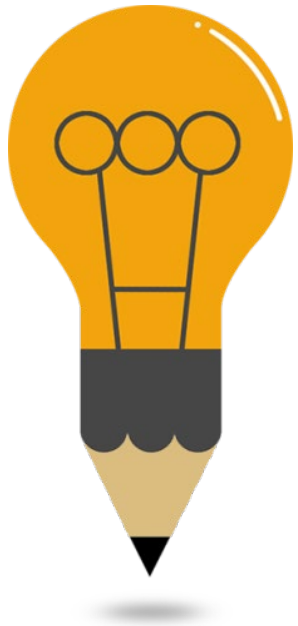
# STRUKTUR DATA

Pertemuan 5


(Tim)

# Agenda Pertemuan

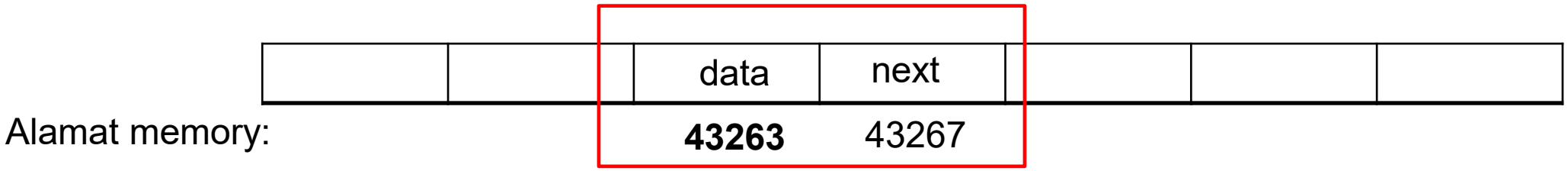


1. Pembahasan PR Latihan Single Linked List (Lanjutan)
2. Double Linked List

# Refreshing: Node dalam Linked List

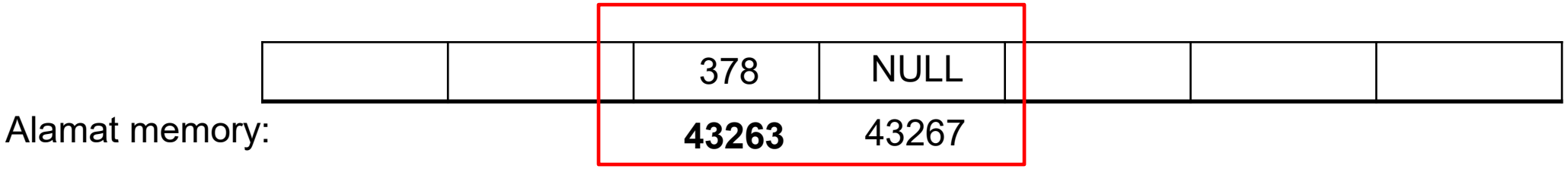
**Apa yang terjadi jika kita mendeklarasikan sebuah node?**

```
struct node{
    int data;
    struct node *next;
};

struct node * p = (struct node *)malloc(sizeof(struct node));
```

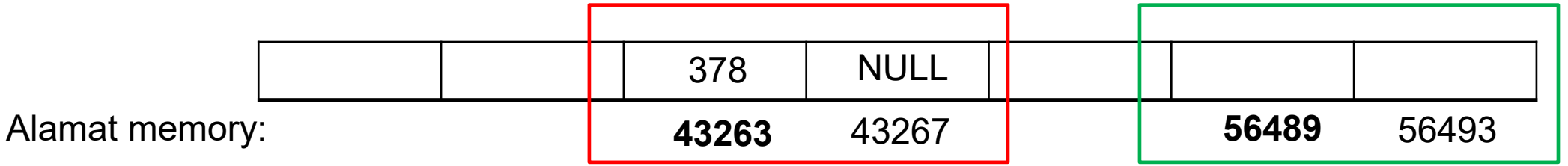



**p** akan berisi alamat pertama dari memory yang dialokasikan (kotak merah) yaitu **43263**

```
p->data = 378;
p->next = NULL ;
```




```
struct node * new_node = (struct node *)malloc(sizeof(struct node));
```



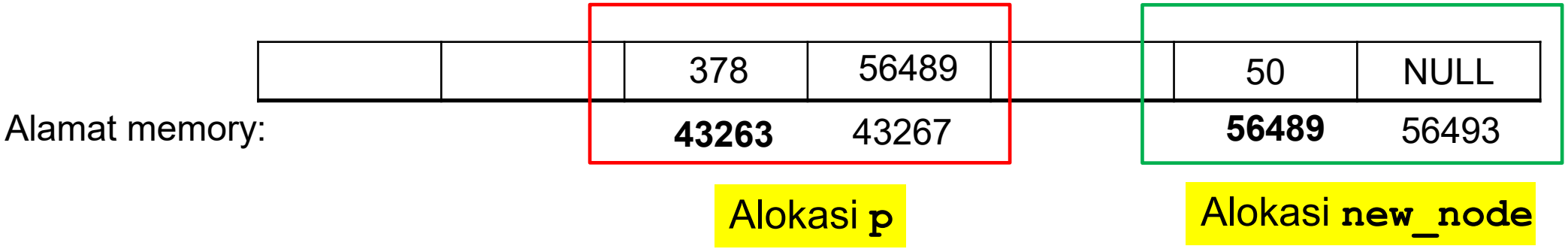

Alokasi **p**

Alokasi **new_node**

Sama seperti **p**, **new_node = 56489**

**Apa yang terjadi jika kita menghubungkan 2 node (p dan new_node)?**

```
new_node->data = 50;
new_node->next = NULL;

p->next = new_node ;
```



