

BAB 5

STRUKTUR POHON (TREE)

5.1. TUJUAN PEMBELAJARAN

Tujuan dari pembelajaran pada pertemuan ini adalah mahasiswa diharapkan mampu :

1. Mampu membuat struktur pohon (tree) dengan menggunakan array.
2. Mampu membuat flowchart pembentukan tree dengan menggunakan struktur datalist.
3. Mampu mengimplementasikan tree sesuai flowchart yang sudah dibuat.

5.2. DASAR TEORI

Struktur pohon (tree) biasanya digunakan untuk menggambarkan hubungan yang bersifat hirarkis antara elemen-elemen yang ada. Contoh implementasi struktur pohon dalam kehidupan sehari-hari, dapat dilihat dari silsilah keluarga, hasil pertandingan yang berbentuk turnamen seperti catur, struktur organisasi dari sebuah perusahaan, dsb. Di dalam C++, terdapat 4 jenis struktur pohon, diantaranya pohon biner (binary tree), complete binary tree, full binary tree, dan skewed binary tree. Namun, pada bab ini yang menjadi fokus pembelajarannya, yaitu binary tree.

Berikut adalah istilah – istilah yang biasa digunakan di dalam struktur pohon (*tree*) :

Tabel 5.1 Tabel istilah dalam *tree*

Predecessor	Node yang berada diatas node tertentu.
Successor	Node yang berada dibawah node tertentu.
Ancestor	Seluruh node yang terletak sebelum node tertentu dan terletak pada jalur yang sama
Descendant	Seluruh node yang terletak setelah node tertentu dan terletak pada jalur yang sama
Parent	Predecessor satu level di atas suatu node.
Child	Successor satu level di bawah suatu node.
Sibling	Node-node yang memiliki parent yang sama
Subtree	Suatu node beserta descendantnya.
Size	Banyaknya node dalam suatu tree
Height	Banyaknya tingkatan dalam suatu tree
Root	Node khusus yang tidak memiliki predecessor.
Leaf	Node-node dalam tree yang tidak memiliki successor.
Degree	Banyaknya child dalam suatu node

5.2.1. Pohon Biner (Binary Tree)

Sebuah binary tree adalah sebuah pengorganisasian secara hirarki dari beberapa buah simpul, dimana masing-masing simpul tidak mempunyai anak lebih dari 2. Simpul yang berada di bawah sebuah simpul dinamakan anak dari simpul tersebut. Simpul yang berada di atas

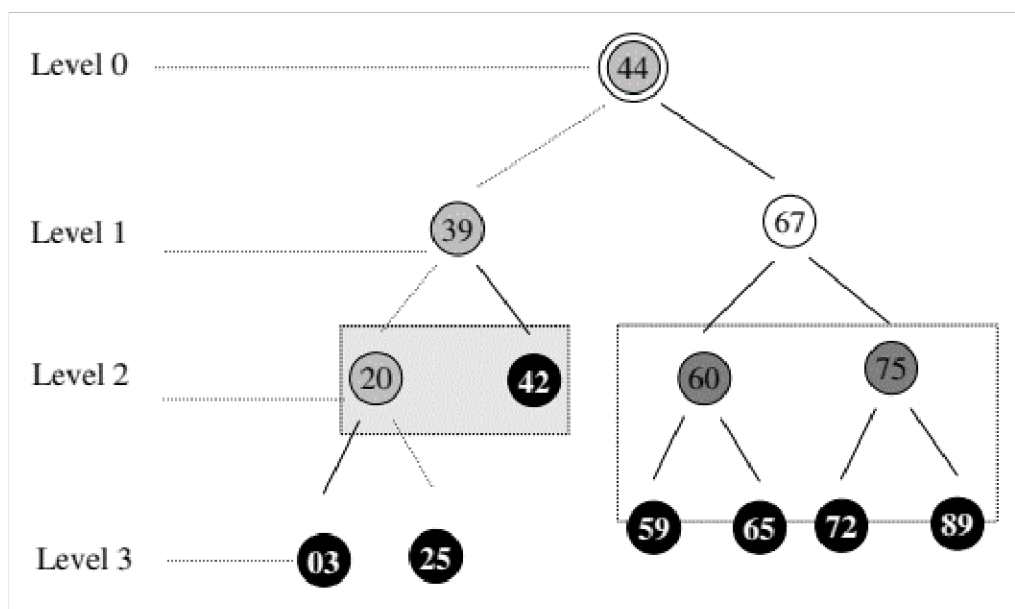
sebuah simpul dinamakan induk dari simpul tersebut. Sederhananya binary tree dapat diartikan sebagai sebuah struktur data yang menggambarkan hubungan hierarki antar elemen-elemennya, contohnya relasi one to many.

