BAB II

TINJAUAN PUSTAKA

2.1. Himpunan

Definisi 2.1.1.

Himpunan merupakan kumpulan dari objek-objek yang memiliki ciri tertentu dan terdefinisikan secara jelas. Benda atau objek di dalam suatu himpunan dinamakan unsur, elemen atau anggota himpunan.

Notasi himpunan adalah {...}. Himpunan S merupakan himpunan benda-benda a,b,c dapat dinyatakan dengan S = {a,b,c}. Adapula himpunan yang tidak memiliki anggota. Himpunan seperti ini dinamakan himpunan kosong.

Definisi 2.1.2.

Misalkan A dan B adalah himpunan. *A himpunan bagian dari B*, dinotasikan $A \subseteq B$ jika dan hanya jika setiap elemen dari A adalah elemen dari B ($A \subseteq B \Leftrightarrow \forall x\, [x \in A \Rightarrow x \in B]$).

Pasangan terurut (a,b) adalah penulisan objek a dan b dalam urutan yang telah ditentukan dengan a menunjukkan urutan pertama dan b menunjukkan urutan kedua. Pasangan terurut (a_1,b_1) dan (a_2,b_2) bernilai sama jika dan hanya jika $a_1 = a_2$ dan $b_1 = b_2$.

Definisi 2.1.3.

Misalkan A dan B himpunan tidak kosong. *Hasil kali kartesius (Cartesian product)* A x B adalah himpunan semua pasangan terurut (a,b) dengan $a \in A$ dan $b \in B$ yang dinotasikan sebagai berikut :

$$A \times B = \{(a,b) \mid a \in A \wedge b \in B\}$$

2.2. Relasi

Definisi 2.2.1.

Relasi adalah himpunan pasangan terurut.

Definisi 2.2.2.

Misalkan A dan B himpunan tak kosong. *Relasi R dari A ke B* adalah himpunan bagian dari A x B. Jika $R \subseteq A \times B$ dan $(a,b) \in R$ dapat dikatakan bahwa a direlasikan ke b oleh R dan dinotasikan aRb. *Relasi pada himpunan A* adalah relasi dari A ke A.

Contoh 2.2.1:

$A = \{1, 2, 3, 4, 5\}$. Relasi R pada himpunan A didefinisikan :

$R = \{(a,b) \in A \times A \mid a < b\}$, sehingga :

$R = \{(1,2), (1,3), (1,4), (1,5), (2,3), (2,4), (2,5), (3,4), (3,5), (4,5)\}$

2.2.1. Matriks Relasi

Suatu relasi dapat direpresentasikan dalam sebuah matriks.

Definisi 2.2.1.1.

Misalkan $A = \{a_1, a_2, \ldots, a_m\}$ dan $B = \{b_1, b_2, \ldots, b_n\}$ adalah himpunan berhingga yang masing – masing mempunyai m dan n anggota, dan R adalah suatu relasi dari A ke B. *Matriks relasi R* adalah matriks $M_R = [m_{ij}]$ yang berukuran m x n dengan :

$$m_{ij} = \begin{cases} 1, \text{jika } (a_i, b_j) \in R \\ 0, \text{lainnya} \end{cases}$$

Matriks relasi ini merupakan matriks boolean sebab masukan matriksnya dalam {1,0}.

Contoh 2.2.1.1 :

Matriks relasi pada contoh 2.2.1 adalah :

$$M_R = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Dari contoh 2.2.1.1 terlihat bahwa matriks relasi pada himpunan A selalu berbentuk matriks bujur sangkar.

Definisi 2.2.1.2.

Misalkan $A = \{a_1, a_2, \ldots, a_n\}$ dan R, S relasi pada himpunan A. *Matriks relasi irisan R dan S* adalah matriks $M_{R \cap S} = [m_{ij}]$ yang berukuran n x n dengan

$$m_{ij} = \begin{cases} 1, \text{jika } (a_i, a_j) \in R \wedge (a_i, a_j) \in S \\ 0, \text{lainnya} \end{cases}$$

Matriks relasi gabungan R dan S adalah matriks $M_{R \cup S} = [m_{ij}]$ yang berukuran n x n dengan

$$m_{ij} = \begin{cases} 1, \text{jika } (a_i, a_j) \in R \vee (a_i, a_j) \in S \\ 0, \text{lainnya} \end{cases}$$

Teorema 2.2.1.1.

Jika R dan S relasi pada himpunan A, maka $M_{R \cap S} = M_R \wedge M_S$ dan $M_{R \cup S} = M_R \vee M_S$.