# Latihan Pertemuan 4

1. Buat fungsi untuk menampilkan nilai dari linked list! (tadi sudah dicontohkan)
2. Buat fungsi untuk menghitung jumlah node dalam sebuah linked list! (looping sama dengan no. 1)
3. Buat program untuk mengkonversi dari array 1D ke linked list!
   - user input ukuran array dan isinya, buat linked list dimulai dengan node head, kemudian loop tambahkan node-node baru ke node head tersebut
4. Buat fungsi untuk membalik nilai dari head ke tail! Contoh: 5->4->3->2->1 menjadi 1->2->3->4->5
   - hanya nilai saja, memory address (pointer node) tetap sama
   - buat temporary pointer node sebagai bantuan: prev, current, next dan loop dari head ke tail
5. Dari 3 fungsi delete (`remove_first, remove_last, remove_middle`), buat sebuah fungsi untuk menghapus node secara umum! (gabungkan 3 fungsi tersebut)
   - **`void remove_node(mynode head, int nilai)`** → asumsi: nilai di dalam linked list berbeda (tidak ada yang sama)
6. Buat program untuk menyimpan data students berisi **`int nim, char nama[50]`** secara dinamis!

# 1. Buat fungsi untuk menampilkan nilai dari linked list!

```c
#include<stdio.h>
#include<stdlib.h>

struct node{
    int value;
    struct node *next;
};

typedef struct node *mynode;

void display_list(mynode head){
    mynode tmp = head;
    while(tmp != NULL){
        printf("%d\n", tmp->value);
        tmp = tmp->next;
    }
    printf("selesai");
}
```

Fungsi untuk iterasi dari head sampai node terakhir dan **printf** value dari setiap node

```c
int main(){
    mynode head = NULL;
    mynode dua = NULL;
    mynode tiga = NULL;
    mynode empat = NULL;
    head = (mynode)malloc(sizeof(struct node));
    dua = (mynode)malloc(sizeof(struct node));
    tiga = (mynode)malloc(sizeof(struct node));
    empat = (mynode)malloc(sizeof(struct node));

    head->value = 10;
    head->next = dua;

    dua->value = 20;
    dua->next = tiga;

    tiga->value = 30;
    tiga->next = empat;

    empat->value = 40;
    empat->next = NULL;

    display_list(head);

    return 0;
}
```

**Output:**

```
10
20
30
40
selesai
```

# 2. Buat fungsi untuk menghitung jumlah node dalam sebuah linked list!

```c
struct node{
    int value;
    struct node *next;
};

typedef struct node *mynode;

void count_node(mynode head){
    int count = 0;
    mynode tmp = head;
    while(tmp != NULL){
        count = count + 1;
        tmp = tmp->next;
    }
    printf("Jumlah node adalah %d", count);
}
```

Fungsi untuk iterasi dari head sampai node terakhir dan menambahkan satu di variabel `count` di setiap iterasi

Statement ini bisa juga ditulis dengan: `count++;`

```c
int main(){
    mynode head = NULL;
    mynode dua = NULL;
    mynode tiga = NULL;
    mynode empat = NULL;
    head = (mynode)malloc(sizeof(struct node));
    dua = (mynode)malloc(sizeof(struct node));
    tiga = (mynode)malloc(sizeof(struct node));
    empat = (mynode)malloc(sizeof(struct node));

    head->value = 10;
    head->next = dua;

    dua->value = 20;
    dua->next = tiga;

    tiga->value = 30;
    tiga->next = empat;

    empat->value = 40;
    empat->next = NULL;

    count_node(head);

    return 0;
}
```

**Output:**
**Jumlah node adalah 4**

# 3. Buat program untuk mengkonversi dari array 1D ke linked list!

```c
#include<stdio.h>
#include<stdlib.h>

struct node{
    int value;
    struct node *next;
};
typedef struct node *mynode;

mynode createNode(int nilai){
    mynode p;
    p = (mynode)malloc(sizeof(struct node));
    p->value = nilai;
    p->next = NULL ;
    return(p);
}
```

```c
void push(mynode *head, int nilai){
    mynode new_node = createNode(nilai);

    if(*head == NULL){
        *head = new_node;
    }
    else{
        mynode tail = *head;
        while(tail->next != NULL)
            tail = tail->next;

        tail->next = new_node;
    }
}

void display_list(mynode head){
    mynode tmp = head;
    while(tmp != NULL){
        printf("%d\n", tmp->value);
        tmp = tmp->next;
    }
    printf("selesai");
}
```

- Jika list kosong (head = null), maka head adalah new_node
- Jika tidak kosong, maka `insert_tail()`

Di blok `else { ..}` adalah fungsi `insert_tail()`

**Kita gunakan fungsi – fungsi yang telah dibuat sebelumnya: `createNode()` dan `display_list()`**

**Panggil fungsi - fungsi tersebut di *main function***

```
int main(){
    int arr[] = {10, 20, 30, 40, 50};
    int n = sizeof(arr) / sizeof(int);

    mynode head = NULL;

    for (int i = 0; i < n; i++)
        push(&head, arr[i]);

    display_list(head);

    return 0;
}
```

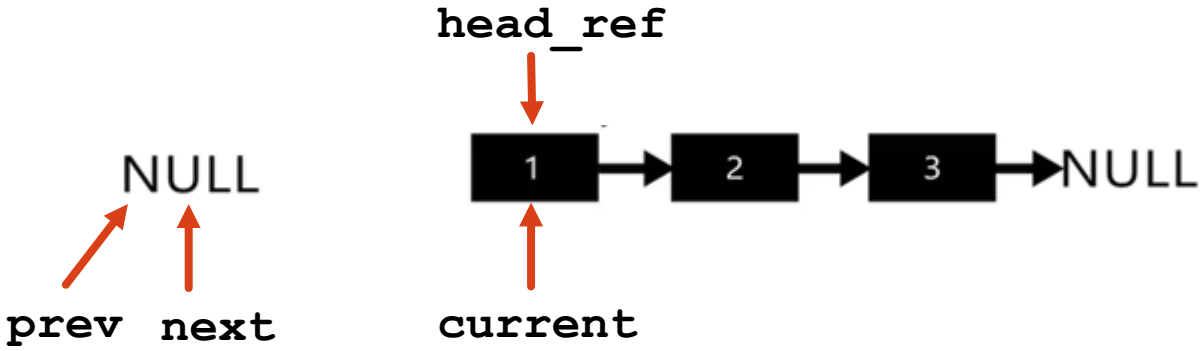
1. Tambahkan node – node ke list dari nilai isi array pertama, kedua, dst yaitu arr[1], arr[2], dst

