BAB II

MATERI PENUNJANG

Dalam bab ini dibahas beberapa materi yang dapat membantu dalam menyelesaikan permasalahan. Materi-materi tersebut adalah Algoritma, Struktur Data, Tipe Data, Tipe Data Abstrak, Pointer, Linked list dan Polinomial.

2.1 Algoritma

Algoritma didefinisikan sebagai himpunan berhingga intruksi-intruksi yang apabila ditelusuri akan menghasilkan suatu penyelesaian dari tugas khusus. Oleh karena itu suatu algoritma harus memenuhi kriteria berikut :

1. Input

   Suatu algoritma dapat mempunyai nol atau banyak input, yaitu suatu kuantitas dimana diberikan nilai awal sebelum algoritma dimulai. Input diperoleh dari himpunan obyek yang telah ditetapkan.

2. Output

   Suatu algoritma harus menghasilkan minimal satu nilai hasil. Nilai output adalah penyelesaian dari persoalan.

3. Keterdefinisian (*definiteness*)

   Setiap intruksi dalam suatu algoritma harus jelas dan tidak mempunyai arti ganda (*ambiguity*).

4. Keberhinggaan (*finiteness*)

   Apabila intruksi-intruksi suatu algoritma ditelusuri, maka untuk semua kasus, algoritma akan berhenti setelah diberikan sejumlah langkah berhingga.

5. Keefektifan (*effectiveness*)

Setiap intruksi dalam suatu algoritma harus dapat dilaksanakan (*executable*) langsung mengarah ke sasaran yang dituju.

2.2. Struktur Data

Secara umum suatu struktur data terdiri dari beberapa bagian seperti himpunan nilai-nilai data dan sejumlah operasi dasar yang bekerja pada data tersebut menurut suatu algoritma tertentu. Sebagai contoh bila nilai data yang dibicarakan adalah bagian dari himpunan bilangan real, operasi-operasi yang dapat dilakukan pada himpunan tersebut adalah operasi tambah, kurang, bagi, dan kali. Sedangkan algoritma yang diterapkan pada himpunan data tersebut mungkin merupakan algoritma untuk mencari nilai maksimum atau minimum dari data yang ada.

Secara lengkap suatu struktur data mempunyai tiga bagian utama, yaitu :

1. Himpunan dari struktur tempat penyimpanan (*storage*), yang merupakan koleksi dari variabel dan hubungan antar suatu variabel dengan variabel yang lain.
2. Himpunan fungsi-fungsi dasar, yang dapat digunakan pada struktur tempat penyimpanan yang ada dan dapat digunakan pada setiap bagian program.
3. Himpunan dari algoritma, yang dapat digunakan untuk pengubahan struktur tempat penyimpanan.

2.3. Tipe Data

Tipe data didefinisikan sebagai koleksi dari obyek (nilai) dan operasi-operasi yang dilakukan pada obyek tersebut.

Sebagai contoh tipe data integer yang terdiri dari obyek {**minint**,..., -4, -3, -2, -1, 0, 1, 2, 3, 4,...,**maxint**} dimana **minint** dan **maxint** adalah masing-masing nilai bilangan bulat terkecil dan terbesar yang dapat direpresentasikan komputer. Operasi-operasi yang dapat dilakukan pada bilangan bulat adalah berupa operasi aritmetika + , -, *, div dan mod. Juga terdapat operasi yang digunakan untuk menguji kesamaan atau ketaksamaan dua bilangan bulat.

2.4. Tipe Data Abstrak

Tipe Data Abstrak (TDA) didefinisikan sebagai suatu tipe data yang terorganisasi sedemikian rupa sehingga spesifikasi dari obyek dan operasi-operasinya terpisah dari representasi obyek dan implementasi operasinya.

Untuk mengilustrasikan konsep TDA, berikut ini diberikan suatu spesifikasi TDA bilangan asli.

Struktur *BilanganAsli* (*NaturalNumbers*) adalah

Obyek : Suatu subrange terurut dari bilangan bulat yang dimulai dari 0 sampai bilangan bulat maksimum (**maxint**) dalam komputer

Functions :

Untuk setiap x,y ∈ *BilanganAsli*; **true,false** ∈ *Boolean*

dan +,-,<,> dan = adalah operasi bilangan asli.