Masing-masing simpul dalam binary tree terdiri dari tiga bagian yaitu sebuah data dan dua buah pointer (pointer kiri dan kanan). Selain itu di dalam simpul juga terdapat *sibling*, *descendants*, dan *ancestors*. **Sibling** merupakan anak lain dari induk simpul tersebut. Sedangkan **descendants** merupakan semua simpul-simpul merupakan cabang (berada di bawah) simpul tersebut. Terakhir **ancestors** merupakan semua simpul yang berada di atas antara simpul tersebut dengan *root*. Biasanya penampilan dari sebuah tree akan ditampilkan dengan berat dari tree tersebut, angka yang menunjukkan jumlah level yang ada di dalamnya.

Tingkat suatu simpul ditentukan dengan pertama kali menentukan akar sebagai bertingkat 1. Jika suatu simpul dinyatakan sebagai tingkat N, maka simpul-simpul yang merupakan anaknya akan berada pada tingkatan $N + 1$.

Tinggi atau kedalaman dari suatu pohon adalah tingkat maksimum dari simpul dalam pohon dikurangi dengan 1 ($N_{maks}-1$).








Ada pun istilah derajat (degree) dari suatu simpul. Derajat suatu simpul dinyatakan sebagai banyaknya anak atau turunan dari simpul tersebut.



Gambar 5.1 Ilustrasi Sebuah Pohon Biner (Binary Tree)

Keterangan:

Tabel 5.1 Keterangan simbol gambar 5.1

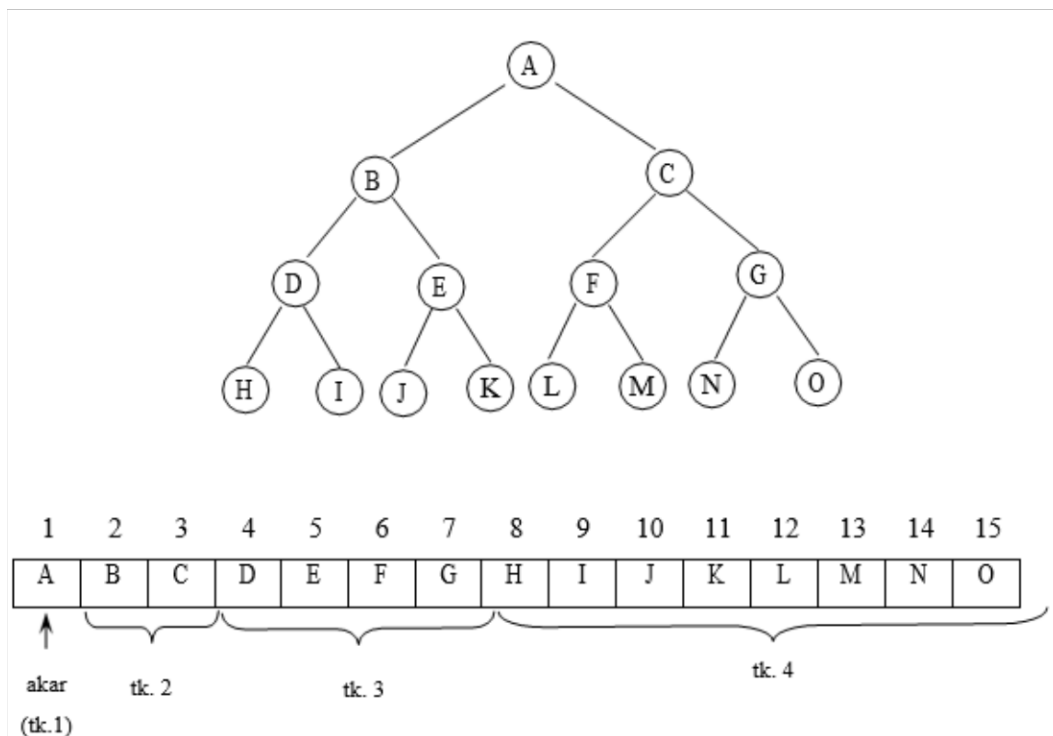
Simbol	Deskripsi	Simbol	Deskripsi
	simpul akar		simpul daun
	induk dari 60 &		descendant dari
	ancestor dari 03 &		sibling
	anak dari 67	cabang

