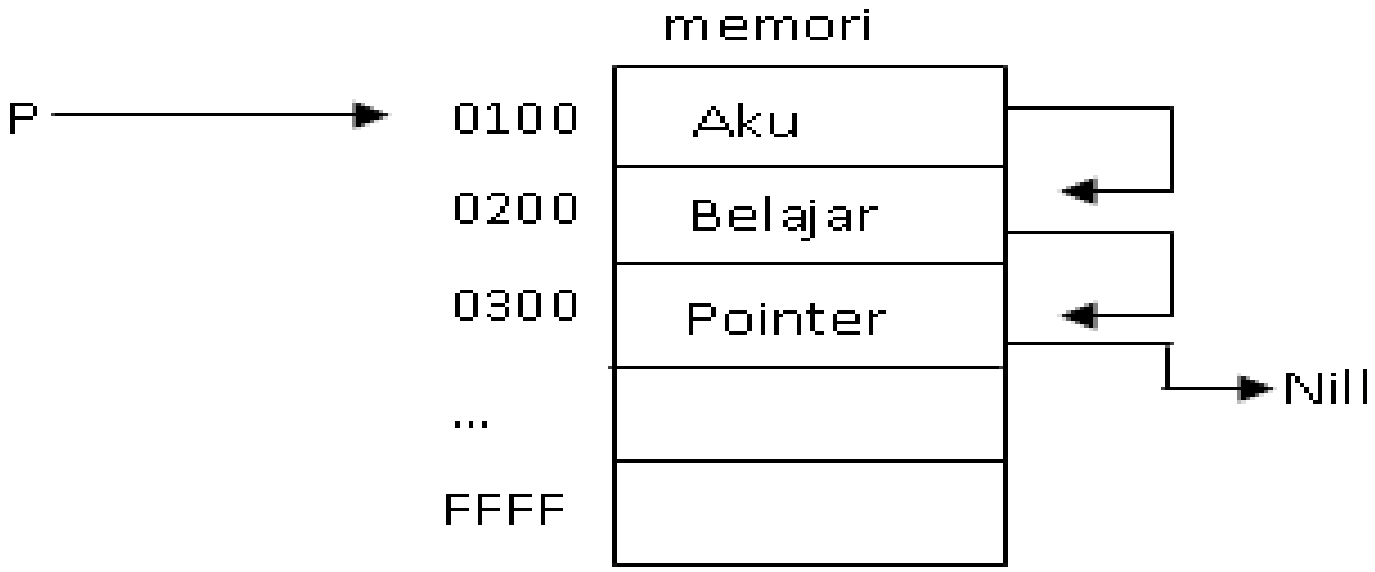
SINGLE LINKED LIST

Struktur Data

PENGERTIAN

- **Pointer** : Setiap kali menambahkan data selalu menggunakan variabel pointer yang baru, otomatis akan membutuhkan banyak variabel pointer (penunjuk)

- **Linked list** : Menggunakan satu variabel pointer untuk menyimpan banyak data



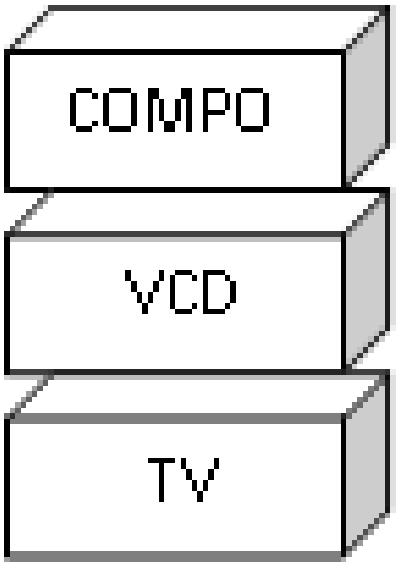
- **Node / Simpul** : Tempat yang disediakan pada suatu area memory tertentu untuk menyimpan data
- Setiap node memiliki pointer (penunjuk) yang menunjuk ke simpul berikutnya sehingga terbentuk suatu untaian yang hanya memerlukan sebuah variable pointer yang disebut **LINKED LIST**

