TUGAS PERTEMUAN 10

STRUKTUR DATA

BINARY TREE

Disusun oleh:

Rama Pramudya Wibisana 2022320019

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS INFORMATIKA

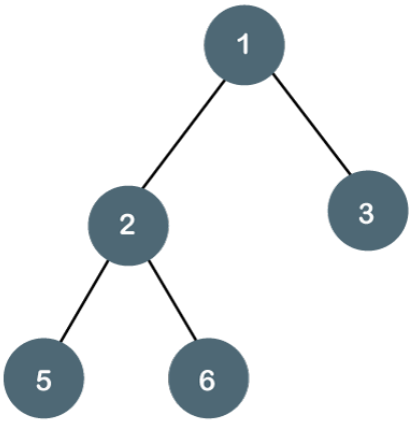
UNIVERSITAS BINA INSANI

BEKASI

2022

A. Binary Tree

Sedikit mengulas Kembali materi pertemuan 9, Binary Tree adalah Tree yang tidak boleh memiliki lebih dari 2 node dalam 1 parent. Simplenya adalah, orang tua tidak boleh memiliki lebih dari 2 anak. Untuk mengetahui lebih jelas, perhatikan contoh simple pada Binary Tree di bawah :



Dalam pemrograman, setiap lingkaran di atas disebut node. Node memiliki nilai, angka yang ada di dalam node merupakan nilai dari node tersebut.

a. Node pertama (1) yang ada di paling atas disebut root. Root itu artinya sebuah Node yang tidak memiliki parent. Dapat dilihat bahwa tidak ada lagi node di atasnya.
b. Node yang ada di level 2 ini disebut Parent (2) & (3). Parent itu ciri-cirinya, dia pasti memiliki node lagi di atasnya (root) dan memiliki anak yang maksimal jumlahnya adalah 2.
c. Node di level terakhir sebutannya bisa ada dua, ada leaf Node, atau children (5) & (6). Tapi saya lebih suka menyebutnya leaf node, node ini mempunyai ciri-ciri yaitu tidak ada lagi node setelahnya, artinya hirarki berhenti di node tersebut.

Itu tadi sedikit kilas balik dari materi pertemuan 9 atau mengenai konsep Binary Tree, selanjutnya saya akan menjelaskan seputar Binary Tree lebih jelas dengan program yang saya buat menggunakan Bahasa C++.

B. Syntax

```
1  #include <stdio.h>
2  #include <conio.h>
3  #include <stdlib.h>
4
5  struct Node
6  {
7      int data;
8      Node *left;
9      Node *right;
10 };
```

Di sini saya melakukan pendeklarasian struct. Variabel data bertipe data integer digunakan untuk menyimpan data pada node. Variabel left pointer yang bertipe data Node digunakan untuk menunjuk node child sebelah kiri. Variabel right pointer yang bertipe data Node, digunakan untuk menunjuk node child sebelah kanan.

```
12 void addData(Node **root, int newData)
13 {
14     if ((*root) == NULL)
15     {
16         Node *newNode;
17         newNode = new Node;
18         newNode->data = newData;
19         newNode->left = NULL;
20         newNode->right = NULL;
21         (*root) = newNode;
22         (*root)->left = NULL;
23         (*root)->right = NULL;
24         printf("Data Bertambah");
25     }
26     else if (newData < (*root)->data)
27         addData(&(*root)->left, newData);
28     else if (newData > (*root)->data)
29         addData(&(*root)->right, newData);
30     else if (newData == (*root)->data)
31         printf("Data Sudah Ada");
32 }
```

Fungsi ini digunakan untuk melakukan penambahan node yang berisi data pada tree. Parameternya adalah variable root pointer bertipe data Node dan variable newData bertipe data integer.