Bukti :

Elemen ke (i,j) dari matriks $M_{R \cap S}$ yang menunjukkan irisan dari R dan S bernilai 1 jika kedua elemen ke (i,j) dari M_R dan M_S

bernilai 1 dan lainnya bernilai 0. Hal ini sama dengan elemen ke (i,j) dari $M_R \wedge M_S$.

Elemen ke (i,j) dari matriks $M_{R \cup S}$ yang menunjukkan gabungan dari R dan S, bernilai 1 jika elemen ke (i,j) dari M_R atau M_S bernilai 1, dan bernilai 0 untuk lainnya. Hal ini sama dengan elemen ke (i,j) dari matriks $M_R \vee M_S$. ♦

Contoh 2.2.1.2.

A = {1, 2, 3}, relasi R dan S pada himpunan A adalah

$R = \{(a,b) \in A \times A \mid a \geq b\}$ dan

$S = \{(a,b) \in A \times A \mid a = b+1 \vee b = a+1\}$, sehingga

R = {(1,1), (2,1), (2,2),(3,1), (3,2), (3,3)} dan

S = {(1,2), (2,1), (2,3), (3,2)}

$$M_R = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \text{ dan } M_S = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Dengan menggunakan teorema 2.2.1.1 maka :

$$M_{R \cap S} = M_R \wedge M_S = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$M_{R \cup S} = M_R \wedge M_S = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Matriks $M_{R \cap S}$ dan $M_{R \cup S}$ di atas terbukti benar sebab $R \cap S = \{(2,1), (3,2)\}$ dan $R \cup S$ = {(1,1), (1,2), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3)}

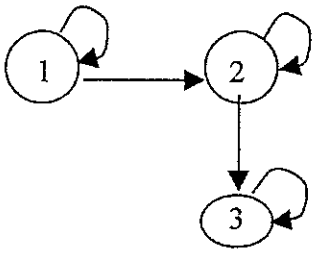
2.2.2. Digraph Relasi

Suatu relasi R pada himpunan A dapat juga digambarkan digraphnya. Vertek – vertek dalam digraph menyatakan elemen dari A, sedangkan edge (rusuk) berarah mewakili anggota relasi R.

Contoh 2.2.2.1 :

A = {1, 2, 3} dan relasi R pada himpunan A adalah

R = {(a,b)∈ A x A | b = a+1 ∨ a = b}, sehingga R = {(1,1), (1,2), (2,2), (2,3), (3,3)}. Digraph dari relasi R adalah :



Gambar 2.2.2.1. Digraph relasi R pada contoh 2.2.2.1

2.2.3. Komposisi Relasi

Definisi 2.2.3.1.

Misalkan R_1 adalah relasi dari A ke B dan R_2 adalah relasi dari B ke C. *Komposisi dari R_1 dan R_2*, dinotasikan R_2 o R_1 adalah relasi dari A ke C yang didefinisikan sebagai berikut :

$R_2 \circ R_1 = \{(a,c) \mid (a,b) \in R_1 \text{ dan } (b,c) \in R_2 \text{ untuk suatu } b \in B\}$

Contoh 2.2.3.1.

A = {1, 2}, relasi pada himpunan A adalah

$R = \{(a,b) \in A \times A \mid a \geq b\}$ dan

$S = \{(a,b) \in A \times A \mid a^2+b^2 > 4\}$, sehingga

$R = \{(1,1), (2,1), (2,2)\}$ dan $S = \{(1,2), (2,1), (2,2)\}$.

$S \circ R = \{(1,2), (2,1), (2,2)\}$

$R \circ S = \{(1,1), (2,1), (2,2)\}$

2.2.4. Path dalam Relasi

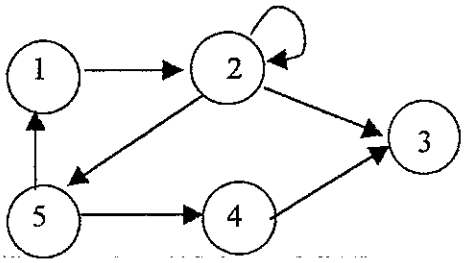
Definisi 2.2.4.1.

Misalkan R relasi pada himpunan A. *Path* dengan panjang n dalam R dari a ke b adalah barisan terbatas $\pi = a, x_1, x_2, \ldots, x_{n-1}, b$ dimulai dengan a dan diakhiri dengan b, sedemikian $aRx_1, x_1Rx_2, \ldots, x_{n-1}Rb$. Panjang path adalah jumlah edge dalam path yang dinotasikan dengan $L(\pi)$.

Path dengan panjang n melibatkan n+1 elemen dari himpunan A, walaupun elemen tersebut tidak perlu berbeda. Untuk path yang elemen-elemennya berbeda dinamakan path sederhana.