2 METODE

- LIFO : Stack(Tumpukan)
- FIFO : QUEUE (Antrian)

LIFO / Last In First Out

- LIFO adalah suatu metode pembuatan linked list, dimana data yang masuk paling akhir adalah data yang keluar paling awal. Hal ini dapat dianalogikan (dalam kehidupan sehari-hari) pada saat anda menumpuk barang, seperti digambarkan di bawah ini :

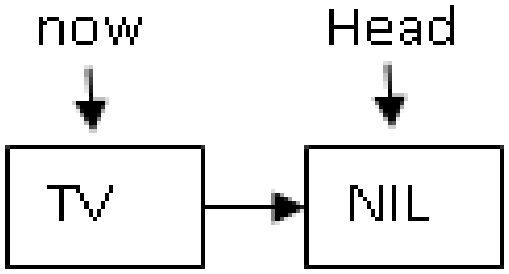


Penambahan / Insert simpul dibelakang

Procedure Insert

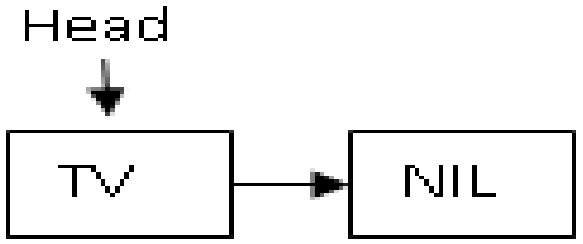
```
Procedure INSERT(elemen:TipeData);
Var Now : Point;
 Begin
  New(Now);
  Now^.Isi:=elemen;
  If Head=Nil then
      Now^.Next:=Nil;
  Else
      Now^.Next:=Head;
  Head:=Now;
End;
```

- **Insert (TV)**
- new (now)
- now^ .isi = elemen → now^.isi = 'TV'