Zero():BilanganAsli ::= 0;

IsZero(x):Boolean ::= **if** x=0 **then** *IsZero* := **true**

else *IsZero* := **false**;

Add(x,y):BilanganAsli	::= **if** x+y <= **maxint then** *Add* := x+y else *Add* := **maxint**;
Equal(x,y):Boolean	::= **if** x=y **then** *Equal* := **true** **else** *Equal* := **False**;
Successor(x):BilanganAsli	::= **if** x=**maxint then** *Successor* :=x **else** *successor* := x+1;
Substract(x,y):BilanganAsli	::= **if** x<y **then** *Substract* := 0 **else** *Substract* := x-y;

Spesifikasi operasi dari TDA terdiri dari nama fungsi, tipe dari setiap argumen dan tipe dari hasil yang diperoleh. Dari contoh TDA *BilanganAsli* dapat dijelaskan sebagai berikut :

Definisi struktur dimulai dengan kata struktur dan singkatannya. Terdapat dua hal utama dalam definisi yaitu obyek dan fungsi-fungsi. Obyek bilangan asli didefinisikan sebagai bagian dari bilangan bulat. Definisi fungsi lebih kompleks, pertama definisi simbol-simbol yang digunakan yakni x dan y untuk menunjukkan dua elemen dari *BilanganAsli*, **true** dan **false** adalah elemen dari *Boolean* dan definisi operasi yang digunakan fungsi. Simbol "::=" dibaca "didefinisikan sebagai" dan ":=" dibaca "sama dengan".

Fungsi yang pertama adalah *Zero* yang tidak mempunyai argumen dan menunjuk kembali ke bilangan 0. Fungsi *IsZero(x)* didefinisikan untuk memeriksa suatu bilangan x sebagai bilangan 0. Fungsi ini bernilai benar jika x adalah 0 dan salah jika tidak.

Fungsi *Add(x,y)* didefinisikan sebagai penjumlahan bilangan asli x dengan y. Jika hasil penjumlahannya lebih kecil atau sama dengan **maxint** maka *Add*=x+y dan jika tidak maka *Add* bernilai **maxint**. Fungsi *Equal(x,y)* didefinisikan untuk

menguji kesamaan dua bilangan asli x dan y. Fungsi ini bernilai benar jika x = y dan salah jika tidak.

Fungsi *Successor(x)* menunjuk ke bilangan asli berikutnya dalam suatu barisan. Jika tidak ada bilangan lain lagi dalam barisan, yaitu jika nilai x adalah **maxint** maka *Successor* didefinisikan sebagai **maxint**.

Fungsi *Substract(x,y)* didefinisikan sebagai pengurangan bilangan asli x oleh y. Jika x lebih kecil dari y maka *Substract* bernilai 0, jika tidak maka *Substract* = x - y.

Implementasi TDA adalah sebuah penerjemahan dari deklarasi yang mendefinisikan struktur data yang bertipe TDA dan prosedur untuk setiap operasi terhadap operasi TDA ke dalam suatu bahasa pemrograman.

2.5. Pointer

Pada saat sebuah program dikompilasi, kompiler akan melihat pada bagian deklarasi variabel untuk mengetahui nama-nama variabel apa saja yang akan digunakan, sekaligus mengalokasikan tempat dalam memori untuk menyimpan nilai data tersebut.

Variabel statis adalah suatu variabel yang tidak dapat diubah selama program dieksekusi. Sedangkan variabel dinamis adalah suatu variabel yang dapat diciptakan atau dihapus selama program dieksekusi.

Variabel dinamis ini diciptakan menggunakan pointer. Pointer merupakan penunjuk yang menunjuk alamat memori dari variabel dinamis dan nilai data yang ditunjuknya disebut sebagai simpul (*node*).

2.5.1. Pendeklarasian Pointer

Suatu tipe pointer dideklarasikan dengan diawali tanda ^ (*caret*) dan diikuti tipe pengenal. Contoh pendeklarasian variabel yang bertipe pointer tersaji berikut ini :

```
Type Pointer    = ^Suku;
     Suku       = Record
                    Koefisien : Real;
                    Pangkat   : Integer;
                    Berikut   : Pointer
                  End;
Var Poly, Poly1 : Pointer;
```

Pada pendeklarasian ini, *Poly* dan *Poly1* merupakan variabel pointer yang menunjuk ke suatu record bertipe *Suku.*

2.5.2. Menciptakan Variabel Dinamis

Dalam Pascal, tersedia prosedur bernama *New* yang digunakan untuk menciptakan variabel dinamis. Bentuk prosedurnya :

```
New (Variabel_Pointer);
```

Sebagai contoh :

```
New (Poly);
```

Pernyataan ini menyebabkan suatu alamat memori dari variabel dinamis bertipe *Suku* disimpan dalam variabel *Poly.*

Setelah variabel dinamis terbentuk, pengisian dapat dilakukan terhadap variabel dinamis tersebut. Sebagai contoh :

```
Poly^.Koefisien := 3;
Poly^.Pangkat   := 14;
Poly^.Berikut   := Nil;
```

Statemen tersebut berarti field *Koefisien* dari record yang ditunjuk oleh *Poly* diisi dengan nilai 3, field *Pangkat* diisi dengan nilai 14 dan field *Berikut* bernilai **nil** karena tidak menunjuk ke simpul yang lain (lihat Gambar 2.1).

