

Praktikum 10

Pohon (*Tree*)

POKOK BAHASAN:

- ✓ Konsep Pohon (*Tree*)
- ✓ Binary Tree
- ✓ Penyajian Tree dengan Array
- ✓ Penyajian Tree dengan Linked List
- ✓ Metode Traversal (Kunjungan Node pada Tree)

TUJUAN BELAJAR:

Setelah melakukan praktikum dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Mampu membuat struktur pohon (*tree*) dengan menggunakan array.
- ✓ Mampu membuat flowchart pembentukan tree dengan menggunakan struktur data list.
- ✓ Mampu mengimplementasikan tree sesuai flowchart yang sudah dibuat.

DASAR TEORI:

Struktur pohon (*tree*) biasanya digunakan untuk menggambarkan hubungan yang bersifat hirarkis antara elemen-elemen yang ada. Contoh penggunaan struktur pohon :

- Silsilah keluarga
- Hasil pertandingan yang berbentuk turnamen
- Struktur organisasi dari sebuah perusahaan

10.1 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.

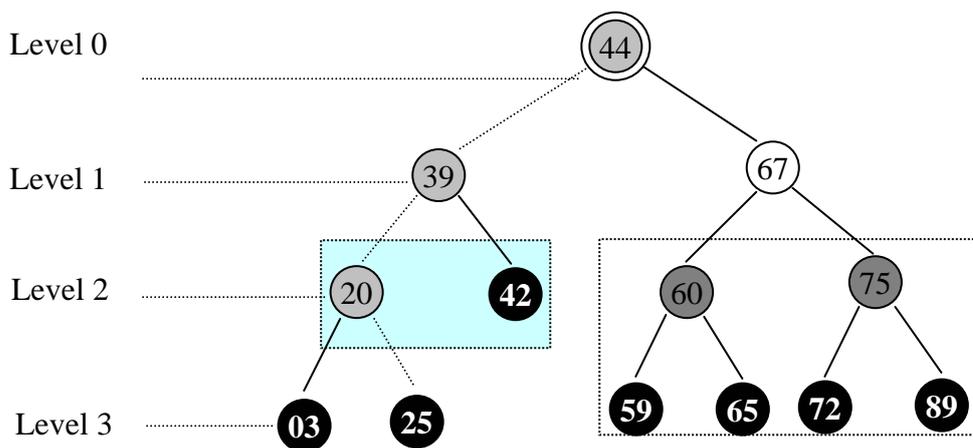
Masing-masing simpul dalam binary tree terdiri dari tiga bagian yaitu sebuah data dan dua buah pointer yang dinamakan pointer kiri dan kanan.

Simpul juga mempunyai *sibling*, *descendants*, dan *ancestors*. *Sibling* dari sebuah simpul adalah anak lain dari induk simpul tersebut. *Descendants* dari sebuah simpul adalah semua simpul-simpul merupakan cabang (berada di bawah) simpul tersebut. *Ancestors* dari sebuah simpul adalah semua simpul yang berada di atas antara simpul tersebut dengan *root*. 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.

Selain tingkat, juga dikenal istilah derajat (degree) dari suatu simpul. Derajat suatu simpul dinyatakan sebagai banyaknya anak atau turunan dari simpul tersebut.



Gambar 10.1 Ilustrasi Sebuah Pohon Biner (*Binary Tree*)

Keterangan:

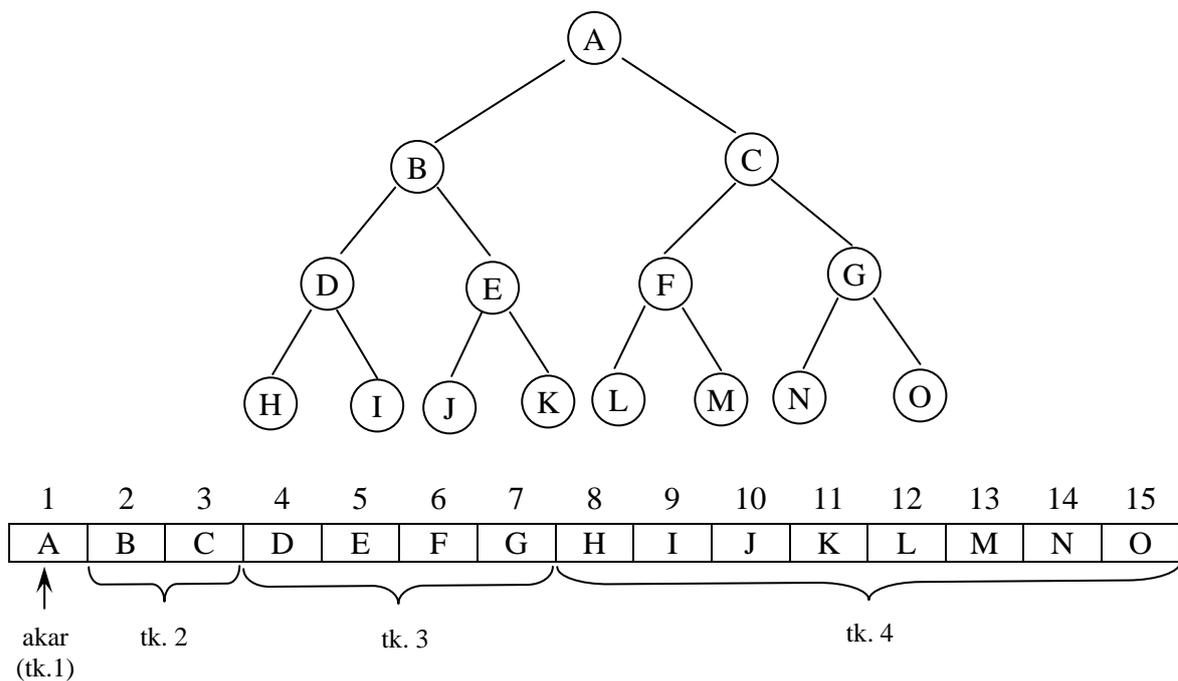
- ◎ simpul akar ○ induk dari 60 & ● ancestor dari 03 & ● anak dari 67
- simpul daun □ descendant dari □ sibling cabang

10.2 DEKLARASI POHON

10.2.1 PENYAJIAN DENGAN ARRAY

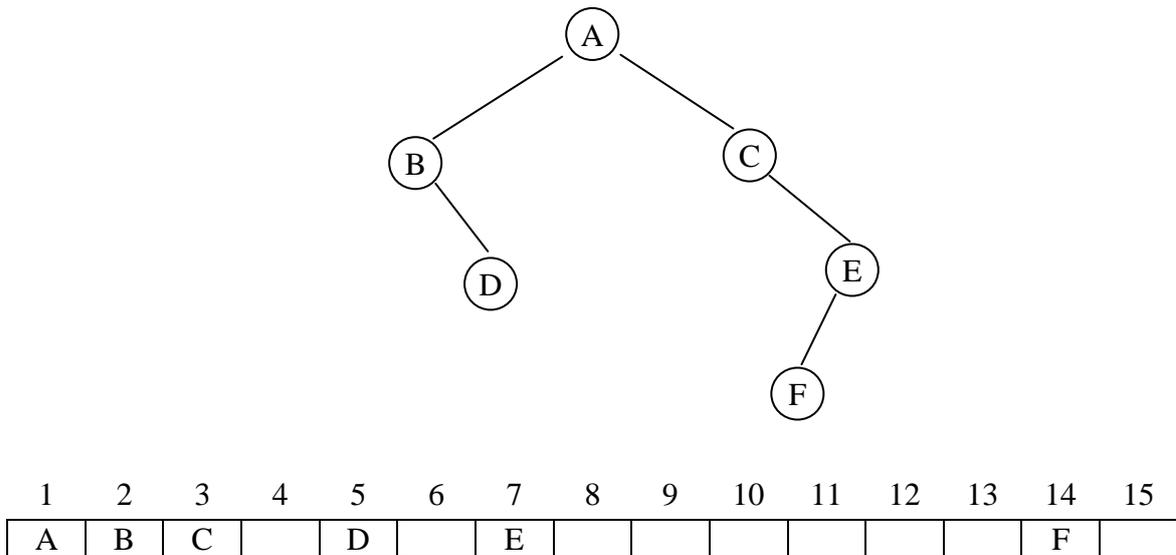
Pohon biner dapat disajikan secara berurutan dengan menggunakan array atau file. Untuk itu diperlukan adanya satu aturan tertentu dengan mengingat definisi pohon biner. Pohon yang mempunyai kedalaman N mempunyai simpul maksimum $2^{(N-1)}$, dengan N adalah kedalaman simpul. Sebagai contoh pohon dengan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