2. Tampilkan list

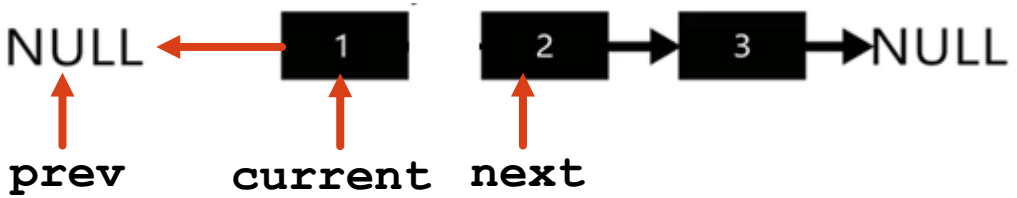
**Output:**

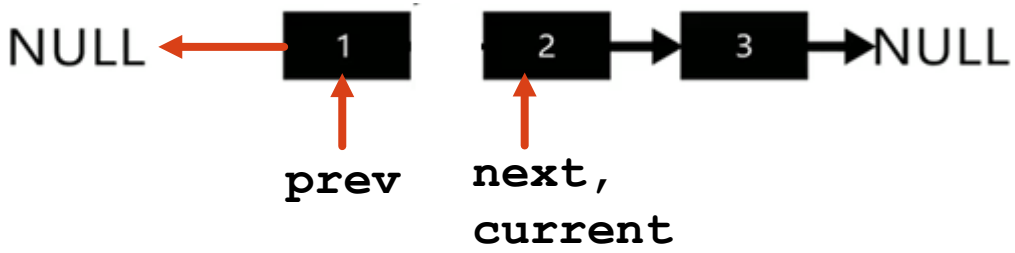
```
10
20
30
40
50
selesai
```

# 4. Buat fungsi untuk membalik nilai dari head ke tail!




- while loop pertama




- (dan seterusnya)

```
void reverse_list(mynode *head_ref){
    mynode prev = NULL;
    mynode current = *head_ref;
    mynode next;
    while (current != NULL)
    {
        next  = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    *head_ref = prev;
}
```

**Panggil fungsi tersebut di *main function* (kita pakai list di *main function* dari pertanyaan no 3 sebelumnya)**

```
int main(){
    int arr[] = {10, 20, 30, 40, 50};
    int n = sizeof(arr) / sizeof(int);

    mynode head = NULL;

    for (int i = 0; i < n; i++)
        push(&head, arr[i]);

    printf("List sebelumnya:\n");
    display_list(head);

    reverse_list(&head);

    printf("\nList setelahnya:\n");
    display_list(head);

    return 0;
}
```

```
Output:
List sebelumnya:
10
20
30
40
50
selesai
List setelahnya:
50
40
30
20
10
selesai
```

## 5. Buat sebuah fungsi untuk menghapus node secara umum, tidak tahu apakah itu head, tail, atau posisinya di tengah - tengah! (Asumsinya adalah nilai di list tidak ada yang sama)

Kasus ini sebenarnya dapat diselesaikan dengan mengembangkan fungsi `remove_middle()` yang sudah kita pelajari. Fungsi `remove_middle` akan menghapus node manapun yang memiliki value `nilai` ***selain node head***. Jadi, kita tinggal menambahkan kemungkinan jika nilai tersebut di node head.

```
void remove_node(mynode *head, int nilai){
    if((*head)->value == nilai)
    {
        remove_first(head);
    }
    else{
        mynode cursor = *head;
        while(cursor != NULL)
        {
            if(cursor->next->value == nilai)
                break;
            cursor = cursor->next;
        }

        if(cursor != NULL)
        {
            mynode tmp = cursor->next;
            cursor->next = tmp->next;
            tmp->next = NULL;
            free(tmp);
        }
    }
}
```

**Panggil `remove_first()` jika value dari head adalah nilai**

**`remove_middle()`**

# 6. Buat program untuk menyimpan data students berisi `int nim, char nama[50]` secara dinamis!

- **Seperti biasa, buat dulu structure untuk node student:**

```c
#include<stdio.h>
#include<stdlib.h>
#include <string.h>

struct student{
    int nim;
    char nama[50];
    struct student *next;
};
typedef struct student *mhsw;
```

- **Buat fungsi untuk menambah node student pada suatu list:**

```c
void tambah_student(mhsw *head, int mhs_nim, char mhs_nama[]){
    mhsw new_student;
    new_student = (mhsw)malloc(sizeof(struct student));
    new_student->nim = mhs_nim;
    strcpy(new_student->nama, mhs_nama);
    new_student->next = NULL;

    if(*head == NULL){
        *head = new_student;
    }
    else{
        mhsw tail = *head;
        while(tail->next != NULL)
            tail = tail->next;

        tail->next = new_student;
    }
}
```

Ini adalah fungsi `createNode()` dari slide sebelumnya

Ini adalah fungsi `push()` atau `insert_tail()` dari slide sebelumnya

- **Buat fungsi untuk membaca isi dari suatu list:**

```
void tampilkan_mahasiswa(mhsw head){
    printf("\nDaftar Mahasiswa :\n");
    mhsw tmp = head;
    while(tmp != NULL){
        printf("NIM: %d\n", tmp->nim);
        printf("Nama: %s\n\n", tmp->nama);
        tmp = tmp->next;
    }
    printf("Selesai");
}
```