Gambar 2.1. Contoh variabel dinamis

2.5.3. Operasi Pada Pointer

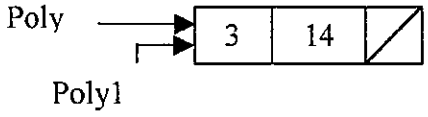
Secara umum ada dua operasi dasar yang dapat dilakukan menggunakan data yang bertipe pointer yaitu mengkopi pointer dan mengkopi isi simpul. Syarat yang harus dipenuhi untuk kedua operasi ini adalah bahwa pointer-pointer yang akan dioperasikan harus mempunyai deklarasi yang sama.

2.5.3.1. Mengkopi Pointer

Operasi ini dilakukan agar dua atau lebih pointer menunjuk ke simpul (lokasi memori) yang sama. Contoh statemen penugasan yang digunakan untuk mengkopi pointer adalah :

```
Poly1 := Poly;
```

yang berarti *Poly1* menunjuk ke simpul yang ditunjuk oleh pointer *Poly*. Dengan demikian baik *Poly* maupun *Poly1* menunjuk lokasi memori yang sama (lihat Gambar 2.2).



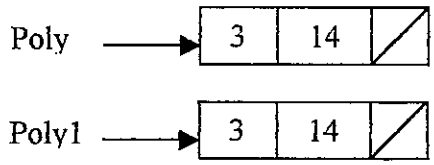
Gambar 2.2. Ilustrasi mengkopi pointer

2.5.3.2. Mengkopi Isi Simpul

Dengan mengkopi isi simpul mengakibatkan dua atau lebih simpul yang ditunjuk oleh pointer yang berbeda mempunyai isi yang sama. Contoh statemen penugasan yang digunakan untuk mengkopi isi simpul adalah :

```
Poly1^ := Poly^;
```

Dengan statemen tersebut isi simpul yang ditunjuk *Poly1* akan bernilai sama dengan isi simpul yang ditunjuk *Poly* (lihat Gambar 2.3).



Gambar 2.3. Ilustrasi mengkopi isi simpul

2.5.4. Operasi Relasi Pada Pointer

Karena hanya berhubungan dengan alamat tertentu dalam memori, maka untuk pointer hanya dapat dioperasikan dengan dua operator yaitu = atau <>. Sedangkan untuk simpul, semua operator relasi (<, <=, =, >=, >, <>) dapat digunakan. Sebagai contoh :

```
If Poly = Nil Then
While Poly <> Nil Do
If Poly^.Pangkat >= Poly1^.Pangkat Then
```

2.5.5. Menghapus Variabel Dinamis

Pada saat program dieksekusi variabel dinamis yang telah diciptakan dapat dihapus kembali. Setelah suatu variabel dinamis dihapus, maka lokasi yang semula ditempatinya akan bebas. Statemen untuk menghapusnya adalah :

```
Dispose (Variabel_Pointer);
```

2.6. Linked list

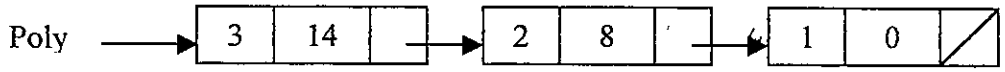
Linked list terdiri atas rangkaian elemen yang saling berhubungan atau berkaitan, dimana setiap elemen dihubungkan dengan elemen lainnya oleh sebuah pointer. Linked list semacam ini dinamakan singly linked list. Karena menggunakan pointer, maka masing-masing elemen linked list tersebut dinamakan dengan simpul (*node*).

Setiap simpul dalam linked list terbagi menjadi dua bagian. Bagian pertama, disebut field informasi, berisi informasi yang akan disimpan dan diolah. Bagian kedua, disebut field penyambung (*link field*). Field penyambung sebenarnya adalah suatu pointer yang menunjuk ke simpul berikutnya, sehingga nilai dari elemen ini adalah alamat suatu lokasi tertentu dalam memori dan digambarkan dengan anak panah. Field penyambung (pointer) dari suatu simpul yang tidak menunjuk simpul lain disebut pointer kosong, yang nilainya dinyatakan sebagai **nil**.

Pada sub bab 2.5.1. telah diberikan pendeklarasian suatu variabel dinamis. Dengan menggunakan variabel dinamis tersebut dapat dimungkinkan untuk membuat struktur data dinamis, yang dapat diperluas atau dipersempit selama eksekusi berlangsung tanpa harus memesan sejumlah memori terlebih dahulu dalam pedeklarasian.

Pada pendeklarasian tersebut, *Poly* merupakan variabel pointer yang menunjuk letak suatu record bertipe *Suku* yang terdiri dari tiga field yaitu *Koefisien*, *Pangkat* dan *Berikut*. Field *Berikut* sendiri berupa variabel pointer yang menunjuk record yang terdiri dari *Koefisien*, *Pangkat* dan *Berikut*. Variabel

Berikut befungsi sebagai pengait atau penghubung dengan simpul berikutnya. Berikut ini contoh linked list yang terbentuk dari pendeklarasian pada sub bab 2.5.1.



