**BAB XII**

POHON BINER

(Binary Tree)

1. **Pohon Biner**

*Linked Lisk*, *Stack*, dan *Queue* yang sudah kita bahas pada bab sebelumnya merupakan struktur data yang bersifat linear. Di dalam terminologi struktur data sebuah pohon (*tree*) adalah suatu struktur yang tak linear yang memiliki sifat khusus. Pohon (tree) merupakan salah satu bentuk implementasi banyak Linked List yang biasanya digunakan untuk menggambarkan hubungan yang bersifat hirarkis antara elemen-elemen yang ada. Secara sederhana pohon bisa didefenisikan sebagai kumpulan elemen yang salah satu elemenya disebut akar (root), dan sisa elemen yang lain diseut simpul terpecah menjadi sejumlah himpunan yang saling tidak terhubung satu sama lain, yang disebut dengan subpohon (subtree), atau juga disebut dengan cabang.

Contoh penggunaan struktrur pohon (tree) adalah :

* + Sisilah keluarga
  + Hasil pertandingan yang berbentuk ternamen
  + Struktur organisasi dari sebuaah perusahaan.

Pohon (tree) terdiri dari sejumlah simpul dengan karakteristik sebagai berikut :

1. Ada sebuah simpul yang berkedudukan sebagai puncak (dan disebut akar atau *root*). semua simpul dapat ditelusuri dari simpul ini.
2. Setiap simpul mempunyai lintasan yang unik terhadap akar.

**1**

 **6**



**2**

1 2

**3 4 5 7**

**8 9** 3 4 5

* 1. (b)

**Gambar 12.1**. Gambar pohon dan graf

Gambar (a) merupakan contoh pohon, sedangkan gambar (b) bukan pohon. Pada gambar (a), pohon memiliki 9 buah simpul, dengan simpul 1 berkedudukan sebagai akar. Simpul 2 dan 6 adalah anak dari simpul 1. Simpul 3, 4, dan 5 adalah anak dari simpul 2, dan sebagainya.

**Level 0 1**

**Level 1 .......................... 2 **   **6**

**Level 2 ............. 3 4 5 7**



**Root**

**Internal Node**

**Leaf**

**Level 3 ........................................................ 8 9**



**Parent of 3, 4, and 5**

**Child of 2 Sub Tree**

**Gambar 12.2**. Terminologi Pohon (tree)

A

B C

D E F G H

I

**Gambar 12.3**. Ilustrasi Pohon (Tree)

Terminologi dalam pohon(tree) adalah sebagai berikut :

1. Node

Sebuah elemen dalam sebuah pohon (tree), berisi sebuah informasi.

1. Parent

Node yang berada di atas node lain secara langsung, B adalah parent dari D dan E

1. Child

Cabang langsung dari sebuah node, D dan E merupakan children dari B.

1. Root

Node teratas yang tidak punya parent.

1. Sibling

Sebuah node lain yang memiliki paren yang sama, sibling dari B adalah C karena memiliki paren yang sama yaitu A.

1. Leaf

Sebuah node yang tidak memiliki children, D, E, F, G, dan I adalah leaf. Leaf bisa disebut dengan *external node,* sedangkan node lainnya disebut delgan *internal node*, B, A, C, dan H adalah *Internal node.*

1. Level

Semua node yang memiliki jara yang sama dari root. A → Level 0, B, dan C → Level 1, D, E, F, G, dan H → Level 2, dan I → Level 3.

1. Depth

Jumlah level yang ada dalam tree.

1. Complete

Semua parent memiliki children yang penuh.

1. Balanced

Semua subtree memilliki dipth yang sama.

**2. Tree Facts**

1. Setiap node, kecuali root, memiliki tepat hanya satu parent.
2. Sekali sebuah link dari sebuah parent ke sebuah child diikuti, tidaklah mungkin untuk kembali ke parentnya dengan mengikuti link yang lain (tidak ada siklus dalam sebuah tree).
3. Kumpulan child-child dari sebuah node, mereka sendiri juga merupakan sebuah tree → disebut sebagai sebutree.