Contoh 2.2.4.1 :

Misalkan suatu relasi mempunyai digraph sebagai berikut :



Gambar 2.2.4.1. Contoh suatu digraph

π_1 = 1, 2, 5, 4, 3 adalah path dengan $L(\pi_1) = 4$ dari vertek 1 ke vertek 3, π_2 = 1, 2, 5, 1 adalah path dengan $L(\pi_2) = 3$ dari vertek 1 kembali lagi ke vertek 1.

Path yang dimulai dan diakhiri di titik yang sama disebut cycle. Dalam contoh 2.2.4.1, π_2 merupakan cycle dengan $L(\pi_2) = 3$.

Definisi 2.2.4.2.

Misalkan R relasi pada himpunan A dan n bilangan bulat positif, maka *relasi R^n pada A* adalah relasi yang menunjukkan path dengan panjang n dalam R. Dengan menuliskan aR^nb berarti terdapat path dengan panjang n dari a ke b dalam R.

Definisi 2.2.4.3.

Misalkan R relasi pada himpunan A. *Relasi* $R^* = \{(a,b) \in A \times A |$ terdapat suatu path dalam R dari a ke b$\}$ disebut relasi keterhubungan (connectivity relation) pada A. Panjang dari path ini secara umum tergantung pada a dan b. Dengan menuliskan aR^*b berarti terdapat path dalam R dengan suatu panjang dari a ke b.

Teorema 2.2.4.1.

Jika R relasi pada himpunan A maka $R^* = \bigcup_{i=1}^{\infty} R^i$.

Bukti :

Dari definisi 2.2.4.2, untuk setiap $n > 0$, $R^n = \{(a,b) \in A \times A |$ terdapat path dalam R dengan panjang n dari a ke b$\}$. Sehingga berdasarkan definisi 2.2.4.3 $R^* = \bigcup_{i=1}^{\infty} R^i$. ♦

Contoh 2.2.4.2.

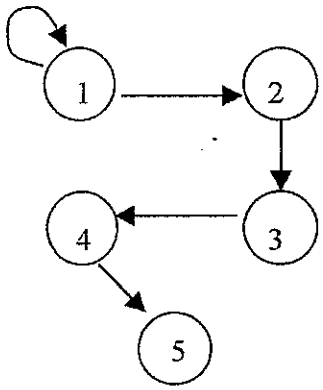
$A = \{1, 2, 3, 4, 5\}$ dan relasi R pada himpunan A adalah

$R = \{(a,b) \in A \times A | a^2+b^2 = 1 \vee b = a+1\}$, sehingga

R = {(1,1), (1,2), (2,3), (3,4), (4,5)} Akan dicari :

a). R^2 b). R*

Digraph R ditunjukkan dalam gambar 2.2.4.2.



Gambar 2.2.4.2. Digraph relasi pada contoh 2.2.4.2.

a). 1 R^2 1 sebab 1 R 1 dan 1 R 1

1 R^2 2 sebab 1 R 1 dan 1 R 2

1 R^2 3 sebab 1 R 2 dan 2 R 3

2 R^2 4 sebab 2 R 3 dan 3 R 4

3 R^2 5 sebab 3 R 4 dan 4 R 5

Jadi R^2 = {(1,1), (1,2), (1,3), (2,4), (3,5)}

b). Untuk menghitung R* diperlukan semua pasangan terurut dari vertek – vertek yang mempunyai path dengan suatu panjang dari vertek pertama ke vertek kedua. Dari gambar 2.2.4.2 terlihat bahwa :

R* = {(1,1), (1,2), (1,3), (1,4), (1,5), (2,3), (2,4), (2,5), (3,4), (3,5) (4,5)}

Sebagai contoh, (1,4)∈R* sebab terdapat path dengan $L(\pi_1) = 3$ dari 1 ke 4, yaitu : π_1 = 1, 2, 3, 4. Dengan cara yang sama,

$(1,5) \in R^*$ sebab terdapat path dengan $L(\pi_2) = 4$ dari 1 ke 5, yaitu : $\pi_2 =$ 1, 2, 3, 4, 5.

2.2.5. Sifat - sifat Relasi

Definisi 2.2.5.1.

Relasi R pada himpunan A disebut *refleksif* jika dan hanya jika aRa untuk setiap $a \in A$.