Gambar 2.4. Contoh linked list dengan 3 elemen

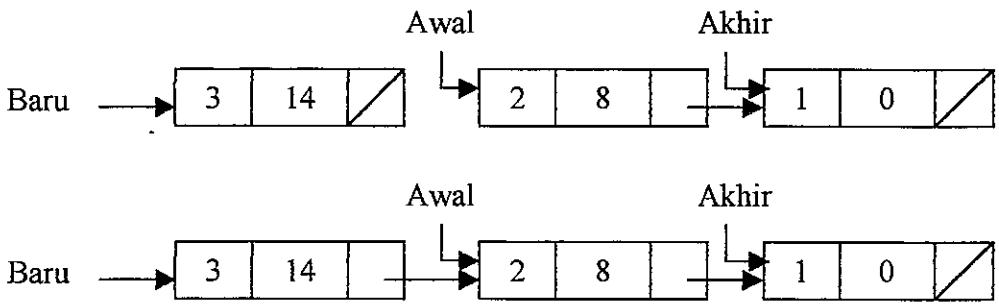
Ada sejumlah operasi yang dapat dilakukan pada sebuah linked list, yaitu menambah simpul dan menghapus simpul.

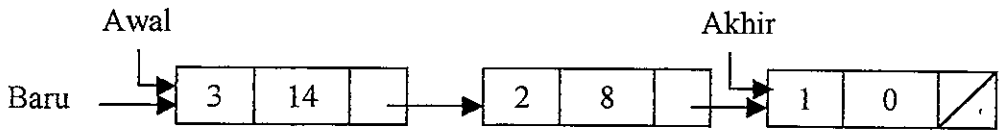
2.6.1. Menambah Simpul

Operasi menambah simpul dapat dipecah berdasarkan posisi simpul baru yang akan disisipkan, yaitu simpul baru selalu diletakkan sebagai simpul pertama, simpul baru menyisip diantara dua simpul yang sudah ada dan simpul baru selalu ditambahkan di belakang simpul terakhir. Pointer *Awal* adalah pointer yang menunjuk ke simpul pertama, pointer *Akhir* menunjuk ke simpul terakhir dan simpul yang ditunjuk oleh pointer *Baru* adalah simpul yang akan ditambahkan

2.6.1.1. Menambah di Depan

Operasi penambahan simpul linked list di depan adalah bahwa simpul baru akan selalu diletakkan di awal linked list. Ilustrasi penambahannya dapat dillihat pada Gambar 2.5.



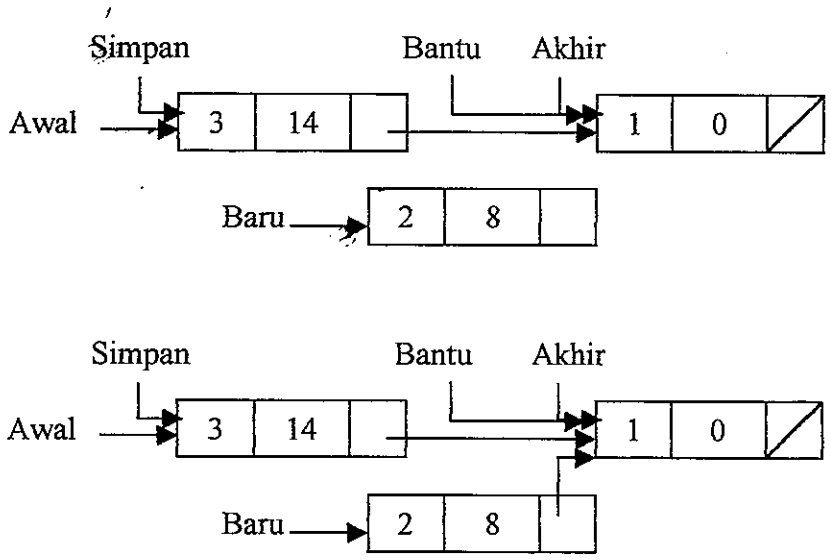


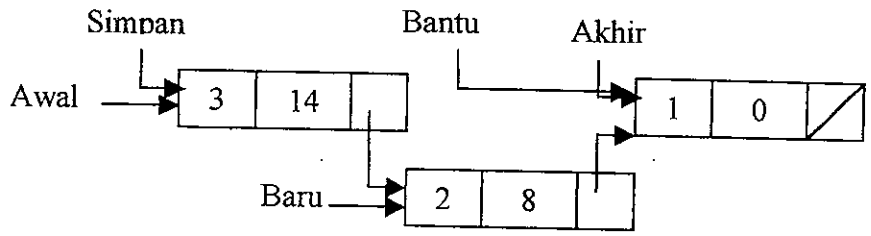
Gambar 2.5. Ilustrasi penambahan simpul di awal linked list

Secara garis besar operasi penambahannya dapat dijelaskan sebagai berikut. Pertama kali pointer pada simpul yang ditunjuk oleh pointer *Baru* dibuat sama dengan *Awal*. Kemudian *Awal* dibuat sama dengan *Baru*. Dengan cara seperti ini simpul *Baru* akan selalu diperlakukan sebagai simpul pertama dalam linked list.