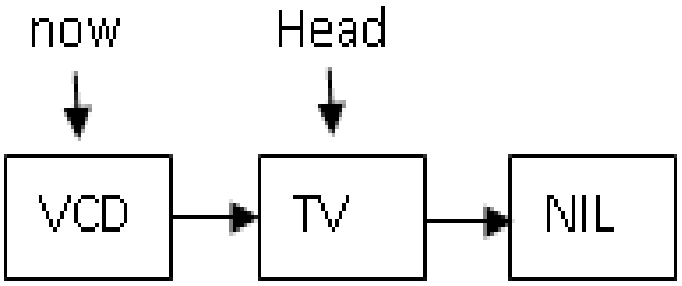


Head = Nil then

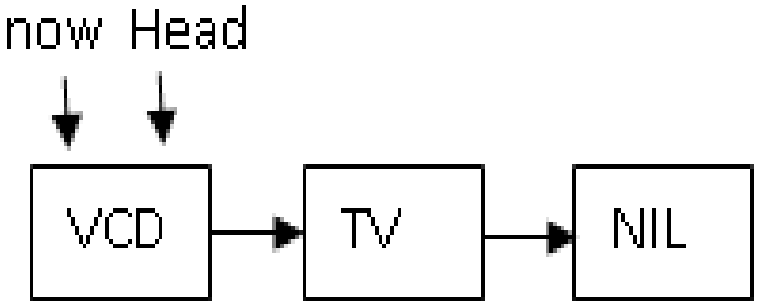
- now^.isi = Nil
- Head = TV



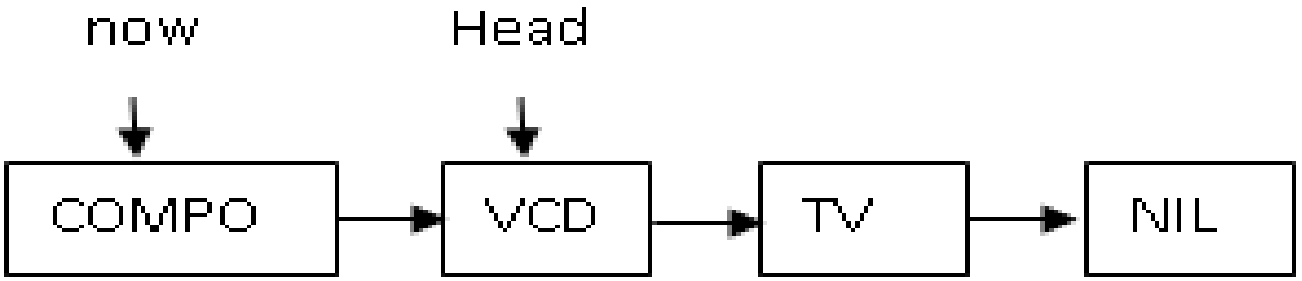
- **Insert (VCD)**
- new(now)
- now^.isi = 'VCD'



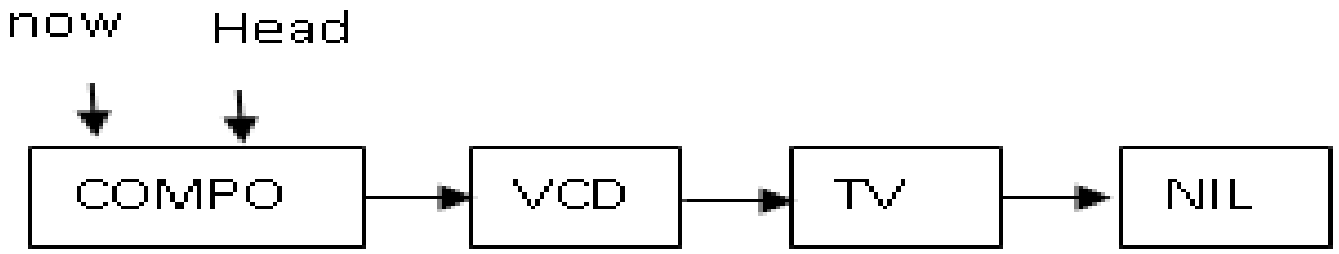
- Head <> Nil
- Head := VCD



- **Insert COMPO**
- new (now)
- now^.isi = ‘COMPO’



- Head < > Nil
- Head := COMPO



Procedure Find First

- Procedure find_first;
  Begin
  - Now := head;
  - Write(now^,isi);

  end;

- now := Head
  now = COMPO
  write(COMPO)

Procedure Find Next

- Procedure find_next;
  Begin
  - If now^.next < > nil then
  - Now := now^.next;
  - Write(now^,isi);

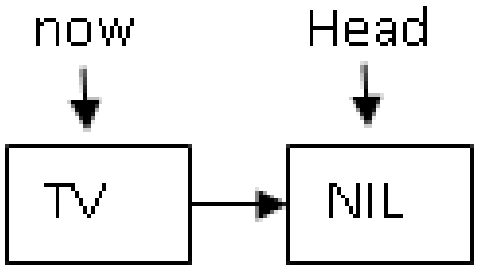
  end;

- now^.isi < > Nil → COMPO < > Nil
  now := now^.next
  now : = VCD
  write (VCD)

