dev++

C. Teori

Pohon (tree) adalah salah satu bentuk graph terhubung yang tidak mengandung sirkuit. Karena merupakan graph terhubung, maka pada pohon selalu terdapat path atau jalur yang menghubungkan setiap dua simpul dalam pohon. Struktur pohon adalah suatu cara merepresentasikan suatu struktur hirarki (one-to-many) secara grafis yang mirip sebuah pohon, walaupun pohon tersebut hanya tampak sebagai kumpulan node-node dari atas ke bawah. Suatu struktur data yang tidak linier yang menggambarkan hubungan yang hirarkis (one-to-many) dan tidak linier antara elemen-elemennya.

Ada dua macam jenis tree, yaitu :

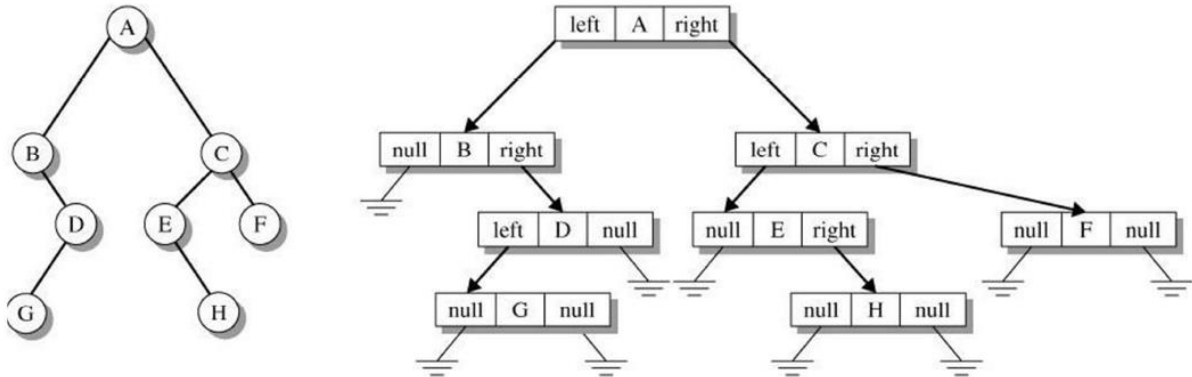
- Tree Statik: isi node-nodenya tetap karena bentuk pohonnya sudah ditentukan.
- Tree Dinamik: isi nodenya berubah-ubah karena proses penambahan (insert) dan penghapusan (delete)

Suatu bentuk pohon dilengkapi dengan apa yang disebut “akar” atau “root”. Pohon yang salah satu simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah dinamakan pohon berakar (rooted tree). Node root dalam sebuah tree adalah suatu node yang memiliki hiarki tertinggi dan dapat juga memiliki node-node anak. Semua node dapat ditelusuri dari node root tersebut. Node root adalah node khusus yang tercipta pertama kalinya. Node-node lain di bawah node root saling terhubung satu sama lain dan disebut subtree. Berikut ini merupakan contoh penggunaan struktur pohon :

- Silsilah keluarga
- Parse Tree (pada compiler)

- Struktur File
- Struktur Organisasi
- Hasil Pertandingan dalam bentuk turnamen

Sebuah tree direpresentasikan pada gambar di bawah ini :



Abstract Tree Model

Tree Node Model

Abstract and tree node model of a binary tree.

D. PRAKTIKUM

Dibuat penerapan tree dari gambar Abstract Tree Model diatas sebagai berikut :

```

1  /* =====
2  == PROGRAM BINARY TREE==
3  ===== */
4
5  #include <stdio.h>
6  #include <stdlib.h>
7
8  struct node
9  {
10     char data;
11     struct node* left;
12     struct node* right;
13 };
14
15 struct node* newNode(int data)
16 {
17     struct node* node = (struct node*)
18     malloc(sizeof(struct node));
19     node->data = data;
20     node->left = NULL;
21     node->right = NULL;
22
23     return(node);
24 }

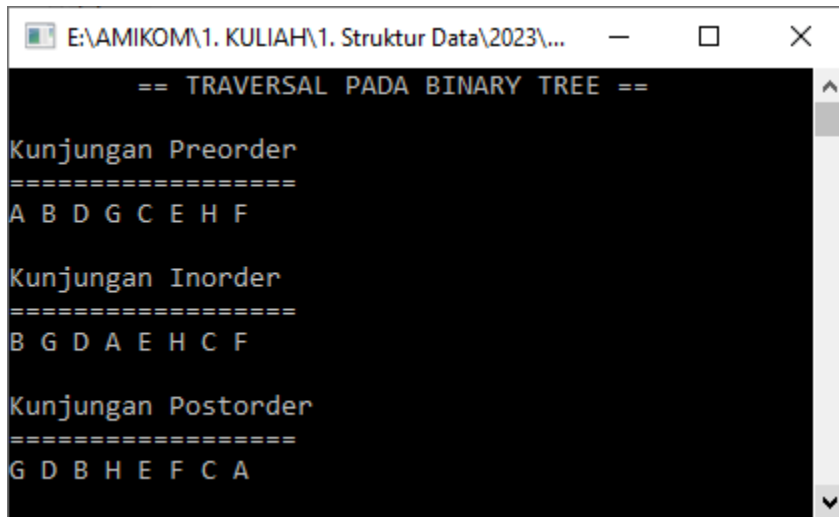
```

```

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 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("== TRAVERSAL PADA BINARY TREE ==");
64
65     printf("\n\nKunjungan Preorder");
66     printf("\n=====\n");
67     printPreorder(root);
68
69     printf("\n\nKunjungan Inorder");
70     printf("\n=====\n");
71     printInorder(root);
72
73     printf("\n\nKunjungan Postorder");
74     printf("\n=====\n");
75     printPostorder(root);
76
77     getchar();
78     return 0;
79 }

```

Hasil output program :

A screenshot of a terminal window with a black background and white text. The window title bar shows the file path "E:\AMIKOM\1. KULIAH\1. Struktur Data\2023\...". The output text is as follows:

```
== TRAVERSAL PADA BINARY TREE ==  
  
Kunjungan Preorder  
=====  
A B D G C E H F  
  
Kunjungan Inorder  
=====  
B G D A E H C F  
  
Kunjungan Postorder  
=====  
G D B H E F C A
```

E. LATIHAN

Urutan Data : 20, 10, 28, 6, 15, 35, 8, 2, 7, 40, 12, 9, 25.

Gambarkan Binary Tree secara manual dari urutan data diatas supaya data terurut.

Susunlah menjadi Binary Tree dalam source code. Kemudian lakukan Traversal Preorder, Inorder, Postorder