**Gambar 12.4**. Pohon biner

Pohon biner (*binary tree*) adalah sebuah pengorganisasian secara hirarki dari beberapa buah node; masing-masing node tidak mempunyai child lebih dari

1. Contoh sebuah pohon biner ditunjukkan pada gambar diatas.

Implementasi binary tree bisa dikatakan menggunakan struktur data linked list, masing-masing node terdiri atas tiga bagian yaitu sebuah data/info dan dua buah pointer yang dinamakan pointer kiri dan pointer kanan.

|  |  |  |
| --- | --- | --- |
| **Kiri** | **Data/info** | **Kanan** |

**Gambar 12.5**. Implementasi pohon biner

Manfaat dari pohon biner yaitu data dengan sendirinya terurutkan. hal ini akan ditunjukkan belakangan.

1. **Balance**

**Gambar 12.6**. Balanced binary tree dan Unbalanced binary tree

* + Sebuah binary tree dikatakan sebagai *Balanced binary* jika setiap level diatas level paling rendah terisi penuh.
  + Sedangkan dikatakan *Unbalanced binary* jika tidak memenuhi keadaan diatas, dan kebanyakan aplikasi menginginkan suatu tree yang seimbang (balanced)

1. **Full Binary tree**

Semua node (kecuali daun) pasti memiliki 2 anak dan setiap subttree memiliki panjang jalur (path) yang sama.

A

B C

D E F G

**Gambar 12.7**. Contoh Full Binary Tree

1. **Complete Binary Tree**

Mirip dengan full binary tree, tapi tipa subtree boleh memiliki panjang path yang berbeda, tipa node (kecuali daun) memiliki 2 anak.

A

B C

D E

**Gambar 12.8**. Contoh Complete Binary Tree

1. **Skewed Binary Tree**

Binary tree yang semua nodenya (kecuali daun) hanya memiliki satu anak.

A D

B E

C F

**Gambar 12.9**. Contoh Skewed Binay Tree

1. **Deklarasi Binary Tree**

* node dalam sebuah binary tree disajikan sebagai berikut :

|  |  |  |
| --- | --- | --- |
| **Kiri** | **Data/info** | **Kanan** |

* sehingga, deklarasi tree untuk sebuah node dalam tree adalah

Type Tree = ^Simpul;

Simpul = record info : char;

{*\* ke cabang kiri \**}

{\* *ke cabang kanan*\*}

kiri,

kanan : tree end;

1. **Pembentukan Binary Tree**

* Dapat dilakukan dengan dua cara : rekursif dan non rekursif.
* Perlu memperhatikan kapan suatu node akan dipasang sebagai node kiri dan kapan sebagai node kanan.
* Misalnya ditentukan, node yang berisi info yang nilainya "lebih besar" dari parentnya akan ditempatkan di sebelah kanan dan yang "lebih kecil" akan ditempatkan disebelah kiri.

1. **Langkah-langkah pembentukan binary tree**
2. Siapkan node baru
   * Alokasikan memory-nya
   * masukkan info-nya
   * Set pointer kiri dan kanan = null
3. sisipkan pada posisi yang tepat
   * **Penelurusan →** untuk menentukan posisi yang tepat, info yang nilainya lebih besar dari parent akan ditelusuri di sebelah kanan, dan yang lebih kecil dari parent akan ditelusuri di sebelah kiri.
   * **Penempatan →** info yang nilainya lebih besar dari parent akan ditempatkan disebelah kanan, yang lebih kecil ditempatkan disebelah kiri.
4. **Algoritma Pembentukan Binary tree**
5. Buat node baru (new).
6. Cek apakah root = NULL
   * Jika ya, maka root = new, melompat ke langkah 9.
   * jika tidak, maka lakukan langkah berikutnya.
7. Mencari posisi yang tepat untuk new, tentukan P = root, Q = root
8. Kerjakan langkah 5 dan 6 selama Q <> NULL dan new → info <> P → info.
9. Tentukan P = Q
10. Cek apakah new → Infor < P → info
    * Jika ya, (teruskan ke cabang kiri), tentukan Q = P → kiri
    * jika tidak, (teruskan ke cabang kanan), tentukan Q = P → kanan
11. Cek apakah new → Infor = P → info
    * jika ya, (tidak perlu disisipkan), tempilkan pesan duplikasi, lompat ke langkah 9.
    * jika tidak, (sisipkan), kerjakan langkah 8.
12. Cek apakah new → Infor > P → info
    * jika ya, (sebagai cabang kiri), P → kiri = new
    * jika tidak, (sebagai cabang kanan) P → kanan = new
13. **Operasi pada Pohon Biner**