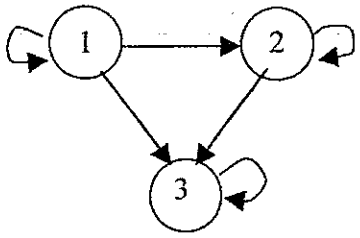
Dengan melihat matriks suatu relasi dapat ditentukan apakah relasi tersebut refleksif atau bukan. Diagonal utama dari matriks relasi refleksif semuanya bernilai 1. Jika dilihat dari digraphnya, maka relasi refleksif mempunyai cycle dengan panjang 1 pada setiap verteknya.

Contoh 2.2.5.1 :

$A = \{1, 2, 3\}$. Relasi $R = \{(a,b) \in A \times A \mid a \leq b\}$ adalah $R = \{(1,1), (1,2), (1,3), (2,3), (2,2), (2,3), (3,3)\}$ merupakan relasi refleksif, sebab untuk setiap $a \in A$, aRa yaitu (1,1), (2,2), dan (3,3), masing-masing berada dalam R.

Matriks relasinya adalah : $M_R = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$

Sedangkan digraphnya :



Gambar 2.2.5.1. Digraph relasi R pada contoh 2.2.5.1

Definisi 2.2.5.2.

Relasi pada himpunan A disebut *simetris* jika dan hanya jika aRb maka bRa, untuk semua a, b $\in$ A.

Matriks relasi simetris $M_R = [m_{ij}]$ mempunyai sifat sebagai berikut :

Jika $m_{ij} = 1$ maka $m_{ji} = 1$ atau jika $m_{ij} = 0$ maka $m_{ji} = 0$.

Jadi matriks tersebut simetris terhadap diagonal utama atau $M_R = M_R^T$.

Digraph dari relasi simetris mempunyai sifat bahwa jika terdapat edge dari vertek a ke vertek b maka terdapat edge pula dari vertek b ke vertek a.

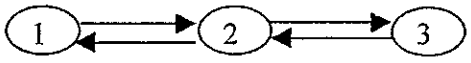
Contoh 2.2.5.2 :

A = {1, 2, 3}. Relasi R = {(a,b) $\in$ A x A | a = b+1 $\vee$ b = a+1}adalah R = {(1,2), (2,1), (2,3), (3,2)} merupakan relasi simetris.

Matriks relasinya adalah:

$$M_R = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Sedangkan digraphnya adalah :



Gambar 2.2.5.2. Digraph relasi R pada contoh 2.2.5.2

Definisi 2.2.5.3.

Relasi R pada himpunan A disebut *transitif* jika dan hanya jika aRb dan bRc maka aRc, untuk semua a, b, c ∈ A.

Dilihat dari matriksnya, menentukan relasi transitif tidaklah semudah pada relasi refleksif dan simetris. Matriks relasi transitif $M_R = [m_{ij}]$ mempunyai ciri – ciri sebagai berikut :

Jika $m_{ij} = 1$ dan $m_{jk} = 1$ maka $m_{ik} = 1$

Sedangkan digraphnya mempunyai sifat bahwa apabila terdapat edge dari a ke b dan dari b ke c maka juga terdapat edge dari a ke c.

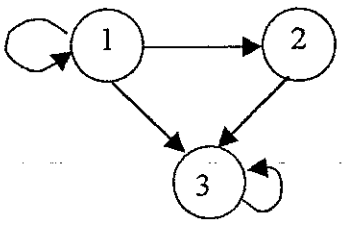
Contoh 2.2.5.3:

$A = \{1, 2, 3\}$ dan relasi $R = \{(a,b) \in A \times A \mid a^2 + b^2 \neq 8 \vee a < b\}$ adalah $R = \{(1,1), (1,2), (1,3), (2,3), (3,3)\}$ merupakan relasi transitif.

Matriks relasinya adalah :

$$M_R = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Sedangkan digraphnya sebagai berikut :

