ARRAY DINAMIS

Sebelumnya telah dijelaskan mengenai variable bertipe array (array statis), suatu tipe data yang bersifat statis (urutan dan ukuran sudah pasti). Kelemahan dari array statis adalah penggunaan ruang memori yang sudah digunakan tidak dapat dihapus apabila nama variable array tersebut sudah tidak digunakan kembali dalam suatu program (penyebab kemubaziran).

Untuk pemecahannya maka digunakan struktur data dinamis dengan menggunakan variable dinamis. Variabel dinamis tidak dapat dideklarasikan secara eksplisit seperti halnya variable statis dan tidak dapat ditunjuk oleh identifier secara langsung, tetapi dapat ditunjuk secara khusus oleh variable dinamis yaitu **POINTER.**

Deklarasi secara umum untuk tipe data POINTER adalah sebagai berikut :

1.
```
Type
        Pengenal = ↑Simpul
        Simpul   = Type
```

2.
```
(Nama var) : (↑Type data)
```

Keterangan :

Pengenal : Nama pengenal yang menyatakan data berupa pointer

Simpul : Menyatakan nama simpul

Type : Tipe data dari simpul

↑ : Tanda yang menyatakan bahwa pengenal memiliki tipe data pointer

Contoh :

1. Kamus :

```
Type
        Point =  ↑Data
        Data  = Record
                < Nama_Mhs : String,
                  Jurusan  : String,
                  Semester : String >
        Endrecord

    DataMhs : Point
```

2. Kamus :

```
    Jumlah_data  :  ↑Integer
    Nama_Siswa :   ↑String
```

Penjelasan:

- DataMhs merupakan variabel bertipe pointer
- Ketika program dikompilasi, variable DataMhs akan menempati lokasi tertentu dalam memori dan variabel tersebut belum menunjuk kesuatu simpul.
- Pointer yang belum menunjuk kesuatu simpul, nilainya dinyatakan sebagai nil guna mengalokasikan simpul dalam memori.
- Statemen yang digunakan untuk membuat sebuah simpul adalah **Alloc (variabel)**

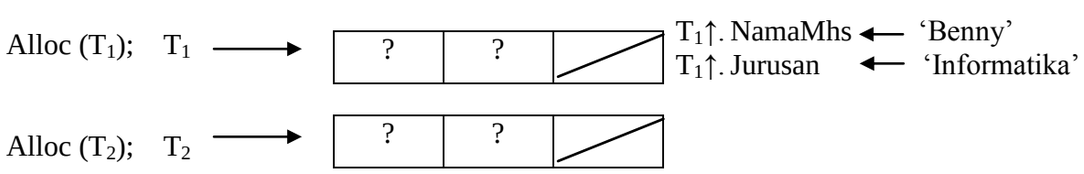
- Catatan : Variabel merupakan nama peubah yang bertipe pointer
- Misal : Alloc (DataMhs); {Berarti kita membuat sebuah simpul yang ditunjuk oleh DataMhs}

Keterangan : Masih tanda tanya karena belum terisi data

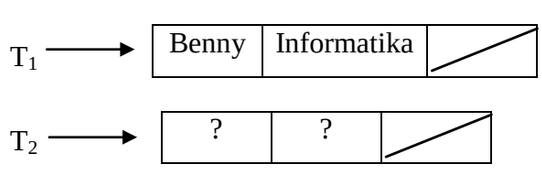
Operasi pada Pointer

1. Mengcopy pointer, artinya bahwa sebuah simpul akan ditunjuk oleh lebih dari sebuah pointer. Contoh :

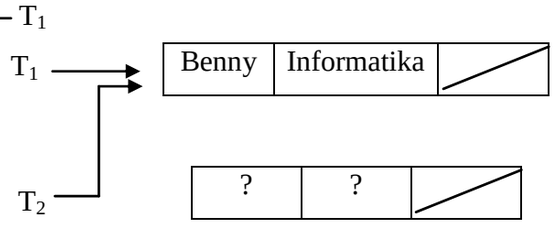
```
Kamus :
  Type
      Point =  ↑Data
      Data  = Record
            < Nama_Mhs : String
              Jurusan      : String>
      Endrecord
 T1, T2 : Point;
```