Seperti terlihat pada deklarasi kelas Pohon biner, terdapat sejumlah fungsi yang nanamnya menyatakan macam operas yang dapat dilakukan pada obyek berkelas ini.

**Menambah Data**

Operasi penambahan data pada pohon biner tidaklah sesederhana operasi yang sama pada senarai berantai. Pada pohon biner, simpul yang pertama kali dibentuk akan menjadi simpul akar. Bila simpul baru ditambahkan, pertama- tama akan dilakukan pengecekan terhadap data. Jika data baru ternyata bernilai lebih kecil daripada data yang terdapat pada simpul, data disisipkan di kiri (pointer kiri). Jika data baru lebih besar dari data yang ada pada simpul, data baru disisipkan di sebelah kanan (pointer kanan).

Proses pembentukan pohon biner ditunjukkan pada gambar dibawah ini :

C C C C C

A A R A R A R

K K

D

**Gambar 12.10**. Pembentukan pohon biner

Data yang disisipkan pertama kali yaitu C. Dengan sendirinya C menjadi akar. Kemudian :

1. A ditambahkan. mengingat A lebih kecil dari pada C, maka A disisipkan pada bagian kiri C.
2. R ditambahkan, R lebih besar dari pada C, maka R disisipkan pada bagian kanan C
3. K ditambahkan, K lebih besar dari pada C, tapi lebih kecil dari pada R, oleh karena itu, K menempati posisi kiri R.
4. D ditambahkan, D lebih besar dari C dan juga lebih kecil dari pada R dan lebih kecil dari pada K, dengan demikian D menempati bagian kiri dari K.
5. U ditambahkan, U lebih besar dari pada C dan juga lebih besar dari R, oleh karenanya, U menempati posisi kanan dari R.

Sekiranya secara berturut-turut data P, B, E, N, X, Z, dan S ditambahkan, pohon biner yang terbenntuk adalah seperti terlihat pada gambar dibawah ini ;

C

A R

B K U

D D S X

E D Z

**Gambar 12.11**. Pohon Biner

1. **Kunjungan Pada Pohon Biner**

Pada sebuah pohon biner bisa dilakukan sejumlah operasi. Salah satu operasi yang paling sering dilakukan adalah melakukan kunjungan pada setiap simpul pada suatu pohon biner tepat satu kali (binari tree traversal). Mengunjungi artinya memrpses data / info yang ada pada node yang bersangkuta. Kunjungan dapat dilakukan dengan tiga cara, yaitu kunjungan secara preorder, inorder, dan postorder. Ketiga kunjungan tersebut dapat dilakukan secara rekursif. Selain itu berdasarkan kedudukan setiap simpul dalam pohon juga bisa dilakukan kunjungan secara levelorder.

Kunjungan pada pohon biner, secara singkat bisa dijelaskan sebagai berikut ;

1. Kunjungan Preorder (*depth first order*), menggunakan urutan :
   * Cetak isi simpul yang dikunjungi
   * Kunjungi cabang kiri
   * Kunjungi cabang kanan

A

G



B

C

D

E

F

**Gambar 12.12**. Skema Kunjungan Preorder

Berdasarkan pohon diatas urutan kunjungan nya adalah : A, B, C, D, E, F, G

Contoh : Misalkan untuk benary tree berikut ini :

0

H

1 2

A K

3 6 7

C J L

5

4

B D

**Gambar 12.13**. Skema Kunjungan Preorder Hasilnya adalah : H, A, C, B, D, K, J, L

1. Kunungan Inorder (*symetric order*), menggunakan urutan :
   * Kunjungi cabang kiri
   * Cetak ini simpul yang dikunjungi
   * Kunjungi cabang kanan