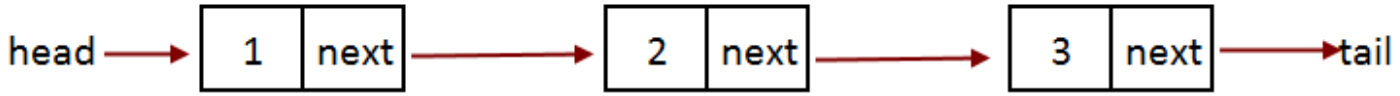
Ini adalah fungsi `display_list ()` dari slide sebelumnya

- **Panggil fungsi - fungsi tersebut di *main function***

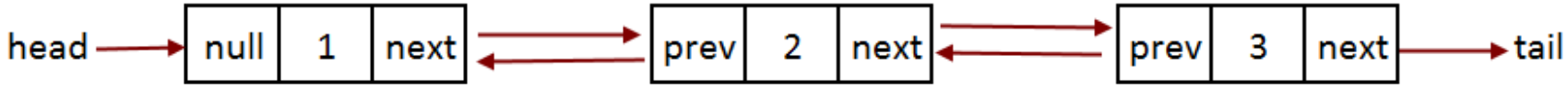
```
int main(){
    int n = 0;
    printf("Masukan jumlah mahasiswa :");
    scanf("%d", &n);

    int nim;
    char nama[50];
    mhsw list_student = NULL;
    for (int i = 0; i < n; i++){
        printf("\nNIM : ");
        scanf("%d", &nim);
        printf("\nNama : ");
        scanf("%s", &nama);
        tambah_student(&list_student, nim, nama);
    }

    tampilkan_mahasiswa(list_student);

    return 0;
}
```

# Double Linked List

- **Single Linked List**
  - Pointer **Next** menyimpan alamat dari node berikutnya
- **Double Linked List**
  - Pointer **Prev** dan **Next** menyimpan alamat dari node sebelumnya dan node berikutnya




Singly Linked List

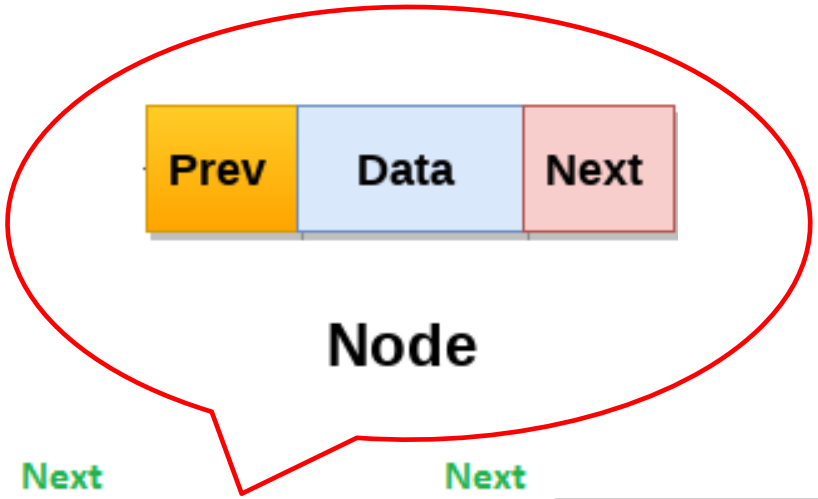


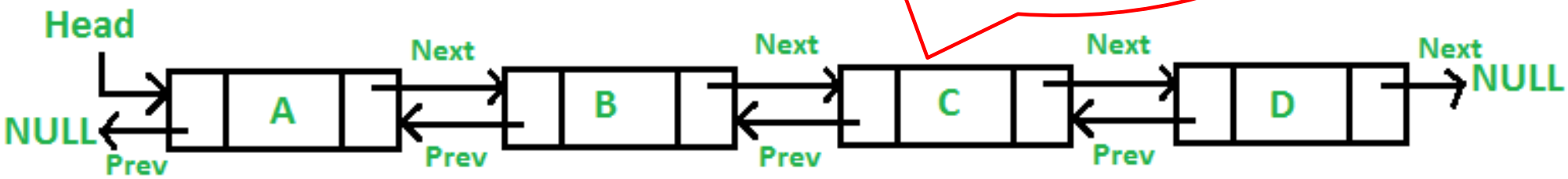

Doubly Linked List

# Double Linked List

- Setiap node terdiri dari 3 bagian:
  - **Data** yang berisi elemen data pada node tersebut
  - **Pointer Next** yang menunjuk ke node berikutnya
  - **Pointer Prev** yang menunjuk ke node sebelumnya

- Pointer **Prev** dari node head (elemen pertama) menunjuk NULL
- Pointer **Next** dari node tail (elemen terakhir) menunjuk NULL



# Deklarasi Double Linked List

- Sama seperti single linked list, setiap node akan berbentuk **`struct`** dan memiliki **dua** buah pointer bertipe **`struct`** yang sama yang berfungsi sebagai pointer **Prev** dan **Next**

```
struct node {
        int data;
        struct node* next;
        struct node* prev;
};
```

Menyimpan alamat node setelahnya

Menyimpan alamat node sebelumnya

***Pointer harus bertipe sama dengan nilai yang disimpan dalam alamat yang ditunjuk***

# Buat Node dengan Alokasi Memory Dinamis

```
struct node{
    int data;
    struct node *next;
    struct node *prev;
};

struct node* head = (struct node*)malloc(sizeof(struct node));
struct node* tail = (struct node*)malloc(sizeof(struct node));

head->data = 40;
head->next = tail;
head->prev = NULL;

tail->data = 50;
tail->next = NULL;
tail->prev = head;
```