Jika data yang dimasukkan lebih kecil dari root, maka data akan diletakkan di node sebelah kiri (baris 26 & 27). Jika data yang dimasukkan lebih besar dari root, maka data akan diletakkan di node sebelah kanan (baris 28 & 29). Dan jika data yang dimasukkan sama dengan data yang sudah dimasukkan sebelumnya, maka akan mencetak string “Data Sudah Ada” (baris 30 & 31).

```
34 void preOrder(Node *root)
35 {
36     if (root != NULL)
37     {
38         printf("%d ", root->data);
39         preOrder(root->left);
40         preOrder(root->right);
41     }
42 }
```

Fungsi ini digunakan untuk mengunjungi node secara pre-order, node yang dikunjungi (induk), kunjungi left, kunjungi right.

Fungsi ini digunakan untuk mengunjungi node secara in-order, kunjungi left, node yang dikunjungi (induk), kunjungi right.

```
void inOrder(Node *root)
{
    if (root != NULL)
    {
        inOrder(root->left);
        printf("%d ", root->data);
        inOrder(root->right);
    }
}
```

```
void postOrder(Node *root)
{
    if (root != NULL)
    {
        postOrder(root->left);
        postOrder(root->right);
        printf("%d ", root->data);
    }
}
```

Fungsi ini digunakan untuk mengunjungi node secara post-order, kunjungi left, kunjungi right, node yang dikunjungi (induk).

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

Selanjutnya saya akan melakukan pendeklarasian variabel yang akan digunakan untuk melakukan penyimpanan dan pemrosesan data, serta digunakan sebagai parameter fungsi.

```
do
{
    system("cls");
    printf("1. Tambah Data\n");
    printf("2. Data Pre-Order\n");
    printf("3. Data In-Order\n");
    printf("4. Data Post-Order\n");
    printf("5. Exit\n");
    printf("Pilih Menu : ");
    scanf("%d", &pil);
```

Selanjutnya saya melakukan perulangan do while yang memiliki kondisi selama nilai variabel pil dari input user tidak bernilai 5.

Dan saya juga membuat menu yang nantinya akan terhubung dengan kondisi switch case di baris selanjutnya. Jika user memilih menu 1, maka semua pernyataan yang ada didalam case 1 akan dijalankan.

```
81  switch (pil)
82  {
83  case 1:
84      printf("\nMasukkan Data Baru : ");
85      scanf("%d", &data);
86      addData(&tree, data);
87      break;
```

Jika kita memilih menu 1, maka pernyataan dalam case 1 akan dijalankan, yaitu memanggil fungsi addData atau menambahkan data.

```
89  case 2:
90      printf("\nPRE-ORDER : ");
91      printf("\n==================\n");
92      if (tree != NULL)
93          preOrder(tree);
94      else
95          printf("Data Kosong");
96      break;
```

Jika kita memilih menu 2, maka pernyataan dalam case 2 akan dijalankan, yaitu menampilkan fungsi preOrder. Jika belum ada data yang dimasukkan, maka akan mencetak string “Data Kosong”.

```
98  case 3:
99      printf("\nIN-ORDER : ");
100     printf("\n==================\n");
101     if (tree != NULL)
102         inOrder(tree);
103     else
104         printf("Data Kosong");
105     break;
```

Jika kita memilih menu 3, maka pernyataan dalam case 3 akan dijalankan, yaitu menampilkan fungsi inOrder. Jika belum ada data yang dimasukkan, maka akan mencetak string “Data Kosong”.

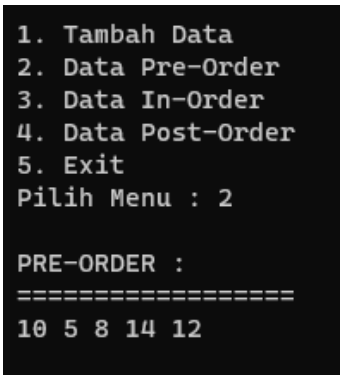
```
107 case 4:
108     printf("\nPOST ORDER : ");
109     printf("\n==================\n");
110     if (tree != NULL)
111         postOrder(tree);
112     else
113         printf("Data Kosong");
114     break;
115 }
```

