Section 04

Linked List

Learning Outcomes

Pada akhir pertemuan ini, diharapkan mahasiswa

akan mampu :

- **mendemonstrasikan jenis dan operasi pada Linked List.**
- **Menerapkan Linked List pada program aplikasi komputer.**

Materi

- Pointer
  - Terminologi
  - Operasi
- Linked List
  - Komponen
  - Jenis
  - Akses
  - Pembangunan
  - Operasi

Pointer

- **POINTER**
  Tipe Data yang isinya berupa address dari data lain. Alokasi data di memory dg menggunakan pointer di memory bersifat DINAMIS.

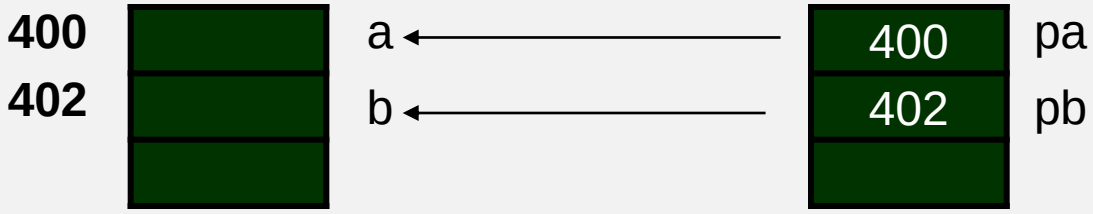
- **OPERASI POINTER**
  Assignment =
  Relational = = , < , >, <=, >=, !=
  Dynamic new, delete (C++), malloc, free (C)

Contoh :

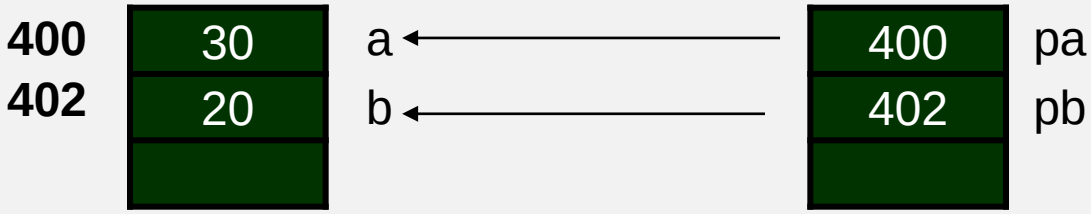
```
            ...
int         a,b;            /* variabel integer */
int         *pa, *pb;       /* variabel pointer */

pa = &a;                    /* pa berisi alamat variabel a */
pb = &b;                    /* pb berisi alamat variabel b */
            ...
```

Pointer (2)

Jika alamat a=400 dan b=402, maka :



Variabel a & b bisa scr tdk langsung diisi (*indirect assignment*) melalui pointernya : *pa= 30; *pb=20;



Linked List

Komponen & Jenis

LINKED LIST

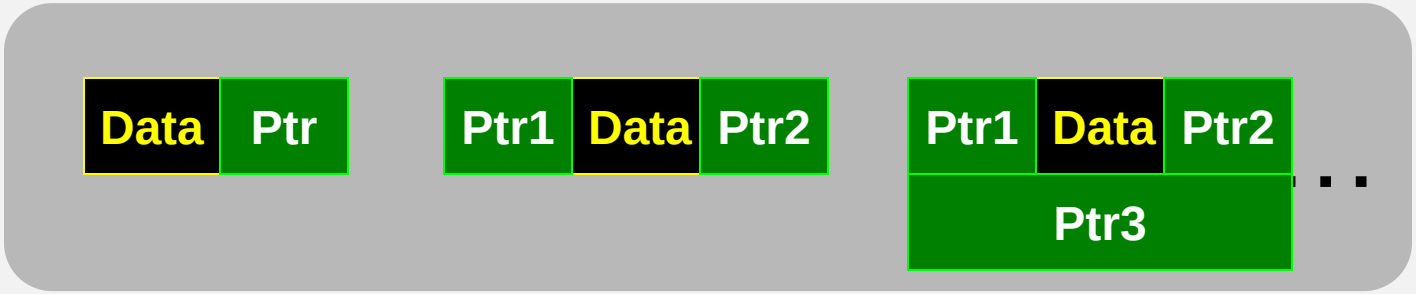
Beberapa simpul/node yg terkait dg bantuan pointer.

