Pertemuan 6

Waktu : 135 menit

Tujuan Pembelajaran : Mahasiswa mampu menjelaskan teknik pemrograman menggunakan Double Linked List.

Substansi Materi : Doubled Linked List, Circullar Double Linked List

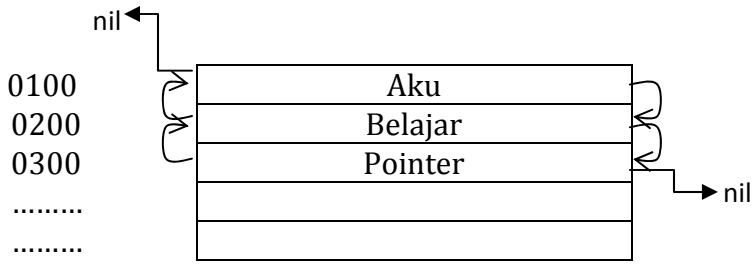
Tabulasi Kegiatan Perkuliahan

No	Tahap Kegiatan	Kegiatan Pengajar	Kegiatan Mahasiswa	Media & Alat	Waktu
1	Pendahuluan	1. Membuka pertemuan 2. Mengulang materi pertemuan sebelumnya	Menyimak Bertanya	Papan Tulis	20 Menit
2	Penyajian Materi	1. Teori Double Linked List 2. Operasi-operasi pada Double Linked List 3. Teori Circullar Double Linked List 4. Operasi-operasi pada Circullar Double Linked List	Menyimak Bertanya Menjawab Pertanyaan	Papan Tulis	80 Menit
3	Penutup	1. Menyimpulkan materi pertemuan 2. Memberikan tugas kecil 3. Menutup pertemuan	Menyimak	Papan tulis	35 Menit

MATERI KULIAH

Double Linked List

Salah satu kelemahan dari single linked list adalah pointer (penunjuk) hanya dapat bergerak satu arah saja, maju atau mundur, kanan atau kiri. Sehingga pencarian data pada single linked list hanya dapat bergerak dalam satu arah saja. Untuk mengatasinya maka digunakan metode double linked list. Linked list seperti ini dikenal dengan nama linked list berpointer ganda atau Double Linked List.



Gambar 1. Ilustrasi Double Linked List

Operasi-operasi pada Double Linked List

Insert

- Insert After

  Procedure insert berguna untuk menambah simpul dibelakang (sebelah kanan) pada sebuah double linked list. Berikut penggalan procedure insert after.

```
Procedure InsertAfter(e:Elemen_Type);
Var Now : Point;
  Begin
        New(now);
        Now^.Isi := e;
        If Head=Nil then
          Begin
                Head := Now;
                Tail := Now;
                Now^.Next := Nil;
                Now^.Prev := Nil;
          End
        Else
          Begin
                Tail^.next := now;
                Now^.Prev := Tail;
                Tail := Now;
                Tail^.Next := Nil;
          End;
  End;
```

- Insert Before

  Sesuai dengan namanya, procedure Insert Before berguna untuk menambah simpul di depan (sebelah kiri). Procedure ini tidak berbeda jauh dengan procedure Insert After.

```
Procedure InsertBefore(e:Elemen_Type);
Var Now : Point;
  Begin
        New(now);
        Now^.Isi := e;
        If Head=Nil then
          Begin
                Head := Now;
                Now^.Next := Nil;
                Now^.Prev := Nil;
          End
        Else
          Begin
                Head^.prev := now;
                Now^.next := head;
                Head := Now;
                Head^.Prev := Nil;
          End;
  End;
```

Delete

Delete After

Procedure Delete After berguna untuk menghapus simpul dari belakang. Procedure ini merupakan kebalikan dari procedure Insert After yang menambahkan simpul dibelakang.

```
Procedure DeleteAfter;
Var Now : Point;
  Begin
        Now := Tail;
        If Now <> Head then
          Begin
              Tail := Now^.Prev;
              Tail^.Next := Nil;
          End
        Else
          Begin
              Tail := Nil;
              Head := Nil;
          End;
        If Now <> Nil then
          Dispose(now);
  End;
```

- **Delete Before**

  Procedure Delete Before merupakan kebalikan dari procedure Delete After yang akan menghapus simpul dari depan (sebelah kiri).

```
Procedure DeleteBefore;
Var Now : Point;
  Begin
        Now := Head;
        If Now <> Head then
          Begin
                Head := Now^.Next;
                Head^.Prev := Nil;
          End
        Else
          Begin
                Tail := Nil;
                Head := Nil;
          End;
        If Now <> Nil then
          Dispose(now);
  End;
```