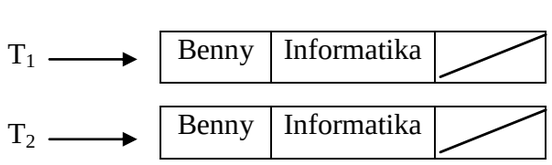
Akibatnya simpul diatas menjadi :



Apabila $T_2 \leftarrow T_1$



2. Mengcopy isi simpul, artinya bahwa dua/lebih simpul ditunjuk leh pointer yang berbeda tetapi mempunyai isi yang sama. Apabila $T_2\uparrow \leftarrow T_1\uparrow$; maka dapat diillustrasikan sebagai berikut:



3. Menghapus pointer

   Apabila pointer dihapus, maka lokasi semula yang ditempati oleh simpul yang ditunjuk oleh pointer tersebut akan bebas sehingga bisa digunakan oleh variable lain.

   Bentuk : Dealloc (Variabel)

   Misal : Dealloc (T_1)

LINKED LIST

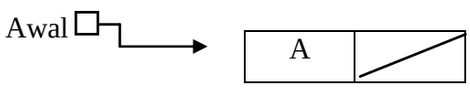
- Struktur ini terdiri dari rangkaian elemen yang saling berhubungan / berkaitan, dimana setiap elemen dihubungkan dengan elemen lainnya oleh sebuah pointer.
- Pointer, sel yang nilainya merupakan alamat sel yang lain dimana sel yang lain itu dapat berupa data atau berupa pointer juga
- Setiap elemen dalam linked list selalu berisi pointer
- Deklarasi Linked List

```
Type
        nama_pointer = ↑Simpul
        Simpul  = Record
           medan_data  : tipedata
           medan_sambungan : Namapointer
        EndRecord
    nama_var_pointer : nama_pointer
```

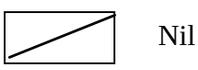
```
Type
      Point =   ↑Data
      Data  = Record
            <Info   : char
             Next  : Point >
       Endrecord
  awal, akhir: Point
```

Istilah – istilah

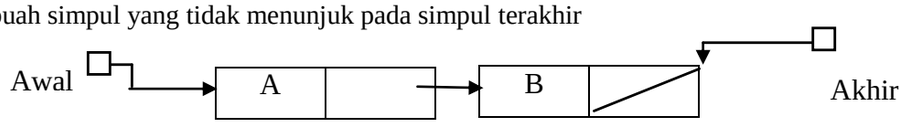
- Simpul, terdiri dari dua bagian :
  a. Bagian/medan data (info)
  b. Bagian/medan sambungan (pointer yang menunjuk kesimpul berikutnya)
- Awal (First), variable yang berisi alamat yang menunjuk lokasi simpul pertama linked list



- Nil / Null, Tidak bernilai yaitu menyatakan tidak mengacu kealamat manapun.



- Akhir, sebuah simpul yang tidak menunjuk pada simpul terakhir



- Linked List kosong dikenali dengan

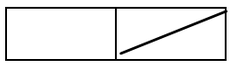
  Awal ← nil

  Awal ⊠

- Elemen terakhir linked list dikenali dengan : Akhir↑.Next ← Nil
- Linked List terdiri dari 3 macam yaitu :
  a. Single Linked List
  b. Double Linked List
  c. Circular Linked List

I. Single Linked List

Adalah linked list dengan simpul berisi satu link / pointer yang mengacu ke simpul berikutnya. Skema Simpul Single Linked List :



Operasi – operasi pada single linked list :

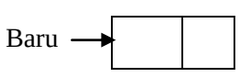
1. Penciptaan

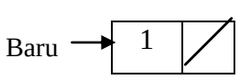
   Awal dan akhir diberi nil.