Dengan demikian, penyajian pohon biner secara berurutan dalam sebuah array adalah sebagai berikut. 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 10.2 Ilustrasi Penyajian Tree dengan Array

Cara penyimpanan seperti ini hanya baik jika pohon binernya berupa pohon biner lengkap. Jika pohon binernya tidak lengkap, pemakaian memori tidak efisien.



Gambar 10.3 Contoh Penyajian Pohon Biner Menggunakan Array yang Tidak Efisien

10.2.2 PENYAJIAN DENGAN LINKED LIST

Simpul dalam pohon biner dapat disajikan dengan list sebagai berikut:



Deklarasi struktur pohon :

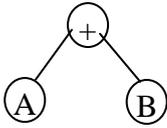
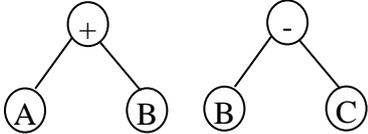
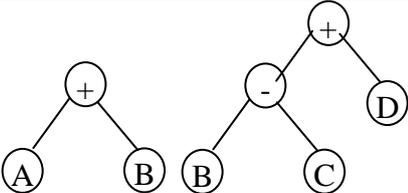
```
typedef char TypeInfo;
typedef struct Simpul *Tree;
struct Simpul {
    TypeInfo Info;
    tree Kiri,      /* cabang kiri */
    tree Kanan;    /* cabang kanan */
};
```

10.3 PEMBENTUKAN POHON BINER

Diketahui suatu persamaan

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

Maka pembentukan pohon biner dapat dilihat pada table di bawah ini :

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

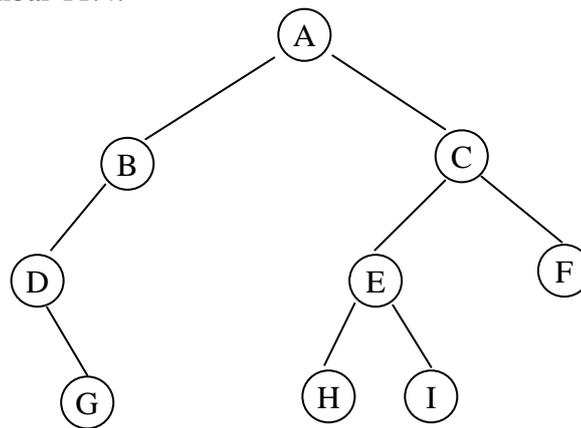
10.4 METODE TRAVERSAL

Proses traversing dari sebuah binary tree artinya melakukan kunjungan pada setiap simpul pada suatu pohon biner tepat satu kali. Kunjungan dapat dilakukan dengan tiga cara, yaitu kunjungan secara preorder, inorder, dan secara postorder. Selain itu, berdasarkan kedudukan setiap simpul dalam pohon, juga bisa dilakukan kunjungan

secara levelorder. Ketiga macam kunjungan yang pertama bisa dilaksanakan secara rekursif.

10.4.1 KUNJUNGAN PREORDER

Kunjungan preorder dilakukan dengan mencetak simpul yang dikunjungi, kunjungan cabang kiri, dan kunjungan cabang kanan. Untuk lebih jelasnya perhatikan pohon biner pada gambar 11.4.