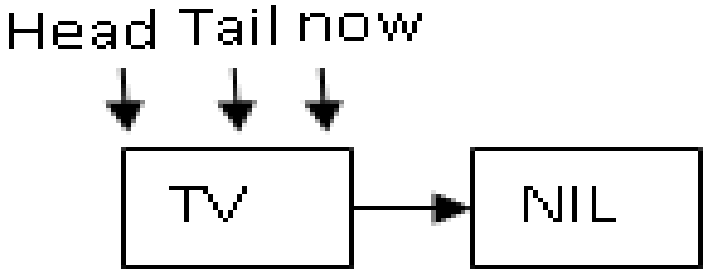
FIFO / First In First Out

- Contoh : Antrian diloket
- Penambahan / Insert simpul di depan

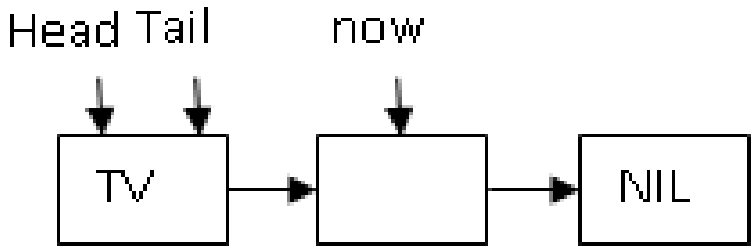
- **Insert (TV)**
- new (now)
- now^.isi = elemen → now^.isi = 'TV'