**Pointer Prev** untuk elemen pertama menunjuk ke NULL

**Pointer Next** untuk elemen terakhir menunjuk ke NULL

# Create Double Linked List

**1. Deklarasikan structure node yang berisi data, pointer next, dan pointer prev**

```
struct node{
    int data;
    struct node *next;
    struct node *prev;
};
typedef struct node* mynode;
```

- Structure node tersebut kemudian bisa didefinisikan dengan **`typedef`**

  → **OPTIONAL** (boleh pakai, boleh tidak, digunakan untuk menyederhanakan)

*Untuk selanjutnya akan dipakai pada slide-slide selanjutnya sebagai global variable*

# Create Double Linked List

**2. Buat fungsi create node dan panggil fungsi tersebut untuk membentuk sebuah double linked list**

```
mynode createNode(int nilai){
    mynode temp;
    temp =(mynode)malloc(sizeof(struct node));
    temp->data = nilai;
    temp->prev = NULL;
    temp->next = NULL;

    return(temp);
}
```

Jika kita punya fungsi di atas, bagaimana memanggilnya di fungsi main?

# Menelusuri Double Linked List (Traversal)

- **Traversal** : membaca elemen-elemen dalam double linked list
- **Forward Traversal**
  - Mulai dari node pertama dan lewati semua node sampai node menunjuk NULL
- **Backward Traversal**
  - Mulai dari node terakhir dan lewati semua node sampai node menunjuk NULL

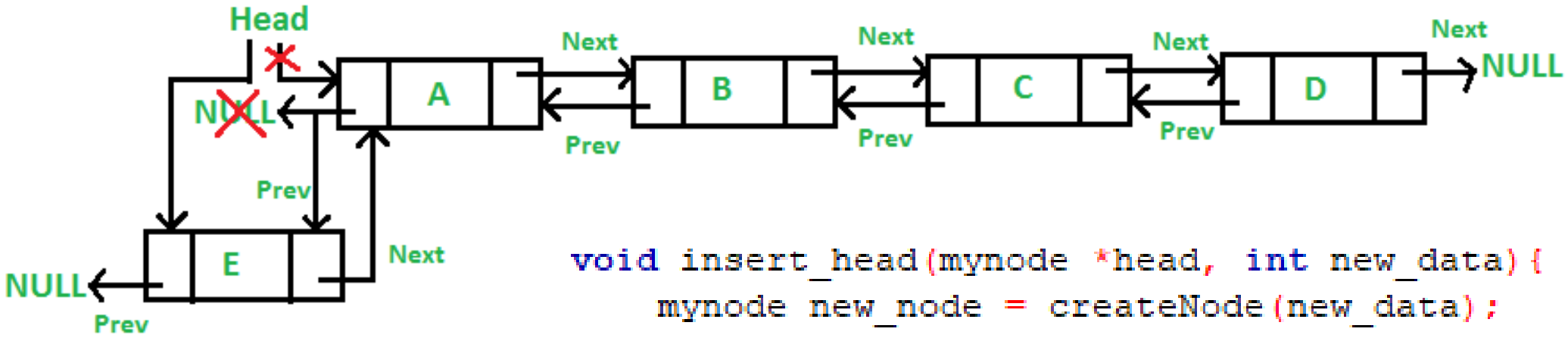
```
void traverse_beg(mynode head){
    mynode tmp = head;
    while(tmp != NULL){
        printf("%d\n", tmp->data);
        tmp = tmp->next;
    }
    printf("selesai");
}
```

```
void traverse_end(mynode tail){
    mynode tmp = tail;
    while(tmp != NULL){
        printf("%d\n", tmp->data);
        tmp = tmp->prev;
    }
    printf("selesai");
}
```

# Operasi pada Double Linked List

1. **Menambahkan node (insert)**
   - Insert sebagai node awal (head)
   - Insert sebagai node akhir (tail)
   - Insert setelah node tertentu
   - Insert sebelum node tertentu
2. **Menghapus node (delete)**
   - Delete node pertama (head)
   - Delete node terakhir (tail)
   - Delete pada node tertentu

# Insert sebagai node awal (head)

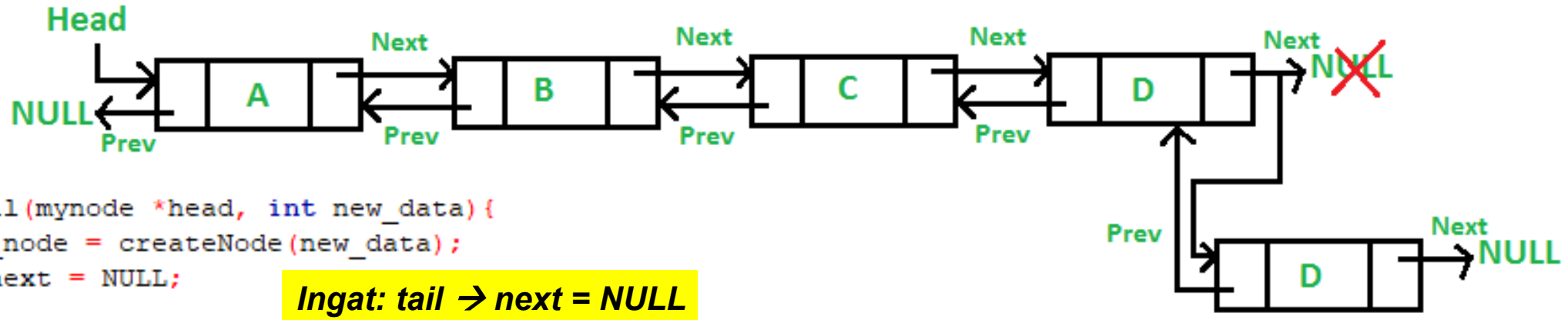



```
void insert_head(mynode *head, int new_data){
    mynode new_node = createNode(new_data);
    new_node->next = *head;
    new_node->prev = NULL;

    if(*head != NULL)
        (*head)->prev = new_node;

    *head = new_node;
}
```