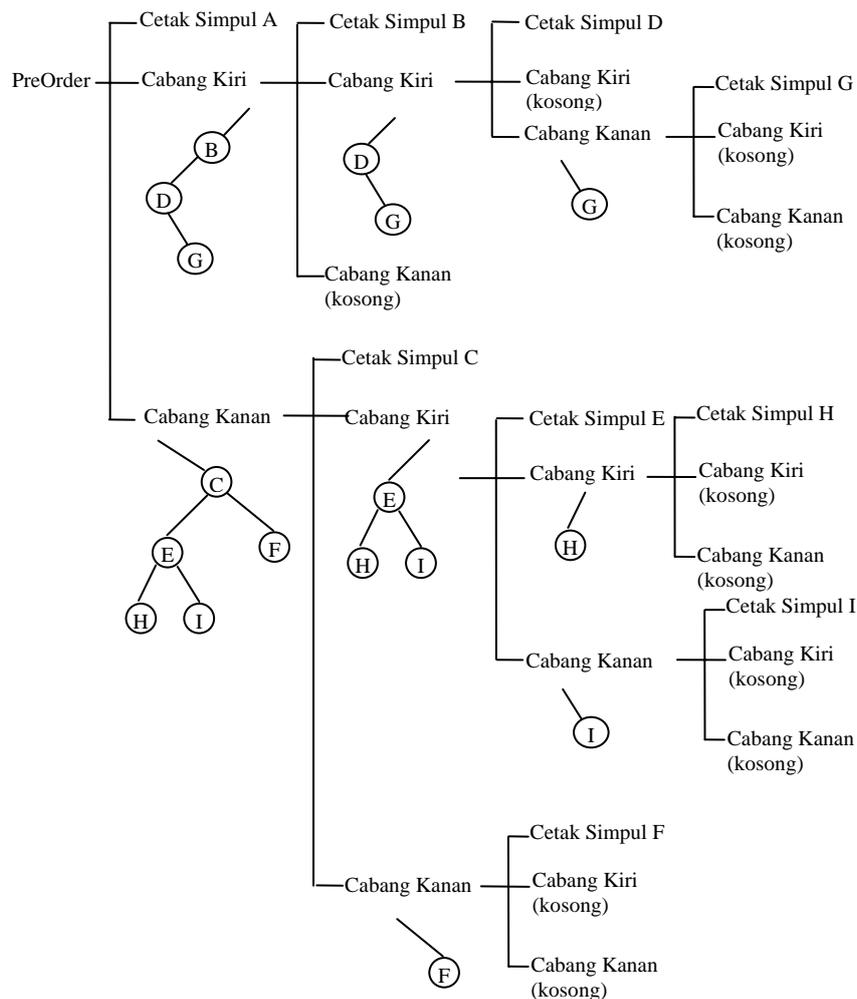


Gambar 10.4 Contoh Pohon Biner

Kunjungan preorder, juga disebut dengan *depth first order*, menggunakan urutan:

- Cetak isi simpul yang dikunjungi
- Kunjungi cabang kiri
- Kunjungi cabang kanan

Dari gambar 10.5 nampak pelacakan kunjungan PreOrder. Hasil dari pelacakan kunjungan secara PreOrder tersebut akan didapatkan hasil 'ABDGCEHIF'.



