SATUAN ACARA PERKULIAHAN
(SAP)

Mata Kuliah : Struktur Data
Kode : TIS3213
Semester : III
Waktu : 2 x 3 x 50 Menit
Pertemuan : 6 & 7

A. Kompetensi

1. Utama

   Mahasiswa dapat memahami tentang konsep pemrograman menggunakan struktur senarai berantai.

2. Pendukung

   Mahasiswa dapat mengetahui operasi-operasi pada senarai berantai.

B. Pokok Bahasan

Senarai Berantai

C. Sub Pokok Bahasan

- Pengertian Senarai Berantai
- Operasi Pada Senarai Berantai
- Senarai Berantai Sebagai Tumpukan

D. Kegiatan Belajar Mengajar

Tahapan Kegiatan	Kegiatan Pengajaran	Kegiatan Mahasiswa	Media & Alat Peraga
Pendahuluan	1. Menjelaskan perkuliahan yang akan dijalani dalam satu semester 2. Menjelaskan materi-materi perkuliahan dan buku-buku acuan yang akan dipergunakan dalam semester ini	Mendengarkan dan memberikan komentar	Notebook, LCD, Papan Tulis
Penyajian	1. Menjelaskan tentang pengertian senarai berantai 2. Menjelaskan tentang operasi-operasi pada senarai berantai 3. Menjelaskan tentang senarai	Memperhatikan, mencatat, dan memberikan komentar. Mengajukan	Notebook, LCD, Papan Tulis

	berantai sebagai tumpukan	pertanyaan.	
Penutup	1. Mengajukan pertanyaan kepada mahasiswa. 2. Memberikan kesimpulan. 3. Mengingatkan akan kewajiban untuk pertemuan selanjutnya.	Memberikan komentar. Mengajukan dan menjawab pertanyaan	Notebook, LCD, Papan Tulis

E. Evaluasi

Evaluasi dilakukan dengan cara memberikan pertanyaan langsung dan tidak langsung kepada mahasiswa.

F. Daftar Referensi

1. P. Insap Santosa, *Struktur Data Menggunakan Turbo Pascal 6.0, Andi Offset, Yogyakarta, 2001*
2. Wirth Niklaus, "Algorithms and Data Structure", Prentice Hall Int. Inc, 1986
3. Antonie Pranata, *Algoritma dan Pemrograman*, J&J Learning Yogyakarta, 2000
4. Dwi Sanjaya, *Bertualang dengan Struktur Data di Planet Pascal*, J&J Learning Yogyakarta, 2001
5. Materi – Materi dari Internet.

RENCANA KEGIATAN BELAJAR MINGGUAN
(RKBM)

Mata Kuliah : Struktur Data
Kode : TIS3213
Semester : III
Waktu : 2 x 3 x 50 Menit
Pertemuan : 6 & 7

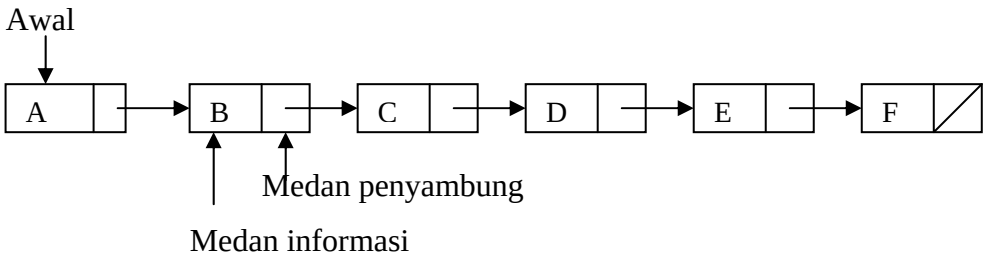
Minggu Ke-	Topik (Pokok Bahasan)	Metode Pembelajaran	Estimasi Waktu (Menit)	Media
1	2	3	4	5
6	4.1 Pengertian Senarai Berantai 4.2 Operasi Pada Senarai Berantai 4.2.1 Menambah Simpul 4.2.2 Menghapus Simpul	Ceramah, Diskusi Kelas	1 x 3 x 50’	Notebook, LCD, Papan Tulis
7	4.3 Senarai Berantai Sebagai Tumpukan	Ceramah, Diskusi Kelas	1 x 3 x 50’	Notebook, LCD, Papan Tulis