5.2.2. Deklarasi Tree

5.2.2.1. Penyajian Tree Dengan Array

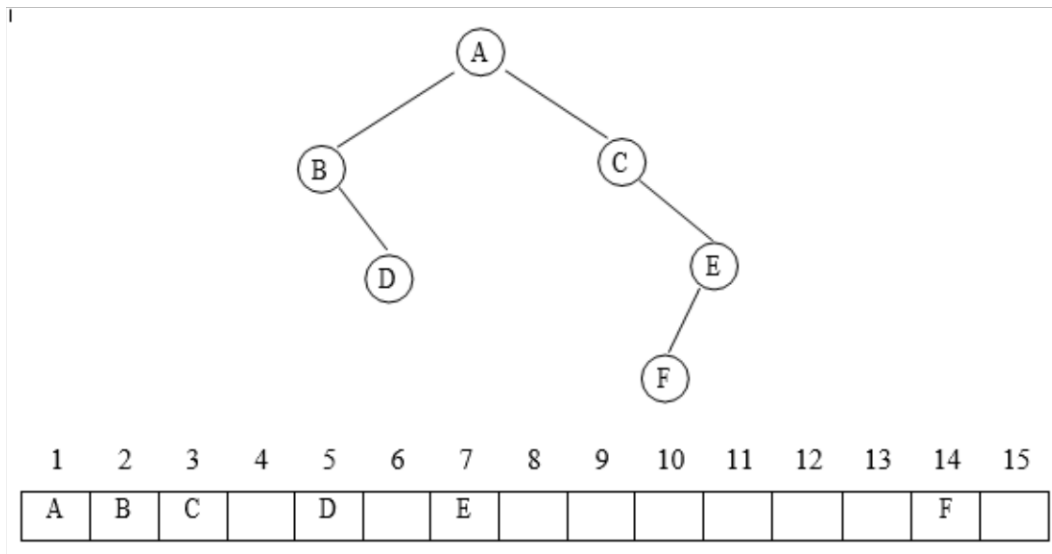
Pohon biner dapat direpresentasikan secara berurutan menggunakan array atau file. Oleh karena itu untuk mengingat definisi dari pohon biner, diperlukan satu aturan tertentu. Di mana pohon yang mempunyai kedalaman N mempunyai simpul maksimum $2^{(N-1)}$, dengan N adalah kedalaman simpul. Sebagai contoh pohon dengan kedalaman 4, mempunyai simpul secara keseluruhan (dari tingkat 1 sampai dengan 4) adalah 15 buah.

Penyajian pohon biner secara berurutan dalam sebuah array dimulai dari akar pohon (simpul tingkat pertama) selalu menempati elemen pertama array. Simpul-simpul tingkat 2 diletakkan sebagai elemen ke-2 dan 3. Simpul-simpul pada tingkat 3 diletakkan sebagai elemen ke-4, 5, 6, 7. Simpul-simpul tingkat 4 diletakkan sebagai elemen ke-8 sampai ke-15.



Gambar 5.2 Ilustrasi Penyajian Tree dengan Array

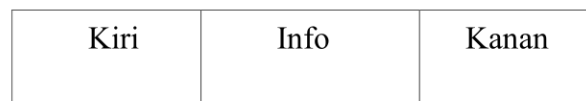
Apabila pohon biner yang dibuat tidak lengkap, maka pemakaian memori akan menjadi tidak efisien.



Gambar 5.3 Ilustrasi Penyajian Tree dengan Array yang kurang efisien

5.2.2.2. Penyajian Dengan Linked List

Representasi simpul dalam pohon biner menggunakan *linked list* dapat, dapat diilustrasikan seperti gambar berikut :



Gambar 5.4 Ilustrasi penyajian dengan *linked list*

Struktur pohon pada c++ dapat dideklarasikan sebagai berikut :

```
typedef char
TypeInfo; typedef
struct Simpul *Tree;
struct Simpul {
    TypeInfo Info;

    tree Kiri; //cabang kiri

    tree Kanan; //cabang kanan
};
```

5.2..3. Pembentukan Pohon Biner

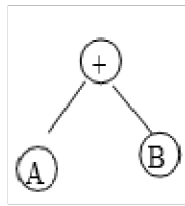
Diketahui suatu persamaan

$$(A + B) * ((B - C) + D)$$

Berikut ketentuan pembentukan tree pada contoh tabel di atas :