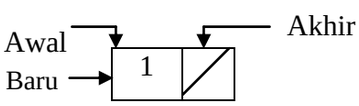
2. Penyisipan

   a. Penyisipan didepan

   - **List kosong {Awal=nil}**

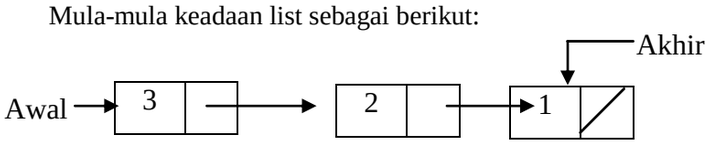


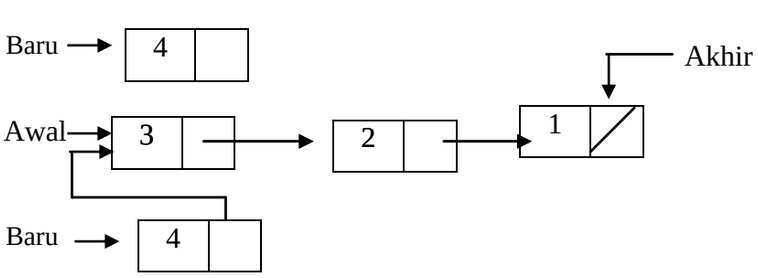


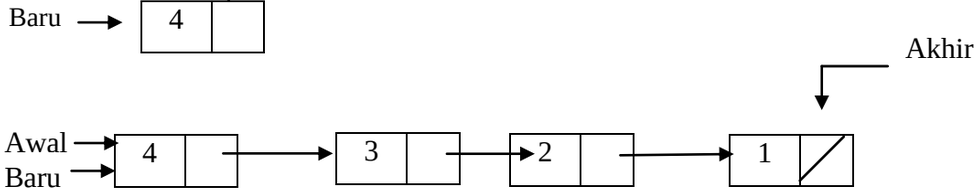


   - **List tidak kosong {Awal ≠ Nil}**

     Mula-mula keadaan list sebagai berikut:





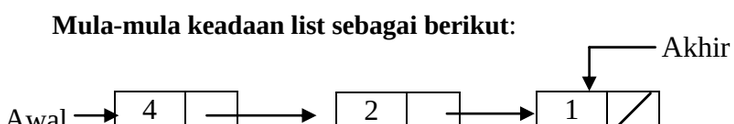


   b. Penyisipan di tengah

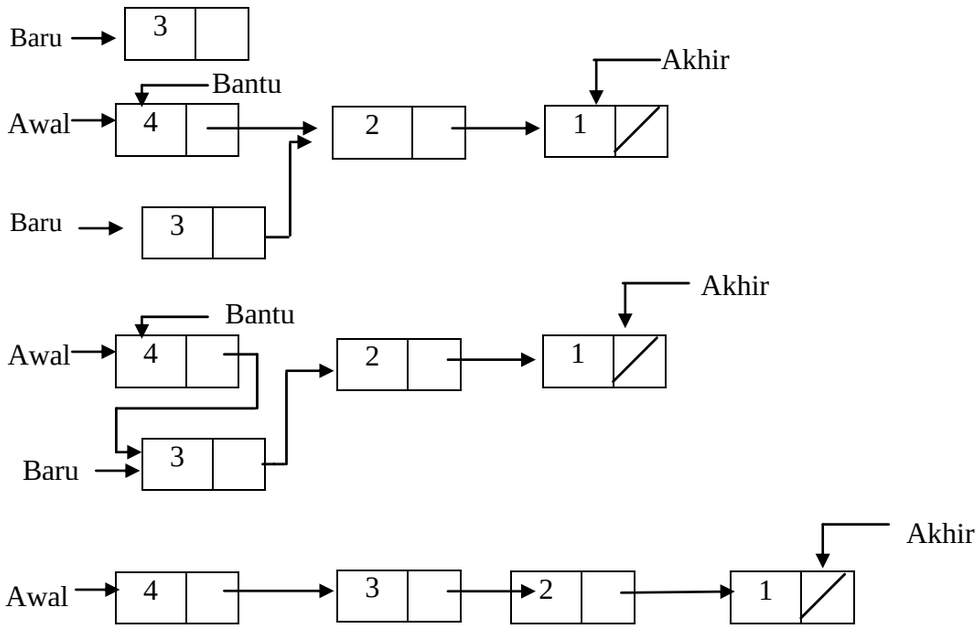
   - List Kosong {Awal = Nil} (Sama dengan penyisipan di depan)