A

G

B

C

D

E

F

**Gambar 12.14**. Skema Kunjungan Inorder

Berdasarkan pohon diatas urutan kunjungan nya adalah : D, B, E, A, F, C, G. Contoh : Misalkan untuk benary tree berikut ini :

0

H

1 2

A K

3 6 7

C J L

4 5

B D

**Gambar 12.15**. Skema Kunjungan Inorder Hasilnya adalah : A, B, C, D, H, J, K, L

1. Kunjungan Postorder, menggunakan urutan
   * Kunjungi cabang kiri
   * Kunjungi cabang kanan
   * Cetak isi simpul yang dikunjungi

A

G



B

C

D

E

F

**Gambar 12.16**. Skema Kunjungan PostOrder

Berdasarkan pohon diatas urutan kunjungan nya adalah : D, E, B, F, G, C, A. Contoh : Misalkan untuk benary tree berikut ini :

0

H

1 2

A K

3 6 7

C J L

5

4

B D

**Gambar 12.17**. Skema Kunjungan PostOrder Hasilnya adalah : B, D, C, A, J, L, K, H

1. **Mengubah Pohon Menjadi Pohon Biner**

Untuk mengubah struktur pohon secara umum menjadi struktur pohon biner, dapat dilakukan dengan langkah-langkah sebagai berikut :

1. tentukan pohon yang akan diubah menjadi pohon biner
2. Simpul-simpul yang mempunyai tingkat yang sama saling dihubungkan satu sama lain, dan cabang pohon digambarkan pada tingkat yang lebih tinggi.
3. Ambil sembarang simpul, simpul yang terletak tepat dibawahnya akan dipasang sebagai cabang kiri, dan simpul yang tepat disebelah kanannya akan dipasang sebagai cabang kanan.

Contoh :

a.

b.

c.



**Gambar 12.18**. Implementasi Pohon biasa menjadi pohon biner :

**Contoh Program 12.1 :**

Program Contoh\_aplikasi\_pohon\_biner; uses crt;

type st1 = string[5]; Tree = ^node; elemen = record niim : string[5];

nama : string[20]; ip : real;

jm\_sdr : integer; umur : integer; end;

node = record info : elemen; kr, kn : tree end;

procedure isi\_data(data:elemen; var p:tree); var lagi:char;

procedure sisip(data:elemen; var p:tree); var baru:tree;

begin

if p = nil then begin

new(baru); baru^.info:=data;

baru^.kr:=nil; baru^.kn:=nil, p:=baru;

end;

end else

if p^.info.nim > data.nim then sisip(data.p^kiri) else if p^.info.nim < data.nim then

sisip(data.p^kn) else

procedure tampilan\_isi;

begin

gotoxy(10, 5); writeln(' ');

gotoxy(10,6); writeln(' Isi Data Mahasiswa'); gotoxy(10,7); writeln(' ');

gotoxy(10,8); writeln(' Nim : '); gotoxy(10,9); writeln(' Nama Mahasiswa : ');

: ');

: ');

gotoxy(10,11); writeln(' Jumlah Saudara gotoxy(10,12); writeln(' Umur

: ');

gotoxy(10,10); writeln(' Ip

end; begin

gotoxy(10,13); writeln(' ');

gotoxy(10,14); writeln(' Isi lagi (Y/T)');

lagi := 'Y';

while upcase(lagi)='Y' do begin

clrscr; tampilan\_isi; with data do begin

gotoxy(29,8); readln(nim); gotoxy(29,9); readln(nama);

end;

gotoxy(29,10); readln(ip); gotoxy(29,11); readln(jm\_sdr); gotoxy(29,12); readln(umur);

end;

end;

sisip(data,p); gotoxy(35,14); readln(lagi);

procedure cari\_data\_mahasiswa(var p:tree); type st=string[5];

var lagi : char;

sdr : integer; ada : boolean; disitu : tree; mnim : st;

procedure tampilan\_cari; begin

gotoxy(10,5); writeln(' ');

gotoxy(10,6); writeln(' Cari Data'); gotoxy(10,7); writeln(' ');

gotoxy(10,8); writeln(' Nim : '); gotoxy(10,9); writeln(' Nama Mahasiswa : ');