1. Buatlah dua buah stack, tumpukan operator dan tumpukan operand.
 - a. Baca ungkapan dalam notasi infix, misalnya S, tentukan panjang ungkapan tersebut, misalnya N karakter, siapkan sebuah stack kosong dan siapkan derajat masing-masing operator, misalnya: ^ berderajat 3, * dan / berderajat2, + dan – berderajat 1 dan (berderajat 0).
 - b. Dimulai dari i = 0 sampai N-1 kerjakan langkah-langkah sebagai berikut:
 - a. $R = S[I]$
 - b. Test nilai R. Jika R adalah:
 - (1) Bila karakter yang dibaca berupa operator ('+', '-', '*', '/') dan tanda '(' maka masukkan karakter tersebut ke stack operator. Buat Simpul dengan Info adalah karakter operator tersebut dengan Kiridan Kanan bernilai null.
 - (2) Bila karakter yang dibaca berupa operand 'A' .. 'Z' maka masukkan ke stack operand. Buat Simpul dengan Info adalah karakter operand tersebut dengan Kiridan Kanan bernilai null.
 - (3) Bila karakter yang dibaca adalah tanda '(' lakukan hal berikut :
 - Pop stack operator, Simpul ini sebagai simpul induk.
 - Pada stack operand, lakukan Pop ujung stack, simpul ini sebagai simpul kanan simpul induk dan pop ujung stack, simpul ini sebagai simpul kiri simpul induk.
 - Kemudian masukkan simpul tersebut ke stack operand.
 - Lakukan proses ini jika di stack operator, terdapat Simpul dengan Info operator. Jika di stack operator adalah Simpul dengan info '(', maka pop isi stack operator.

- c. Jika di akhir, isi stack operand dan operator belum kosong, maka lakukan langkah b3, sampai isi stack operand dan operator kosong.



Tabel 5.2 Tabel pembentukan pohon biner di atas

Karakter yang dibaca	Tumpukan operator	Tumpukan operand	Pohon biner yang terbentuk
((
A	(A	
+	(+	A	
B	(+	AB	
)		+	
*	*	+	
(* (+	
(* ((+	
B	* ((+B	
-	* ((-	+B	
C	* ((-	+BC	
)	(*	+ -	
+	* (+	+ -	
D	* (+	+ - D	
)	*	++	

--	--	--	--

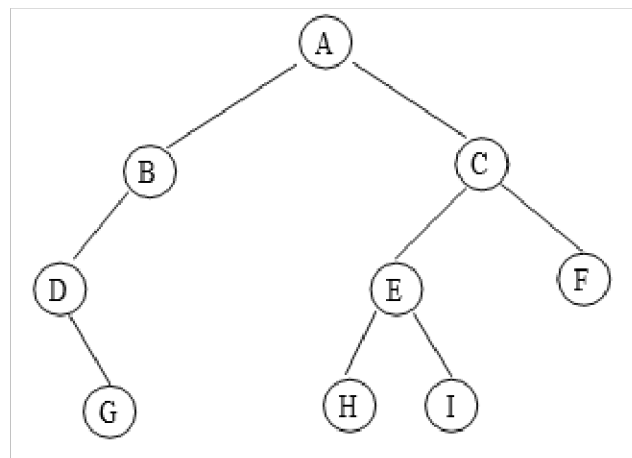
5.2.2.4. Metode Transversal

Proses traversing dari sebuah binary tree merupakan proses melakukan kunjungan pada setiap node pada suatu binary tree tepat satu kali. Kunjungan dapat dilakukan dengan tiga cara, yaitu kunjungan secara pre-order (kiri, root, kanan), in-order (root, kiri, kanan), dan secara post-order (kiri, kanan, root).

Selain itu, berdasarkan kedudukan setiap simpul dalam pohon, juga bisa dilakukan kunjungan secara level-order. Ketiga macam kunjungan yang pertama bisa dilaksanakan secara rekursif.