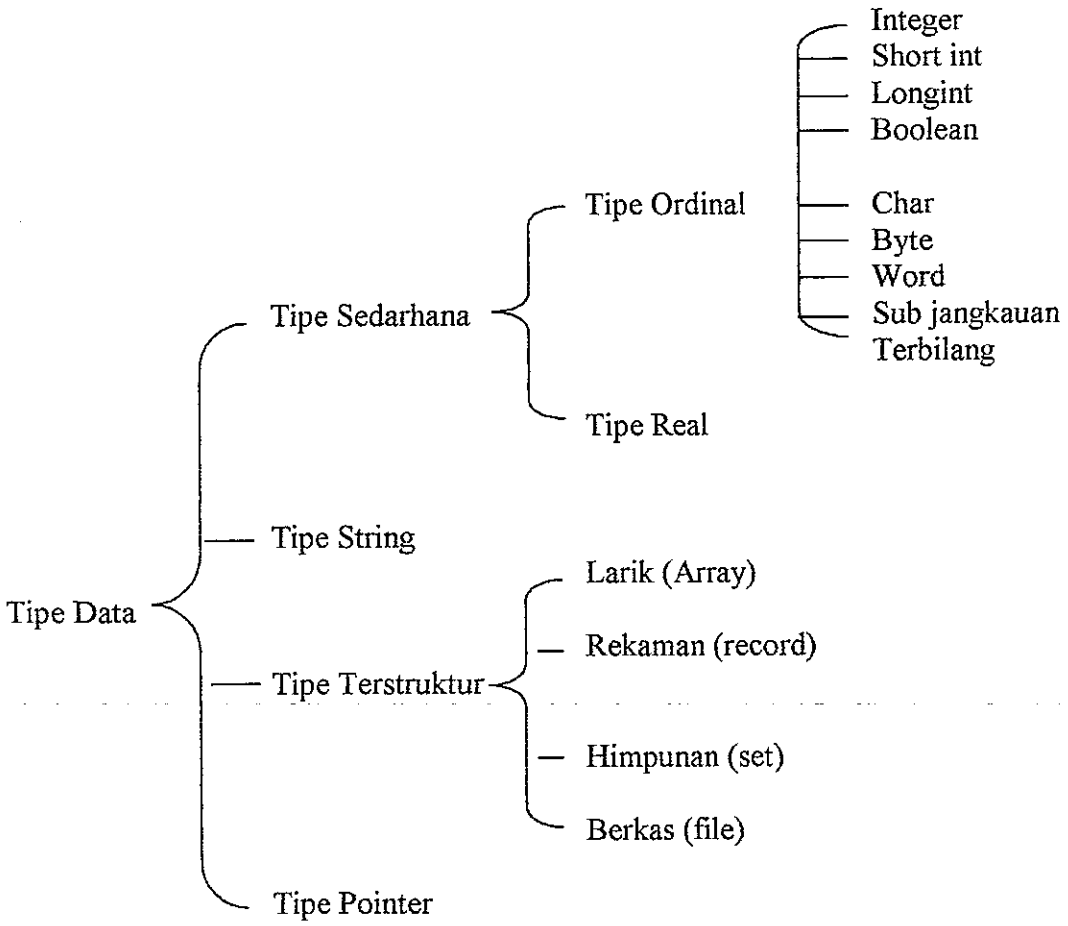


Gambar 2.2.5.3. Digraph relasi R pada contoh 2.2.5.3.

2.3. Struktur Data Dinamis

Untuk memecahkan suatu masalah, komputer memerlukan informasi atau data. Informasi adalah pengetahuan yang dapat disampaikan, termasuk ide dan konsep, sedangkan data adalah informasi dalam bentuk yang dapat digunakan komputer. Misalnya angka dan huruf. Data ini mempunyai beberapa tipe, yang biasa dinamakan tipe data.

Dalam Pascal, semua variabel yang akan dipakai harus sudah ditentukan tipe datanya. Dengan menentukan tipe data suatu variabel, dapat sekaligus menentukan batasan nilai dan jenis operasi yang bisa dilaksanakan atas variabel tersebut. Secara lengkap, tipe data dalam Turbo Pascal dapat digambarkan sbb:

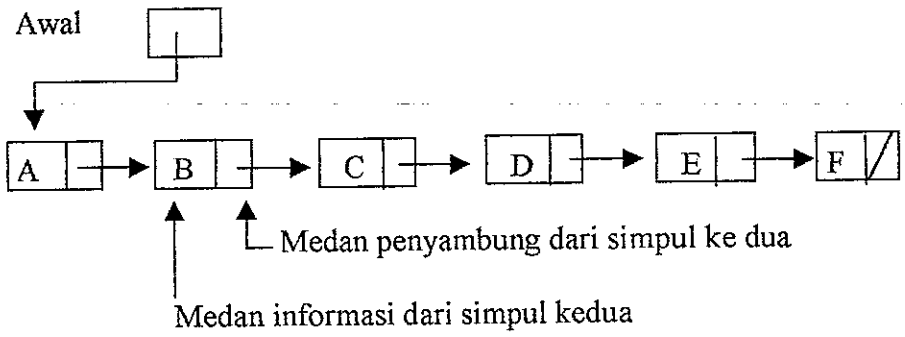


Variabel dengan tipe data pointer merupakan variabel dinamis, yaitu suatu variabel yang akan dialokasikan hanya pada saat diperlukan setelah program dieksekusi. Variabel inilah yang nantinya digunakan untuk membentuk senarai berantai.

