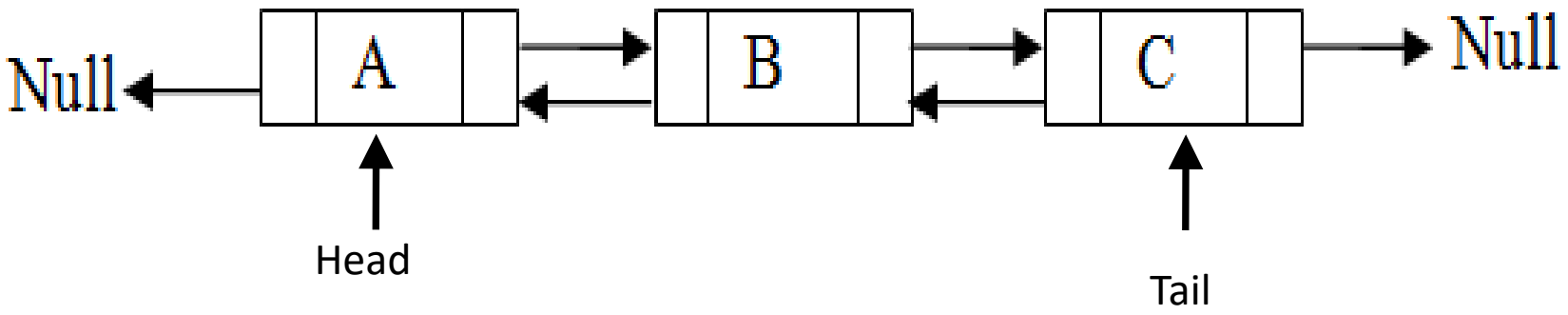
# DOUBLE LINKED LIST

DOSEN : SULISTYOWATI, ST., M.KOM.

# DOUBLE LINKED LIST

Pada Single Linked List, pointer hanya dapat bergerak satu arah saja, yakni ke kiri atau kanan saja. Untuk mengatasi hal ini maka digunakan Double Linked List, yaitu satu simpul/node memiliki 2 buah variabel pointer yang digunakan untuk menunjuk pada simpul/node berikutnya atau sebelumnya.

Ilustrasi Double Linked List :



# DOUBLE LINKED LIST

- Jadi sebuah simpul pada Double Linked List terdiri dari 3 bagian, yaitu :
  1. Medan informasi (info), berisi informasi atau data yang akan disimpan/diolah.
  2. Medan penyambung/pointer kiri, berisi alamat memori dari simpul/node sebelah kiri/sebelumnya.
  3. Medan penyambung/pointer kanan, berisi alamat memori dari simpul/node sebelah kanan/berikutnya.
- **HEAD** adalah pointer yang menunjuk ke simpul paling depan dan **TAIL** adalah pointer yang menunjuk ke simpul paling belakang.
- Simpul A, pointer kirinya adalah NULL dan simpul C pointer kanannya adalah NULL. Ini karena pointer tersebut tidak menunjuk/mengait ke simpul manapun.

# DEKLARASI DOUBLE LINKED LIST

Bentuk umum deklarasi *double linked list* :

```
struct tipe_data pointer
{
        tipe_data nama_medan_info;
        tipe_data_pointer *nama_medan_penyambung_kiri;
        tipe_data_pointer *nama_medan_penyambung_kanan;
};
tipe_data_pointer *nama_variabel_pointer;
```

# DEKLARASI DOUBLE LINKED LIST

**Keterangan :**

- tipe_data_pointer = tipe data buatan yang berupa simpul/node
- nama_medan_info = nama medan informasi
- nama_medan_penyambung_kiri = nama medan penyambung/pointer kiri
- nama_medan_penyambung_kanan = nama medan penyambung/pointer kanan
- nama_variabel_pointer = nama variabel yang bertipe pointer

# CONTOH DOUBLE LINKED LIST

Contoh :

```
struct simpul
{
        char info;      simpul *kiri, *kanan;
};
simpul *head, *tail, *baru, *bantu;
```

Dari contoh diatas, berarti telah dibuat sebuah tipe data buatan berupa struktur yang diberi nama “simpul”.