***Jika DLL tidak kosong, hubungkan dengan node baru***

**Ingat: `mynode *head` sama saja dengan `struct node **head` → double pointer (menyimpan alamat memory pointer)**

# Insert sebagai node akhir (tail)

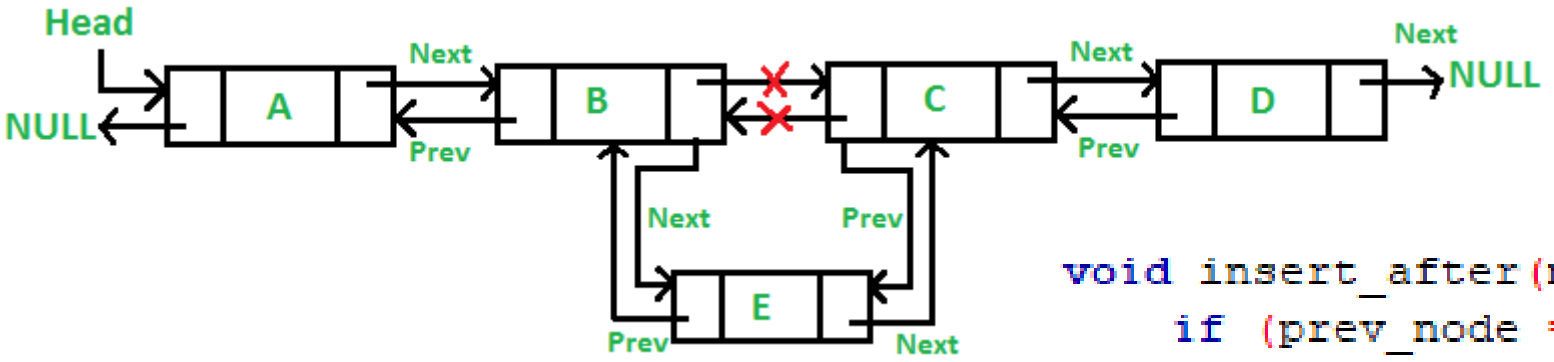



```
void insert_tail(mynode *head, int new_data){
    mynode new_node = createNode(new_data);
    new_node->next = NULL;

    if(*head == NULL){
        new_node->prev = NULL;
        *head = new_node;
        return;
    }

    mynode cursor = *head;
    while(cursor->next != NULL)
        cursor = cursor->next;

    cursor->next = new_node;
    new_node->prev = cursor;
}
```

***Ingat: tail → next = NULL***

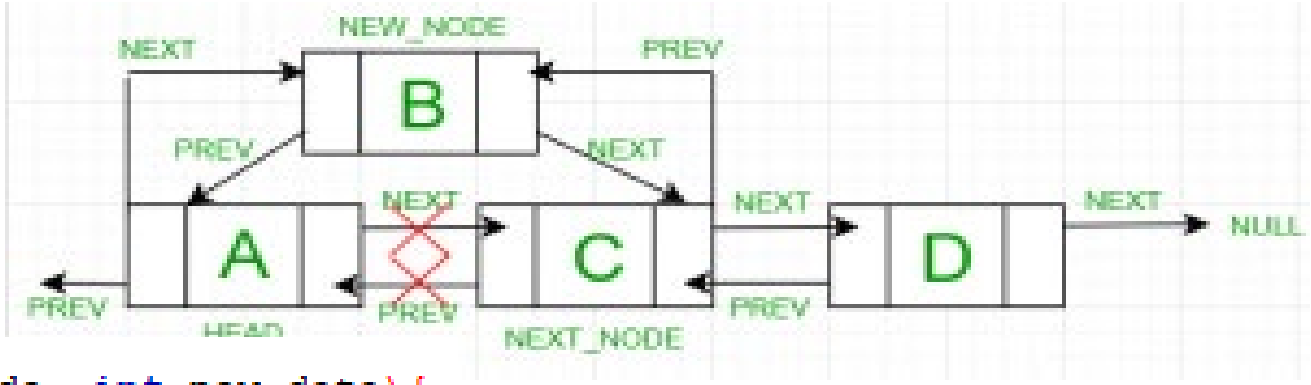
***Mencari node tail dari head***

# Insert setelah node tertentu




```
void insert_after(mynode prev_node, int new_data){
    if (prev_node == NULL) {
        printf("Previous Node tidak boleh NULL");
        return;
    }

    mynode new_node = createNode(new_data);
    new_node->next = prev_node->next;
    prev_node->next = new_node;
    new_node->prev = prev_node;

    if (new_node->next != NULL)
        new_node->next->prev = new_node;
}
```