Setiap simpul berisi :

- **Field Data / informasi**
- **Field Pointer untuk menunjuk simpul berikutnya**

Berdasarkan banyaknya POINTER, LINKED LIST dibedakan menjadi :

- **Single Linked List**
- **Double Linked List**
- **Multiple Linked List**



Cara Akses

Cara Access Link List

- **SEQUENTIAL**

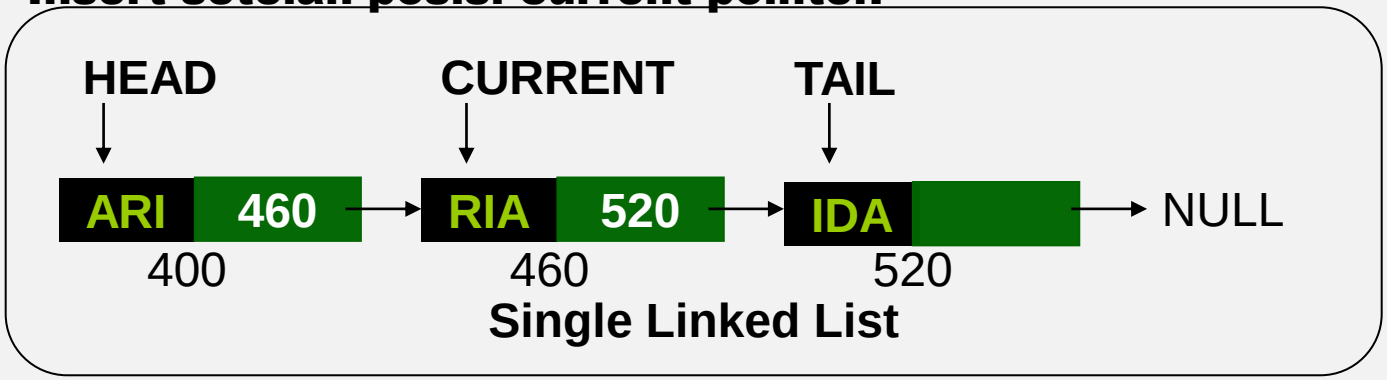
Tiga pointer yang umumnya digunakan :

- **HEAD : Pointer yg menunjuk elemen pertama.**
- **CURRENT : Pointer yg menunjuk ke elemen yang sedang aktif.**
- **TAIL : Pointer yg menunjuk elemen terakhir.**

Membangun Linked List

A.Single Linked list dapat dibangun dengan 3 cara:

- **Insert depan, node baru selalu berada didepan (menjadi Elemen yang ditunjuk oleh pointer Head).**
- **Insert belakang, node baru selalu berada di belakang (menjadi elemen yang ditunjuk oleh pointer Tail).**
- **Insert setelah posisi current pointer.**



B.Double Linked list dapat dibangun dengan 2 cara:

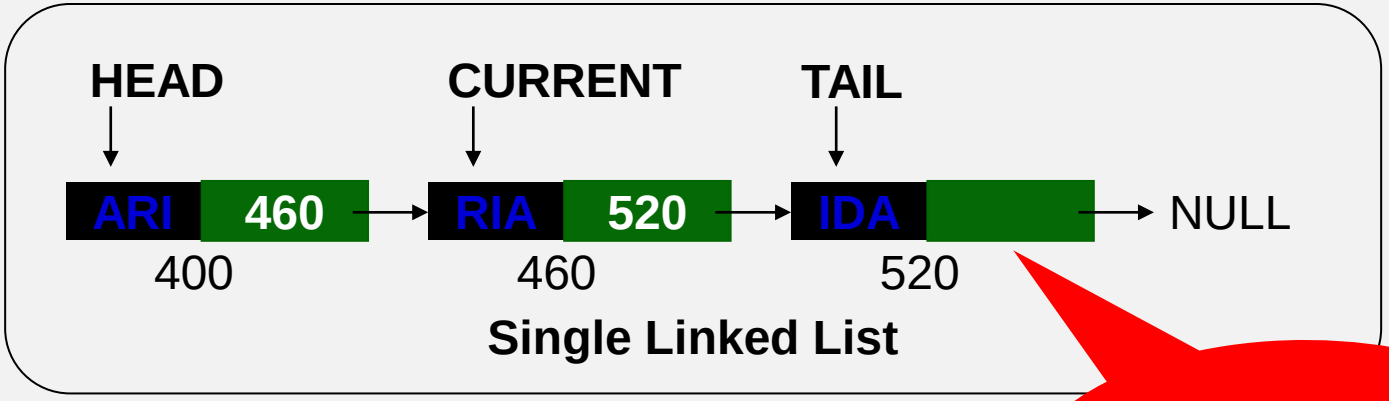
- **InsertBefore, insert sebelum posisi Current.**
- **InsertAfter, insert setelah posisi Current.**

Operasi

- **Create();**
  Untuk menciptakan Linked List yang baru dan kosong.
- **Insert(type e)**
  Untuk menambahkan 1 elemen/data ke dalam Linked List.
- **int Empty()**
  Untuk mengecek apakah Linked List masih kosong atau sudah berisi data.
- **Retrieve (type *e)**
  Untuk mengubah isi elemen yang ditunjuk oleh pointer current dengan isi dari variable yang dikirim (variable e).
- **Find_First()**
  Untuk mencari elemen pertama : yaitu dengan memposisikan pointer Current ke posisi HEAD.
- **Find_Next()**
  Untuk mencari elemen berikut yang ditunjuk oleh Pointer Current.
- **Delete()**
  Untuk menghapus elemen yang ditunjuk oleh Pointer Current.

Contoh Single Linked List

```
struct NODE {
        char          Nama[40];
        struct NODE     *next;
  } *HEAD, *CURRENT, *TAIL;
```



Bagaimana implementasi insert di Tail ?

Deklarasi dan alokasi memori :

```
    ...
NODE *P;
P  = (NODE*) malloc( sizeof( struct NODE ) );
/*  Set DATA  dan POINTER  */
    ...
```

Operasi

void Create()

// Untuk menciptakan Linked List yang baru dan kosong.

```
{ HEAD = null;
  TAIL  = null;
}
```

Operasi

```
Void Insert_Dpn(int e)/
// Untuk menambahkan 1 elemen/data ke dalam Linked List.
//Sisip di depan, di lokasi yang ditunjuk Head
{
Struct node *P;
P= (NODE *)malloc(struct node sizeof(struct node));
P -> Nama = 100;
P -> Next= HEAD;
HEAD = P;
}
```

Operasi

int Empty()

```
//Untuk mengecek apakah Linked List masih kosong atau
//sudah berisi data
{ if(HEAD== null) return 1;
  else return 0;
}
```

Find_First()

```
// Untuk mencari elemen pertama : yaitu dengan memposisikan
// pointer Current ke posisi HEAD.
{  CURRENT = HEAD; }
```

Find_Next()

```
//Untuk mencari elemen berikut yang ditunjuk oleh
//Pointer Current.
{  CURRENT = CURRENT -> NEXT; }
```

Operasi

```
Void Insert_blk(type e)
// Menambah 1 elemen ke dalam Linked List.
//Sisip di belakang, di lokasi yang ditunjuk TAIL
{
Struct node *P;
P= (NODE *)malloc(struct NODE sizeof(struct NODE));
P->DT = 200;
P->NEXT=null;
TAIL->NEXT = P;
TAIL = P;
}
```

Operasi

Retrieve (int *e)

```
//Untuk mengambil isi elemen yang ditunjuk oleh pointer
//current dan dimasukkan ke variabel yang ditunjuk oleh  e.
{
        *e = CURRENT -> DT;
}
```

LATIHAN

Cobalah untuk membuat fungsi Delete(); Untuk menghapus elemen yang ditunjuk oleh Pointer Current.

Selesai