1. Kunjungan pre-order

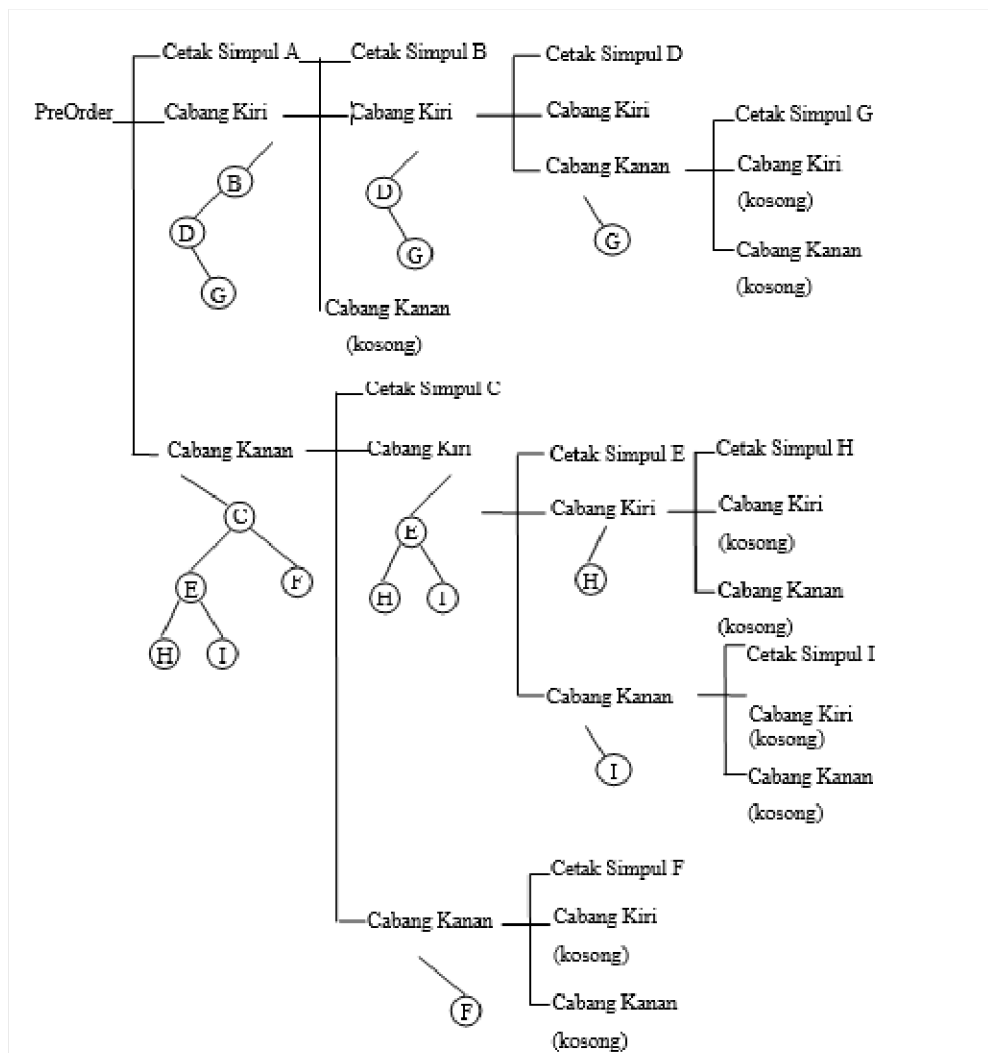
Kunjungan preorder dilakukan dengan mencetak simpul yang dikunjungi, kunjungan cabang kiri, dan kunjungan cabang kanan. Untuk lebih jelasnya perhatikan pohon biner pada gambar 5.4.



Gambar 5.4 Ilustrasi Pohon Biner

Kunjungan preorder, juga disebut dengan *depth first order*, dengan alur yang pertama, yaitu mencetak isi simpul yang dikunjungi, kemudian mengunjungi cabang kiri, terakhir mengunjungi cabang kanan.

Dari gambar 5.4 nampak pelacakan kunjungan PreOrder. Hasil dari pelacakan kunjungan secara PreOrder tersebut akan didapatkan hasil 'ABDGCEHIF'.