BAB IV
SENARAI BERANTAI

4.1 PENGERTIAN SENARAI BERANTAI

Dalam pemakaian sehari-hari istilah senarai (*list*) adalah kumpulan linear sejumlah data. Sedangkan senarai berantai adalah kumpulan komponen yang disusun secara berurutan dengan bantuan pointer. Masing-masing komponen dinamakan dengan simpul (*node*). Dimana 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.

Berikut contoh senarai berantai :



Gambar 4.1 Senarai berantai dengan 6 simpul

4.2 OPERASI PADA SENARAI BERANTAI

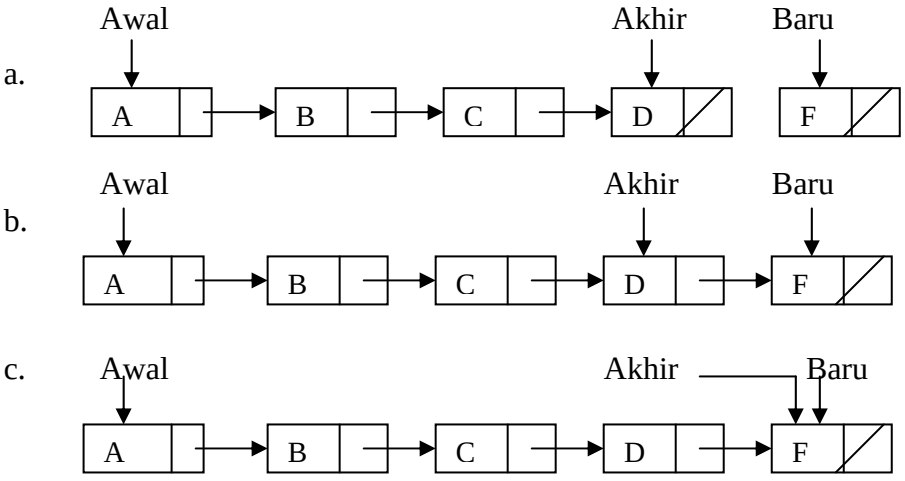
Ada sejumlah operasi yang bisa dilakukan pada sebuah senarai berantai, yaitu operasi membaca isi senarai (*traversal*), menambah simpul, menghapus simpul dan mencari informasi pada suatu senarai berantai.

4.2.1 Menambah Simpul

4.2.1.1 Menambah di Belakang

Dalam hal ini simpul-simpul baru yang ditambahkan selalu akan menjadi simpul terakhir. Ilustrasi penambahannya digambarkan pada Gambar 4.2. Pointer Awal adalah pointer yang menunjuk ke simpul pertama, pointer Akhir menunjuk ke simpul terakhir dan pointer Baru adalah simpul yang akan ditambahkan (Gambar 4.2.a).

Untuk menyambung simpul yang ditunjuk oleh Akhir dan Baru, pointer pada simpul yang ditunjuk oleh simpul Akhir dibuat sama dengan Baru (Gambar 4.2.b), kemudian pointer Akhir dibuat sama dengan pointer Baru (Gambar 4.2.c)



Gambar 4.2 Ilustrasi penambahan simpul di belakang

Prosedur untuk menambah simpul dibelakang senarai berantai tersaji pada Program 4.1 berikut ini:

```
procedure TAMBAH_BELAKANG (var Awal, Akhir : simpul; Elemen : char);
var Baru : simpul;
begin
   new (Baru); Baru^.info := Elemen;
   if Awal = nil then
         Awal := Baru
   else
         Akhir^.Berikut := Baru;        {* Gambar 4.2.b *}
   Akhir := Baru;                       {* Gambar 4.2.c *}
   Akhir^.Berikut := nil
end;
```