Kumpulan variabel, yang mungkin dari beberapa tipe data yang berbeda, dihubungkan dengan berbagai cara disebut struktur data. Struktur data yang dapat digunakan untuk mengalokasikan dan mengdealokasikan memori secara dinamis, yaitu sesuai dengan kebutuhan pada saat suatu program dieksekusi disebut struktur data dinamis.

Salah satu struktur data dinamis yang paling sederhana adalah senarai berantai (linked list), yaitu kumpulan komponen yang disusun secara berurutan dengan bantuan pointer. Pointer adalah suatu sel yang nilainya menunjuk ke sel yang lain. Masing-masing komponen dalam senarai berantai dinamakan dengan simpul (node). Setiap simpul terbagi menjadi dua bagian. Bagian pertama disebut medan informasi, berisi informasi yang akan disimpan dan diolah. Bagian kedua disebut medan penyambung (link field), berisi alamat simpul berikutnya.

Contoh 2.3.1.



Gambar 2.3.1. Contoh senarai berantai dengan 6 simpul

Contoh 2.3.1 menunjukkan diagram skematis dari senarai berantai dengan 6 simpul. Setiap simpul di gambar dalam dua bagian. Bagian kiri adalah medan informasi. Bagian kanan berupa medan penyambung, sehingga dalam diagram digambarkan sebagai anak panah. Medan penyambung sebenarnya adalah sebuah pointer yang menunjuk ke simpul berikutnya, sehingga nilai dari medan ini adalah alamat suatu lokasi tertentu dalam memori.

Misalkan A, B, C, D, E Dan F adalah variabel yang bertipe pointer dengan simpul-simpulnya bertipe rekaman, maka dekalarasi tipe pointernya untuk membentuk senarai berantai seperti contoh 2.3.1 adalah sebagai berikut :

```
type variabel     = ^simpul;
     simpul       = record
                      info          : tipe;
                      berikut       : variabel;
                    end;
```

dengan variabel : nama variabel yang bertipe ponter

simpul : nama simpul

info : nama medan dari data yang bisa bertipe sembarang

tipe : tipe data dari masing-masing medan

berikut : nama medan yang bertipe pointer

Dari contoh 2.3.1, pointer Awal yang bukan merupakan bagian dari senarai menunjuk ke simpul pertama dari senarai tersebut. Medan penyambung (pointer) dari suatu simpul yang tidak menunjuk simpul lain disebut dengan pointer kosong yang nilainya dinyatakan sebagai **nil** (nil adalah kata baku Pascal yang berarti pointer tidak menunjuk ke suatu simpul, dan nilainya sama dengan 0 atau bilangan negatif).

Operasi yang dapat dilakukan pada senarai berantai diantaranya adalah menambah simpul dan membaca isi senarai. Untuk lebih jelasnya, dimisalkan deklarasi pointer dan simpul yang digunakan adalah sebagai berikut :

```
type Simpul = ^Data;
    Data    = record
                Info        : char;
                Berikut     : Simpul;
              end;
var Elemen              : char;
    Awal, Akhir, Baru   : Simpul;
```

Berikut prosedur untuk menyisipkan simpul dan membaca isi senarai dari kiri ke kanan (membaca maju).

```
Procedure SISIP_ELEMEN(var Awal, Akhir : Simpul; Elemen :
char);
var Baru, Bantu : Simpul;
begin
repeat
   write ('Masukkan datanya : ');readln(Elemen);
   new(Baru);
   Baru^.Info   := Elemen;
   if Awal = nil then {Senarai masih kosong}
      begin
         Awal :=Baru;
         Akhir := Baru
      end
   else
      begin
         {Mencari lokasi yang sesuai}
         Bantu :=Awal;
         while Elemen > Bantu^.Berikut^.Info do
             Bantu := Bantu^.Berikut;
```

```
            {Menyisipkan simpul baru}
            Baru^.Berikut := Bantu^.Berikut;
            Bantu^.Berikut := Baru;
         end;
      write ('Akan Memasukkan Lagi ? (Ya / Tidak) :');
      readln(Jawab);writeln;
until Jawab in ['T','t']
end;
```

Program 2.3.1. Prosedur untuk menyisipkan simpul di senarai berantai

```
Procedure BACA_MAJU (Awal : Simpul);
var Bantu : Simpul ;
begin
   Bantu := Awal ;
   write('Awal --> ');
   while Bantu <> nil do
    begin
      write(Bantu^.Info,' --> ');
      Bantu := Bantu^.Berikut;
    end;
   write('nil'); readln
end;
```