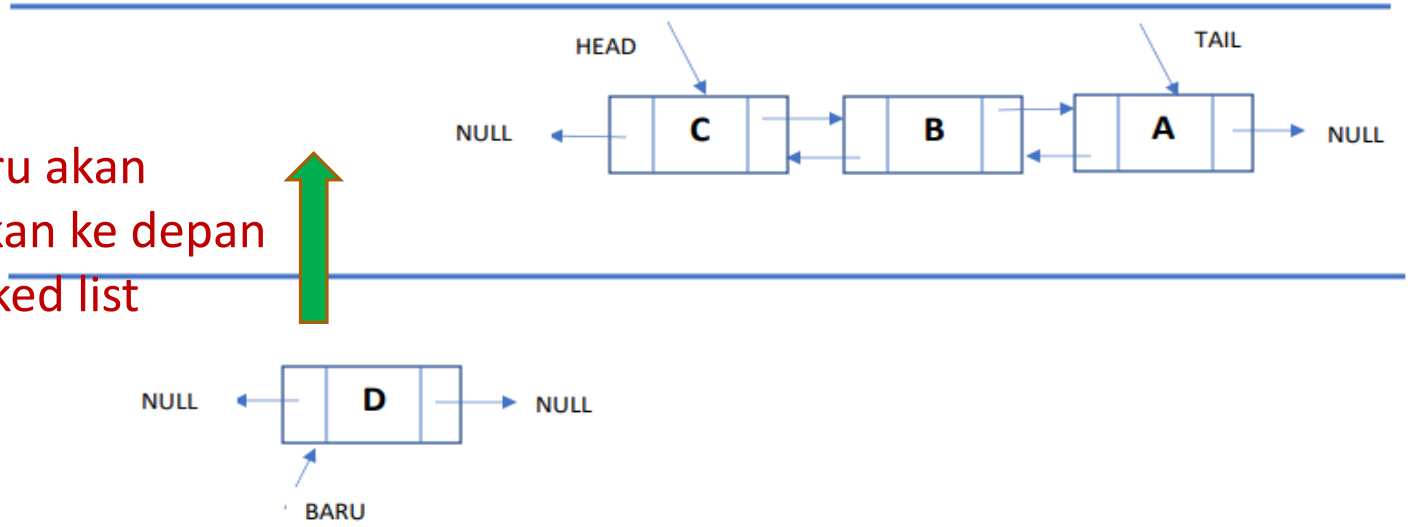
Dimana field yang terbentuk ada 3, yaitu :

data → untuk menyatakan meda informasi
kiri → untuk menyatakan medan penyambung kiri (harus bertipe pointer).
kanan → untuk menyatakan medan penyambung kanan (harus bertipe pointer).

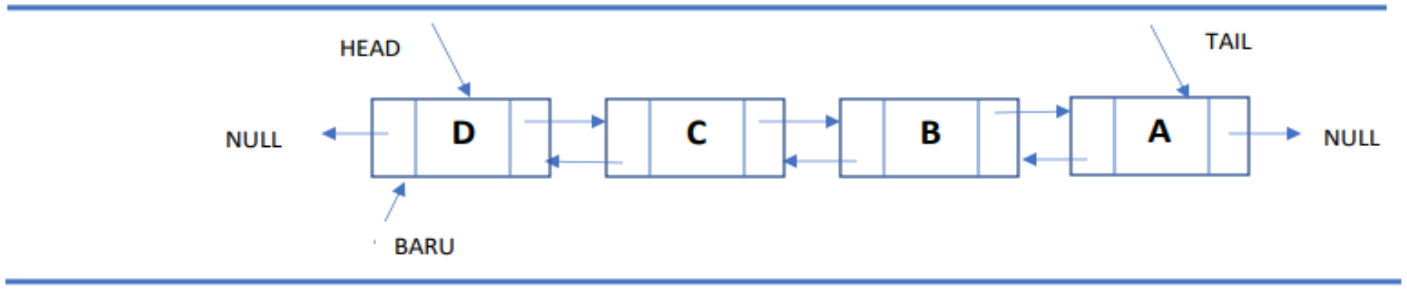
Dan double linked list yang terbentuk bernama DB

# MENAMBAH SIMPUL DI DEPAN DOUBLE LINKED LIST

Kondisi awal :




Kondisi akhir :