2.6.1.2. Menambah di Tengah

Untuk menambah simpul di tengah linked list diperlukan bantuan dua buah pointer, misalnya *Bantu* dan *Simpan*. Dalam hal ini simpul baru akan diletakkan sebelum simpul yang ditunjuk oleh pointer *Bantu*. Ilustrasi penambahannya digambarkan seperti terlihat pada Gambar 2.6.



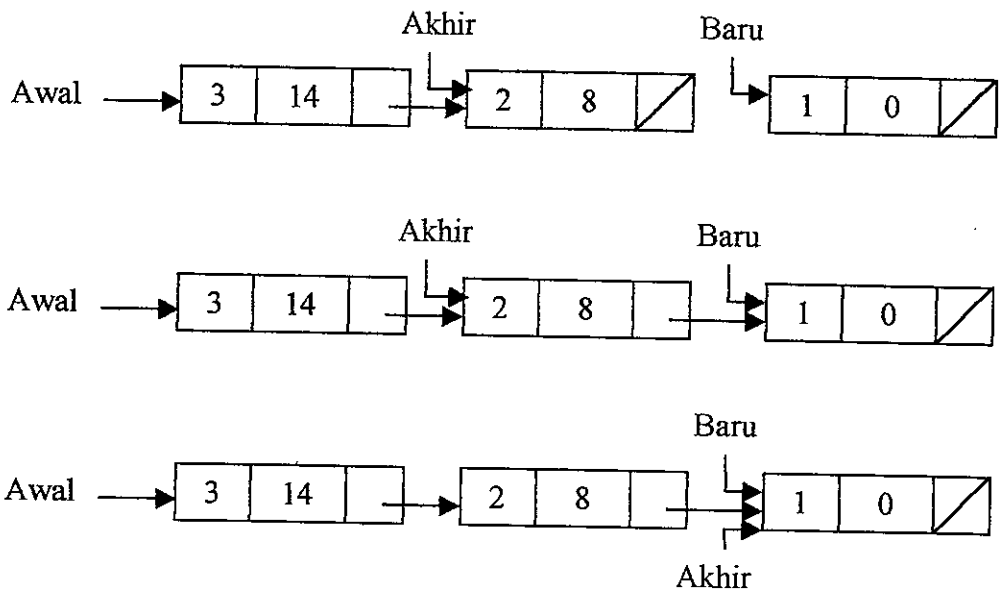


Gambar 2.6. Ilustrasi penambahan simpul di tengah linked list

Pertama kali ditentukan tempat dimana simpul *Baru* akan ditambahkan, yaitu dengan menempatkan pointer *Bantu* pada suatu tempat. Kemudian pointer pada simpul yang ditunjuk oleh *Baru* dibuat sama dengan *Bantu*. Selanjutnya pointer pada simpul yang ditunjuk oleh simpul *Simpan* dibuat sama dengan *Baru*.

2.6.1.3. Menambah di Belakang

Operasi penambahan simpul di akhir suatu linked list adalah bahwa simpul-simpul baru yang ditambahkan selalu akan menjadi simpul terakhir. Ilustrasi penambahannya dapat dilihat pada Gambar 2.7.



Gambar 2.7. Ilustrasi penambahan simpul di akhir linked list

Operasi penambahannya dapat dijelaskan sebagai berikut. Pointer pada simpul yang ditunjuk oleh simpul *Akhir* dibuat sama dengan *Baru*. Kemudian pointer *Akhir* dibuat sama dengan pointer *Baru*.