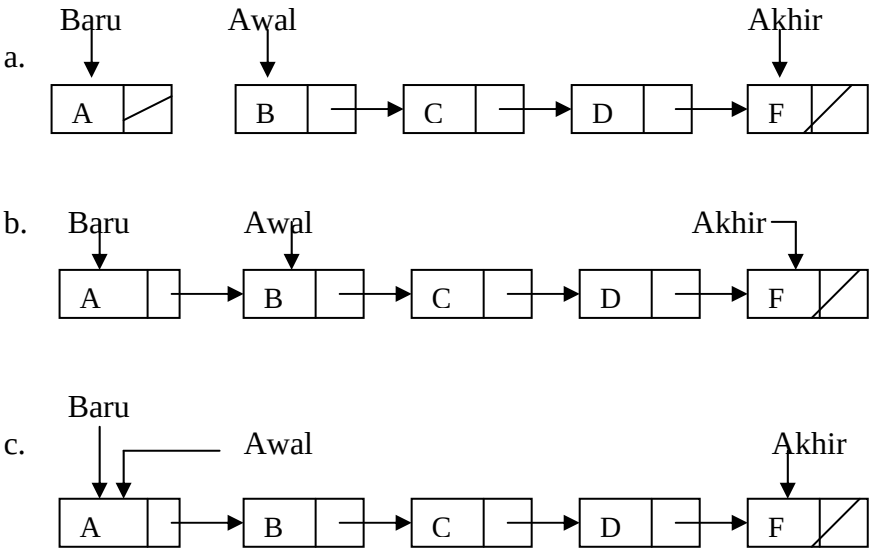
Program 4.1 Prosedur menambah simpul di belakang senarai berantai

Prosedur diatas bisa dipanggil dengan pemanggil prosedur :

TAMBAH_BELAKANG (Awal, Akhir, Elemen);

4.2.1.2 Menambah di Depan

Secara garis besar operasi penambahannya bisa dijelaskan sebagai berukut. Pertama kali pointer pada simpul yang ditunjuk oleh pointer Baru dibuat sama dengan Awal (Gambar 4.3.b). Kemudian Awal dibuat sama dengann Baru (Gambar 4.3.c). Dengan demikian simpul baru akan selalu diperlakukan sebagai simpul pertama dalam senarai berantai.



Gambar 4.3 Ilustrasi penambahan simpul di awal senarai berantai

Prosedur untuk menambah simpul didepan senarai berantai tersaji pada Program 4.2 :

```
procedure TAMBAH_DEPAN (var Awal, Akhir : simpul; Elemen : char);
var Baru : simpul;
begin
   new (Baru); Baru^.info := Elemen;
   if Awal = nil then
         Akhir := Baru
   else
         Baru^.Berikut := Awal;          {* Gambar 4.3.b *}
   Awal := Baru;                         {* Gambar 4.3.c *}
end;
```

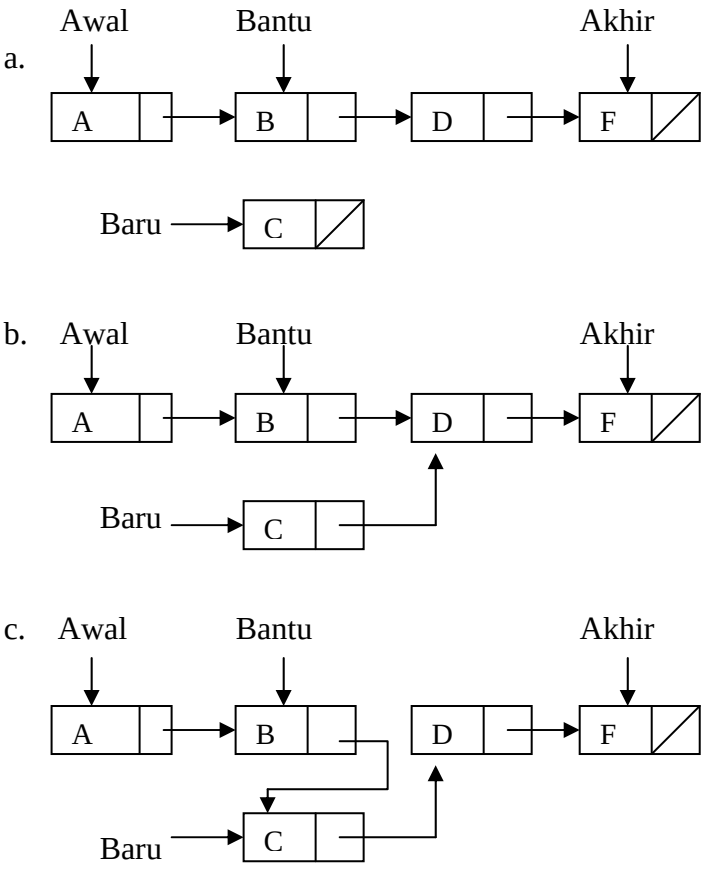
Program 4.2 Prosedur menambah simpul di depan senarai berantai