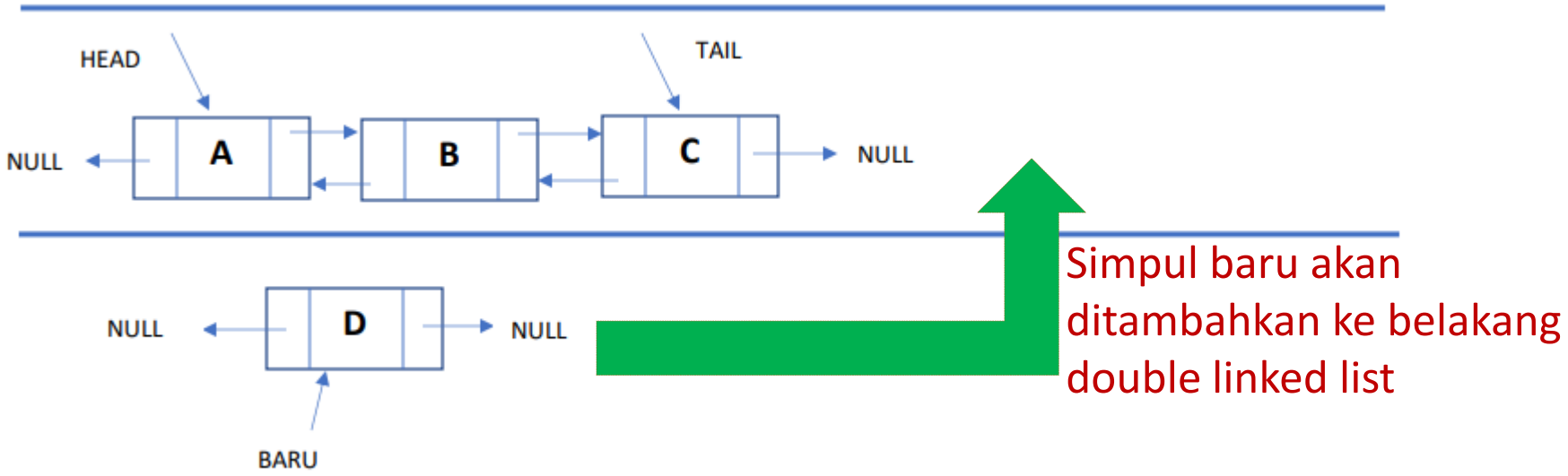
# MENAMBAH SIMPUL DI DEPAN DOUBLE LINKED LIST

Algoritma :

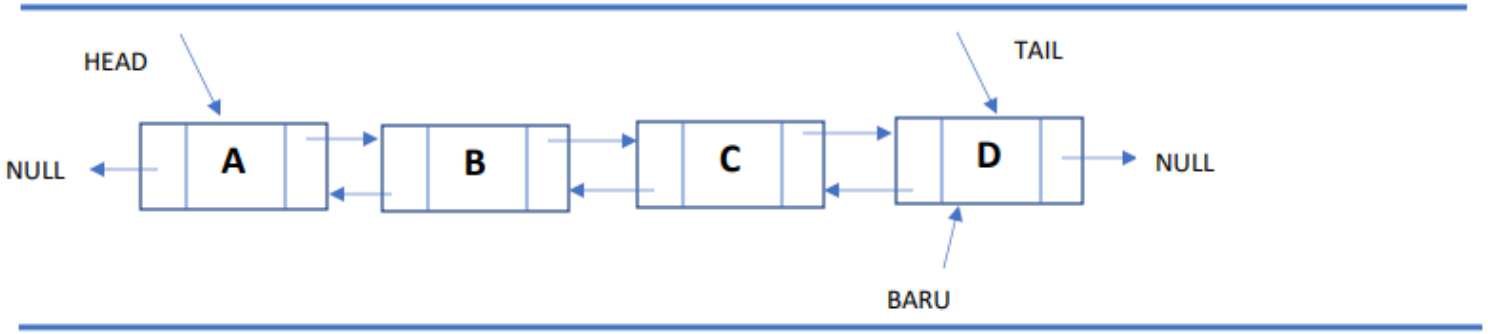
```
baru=new simpul;
cin>>baru->info;
baru->kiri=NULL;
baru->kanan=NULL;
if (head=NULL)
        tail=baru;
else
{
        baru->kanan=head;
        head->kiri=baru;
}
head=baru;
```

# MENAMBAH SIMPUL DI BELAKANG DOUBLE LINKED LIST

Kondisi awal :




Kondisi akhir :