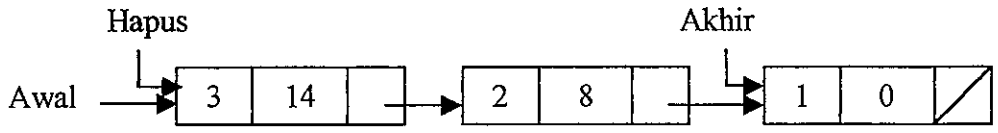
2.6.2. Menghapus simpul

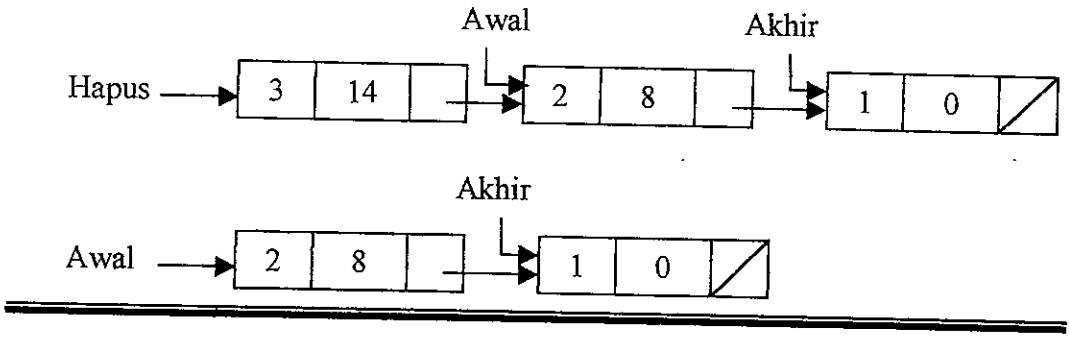
Dalam menghapus simpul ada satu hal yang perlu diperhatikan, yaitu bahwa simpul yang dapat dihapus adalah simpul yang berada sesudah simpul yang ditunjuk oleh suatu pointer (dalam gambar adalah simpul yang berada di sebelah kanan simpul yang ditunjuk oleh suatu pointer), kecuali simpul pertama.

Seperti halnya pada penambahan simpul, penghapusan simpul juga dapat dibagi menjadi menghapus simpul pertama, menghapus simpul yang ada di tengah dan menghapus simpul terakhir. Untuk menghapus simpul yang ada di tengah, diperlukan bantuan sebuah pointer yang menunjuk ke simpul sebelum (sebelah kiri) simpul yang akan dihapus. Misalkan simpul yang akan dihapus adalah simpul yanng ditunjuk oleh pointer *Hapus*.

2.6.2.1. Menghapus Simpul Pertama

Untuk menghapus simpul pertama, maka pointer *Hapus* dibuat sama dengan pointer *Awal*. Kemudian pointer *Awal* ditunjukkan ke simpul berikutnya setelah simpul yang ditunjuk oleh pointer *Awal*. Selanjutnya, simpul yang ditunjuk oleh pointer *Hapus* di **dispose**. Ilustrasi penghapusannya dapat dilihat pada Gambar 2.8.