Prosedur diatas bisa dipanggil dengan pemanggil prosedur :

TAMBAH_DEPAN (Awal, Akhir, Elemen);

4.2.1.3 Menambah di Tengah

Untuk menambah simpul ditengah senarai berantai, diperlukan bantuan sebuah pointer lain, misalnya Bantu. Dalam hal ini simpul Baru akan diletakkan setelah simpul yang ditunjuk oleh pointer Bantu.



Gambar 4.4 Ilustrasi penambahan simpul di tengah senarai berantai

Pertama kali tentukan dimana simpul baru akan ditambahkan, yaitu dengan menempatkan pointer Bantu pada suatu tempat. Kemudian pointer pada simpul yang ditunjuk oleh Baru dibuat sama dengan pointer pada simpul yang ditunjuk oleh Bantu (Gambar 4.4.b). Selanjutnya pointer pada simpul yang ditunjuk oleh simpul Bantu dibuat sama dengan Baru (Gambar 4.4.c). Prosedur selengkapnya tersaji pada Program 4.3 berikut :

```
procedure TAMBAH_TENGAH (var Awal, Akhir : simpul; Elemen : char);
var Baru, Bantu : simpul;
begin
   new (Baru); Baru^.info := Elemen;
   if Awal = nil then
      begin
         Awal := Baru;
         Akhir := Baru
      end
   else
      begin
         {* Mencari lokasi yang sesuai *}
         Bantu := Awal;
         while Elemen > Baru^.Berikut^.Info do
               Bantu := Bantu^.Berikut;

         {* Menyisipkan simpul baru *}
         Baru^.Berikut := Bantu^.Berikut;      {* Gambar 4.4.b *}
         Bantu^.Berikut := Baru;               {* Gambar 4.4.c *}
      end;
end;
```

Program 4.3 Prosedur menambah simpul di tengah senarai berantai

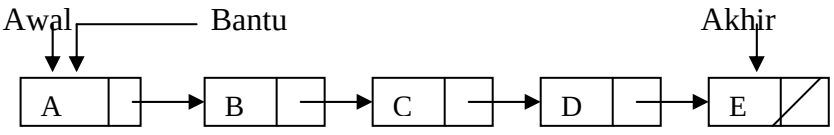
4.2.2 Menghapus Simpul

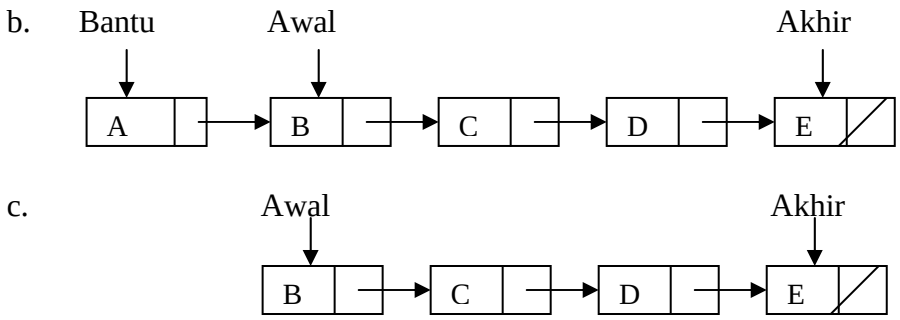
Dalam menghapus simpul ada satu hal yang harus diperhatikan, yaitu bahwa simpul yamg bisa dihapus adalah simpul yang berada sesudah simpul yang ditunjuk oleh suatu pointer, kecuali untuk simpul pertama.