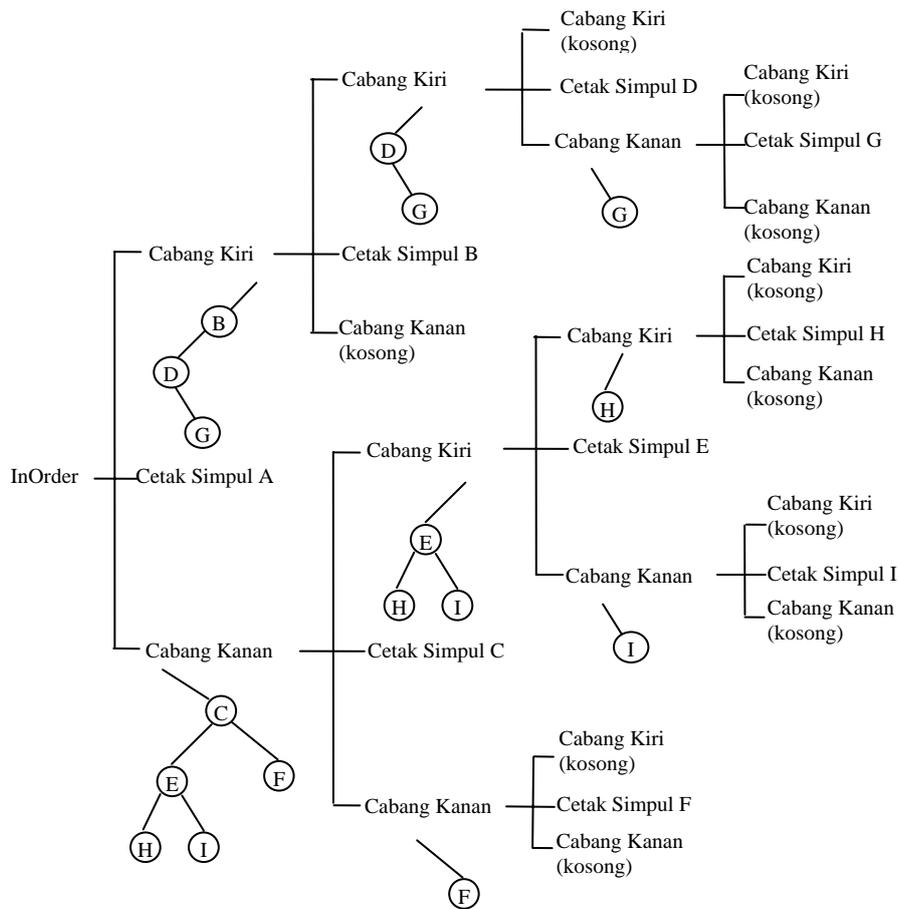
Gambar 10.5 Kunjungan PreOrder dari pohon biner pada Gambar 10.4

10.4.2 KUNJUNGAN INORDER

Kunjungan secara inorder, juga sering disebut dengan *symmetric order*, menggunakan urutan:

- Kunjungi cabang kiri
- Cetak isi simpul yang dikunjungi
- Kunjungi cabang kanan

Dari gambar 10.6 nampak pelacakan kunjungan InOrder. Hasil dari pelacakan kunjungan secara InOrder tersebut akan didapatkan hasil 'DGBAHEICF'.



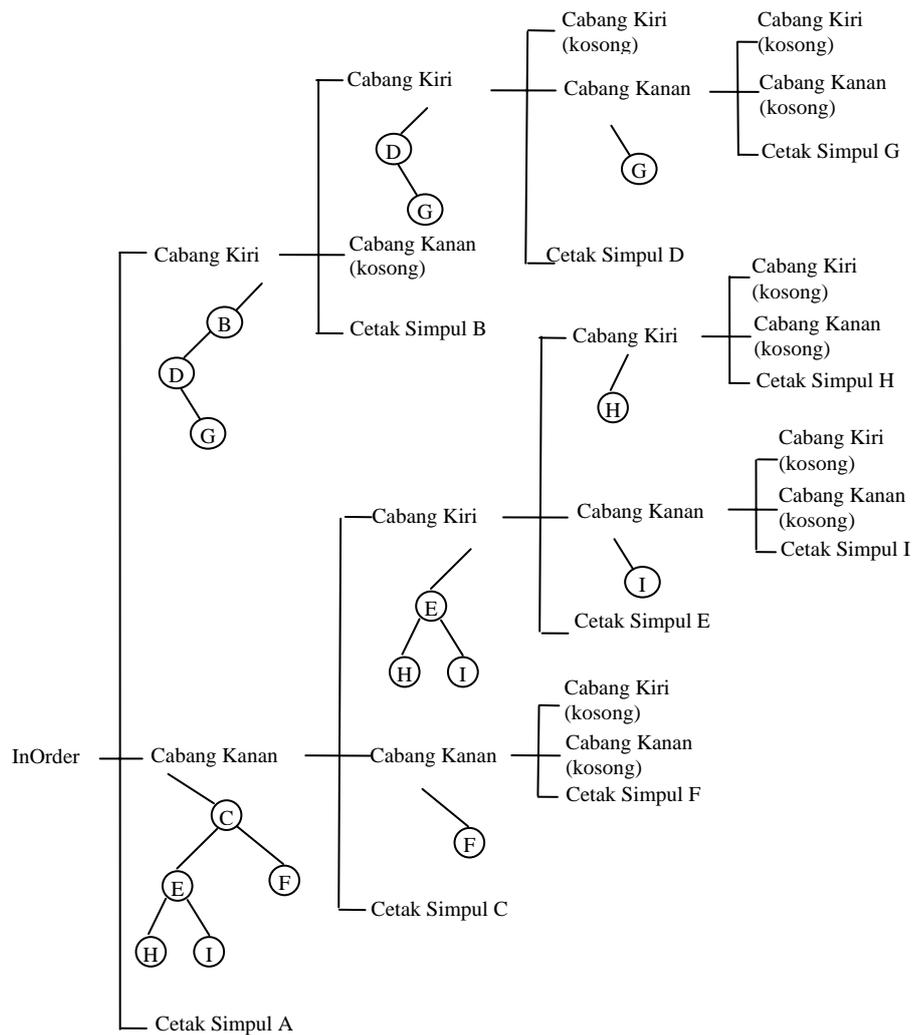
Gambar 10.6 Kunjungan InOrder dari pohon biner pada Gambar 10.4