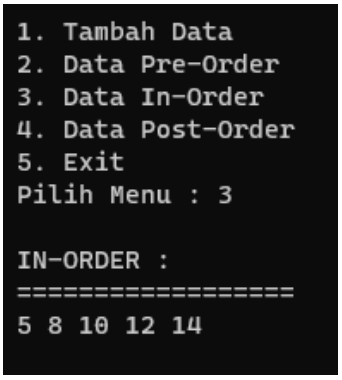
Jika kita memilih menu 4, maka pernyataan dalam case 4 akan dijalankan, yaitu menampilkan fungsi postOrder. Jika belum ada data yang dimasukkan, maka akan mencetak string “Data Kosong”.

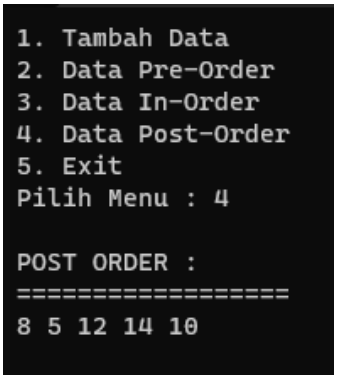
```
116         _getch();
117     } while (pil != 5);
118     return EXIT_FAILURE;
119 }
120
```

Jika memilih menu 5, maka program akan berakhir atau exit.

C. Running Program







Capture di atas adalah data yang sudah saya masukkan ke dalam program yang tadi sudah saya buat. Saya memilih menu 2 maka data yang dimasukkan akan ditampilkan dalam keadaan pre-order. Saya memilih menu 3 maka data yang dimasukkan akan ditampilkan dalam keadaan in-order. Saya memilih menu 4 maka data yang dimasukkan akan ditampilkan dalam keadaan post-order.

D. Penjelasan

Input data pertama adalah 10

Maka akan menjalankan operasi kondisi if((*root) == NULL). Karena kondisi bernilai benar maka pernyataan yang ada di dalamnya akan dijalankan, yaitu pendeklarasian node baru bernama ‘newNode’, kemudian dialokasikan memori untuk node yang baru dibuat.

Selanjutnya, dilakukan proses inisialisasi pada node yang baru dibuat
newNode->data = newData; -> baru-> data diisi nilai dari bilangan yang diinput (nilai dari variabel data) yaitu berisi 10.
newNode->left = NULL;
newNode->right = NULL;

Node newNode->left dan newNode->right diinisialisasi bernilai NULL, artinya node yang baru dibuat belum memiliki child.

Kemudian node yang baru dibuat (10) dijadikan sebagai root melalui syntax,

```
(*root) = newNode;
(*root)->left = NULL;
(*root)->right = NULL;
```

Setelah pembuatan node baru pada tree selesai dilakukan, akan tercetak string pada layar “Data bertambah”.

Input data kedua adalah 5

Maka akan menjalankan operasi kondisi else if (newData < (*root)->data). Karena 5 lebih kecil dari 10 maka letaknya berada di kiri dan 5 adalah child node dari root 10 dan akan menjadi parent.

Input data ketiga adalah 8

Maka akan menjalankan operasi kondisi else if (newData > (*root)->data). Karena 8 lebih besar dari 5 maka letaknya berada di sebelah kanan dan 8 adalah leaf node dari parent 5.

Input data keempat adalah 14

Maka akan menjalankan operasi kondisi else if (newData > (*root)->data). Karena 14 lebih besar dari 10 maka letaknya berada di kanan dan 14 adalah child node dari root 10 dan akan menjadi parent.

Input data kelima adalah 12

Maka akan menjalankan operasi kondisi else if (newData < (*root)->data). Karena 12 lebih kecil dari 14 maka letaknya berada di sebelah kiri dan 12 adalah leaf node dari parent 14.

E. Bentuk Binary Tree berdasarkan data yang diinput di atas