: ');

gotoxy(10,10); writeln(' Ip

end;

gotoxy(10,13); writeln(' ');

gotoxy(10,14); writeln(' Cari Lagi (Y/T)'); gotoxy(10,15); writeln(' ');

gotoxy(10,11); writeln(' Umur : '); gotoxy(10,12); writeln(' Jumlah Saudara : ');

procedure cari (x:st; p:tree; var tamu:tree; var ketemu:boolean); var baru:tree;

begin

if p=nil then

ketemu := false;

begin

end else

if p^.info.nim > x then

cari(x, p^.kr, temu, ketemu)

end; begin

lagi := 'Y';

else

cari(x, p^.kn, temu, ketemu);

while upcase(lagi)='Y' do begin

clrscr; disitu := nil; tampilan\_cari;

gotoxy(29, 8); readln(mnim); cari(mnim, p, disitu, ada);

if ada then begin

tampilan\_cari; with disitu^.info do

gotoxy(29,8); writeln(nim);

begin

end;

gotoxy(29,10); writeln(nama);

gotoxy(29,11); writeln(ip:4:2); gotoxy(29,12); writeln(umur); gotoxy(29,13); writeln(jm\_sdr);

end else

gotoxy(35,14); readln(lagi);

begin

gotoxy(20,15); writeln(' Data tidak ada ! '); readln

end;

end;

end;

procedure cetak\_header\_laporan; var lagi : char;

i : integer;

procedure cetak\_header\_laporan; begin

writeln(' Laporan Data Mahasiswa ');

writeln(' ');

writeln(' No. Jum.Sdr Nim Nama Mahasiswa Ip Umur'); writeln ( );

end;

procedure inorder(p:tree);

inorder(p^.kr);

begin

if p <> nil then

begin

inc(i);

with p^.info do

writeln(' ', i:4, ' ', jm\_sdr:2, ' ', nim:5, ' ', nama:20, ' ',

ip:4:2, ' ', umur:2);

inorder(p^kn);

end;

end;

end;

clrscr; i:0; cerak\_header\_laporan; inorder(p);

writeln(' ');

write(' Ingin lihat laporan lagi (Y/T) : '); readln(lagi);

procedure hapus\_dat(var p:tree); type st = string[5];

var lagi : char;

ada : boolean; disitu : tree; mnim : st;

procedure tampilan\_hapus; begin

gotoxy(10,5); writeln(' ');

gotoxy(10,6); writeln(' Hapus Data Mahasiswa');

gotoxy(10,7); writeln(' ');

:');

gotoxy(10,11); writeln(' Umur

:');

gotoxy(10,10); writeln(' Ip

:');

gotoxy(10,9); writeln(' Nama

: ');

gotoxy(10,8); writeln(' Nim

end;

gotoxy(10,12); writeln(' Jumlah Saudara :');

gotoxy(10,13); writeln(' ');

gotoxy(10,14); writeln(' Hapus (Y/T) : ');

gotoxy(10,15); writeln(' ');

procedure cari(x:st; p:tree; var temu:tree; var ketemu:boolean); var baru : tree;

begin

if p= nil then begin

ketemu := false;

end else

if p^.info.nim = x then begin ketemu := tree; temu:=p; end

else

if p^.info.nim > x then cari(x,p^.kr, temu, ketemu)

else

cari(x,p^.kn, temu, ketemu);

end;

end;

clrscr; i:= 0; cetak\_header\_laporan; inorder(p); writeln(' ');

write(' Ingin lihat laporan lagi (Y/T) : '); readln(lagi);

procedure hapus\_data(var p:tree); type st=string[5];

var lagi:char; ada : boolean; disitu : tree; mnim : st;

procedure tampilan\_hapus; begin

gotoxy(10,5); writeln(' ');

gotoxy(10,6); writeln(' Hapus Data Mahasiswa');

gotoxy(10,7); writeln(' ');