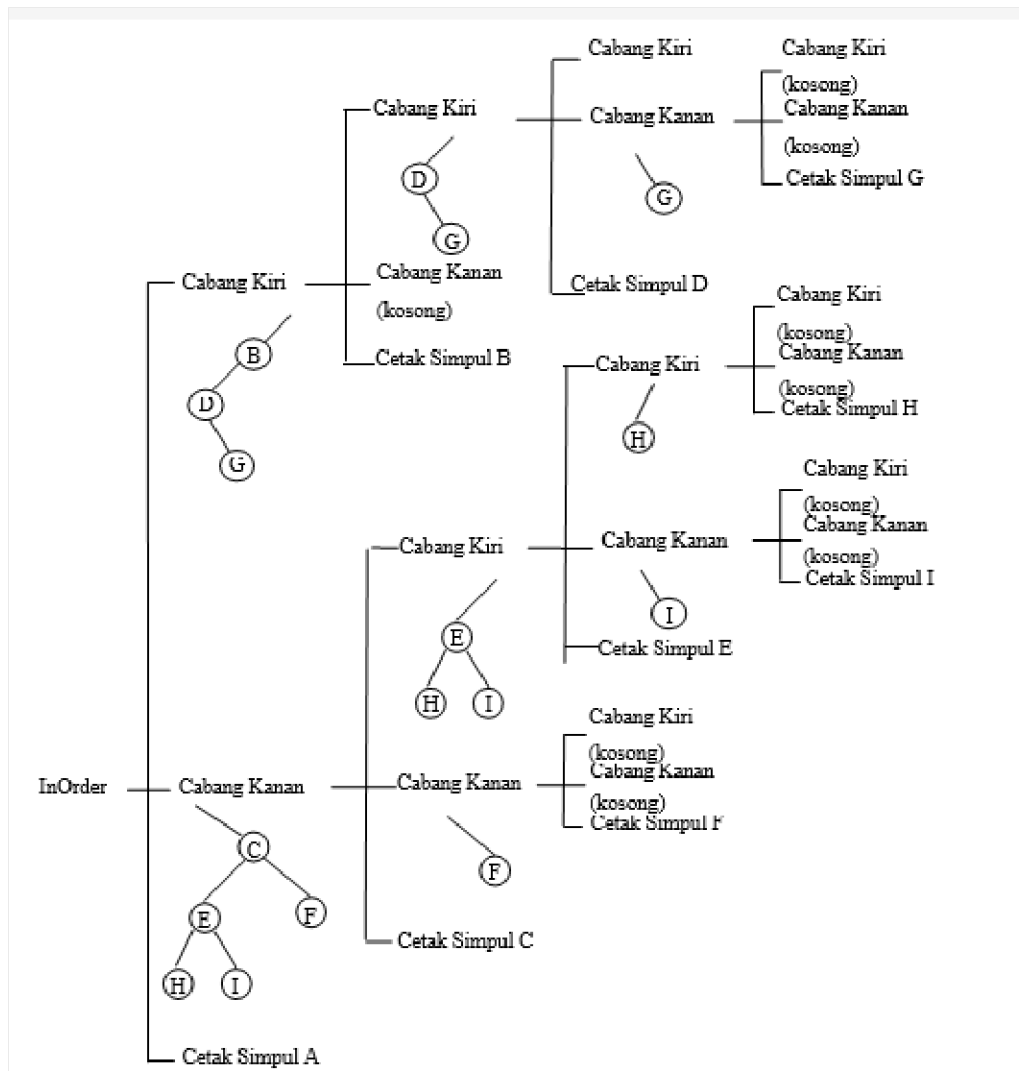


Gambar 5.5 Kunjungan PreOrder dari pohon biner pada Gambar 5.4

2. Kunjungan in-order

Kunjungan secara inorder, juga sering disebut dengan symmetric order, menggunakan urutan yang pertama mengunjungi cabang kiri, kemudian mencetak isi simpul yang dikunjungi, terakhir mengunjungi cabang kanan.