4.2.2.1 Menghapus Simpul Pertama

Untuk menghapus simpul pertama, maka pointer Bantu dibuat sama dengan pointer Awal (Gambar 4.5.a). Kemudian pointer Awal dipindahkan ke simpul yang ditunjuk oleh pointer pada simpul yang ditunjuk oleh pointer Bantu (Gambar 4.5.b). Selanjutnya simpul yang ditunjuk oleh pointer Bantu di **dispose** (Gambar 4.5.c).

a.

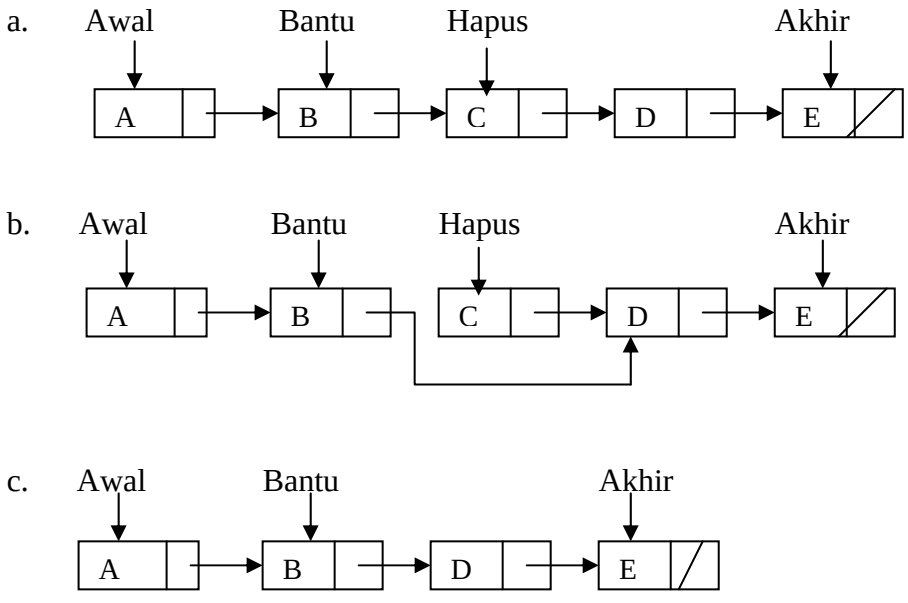




Gambar 4.5 Ilustrasi menghapus simpul pertama

4.2.2.2 Menghapus Simpul di Tengah atau Terakhir

Untuk menghapus simpul yang ada ditengah senarai berantai, pertama kali letakkan pointer Bantu pada simpul di sebelah kiri simpul yang akan dihapus. Simpul yang akan dihapus ditunjuk dengan pointer lain, misalnya Hapus (Gambar 4.6.a). Kemudian pointer pada simpul yang ditunjuk oleh Bantu ditunjukkan pada simpul yang ditunjuk oleh pointer pada simpul yang akan dihapus (Gambar 4.6.b). Selanjutnya simpul yang ditunjuk pointer Hapus di **dispose** (Gambar 4.6.c).



Gambar 4.6 Ilustrasi menghapus simpul yang ada ditengah senarai berantai atau simpul terakhir

Prosedur selengkapnya tersaji pada Program 4.4. Perubah Elemen berisi data yang akan dihapus. Prosedur ini dipanggil dengan statemen :