   - List tidak kosong {Awal ≠ Nil}

     Mula-mula keadaan list sebagai berikut:



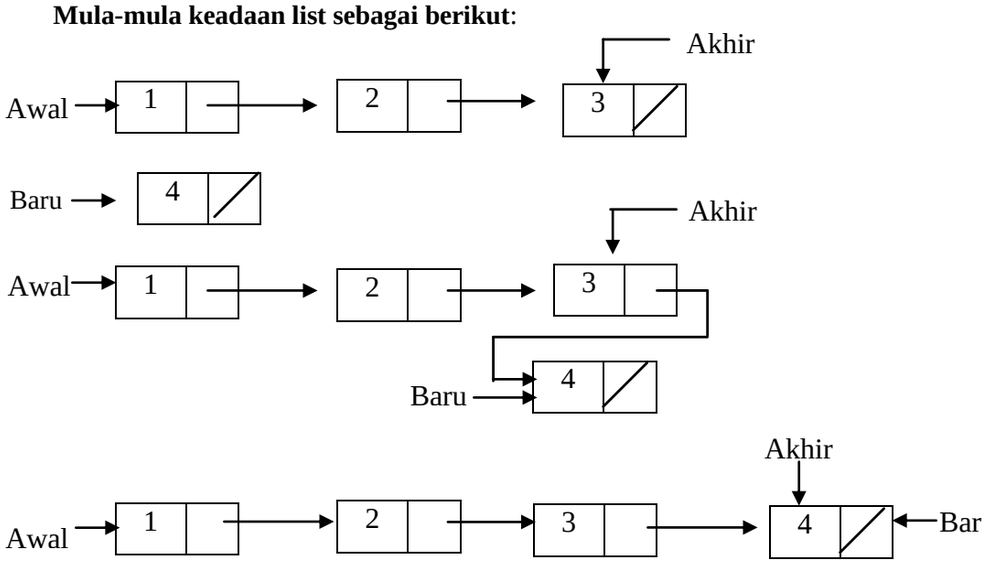
Asumsi data : 3 akan disisipkan setelah data ‘4’



c. <u>Penyisipan di akhir</u>

- List kosong {Awal = Nil} (Sama dengan penyisipan di depan)
- List tidak kosong (Awal ≠ Nil}