10.4.3 KUNJUNGAN POSTORDER

Kunjungan secara postorder menggunakan urutan:

- Kunjungi cabang kiri
- Kunjungi cabang kanan
- Cetak isi simpul yang dikunjungi

Dari gambar 10.7 nampak pelacakan kunjungan PostOrder. Hasil dari pelacakan kunjungan secara PostOrder tersebut akan didapatkan hasil 'GDBHIEFCA'.



Gambar 10.7 Kunjungan PostOrder dari pohon biner pada Gambar 10.4

TUGAS PENDAHULUAN

Pada dasarnya pembentukan pohon (tree) menggunakan implementasi tumpukan (stack) dengan double linked list. Buatlah flowchart untuk pembentukan pohon biner untuk suatu persamaan yang diinputkan (asumsi : persamaan yang diinputkan harus selalu benar). Lihat pembentukan tree pada contoh table diatas dengan ketentuan sebagai berikut :

- Buatlah dua buah stack, tumpukan operator dan tumpukan operand.
- Cek karakter yang diinputkan :

1. Bila karakter yang dibaca berupa operator ('+', '-', '*', '/') dan tanda '(' maka masukkan karakter tersebut ke tumpukan operator.
2. Bila karakter yang dibaca berupa operand 'A' .. 'Z' maka masukkan ke tumpukan operand.
3. Bila karakter yang dibaca adalah tanda '(' lakukan hal berikut :
 - o Pop tumpukan operator sampai dengan tanda ')' dan tandai operator yang berada diantaranya ('+', '-', '*', '/').
 - o Bentuklah sebuah Simpul dengan Info adalah karakter operator tersebut dengan Kiri dan Kanan bernilai null.
 - o Pada tumpukan operand, pop 2 karakter teratas.
 - Apabila karakter pertama berupa karakter 'A' .. 'Z' bentuk Simpul dengan Info adalah karakter operand dan pointer Kiri dan Kanan bernilai null. Apabila berupa operator, bentuk pohon dengan menunjuk pointer Kiri dari Simpul operator ke Simpul karakter pertama.
 - Cek untuk karakter kedua dengan cara yang sama. Apabila karakter kedua berupa operator, bentuk pohon dengan menunjuk pointer Kanan dari Simpul operator ke Simpul karakter kedua.
 - o Kemudian karakter operator hasil pop yang sudah ditandai masukkan ke tumpukan operand.
 - o Ulangi langkah 3 sampai pembentukan pohon selesai.
4. Hasil output berupa penelusuran linked list dari pohon teratas secara preorder.