# MENAMBAH SIMPUL DI BELAKANG DOUBLE LINKED LIST

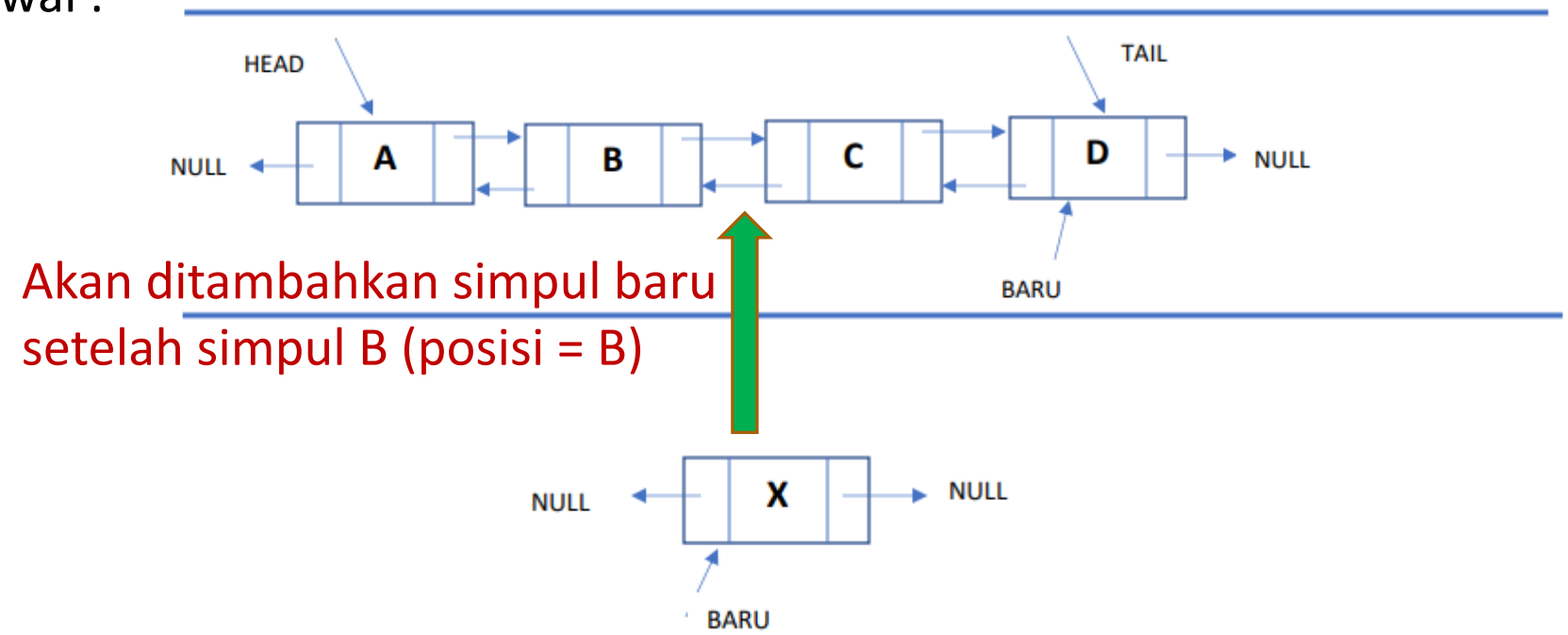
Algoritma :

```
baru=new simpul;
cin>>baru->info;
baru->kiri=NULL;
baru->kanan=NULL;
if (head=NULL)
        head=baru;
else
{
        tail->kanan=baru;
        baru->kiri=tail;
}
tail=baru;
```

# MENAMBAH SIMPUL DI TENGAH DOUBLE LINKED LIST

Untuk menambahkan simpul di tengah double linked list, maka harus diketahui terlebih dahulu posisi dimana simpul yang baru akan ditambahkan.

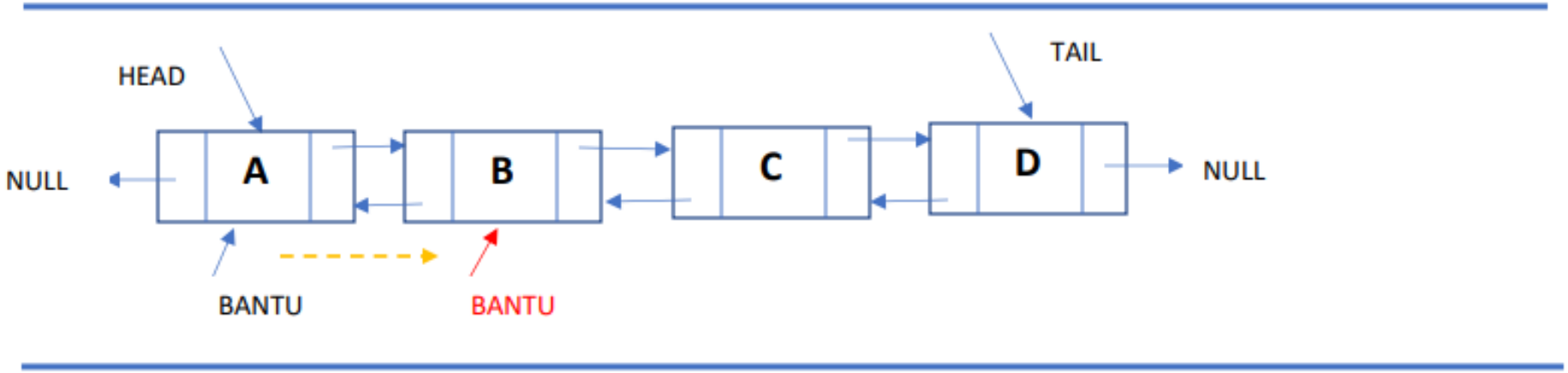
Kondisi awal :