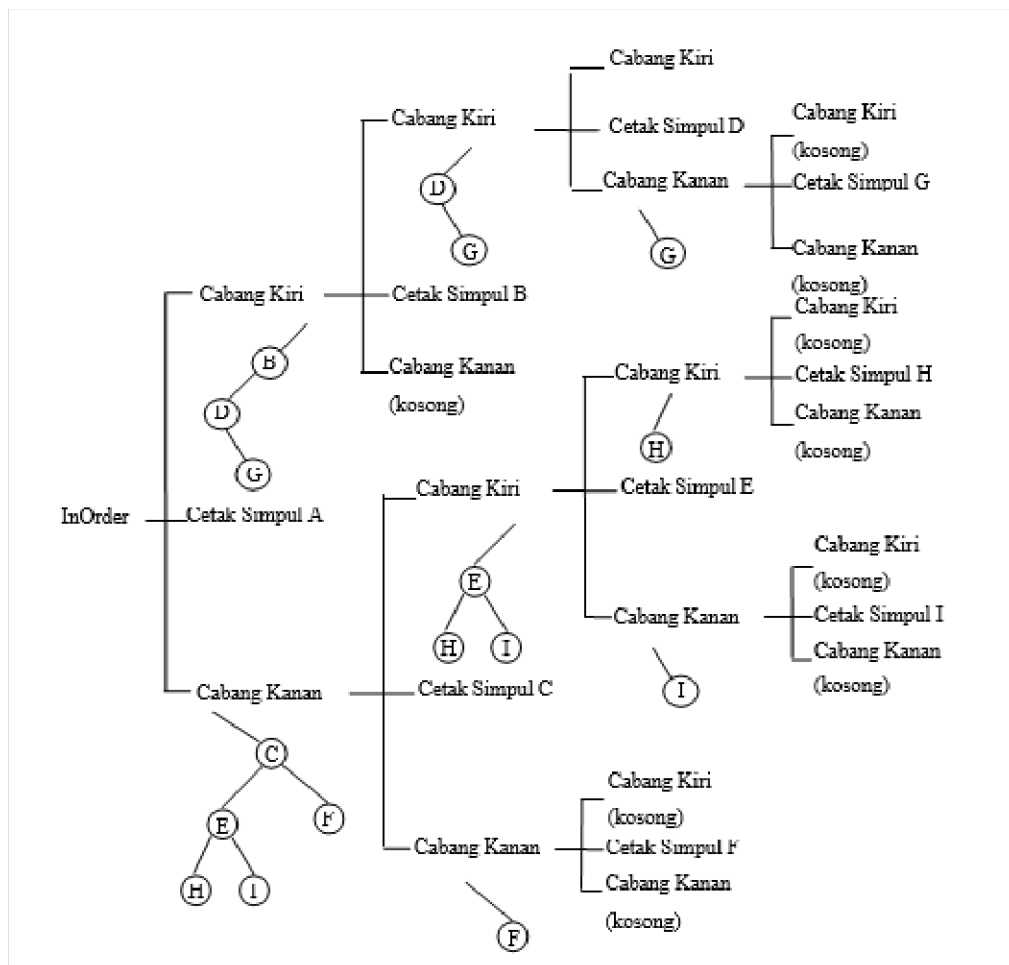
Dari gambar 5.6 nampak pelacakan kunjungan InOrder. Hasil dari pelacakan kunjungan secara InOrder tersebut akan didapatkan hasil ‘DGBAHEICF’



Gambar 5.6 Kunjungan InOrder dari pohon biner pada Gambar 5.4

Kunjungan secara postorder menggunakan urutan yang dimulai dari mengunjungi cabang kiri, kemudian mengunjungi cabang kanan, dan terakhir mencetak isi simpul yang dikunjungi

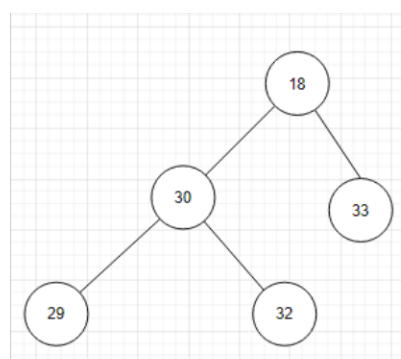
Dari gambar 5.7 nampak pelacakan kunjungan PostOrder. Hasil dari pelacakan kunjungan secara PostOrder tersebut akan didapatkan hasil 'GDBHIEFCA'.



Gambar 5.7 Kunjungan PostOrder dari pohon biner pada Gambar 5.4

5.3. STUDI KASUS

Analisalah dan tentukan kunjungan pre-order, in-order, dan post-order pada *binary tree* dibawah ini, jika terdapat angka 15, 30, 27, 25, 29 :



Gambar 5.8 Studi kasus pohon biner

Berdasarkan gambar 5.8, maka hasil yang muncul ketika dicetak, diantaranya:

1. Pre-order : Root pertama 18, ke kiri 30, punya kiri lagi 29. Setelah itu ke kanan 32, ke kanan yang atasnya 33. Jadi akan muncul data : 18, 30, 29, 32, 33.
2. In-order : Kiri pertama adalah 29, naik ke root 30, ke kanan 32. Karena sudah tidak ada di kiri, baik ke root 18 dan kanan 33. Jadi akan muncul data : 29, 30, 32, 18, 33.
3. Post-order : Kiri pertama 29, kanan 32, rootnya 30 tapi juga jadi kiri, kanan 33 dan rootnya 18. Jadi akan muncul data : 29, 32, 30, 33, 18.