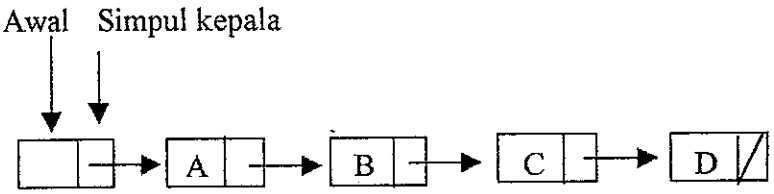
Program 2.3.2. Membaca senarai berantai dari kiri ke kanan (membaca maju)

2.3.1. Senarai Berantai Berkepala

Senarai berantai berkepala (headed linked list) adalah senarai berantai yang simpul pertamanya merupakan simpul kepala (header node). Simpul ini tidak berisi informasi seperti halnya simpul-simpul

lain dalam senarai berantai, tetapi keberadaannya sangat diperlukan untuk lebih mempercepat proses eksekusi.

Contoh 2.3.1.1.



Gambar 2.3.1.1. Contoh pemakaian simpul kepala

Pada contoh 2.3.1.1, bagian informasi pada simpul kepala dibiarkan kosong. Prosedur untuk proses inisialisasi senarai berantai yang bertujuan untuk membentuk simpul kepala adalah sebagai berikut :

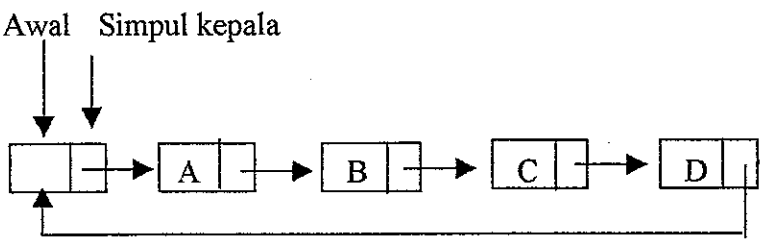
```
Procedure AWALAN_SBK(var Awal : simpul);
begin
   new(Awal);
   Awal^.Berikut := nil;
end;
```

Program 2.3.1.1. Pembentukan simpul kepala

2.3.2. Senarai Berantai Berputar

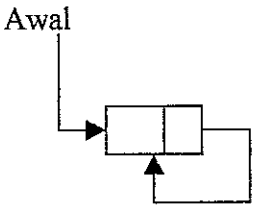
Senarai berantai berputar (circular linked list) adalah senarai yang pointer pada simpul terakhir diarahkan kembali ke simpul pertama. Sehingga tidak akan ada simpul yang berisi pointer yang bernilai nil. Keuntungannya adalah bahwa untuk membaca seluruh isi senarai berantai tidak harus dilakukan dari simpul pertama. Jika menggunakan simpul kepala, maka dinamakan dengan senarai berantai berputar berkepala (headed circular linked list). Dalam hal ini pointer dalam simpul terakhir diarahkan kembali ke simpul kepala.

Contoh 2.3.2.1.



Gambar 2.3.2.1. Contoh Senarai berantai berputar berkepala dengan lima simpul (termasuk simpul kepala)

Simpul kepala pada senarai berantai berputar digambarkan sebagai berikut :



Gambar 2.3.2.2. Simpul kepala pada senarai berantai berputar

Untuk membentuk simpul kepala yang demikian, prosedur inisialisasi disajikan dalam program di bawah ini :

```
Procedure INISIALISASI_SBPK (var Kepala : simpul);
begin
   new(kepala);
   Kepala^.Berikut := Kepala;
end;
```

Program 2.3.2.1. Inisialisasi simpul kepala pada senarai berantai berputar

2.3.3. Senarai Berantai Banyak

Senarai berantai banyak (multilist) sering disebut dengan senarai dari senarai (list of list). Salah satu contoh pemakaian senarai berantai