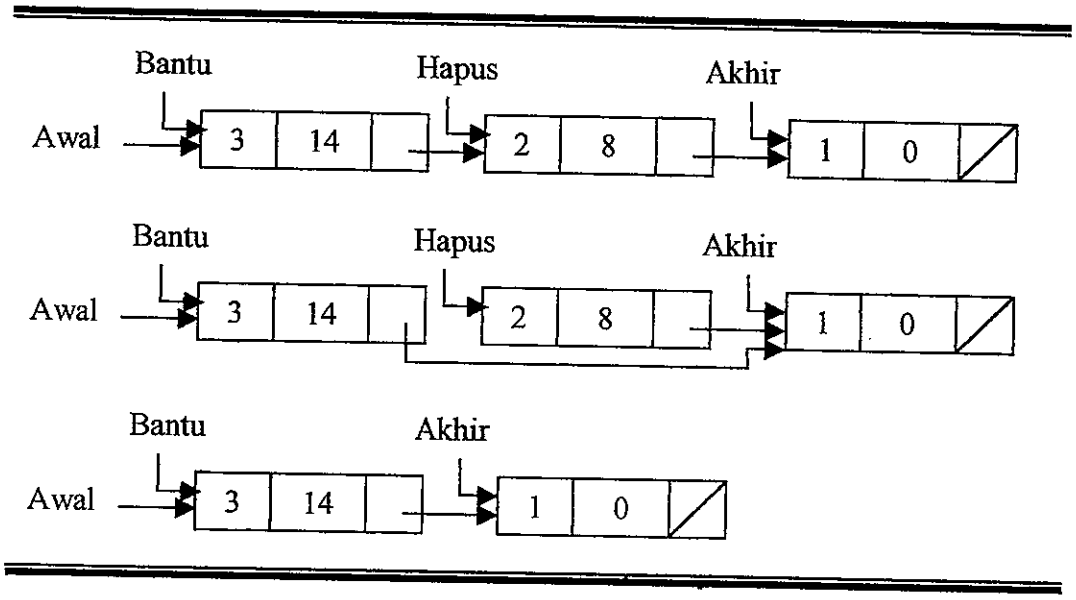




Gambar 2.8. Ilustrasi menghapus simpul pertama

2.6.2.2. Menghapus simpul di Tengah

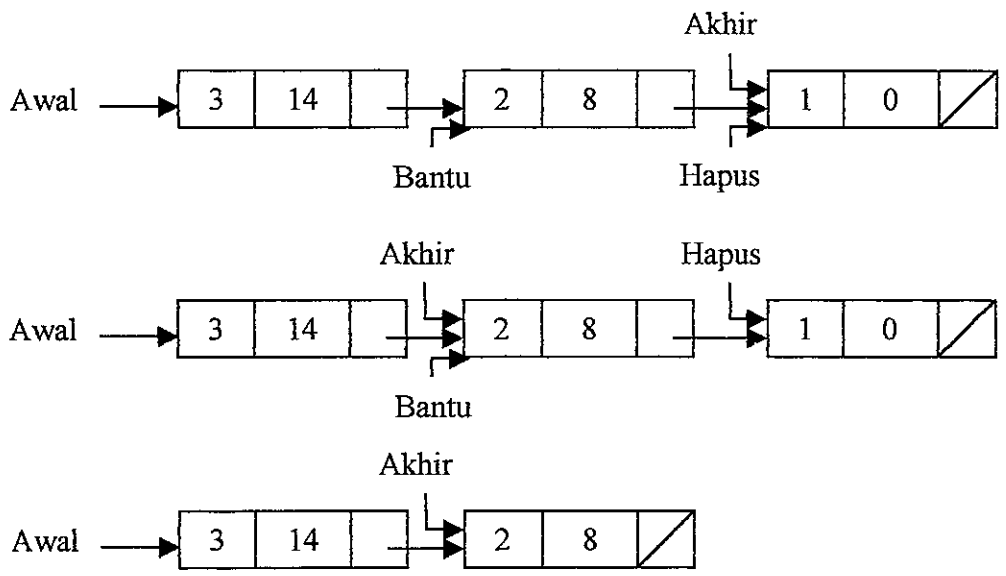
Untuk menghapus simpul yang ada di tengah linked list, pertama kali pointer *Bantu* diletakkan pada simpul yang berpointer menunjuk ke simpul yang akan dihapus. Kemudian, pointer pada simpul yang ditunjuk oleh *Bantu* ditunjukkan pada simpul yang ditunjuk oleh pointer pada simpul yang ditunjuk *Hapus*. Selanjutnya simpul yang ditunjuk oleh pointer *Hapus* di **dispose**. Ilustrasinya penghapusannya disajikan pada Gambar 2.9.



Gambar 2.9. Ilustrasi menghapus simpul di tengah

2.6.2.3. Menghapus Simpul di Akhir

Untuk menghapus simpul yang ada di akhir linked list, pertama kali pointer *Hapus* dibuat sama dengan *Akhir*, dan menempatkan pointer *Bantu* pada simpul yang berpointer menunjuk ke simpul yang akan dihapus. Kemudian pointer *Akhir* dibuat sama dengan *Bantu*. Pointer pada simpul yang ditunjuk pointer *Akhir* diberi nilai **nil** dan selanjutnya simpul yang ditunjuk oleh pointer *Hapus* di **dispose**. Ilustrasi penghapusannya disajikan pada Gambar 2.10.



Gambar 2.10. Ilustrasi menghapus simpul di akhir

2.7. Polinomial

Suatu polinomial A dalam x dengan koefisiennya elemen bilangan riil didefinisikan sebagai : $A(x) = a_m x^m + a_{m-1}x^{m-1} + ... + a_1 x + a_0$ dimana m adalah bilangan bulat nonnegatif dan setiap a_i elemen bilangan riil. Dalam ekspresi di atas $a_i x^i$ adalah suku-suku dari polinomial A(x) dan a_i adalah koefisien dari x^i dalam suku $a_i x^i$.

Polinomial $A(x) = a_m x^m + a_{m-1}x^{m-1} + \ldots + a_1 x + a_0$ dan $B(x) = b_n x^n + b_{n-1}x^{n-1} + \ldots + b_1 x + b_0$ didefinsikan sama jika dan hanya jika $a_i = b_i$ untuk setiap i.

Polinomial $A(x) = a_m x^m + a_{m-1}x^{m-1} + \ldots + a_1 x + a_0$ dengan $a_m \neq 0$, maka m adalah derajat (*degree*) dari polinomial A(x). Polinomial A(x) = 0 disebut polinomial nol.

2.7.1. Penjumlahan polinomial

Penjumlahan polinomial $A(x) = a_m x^m + a_{m-1}x^{m-1} + \ldots + a_1 x + a_0$ dan polinomial $B(x) = b_n x^n + b_{n-1}x^{n-1} + \ldots + b_1 x + b_0$ dengan $m \geq n$ adalah :

$$A(x) + B(x) = a_m x^m + a_{m-1}x^{m-1} + \ldots + a_{n+1}x^{n+1} + (a_n+b_n)\, x^n + (a_{n-1}+b_{n-1})\, x^{n-1} + \ldots + (a_1+b_1)\, x + (a_0+b_0)$$

Contoh : $A(x) = 3x^{14}+2x^8+1$

$B(x) = 8x^{14}-3x^{10}+10x^6$

Hasil penjumlahannya adalah :