:');

gotoxy(10,11); writeln(' Umur

:');

gotoxy(10,10); writeln(' Ip

:');

gotoxy(10,9); writeln(' Nama

: ');

gotoxy(10,8); writeln(' Nim

end;

gotoxy(10,12); writeln(' Jumlah Saudara :');

gotoxy(10,13); writeln(' ');

gotoxy(10,14); writeln(' Hapus (Y/T) : ');

gotoxy(10,15); writeln(' ');

procedure cari(x:st; p:tree; var temu:tree; var ketemu:boolean); var baru : tree;

begin

if p= nil then begin

ketemu := false;

end else

if p^.info.nim = x then begin ketemu := tree; temu:=p; end

else

if p^.info.nim > x then cari(x,p^.kr, temu, ketemu)

end; begin

lagi := 'Y';

else

cari(x,p^.kn, temu, ketemu);

while upcase (lagi) = 'Y' do begin

clrscr; i:= 0; cetak\_header\_laporan; inorder(p); writeln(' ');

end;

end;

write(' Ingin lihat laporan lagi (Y/T) : '); readln(lagi);

var hilang : char;

lagi := 'Y'; hilang := 'T';

begin

while := upcase (lagi) = 'Y' do begin

clrscr; disitu := nil; tampilan\_hapus; gotoxy(29,8); readln(mnim);

if mnim = ' ' the n begin lagi := 'Y'; exit end; cari(mnim, p, disitu, ada);

if ada then begin

tampilan\_hapus; with disitu^.info do begin

gotoxy(29,8); writeln(nim); gotoxy(29,9); writeln(nama); gotoxy(29,10); writeln(ip:4:2); gotoxy(29,11); writeln(umur); gotoxy(29,12); writeln(jm\_sdr);

end else

end;

gotoxy(35,14); readln(hilang);

if upcase(hilang) = 'Y' then penghapusan(mnim, p); gotoxy(10,14); writeln(' Hapus lagi (Y/T) : '); gotoxy(35,14); readln(lagi);

begin

gotoxy(20,15); writeln(' Data Tidak Ada ! '); readln;

end;

end;

end;

procedure tampilan; begin

gotoxy(' ');

gotoxy(' Pengolahan Data Mahasiswa');

gotoxy(' ');

gotoxy(' 1. Masukkan Data Mahasiswa '); gotoxy(' 2. Mencari Data Mahasiswa'); gotoxy(' 3. Hapus Data Mahasiswa'); gotoxy(' 4. Laporan Data Mahasiswa');

gotoxy(' ');

gotoxy(' Pilih : ');

end;

var a:tree;

c : elemen; terus : boolean; pil : integer;

terus := true;

begin

while terus do

begin

clrscr; tampilan; pil := 0;

while (pil < 1) or (pil > 5) do begin

gotoxy(25,14); readln(pil);

end;

case pil of

* 1. : isi\_data(c,a);
  2. : cari\_data\_mahasiswa(a);
  3. : hapus\_data(a);
  4. : laporan(a);
  5. : terus := false;

end.

end;

end;

**DAFTAR PUSTAKA**

Munir, Rinaldi. 2007. *Algoritma & Pemrograman Dalam Bahasa Pascal dan C.*

Informatika. Bandung.

Hasbi, M. 2003. *Struktrur Data Dan Algoritma Dalam Bahasa Pemrograman Turbo Pascal.* Gava Media. Yogyakarta.

Rosa A, S, Shalahuddin, M. 2010. *Algoritma dan Pemrograman.*Modula. Bandung Santosa, P. Isap. Ir.Msc. 1993, *Struktur Data Menggunakan Turbo Pascal 6.0.* Andi Offset.

Yogyakarta.

Kadir, Abdul. 2003. *Pemrograman C++.* Andi Offset. Yogyakarta.

Purwanto, Eko. 2011. *Bahasa Pemrograman Pascal.* Duta Publishing. Surakarta.