# MENAMBAH SIMPUL DI TENGAH DOUBLE LINKED LIST

Langkah :

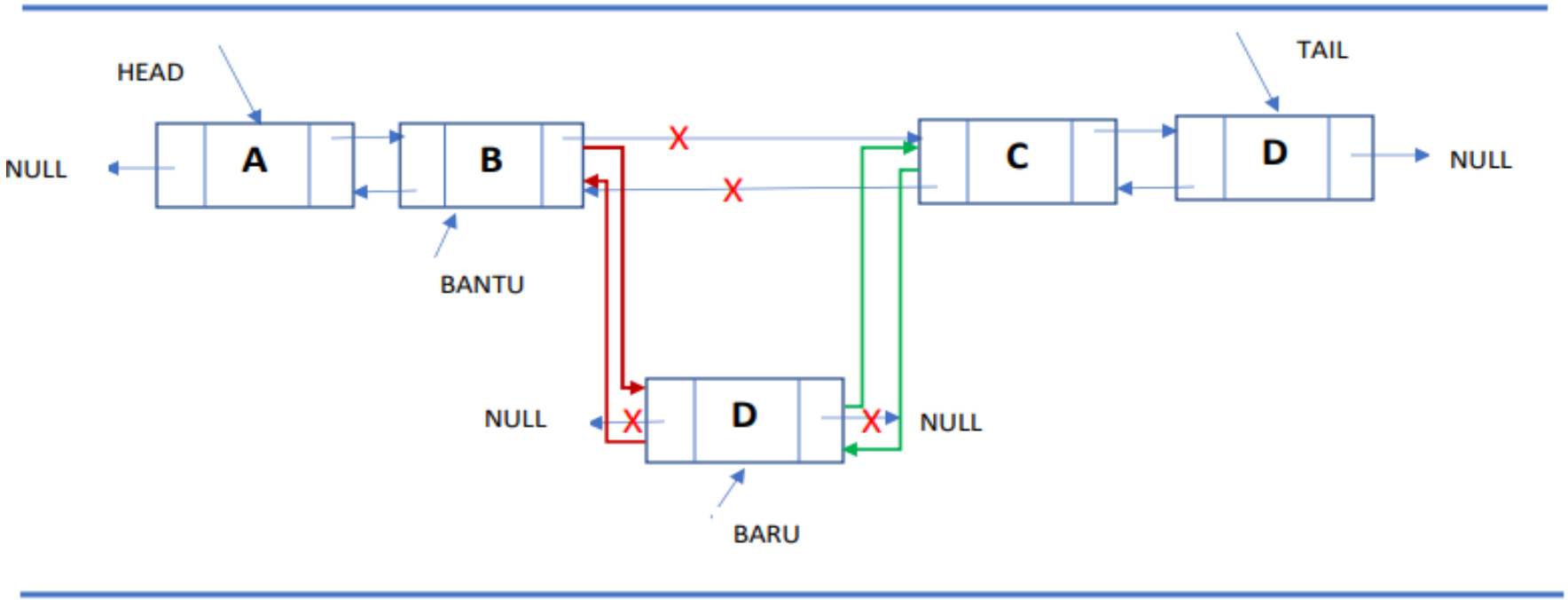
Dibutuhkan sebuah pointer bantuan, yang diberi nama pointer bantu. Pointer bantu semula diletakkan di posisi simpul paling depan atau sama dengan head. Kemudian pointer bantu bergerak terus ke belakang sampai menemukan posisi yang dicari.



# MENAMBAH SIMPUL DI TENGAH DOUBLE LINKED LIST

Langkah :

Jika sudah ditemukan, pointer bantu berhenti bergerak. Kemudian sisipkan simpul yang baru.



# MENAMBAH SIMPUL DI TENGAH DOUBLE LINKED LIST

Kondisi akhir :