Mula-mula keadaan list sebagai berikut:



```
Procedure sisip_depan_single(Input  elemen : tipedata, I/O  awal, akhir : nama_pointer)
{I.S. : data yang akan disisipkan (elemen), pointer penunjuk awal dan pointer penunjuk akhir sudah terdifinisi}
{F.S. : menghasilkan satu simpul yang disisipkan di depan pada single linked list}
Kamus :
        baru : nama_pointer
Algoritma :
        Alloc(baru)
        baru↑.info ← elemen
        If (awal = nil)
          Then
                baru↑.next ← nil
                akhir ← baru
          Else
                baru↑.next ← awal
        EndIf
        awal ← baru
EndProcedure

Procedure sisip_belakang_single(Input  elemen : tipedata, I/O  awal, akhir : nama_pointer)
{I.S. : data yang akan disisipkan (elemen), pointer penunjuk awal dan pointer penunjuk akhir sudah terdifinisi}
{F.S. : menghasilkan satu simpul yang disisipkan di belakang pada single linked list}
Kamus :
        baru : nama_pointer
Algoritma :
        Alloc(baru)
        baru↑.info ← elemen
        baru↑.next ← nil
        If (awal = nil)
          Then
                awal ← baru
          Else
                akhir↑.next ← baru
        EndIf
        akhir ← baru
EndProcedure

Procedure sisip_tengah_single(Input  elemen : tipedata, I/O  awal, akhir : nama_pointer)
{I.S. : data yang akan disisipkan (elemen), pointer penunjuk awal dan pointer penunjuk akhir sudah terdifinisi}
{F.S. : menghasilkan satu simpul yang disisipkan di tengah pada single linked list}
Kamus :
        Procedure sisip_belakang_single(Input  elemen : tipedata, I/O  awal, akhir : nama_pointer)
        baru,bantu : nama_pointer
        ketemu : boolean
        datasisip : tipedata
```

```
Algoritma :
        If (awal = nil)
         Then
                Alloc(baru)
                baru↑.info ← elemen
                baru↑.next ← nil
                awal ← baru
                akhir ← baru
         Else
                {Pencarian menggunakan metode Sequential dengan boolean}
                Input(datasisip)
                bantu ← awal
                ketemu ← false
                While (not ketemu and bantu ≠ nil) do
                        If (datasisip = bantu↑.info)
                         Then
                                ketemu ← true
                         Else
                                bantu ← bantu↑.next
                        EndIf
                EndWhile
                If (ketemu)
                 Then
                        Alloc(baru)
                        baru↑.info ← elemen
                        If (bantu = akhir)
                         Then
                                sisip_belakang_single(elemen,awal,akhir)
                         Else
                                baru↑.next ← bantu↑.next
                                bantu↑.next ← baru
                        EndIf
                 Else
                        Output("Data yang akan disisipkan tidak ada");
                EndIf
        EndIf
EndProcedure

Procedure isi_elemen(Output  elemen : tipedata)
{I.S. : user memasukkan data yang akan disisipkan (elemen)}
{F.S. : mengahsilkan data yang akan disisipkan (elemen)}
Kamus :

Algoritma :
        Input(elemen)
EndProcedure
```

```
{Algoritma Utama}
Algoritma_penyisipan_Single_Linked_List
Kamus :
        {prototype}
        Procedure isi_elemen(Output  elemen : tipedata)
        Procedure sisip_depan_single(Input  elemen : tipedata, I/O  awal, akhir : nama_pointer)
        Procedure sisip_belakang_single(Input  elemen : tipedata, I/O  awal, akhir : nama_pointer)
        Procedure sisip_tengah_single(Input  elemen : tipedata, I/O  awal, akhir : nama_pointer)

        {deklarasi}
        nama_pointer = ↑simpul
        simpul = Record
        < info : tipedata,
          next : nama_pointer>
        EndRecord

        awal, akhir : nama_pointer        {pointer penunjuk}
        menu : integer
        elemen : tipedata
Algoritma :
        {penciptaan list}
        awal ← nil
        akhir ← nil

        {penyisipan}
        Repeat
                output("Menu Pilihan")
                output("==========")
                output("1. Sisip Depan")
                output("2. Sisip Tengah")
                output("3. Sisip Belakang")
                output("0. Keluar")
                Input(menu)
                Depend on (menu)
                        (menu = 1)      : isi_elemen(elemen)
                                          sisip_depan_single(elemen, awal, akhir)
                        (menu = 2)      : isi_elemen(elemen)
                                          sisip_tengah_single(elemen, awal, akhir)
                        (menu = 3)      : isi_elemen(elemen)
                                          sisip_belakang_single(elemen, awal, akhir)
                EndDepend
        Until (menu = 0)
```