PERCOBAAN

1. Buatlah workspace untuk praktikum Struktur Data dengan menggunakan Visual C++.
2. Buatlah project untuk praktikum KESEPULUH.
3. Cobalah untuk masing-masing percobaan di bawah ini.
4. Implementasikan flowchart yang anda buat pada Tugas Pendahuluan.

Percobaan 1 : Membangun Tree Baru

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct tree *pohon;

/* Deklarasi dari tree dengan menggunakan linked list */
struct tree{
    char info;
    pohon kiri;
    pohon kanan;
};

/* Fungsi untuk membuat simpul yang pertama */
pohon baru(char hrf)
{
    pohon br;
    br=(pohon)malloc(sizeof(struct tree));
    br->info=hrf;
    br->kiri=NULL;
    br->kanan=NULL;
    return (br);
}

/*
Fungsi untuk menyisipkan simpul pada tree yang sudah dibangun
*/
void sisip (pohon ph, pohon sp)
{
    pohon P,Q;
    P = ph;
    Q = ph;
    while((sp->info != ph->info)&&(Q!=NULL))
    {
        P = Q;
        if (sp->info < P->info)
            Q = P->kiri;
        else
            Q = P->kanan;
    }
    if(sp->info == P->info)
        printf("Sudah ada");
    else
        if(sp->info < P->info)
            P->kiri=sp;
        else
            P->kanan=sp;
}
}
```

```
/*
Fungsi-fungsi untuk metode traversal secara rekursif
*/

void preorder(pohon ph)
{
    if (ph != NULL)
    {
        printf("%c ", ph->info);
        preorder(ph->kiri);
        preorder(ph->kanan);
    }
}

void inorder(pohon ph)
{
    if (ph != NULL)
    {
        inorder(ph->kiri);
        printf("%c ", ph->info);
        inorder(ph->kanan);
    }
}

void postorder(pohon ph)
{
    if (ph != NULL)
    {
        postorder(ph->kiri);
        postorder(ph->kanan);
        printf("%c ", ph->info);
    }
}

main()
{
    int j=0;char input,jawab[2];
    pohon br, ssp;
    while(1)
    {
        printf("Ketikkan huruf :");scanf("%c",&input);
        fflush(stdin);
        if (j==0) br = baru(input);
        else
        {
            ssp = baru(input);
            sisip(br,ssp);
        }
        printf("Ada data lagi(y/t):"); scanf("%s",&jawab);
        fflush(stdin);
    }
}
```

```
if((strcmp(jawab,"Y")==0)|| (strcmp(jawab,"y")==0))
    {
        j++;continue;
    }
else if ((strcmp(jawab,"T")==0)|| (strcmp(jawab,"t")==0))
    break;
}
preorder(br); printf("\n");
inorder(br); printf("\n");
postorder(br);
}
```

LATIHAN

1. Implementasikan tree untuk persamaan matematika seperti yang ada dalam Dasar Teori, yang mana flowchart-nya sudah anda buat dalam Tugas Pendahuluan, dengan langkah sebagai berikut:
 - a) Buatlah struktur data stack untuk tumpukan operator.
 - b) Buatlah struktur data stack untuk tumpukan operand.
 - c) Buatlah struktur data tree seperti pada dasar teori.
 - d) Pada program utama inputkan persamaan yang akan dibentuk tree nya :
 - i) POHON BINER UNTUK PERSAMAAN MATEMATIKA
 - ii) Inputkan Persamaan : _
 - e) Buatlah prosedur InitializeStack untuk inialisasi tumpukan operator dan tumpukan operand.
 - f) Buatlah prosedur InitializeTree untuk inialisasi pohon yang dibentuk.
 - g) Buatlah fungsi Empty untuk memeriksa apakah stack dalam keadaan kosong.
 - h) Buatlah fungsi Full untuk memeriksa apakah stack dalam keadaan penuh.
 - i) Buatlah prosedur Tree_Process untuk pembentukan tree.
 - j) Buatlah prosedur Tree_Preorder untuk menampilkan tree secara preorder.
 - k) Buatlah prosedur Tree_Postorder untuk menampilkan tree secara postorder.
2. Buatlah fungsi yang menampilkan isi dari tree dengan metode traversal tanpa proses rekursif.