- **Delete at Position**

  Procedure at Position, sesuai dengan namanya, berguna untuk menghapus simpul pada posisi yang diinginkan. Untuk melakukannya diperlukan bantuan 2 variabel pointer yang pada modul ini diberi nama Bantu1 dan Bantu2. Nama tersebut boleh diganti.

```
Procedure DeleteAtPos;
Var Bantu1, Bantu2 : Point;
  Begin
        Bantu1 := Now^.Prev;
        Bantu2 := Now^.Next;

        If Bantu1 <> Nil then
          Bantu1^.Next := Bantu2;
        Else
          Head := Bantu2;
        If Bantu2 <> Nil Then
          Bantu2^.Prev := Bantu1;
        Else
          Tail := Bantu1;
        If Now <> Nil Then
          Dispose(Now);
  End;
```

Circullar Double Linked List

Merupakan double linked list yang simpul terakhirnya menunjuk ke simpul awal dan simpul awalnya menunjuk ke simpul akhir sehingga membentuk suatu lingkaran.

Operasi-operasi pada Circullar Double Linked List

- **Insert**
  - **Insert After**

    Procedure Insert After berguna untuk menambah simpul di belakang (sebelah kanan) pada sebuah double linked list.

```
Procedure InsertAfter(e:Elemen_Type);
Var Now : Point;
  Begin
        New(now);
        If Head=Nil then
          Begin
                Head := Now;
                else
                Now^.Prev := Tail;
                Tail^.Next := Now;
          End;
                Now^.Isi := e;
                Tail := Now;
                Tail^.next := Head;
                Head^.Prev := Tail;
End;
```

- **Insert Before**

  Procedure Insert Before berguna untuk menambahkan simpul di depan (sebelah kiri). Procedure ini tidak berbeda jauh dengan procedure Insert After yang telah dijelaskan sebelumnya.

```
Procedure InsertBefore(e:Elemen_Type);
  Begin
        If Head=Nil then
          Begin
                Head := Now;
                Tail := Now;
                End else
                Begin
                Now^.Prev :=Head;
                Head^.Next := Now;
          End;
                Now^.Isi := e;
                Now^.Next := Tail;
                Tail^.Prev := Now;
                Tail := Now;
End;
```

Delete

Delete After

Procedure Delete After berguna untuk menghapus simpul dari belakang. Procedure ini merupakan kebalikan dari Procedure Insert After yang menambah simpul di belakang.

```
Procedure DeleteAfter;
Var Now : Point;
  Begin
        Now := Tail;
        If Head=Nil then
          Begin
              Tail := Nil;
              Head := Nil;
              End else
              Begin
              Tail := Now^.Prev;
              Tail^.Next := Head;
              Head^.Prev := Tail;
          End;
  If Now <> nil then Dispose(Now);
End;
```

- **Delete Before**

  Procedure Delete Before merupakan kebalikan dari procedure Delete After yang akan menghapus simpul dari belakang, sedangkan Delete Before akan menghapus simpul dari depan (sebelah kiri).

```
Procedure DeleteBefore;
Var Now : Point;
  Begin
        Now := Tail;
        If Head=Tail then
          Begin
              Tail := Nil;
              Head := Nil;
           End else
           Begin
              Head := Now^.Next;
              Head^.Prev := Tail;
              Tail^.Next := Head;
          End;
  If Now <> nil then Dispose(Now);
End;
```

- **Delete at Position**

  Procedure Delete at Position berguna untuk menghapus simpul pada posisi yang diinginkan. Untuk itu diberikan bantuan 2 buah variabel pointer yang diberi nama Bantu1 dan Bantu2.

```
Procedure DeleteAtPos;
Var Bantu1, Bantu2 : Point;
  Begin
        Bantu2 := Now^.Next;
        Bantu1 := Now^.Prev;

        If Bantu1 <> Now  then
        Begin
          Bantu1^.Next := Bantu2;
          Bantu2^.Prev := Bantu1;
          If Bantu2 = Tail then Head := Bantu1;
          If Bantu1 = Head then Tail := Bantu2;
        End else
        Begin
          Head := Nil;
          Tail := Nil;
          End;
        If Now <> Nil then
          Dispose(Now);
  End;
```

- **Update**

Procedure update berguna untuk mengganti isi suatu simpul dengan data yang lain. Procedure update ini memanfaatkan suatu procedure cari untuk mencari posisi simpul yang akan diganti isinya tersebut. Setelah ketemu, barulah diganti isinya.

```
Procedure Update(x,y : elemen_Type);
Begin
  Cari(x);
  Now^.isi := y;
End;
```