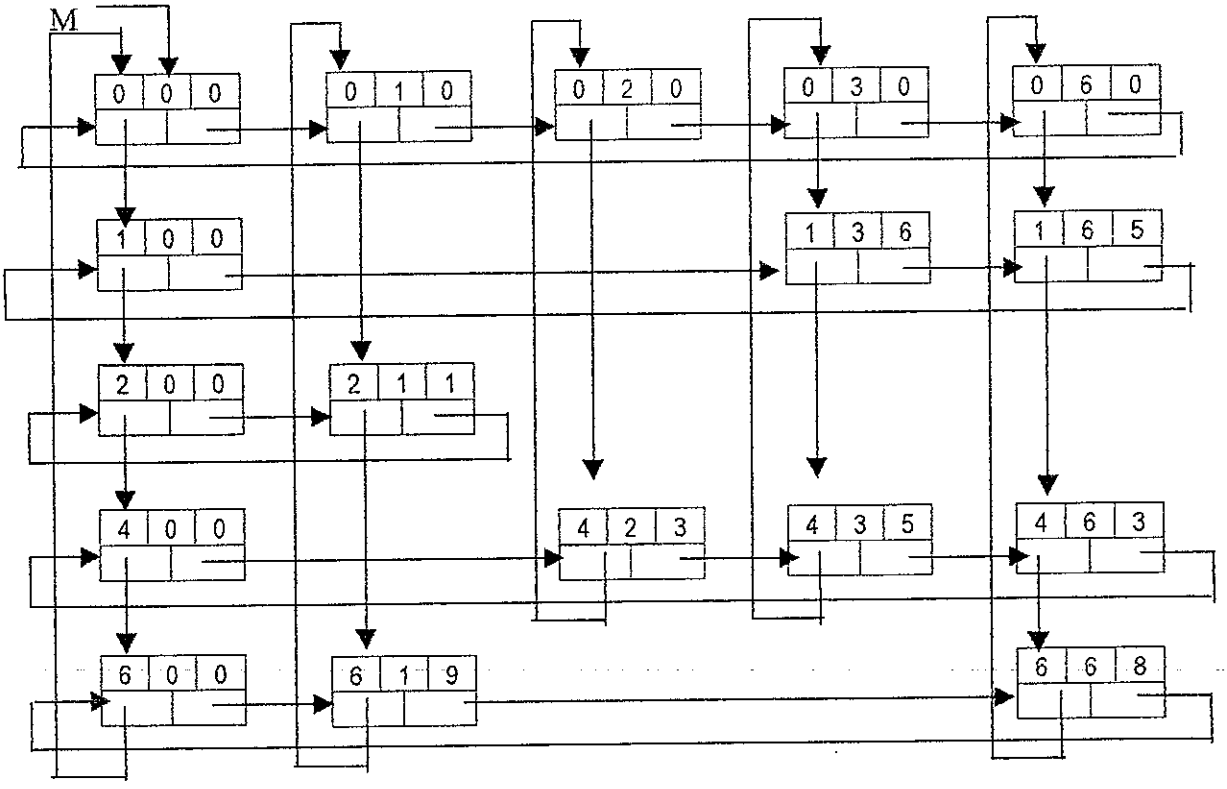
banyak adalah pada matriks jarang (sparse matrix). Matriks jarang adalah suatu matriks yang elemen tak nolnya sangat sedikit dibanding dengan elemen nolnya.

Contoh 2.3.3.1.

Salah satu contoh matrik jarang :

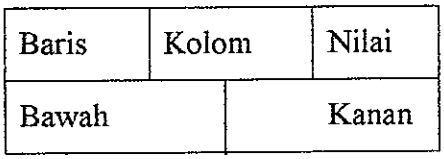
$$\begin{bmatrix} 0 & 0 & 6 & 0 & 0 & 5 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 5 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 8 \end{bmatrix}$$

Dengan senarai berantai banyak, penyajian matriks di atas menjadi seperti di bawah ini :



Gambar 2.3.3.1. Penyimpanan matriks jarang dengan senarai berantai banyak

Struktur setiap simpul untuk matriks jarang adalah :



Gambar 2.3.3.2. Struktur simpul untuk matriks jarang

Baris, Kolom dan Nilai menunjukkan nomor baris, nomor kolom dan nilai elemen tak nol. Sedangkan Bawah dan Kanan masing-masing adalah pointer yang bertipe sama dengan struktur simpul di atas. Pointer Bawah menunjuk ke nomor baris berikutnya, pointer Kanan menunjuk ke nomor kolom berikutnya.

Untuk mempermudah penambahan elemen baru, setiap baris dan kolom mempunyai simpul kepala. Keseluruhan senarai juga mempunyai simpul kepala yang ditunjuk oleh pointer M, yang mempunyai Baris = 0, Kolom = 0 dan Nilai = 0. Simpul kepala dari kolom pertama ditunjuk oleh M^.Kanan, sedangkan simpul kepala dari baris pertama ditunjuk oleh M^.Bawah. Simpul kepala dari suatu baris selalu berisi Kolom = 0 dan Nilai = 0. Simpul kepala dari suatu kolom selalu berisi Baris = 0 dan Nilai = 0.

Deklarasi simpul seperti di atas adalah

```
type Mat = ^ Elemen ;
   Elemen = record
                  Baris, {  Nomor baris }
                  Kolom : Integar { Nomor kolom }
                  Nilai = real ; { Nilai elemen }
                  Kanan,  { ke kolom berikutnya }
                  Bawah : Mat { ke baris berikutnya }
               end ;
```