HAPUS_SIMPUL (Awal, Akhir, Elemen);

```
procedure HAPUS_SIMPUL (var Awal, Akhir : simpul; Elemen : char);
var Bantu, Hapus : simpul;
begin
   if Awal = nil then
      writeln ('Senarai Masih Kosong');

   else if Awal^.Info = Elemen then          {* Simpul pertama dihapus *}
     begin
        Hapus := Awal;
        Awal := Hapus^.Berikut;
        dispose (Hapus)
      end

   else                              {* menghapus simpul tengah atau terakhir *}
      begin
        Bantu := Awal          {* Mencari simpul yang akan dihapus *}
        while (Elemen <> Bantu^.Info) and (Bantu^.Berikut <> nil) do
              Bantu := Bantu^.Berikut;
         Hapus := Bantu^.Berikut;
         if Hapus <> nil then          {* Simpul yang akan dihapus ketemu *}
           begin
              if Hapus <> Akhir then        {* Simpul tengah dihapus *}
                 Bantu^.Berikut := Hapus^.Berikut
              else                          {* Simpul terakhir dihapus *}
                 begin
                    Akhir := Bantu;
                    Akhir^.Berikut := nil
                 end;
              dispose (Hapus)
           end
         else                    {* Simpul yang akan dihapus tidak ketemu *}
             writeln ('Simpul yang akan dihapus tidak ada')
       end
end;
```

Program 4.4 Prosedur untuk menghapus simpul

4.3 SENARAI BERANTAI SEBAGAI TUMPUKAN

Berikut akan disajikan contoh lain penggunaan tumpukan, yaitu untuk menghitung nilai akhir suatu ungkapan postfix. Algoritma untuk menghitung nilai ungkapan postfix tersaji dibawah ini. (Untuk meneyederhanakan penjelasan, operand yang akan dioperasikan hanya terdiri dari 1 digit).

Algoritma HITUNG_POSTFIX

Langkah 0 Baca ungkapan dalam notasi infix kemudian konversikan menjadi notasi postfix. Ungkapan postfix disimpan sebagai perubah Post (yang bertipe **string**).

Langkah 1 Untuk I = 1 sampai N (N adalah panjang ungkapan postfix), kerjakan langkah 2 berikut ini.

Langkah 2 Test nilai Post [I] :

Jika Post [I] adalah :

operator : pop dua elemen dari tumpukan, operasikan sesuai dengan operatornya dan hasilnya dipush kembali kedalam tumpukan.

operand : push kedalam tumpukan.

Langkah3 Setelah langkah 2 selesai, dalam tumpukan hanya tinggal satu elemen, yaitu hasil akhir dari ungkapan postfix yang akan dihitung. Pop elemen ini dan cetak hasilnya.

Contoh : Ungkapan yang akan dihitung (dalam notasi infix)

(2 + 3 * (6 – 2)) / ((1 + 3) / 2)

Notasi postfix dari ungkapan diatas adalah :

2 3 6 2 - * + 1 3 + 2 / /

Tabel berikut ini menunjukkan proses pelacakan penghitungan ungkapan dengan menggunakan notasi postfix sesuai algoritma diatas.

Isi Tumpukan sebelumnya	Karakter dibaca	Jenis	Isi Tumpukan sesudahnya
(kosong)	2	Operand	2
2	3	Operand	3 2
3 2	6	Operand	6 3 2
6 3 2	2	Operand	2 6 3 2
2 6 3 2	-	Operator	4 3 2
4 3 2	*	Operator	12 2
12 2	+	Operator	14
14	1	Operand	1 14
1 14	3	Operand	3 1 14
3 1 14	+	Operator	4 14
4 14	2	Operand	2 4 14
2 4 14	/	Operator	2 14
2 14	/	Operator	7
7			

Setelah semua karakter dalam notasi postfix selesai dibaca, isi tumpukan tinggal sebuah elemen, dan inilah hasil akhir yang kita peroleh (dalam contoh diatas = 7).

--ooOOOoo--

Soal & Pembahasan :

Soal :

1. Apakah yang dimaksud dengan senarai berantai (*linked list*).
2. Sebuah simpul pada senarai berantai terdiri atas 2 bagian, sebutkan.

Pembahasan :

1. Senarai berantai adalah kumpulan komponen yang disusun secara berurutan dengan bantuan pointer
2. Simpul terdiri atas :
    - Bagian pertama disebut **medan informasi,** berisi informasi yang akan disimpan dan diolah.
    - Bagian kedua disebut **medan penyambung** (*link field*), berisi alamat simpul berikutnya.