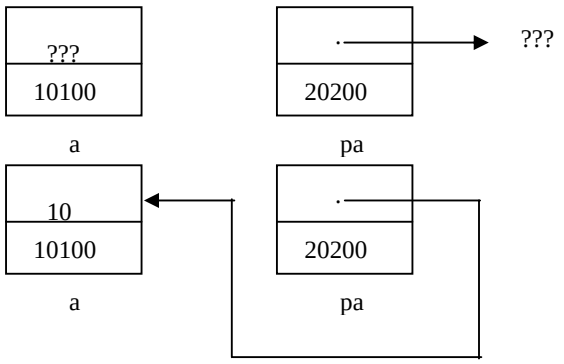
BAB VIII
POINTER

Tipe data pointer digunakan pada pemrograman dinamis, artinya kebutuhan memori tergantung saat program dieksekusi. Berbeda dengan pemrograman yang menggunakan predefined-types sebelumnya, dimana kebutuhan memori ditetapkan saat pendefinisian tipe data sehingga bisa berakibat memori yang teralokasi tidak semuanya terpakai, seperti penggunaan array (larik). Pemrograman seperti ini dikenal sebagai pemrograman memori statis.

8.1 Pengertian Pointer

Lihat potongan program di bawah ini :

```
Type pointer : ^integer;
Var a  : integer;
    pa : pointer;
Begin
    a    : = 10;
    pa^ : = a;
    …..
```



Sebelum a diberi nilai, a ditempatkan oleh sistem ke suatu alamat memori tertentu. Demikian juga pointer pa, sebelum diberikan nilai, pa menunjuk kepada suatu tipe data integer tak tentu. Nilai a adalah suatu nilai yang terletak didalam jangkauan tipe integer, sedangkan nilai pa adalah alamat memori suatu variabel/pengenal dengan tipe integer (lihat gambar). Oleh karena itu untuk mengakses sebuah pointer untuk kasus diatas digunakan operator isi ^ (atap atau simpul).

8.2 Deklarasi Pointer

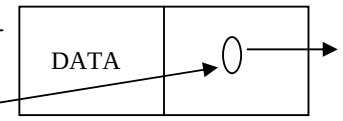
```
Type pengenal = ^simpul;

        Simpul = tipe;
Contoh : type bil_bulat = ^integer;
          var i, j : bil_bulat;
Umumnya tipe berupa rekaman (struktur) record, misal deklarasi :
Type str30 = string[30];
     point = ^data;
     data  = record
                nomer      : integer;
                nama_peg : str30;
                alamat       : str30;
                pekerjaan  : str30;
             end;
Kemudian bentuk definisinya :
        var P1, P2              : point;
            a,b,c      : str30;
```

Agar nampak bahwa P1 dan P2 bersifat dinamis terhadap kebutuhan memori, deklarasi di atas dimodifikasi sebagai berikut :

```
Type str30 = string[30];
     point = ^data;
```

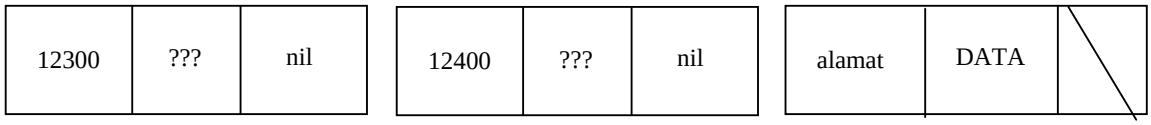
```
data  = record
          nomer     : integer;
          nama_peg : str30;
          alamat     : str30;
          pekerjaan  : str30;
          next       : point; {elemen ini baru}
        end;
```



Setelah pendefinisian, kemudian minta memori kepada sistem dengan perintah :

new (P1); new (P2);

maka P1 dan P2 sekarang siap diakses. Di sini ada dua elemen yang perlu dibedakan. Pertama untuk isi data sedangkan komponen kedua untuk pointer yang menunjuk kepada elemen berikutnya. Karena P1 dan P2 dalam hal ini hanya masih berdiri sendiri, maka elemen kedua P1 dan P2 sesuai definisi masing-masing menunjuk ke NIL (Not In List, tak ada elemen yang mengikuti).



P1 P2 Pointer

8.3 Operasi pada Pointer

Setelah pengalokasian memori dilakukan, langkah kedua pemberian nilai. Misalkan data P1 diberikan inisialisasi seperti berikut :

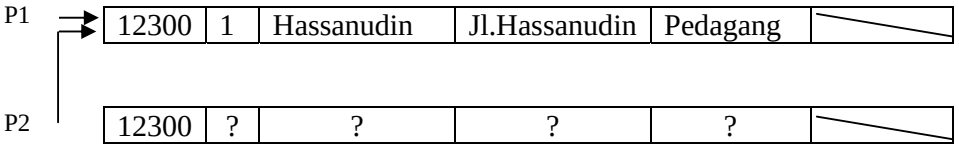
P1^.nomer := 1; P1^.nama := ‘Hasanuddin’; P1^.alamat := ‘Jl.Hasanudin 17; P1^pekerjaan := ‘Pedagang’;

Ilustrasi :

Ada 2 operasi yang sering diterapkan :

1. Mengkopi pointer

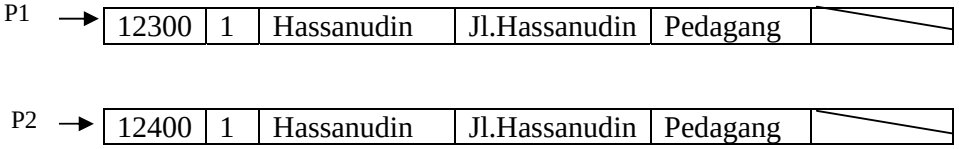
Contoh : P2:=P1;



2. Mengkopi isi(data)

Dianggap bagian pertama tidak dilakukan, contoh ;

P2^:=P1^;



Operasi pada pointer hanya bisa dilakukan jika tipe keduanya sama. Operator lainnya seperti operator biner (<,>,<>,<=,>=) dapat digunakan.

Contoh :

```
If (T1 = T2) then {apakah lokasi yang ditunjuk T1 dan T2 sama}
If (T1 = nil) then {apakah sampai elemen terakhir}
While (T2<>nil) do
```

Demikian juga untuk operasi pada isi bisa digunkan operator-operator di atas, misal :

```
If (T1^=T2^) then
While (T1^<>T2^) do
```

8.4 Menghapus Pointer

Pointer dapat mendealokasikan (memberikan kembali) memori yang diminta kepada sistem jika memang tidak diperlukan lagi. Setelah pendealokasian memori, maka pointer tersebut tidak dapat lagi diakses. Logis!Artinya data sudah tidak eksis lagi karena ikut terhapus.

Statement pendealokasian :

```
Dispose (perubah);
```

Contoh :

```
Dispose (P1);
```