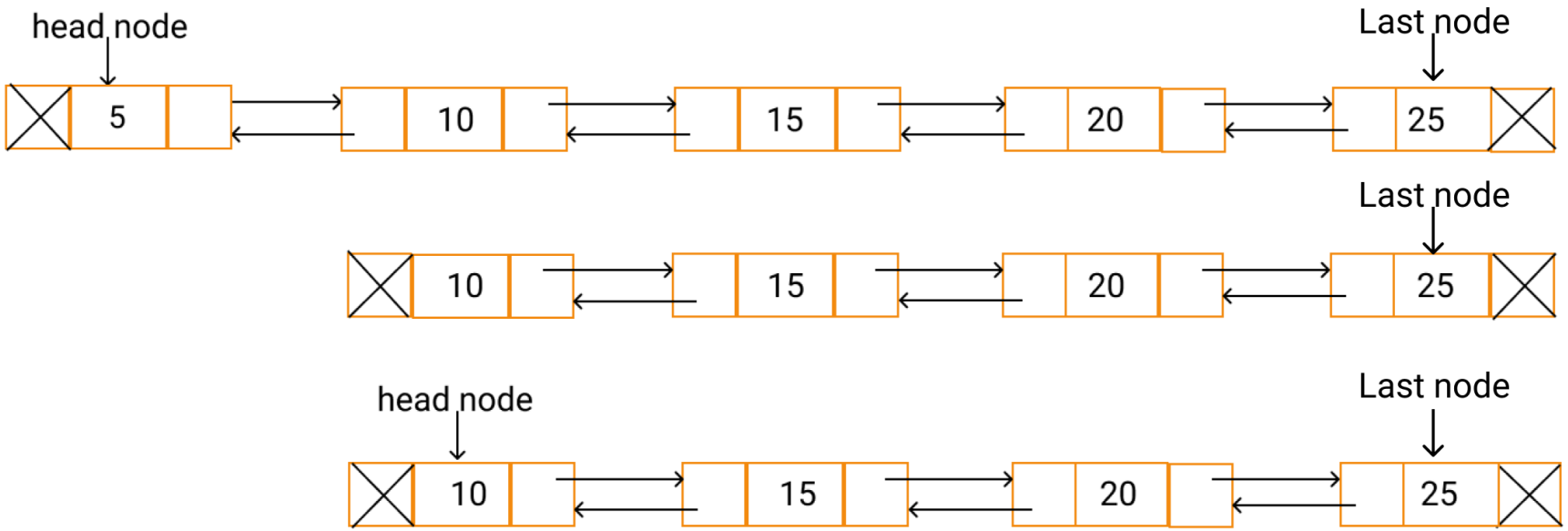
# Insert sebelum node tertentu




```
void insert_before(mynode next_node, int new_data){
    if (next_node == NULL) {
        printf("Next Node tidak boleh NULL");
        return;
    }

    mynode new_node = createNode(new_data);
    new_node->prev = next_node->prev;
    next_node->prev = new_node;
    new_node->next = next_node;

    if (new_node->prev != NULL)
        new_node->prev->next = new_node;
}
```

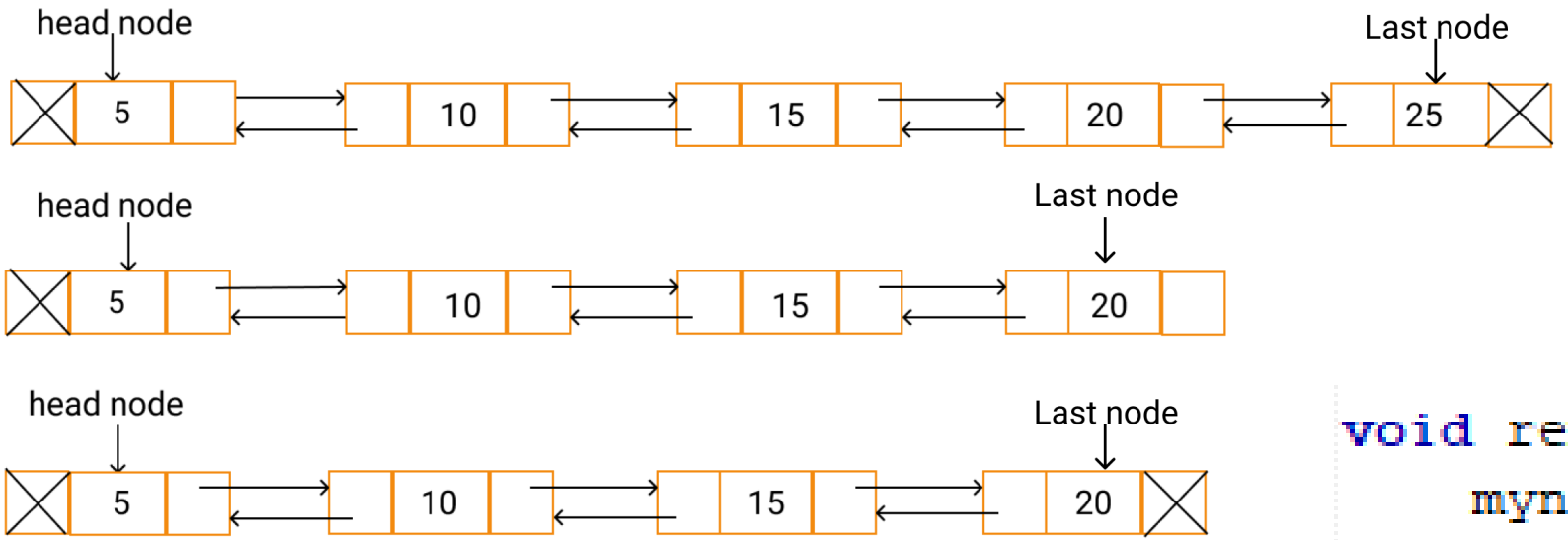
***Ubah pointer Next dari previous node ke node baru***