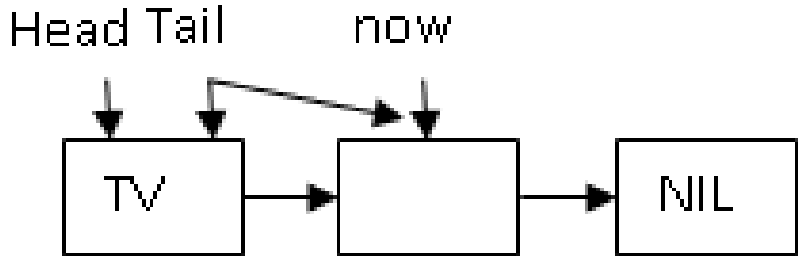
- Head = Nil then
- Head = TV

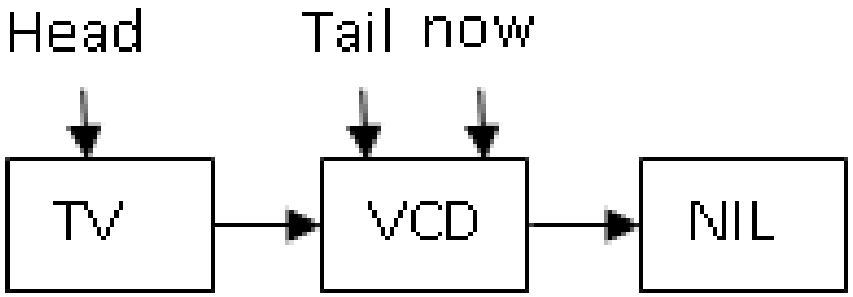


- **Insert (VCD)**
- new (now)
- Head < > Nil then
- Tail^.next = now

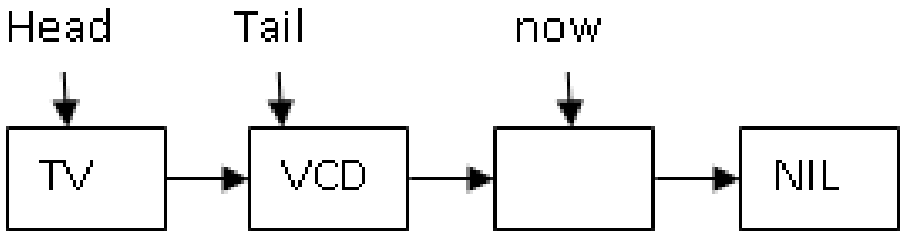


- Tail := now

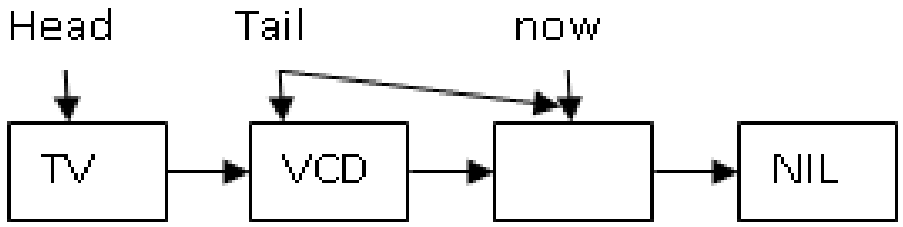


- Now^.isi = VCD
- 

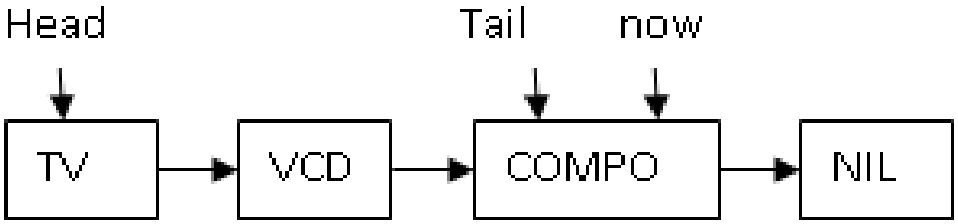
- **Insert (COMPO)**
- new(now)
- Head< > Nil then
- Tail^.next = now



- Tail := now
- Tail^.next=Nil



- now^.isi = COMPO



Procedure Find First

- Procedure find_first;

  Begin
  - Now := head;
  - Write(now^,isi);

  end;

- now := TV

  write(TV)

Procedure Find Next

- Procedure find_next;
  Begin
  - If now^.next < > nil then
  - Now := now^.next;
  - Write(now^,isi);

  end;

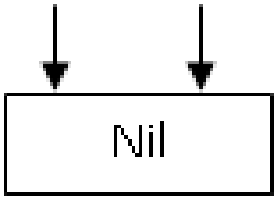
- now^isi.<> Nil → TV <> Nil
  now := now^.next
  now := VCD
  write(VCD)

Procedure dan Function Linked List Lainnya

- Create : Membuat sebuah linked list yang baru dan masih kososng. (ket: procedure ini wajib dilakukan sebelum menggunakan linked list)
- Procedure Create;

```
Begin
        Head:=nil;
        Tail:=nil;
End;
```

- Head Tail



- Empty : Function untuk menentukan apakah linked list kosong atau tidak
- Function Empty : Boolean;

```
Begin
        If head = nil then
                Empty:= true
        else
                empty:= false;
end;
```

- Retrieve : Mengambil elemen yang ditunjuk oleh now. Elemen tersebut lalu ditampung pada suatu variabel (di bawah dicontohkan variabel r).
- Procedure Retrieve(var r: TipeData);

```
Begin
        R:= Now^.isi;
 End;
```

- Update : Mengubah elemen yang ditunjuk oleh now dengan isi dari suatu variabel (di bawah dicontohkan variabel u).
- Procedure Update(u: TipeData);

```
Begin
        Now^.isi:=u;
 End;
```

- Delete Now : Menghapus elemen yang ditunjuk oleh now. Jika yang dihapus adalah elemen pertama dari linked list(head), maka head akan berpindah ke elemen berikut
- Procedure DeleteNow;

```
Var x : point;
        Begin
                If now<>head then
                  Begin
                    x:=head;
                    while x^.next<>now do
                            x:=x^.next;
                    x^.next:=now^.next;
                   end
                else head:= head^.next;
                            dispose(Now);
                Now:= head;
End;
```

- Delete Head : Menghapus elemen yang ditunjuj head. Head berpindah ke elemen sesudahnya.
- Procedure DeleteHead;

```
Begin
    If head<>nil then
      Begin
        Now:=head;
        Head:=Head^.next;
        Dispose(Now);
        Now:=Head;
    End;
End;
```

- Clear : Untuk menghapus linked list yang sudah ada.wajib dilakukan bila ingin mengakhiri program yang menggunakan linked list. Jika tidak data-data yang dialokasikan ke memori pada program sebelumnya akan tetap tertinggal di dalam memori.
- Procedure Clear;

```
Begin
 While head <> nil do
    Begin
       Now:=head;
       Head:=head^.next;
       Dispose(Now);
    End;
End;
```

- **Latihan :**

  Buatlah ilustrasi untuk procedure insert, find first dan find next untuk data Antrian : Alya, Alby, Shaina

  Tugas :

  Buat program tumpukan dengan menggunakan procedure Insert dan update !