$A(x)+B(x) = (3+8)x^{14}-3x^{10}+2x^8+10x^6+1 = 11x^{14}-3x^{10}+2x^8+10x^6+1.$

2.7.2. Pengurangan Polinomial

Jika terdapat polinomial $A(x) = a_m x^m + a_{m-1}x^{m-1} + \ldots + a_1 x + a_0$ dan polinomial $B(x) = b_n x^n + b_{n-1}x^{n-1} + \ldots + b_1 x + b_0$ dengan $m \geq n$ maka pengurangan polinomial A(x) oleh polinomial B(x) didefinisikan sebagai :

$$A(x) - B(x) = a_m x^m + a_{m-1}x^{m-1} + \ldots + a_{n+1}x^{n+1} + (a_n-b_n)\, x^n + (a_{n-1}-b_{n-1})\, x^{n-1} + \ldots + (a_1-b_1)\, x + (a_0-b_0)$$

Contoh : $A(x) = 3x^{14}+2x^8+1$

$B(x) = 8x^{14}-3x^{10}+10x^6$

Hasil pengurangan polinomial A(x) oleh polinomial B(x) adalah

$A(x)-B(x) = (3-8)x^{14}+(0-(-3))x^{10}+2x^{8}+(0-10)x^{6}+1 = -5x^{14}+3x^{10}+2x^{8}-10x^{6}+1.$

2.7.3. Perkalian Polinomial

Perkalian polnomial $A(x) = a_m x^m + a_{m-1}x^{m-1} + \ldots + a_1 x + a_0$ dengan polinomial $B(x) = b_n x^n + b_{n-1}x^{n-1} + \ldots + b_1 x + b_0$ didefinisikan sebagai :

$$A(x)\,B(x) = (a_m x^m + a_{m-1}x^{m-1} + \ldots + a_1 x + a_0)(b_n x^n + b_{n-1}x^{n-1} + \ldots + b_1 x + b_0)$$

$$= a_m b_n x^{m+n} + a_m b_{n-1} x^{m+n-1} + \ldots + a_m b_0 x^m + a_{m-1} b_n x^{m-1+n} + \ldots$$

$$+ a_{m-1} b_0 x^{m-1} + .. + a_0 b_n x^n + \ldots + a_0 b_0$$

$$= a_m b_n x^{m+n} + (a_m b_{n-1} + a_{m-1} b_n) x^{m+n-1} + (a_m b_{n-2} + a_{m-1} b_{n-1} + a_{m-2} b_n) x^{m+n-2} + \ldots$$

$$+ (a_1 b_0 + a_0 b_1)x + (a_0 b_0)$$

Contoh : $A(x) = 3x^{14}+2x^{8}+1$

$B(x) = 8x^{14}-3x^{10}+10x^{6}$

Hasil perkaliannya adalah :

$$A(x)B(x) = (3x^{14}+2x^{8}+1)(8x^{14}-3x^{10}+10x^{6})$$

$$= (3x^{14}+2x^{8}+1)8x^{14} + (3x^{14}+2x^{8}+1)(-3x^{10}) + (3x^{14}+2x^{8}+1)10x^{6}$$

$$= (24x^{28}+16x^{22}+8x^{14}) + (-9x^{24}-6x^{18}-3x^{10}) + (30x^{20}+20x^{14}+10x^{6})$$

$$= 24x^{28}-9x^{24}+16x^{22}+30x^{20}-6x^{18}+28x^{14}-3x^{10}+10x^{6}$$

2.7.4. Pembagian Polinomial

Membagi polinomial A(x) oleh B(x) dengan B(x) bukan polinomial nol, adalah mencari polinomial C(x) sedemikian sehingga A(x)/B(x) = C(x) yang berarti A(x)=B(x)*C(x) dengan B(x) dan C(x) merupakan faktor dari A(x), dan

$$deg(A(x)) = deg(B(x)) + deg(C(x)) \geq deg(B(x))$$

dimana *deg*(A(x)) adalah derajat dari polinomial A(x).

Dalam situasi yang umum dimana B(x) bukan merupakan faktor dari A(x) dapat diekspresikan dengan :

$$A(x) = B(x) * C(x) + D(x) \text{ dan } deg(D(x)) < deg(B(x)).$$

Contoh : $A(x) = 3x^{14}+2x^{8}+1$

$B(x) = 8x^{14}-3x^{10}+10x^{6}$

Hasil pembagiannya adalah :

$3x^{14}+2x^{8}+1 = (8x^{14}-3x^{10}+10x^{6})*0,375 + (1,125x^{10}+2x^{8}-3,75x^{6}+1)$

didapat bahwa :

$C(x)= 0,375$ dan

$D(x)= 1,125x^{10}+2x^{8}-3,75x^{6}+1$ dimana $deg(D(x)) < deg(B(x))$