# Delete node pertama (head)



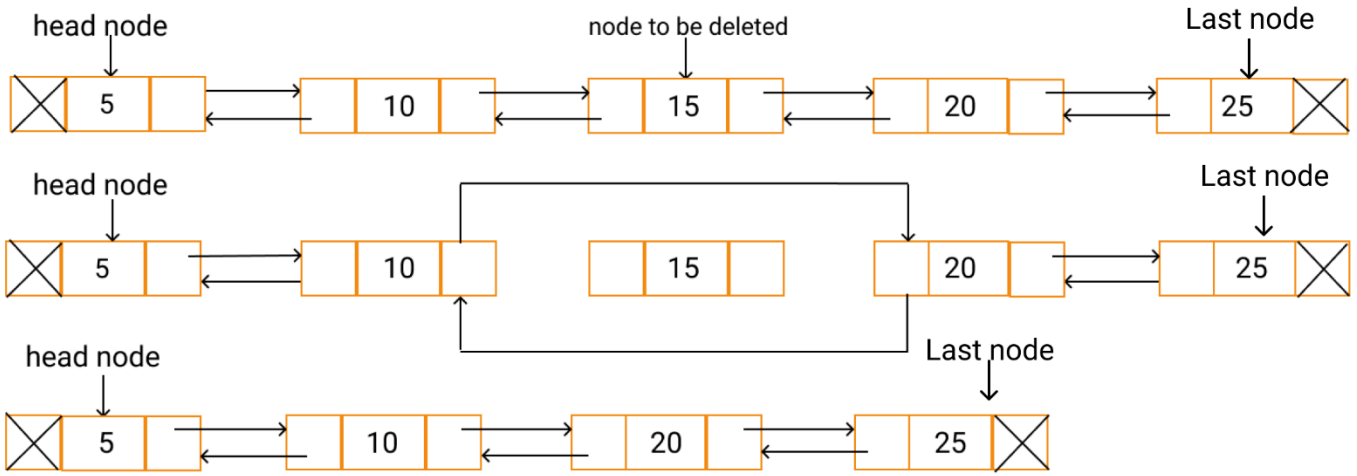

```
void remove_first(mynode *head){
     mynode temp = *head;
     *head = (*head)->next;
     (*head)->prev = NULL;

     free(temp);
}
```

# Delete node terakhir (tail)




```
void remove_end(mynode tail){
    mynode temp = tail;
    tail = tail->prev;

    if(tail != NULL)
        tail->next = NULL;

    free(temp);
}
```

# Delete pada node tertentu




```
void remove_middle(mynode head, int position){
    mynode temp = head;
    for(int i=1; i<position && temp!=NULL; i++)
    {
        temp = temp->next;
    }

    if(temp != NULL)
    {
        temp->prev->next = temp->next;
        temp->next->prev = temp->prev;

        free(temp);
    }
    else
    {
        printf("Posisi tidak valid!\n");
    }
}
```

# Aplikasi Linked List di Dunia Nyata

- Image viewer
- Previous & next page di web browser
- Playlist di music player

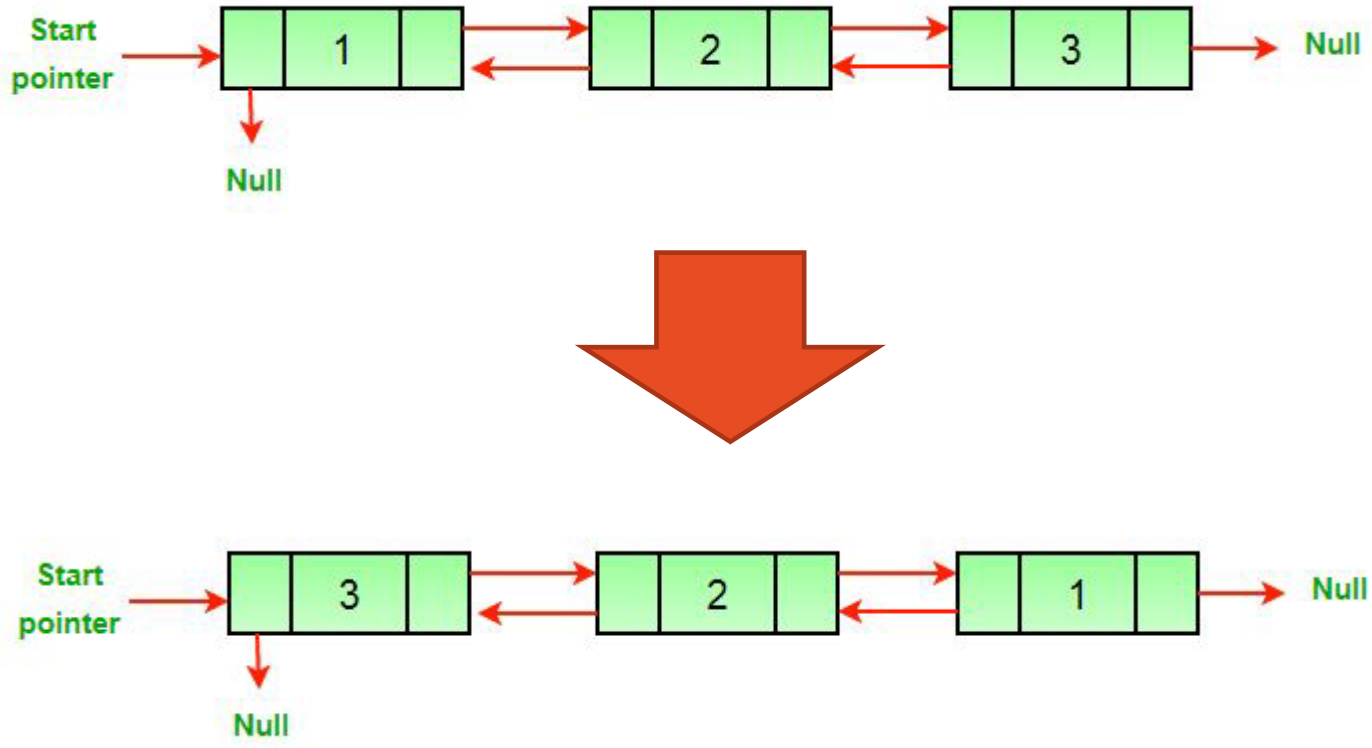
# Latihan 1 (Pemanasan!)

- Buat sebuah program untuk menampilkan output di bawah ini menggunakan double linked list!

```
Input the number of nodes : 3
Input data for node 1 : 2
Input data for node 2 : 5
Input data for node 3 : 8

Data entered in the list are :
node 1 : 2
node 2 : 5
node 3 : 8
Input data for the first node : 1

After insertion the new list are :
node 1 : 1
node 2 : 2
node 3 : 5
node 4 : 8
```

# Latihan 2: Membalik Double Linked List

- Bagaimana untuk membalik nilai-nilai dalam double linked list (tail ke head)?



# TERIMA KASIH