5.4. PERCOBAAN

1. Buatlah workspace menggunakan Replit.
2. Buatlah project baru TREE yang berisi file C++ source untuk pembuatan program sederhana tree dan tree dengan array
3. Cobalah untuk masing-masing percobaan di bawah

Percobaan 1 : Program sederhana tree

```
//Program Tree
#include <iostream>
using namespace std;
class Node {
public:
    int data;
    Node* left;
    Node* right;

    Node(int value) {
        data = value;
        left = NULL;
        right = NULL;
    }
};

class Tree {
public:
    Node* root;
```

```

Tree() { root = NULL; }

void insert(int value) {
    root = insert(root, value);
}

Node* insert(Node* node, int value) {
    if (node == NULL) {
        node = new Node(value);
    } else if (value <= node->data) {
        node->left = insert(node->left, value);
    } else {
        node->right = insert(node->right, value);
    }
    return node;
}

void inorder() { inorder(root); }

void inorder(Node* node) {
    if (node == NULL) return;
    inorder(node->left);
    cout << node->data << " ";
    inorder(node->right);
}
};

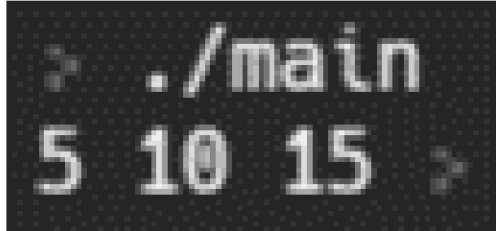
int main() {
    Tree tree;
    tree.insert(10);
    tree.insert(5);
    tree.insert(15);
    tree.inorder();
    return 0;
}

```

```
}
```

OUTPUT

5 10 5



Percobaan 2 : Program tree dengan array

```
#include <iostream>
using namespace std;

const int MAX_NODES = 100;

int tree[MAX_NODES];
int n;

void insertNode(int x) {
    int curr = 0;
    while (tree[curr] != -1) {
        if (x < tree[curr]) {
            curr = 2 * curr + 1;
        } else {
            curr = 2 * curr + 2;
        }
    }
    tree[curr] = x;
    n++;
}

void inorderTraversal(int curr) {
    if (tree[curr] != -1) {
```

```

        inorderTraversal(2 * curr + 1);
        cout << tree[curr] << " ";
        inorderTraversal(2 * curr + 2);
    }
}

int main() {
    for (int i = 0; i < MAX_NODES; i++) {
        tree[i] = -1;
    }
    n = 0;

    int x;
    char pilihan;

    do {
        cout << "Masukan nilai pada binary tree ";
        cin >> x;
        insertNode(x);
        cout << "Ingin memasukan nilai lain(y/n)? ";
        cin >> pilihan;
    } while (pilihan == 'y' || pilihan == 'Y');

    cout << "In-order traversal: ";
    inorderTraversal(0);

    return 0;
}

```

OUTPUT

```

Masukan nilai pada binary tree 15
Ingin memasukan nilai lain(y/n)? y
Masukan nilai pada binary tree 19
Ingin memasukan nilai lain(y/n)? y
Masukan nilai pada binary tree 25

```

Ingin memasukan nilai lain(y/n)? y

Masukan nilai pada binary tree 30

Ingin memasukan nilai lain(y/n)? y

Masukan nilai pada binary tree 33

Ingin memasukan nilai lain(y/n)? n

In-order traversal: 15 19 25 30 33

```
> ./treedenganarray
Masukan nilai pada binary tree 15
Ingin memasukan nilai lain(y/n)? y
Masukan nilai pada binary tree 19
Ingin memasukan nilai lain(y/n)? y
Masukan nilai pada binary tree 25
Ingin memasukan nilai lain(y/n)? y
Masukan nilai pada binary tree 30
Ingin memasukan nilai lain(y/n)? y
Masukan nilai pada binary tree 33
Ingin memasukan nilai lain(y/n)? n
In-order traversal: 15 19 25 30 33 >
```

5.5. TUGAS DAN LATIHAN

Buatlah sebuah program c++ mengenai konsep penerapan tree, dengan kriteria :

1. Bilangan/value diinput diprogram
2. Bilangan yang diinput ke program 12, 16, 20, 24, 32
3. Dapat mengurutkan bilangan yang diinput secara pre-order
4. Dapat mengurutkan bilangan yang diinput secara in-order
5. Dapat mengurutkan bilangan yang diinput secara post-order