# Belajaritma

## Data Structure

## Sesi II : Linked List
## (PDF Example)

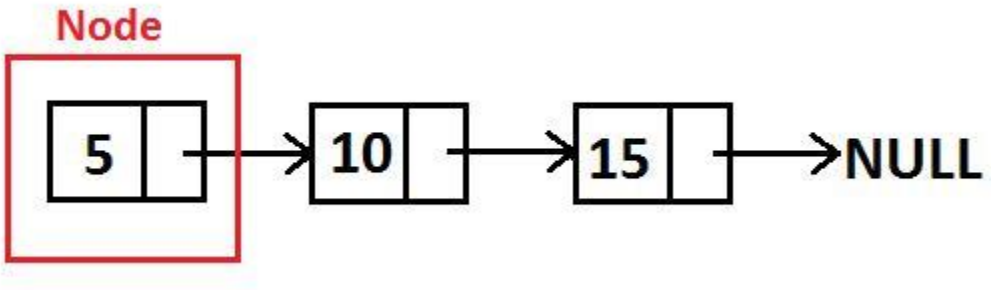
Himpunan Mahasiswa
HIMTI (Teknik Informatika)
BINUS
UNIVERSITY

Dalamrangkumanmateriini, pembahasan code menggunakan Bahasa Pemrograman C dengan *library malloc.h dan stdio.h*.

# III. Single Linked List

## A. Pengertian Single Linked List

Single linked list adalah linked list yang hanya memiliki pointer 1 arah, yaitu ke arah node setelahnya




## B. Operasi dalam Linked List

Linked list secara umum memiliki 2 jenis operasi, yaitu push dan pop

1. Push

Push adalah cara kita untuk memasukkan suatu data, biasa dalam bentuk struct ke dalam suatu linked list.

Push sendiri terdiri menjadi tiga, yaitu

a. Push head memungkinkan kita memasukkan suatu data ke paling depan,

contoh:

5 4 2 1 di push head menjadi -----> 1 2 4 5

Code untuk push head adalah:

```
void pushDepan(int x){
	curr = (struct data*)malloc(sizeof(struct data));
	curr->x = x;
	if(head==NULL){
		head = tail = curr;
	}else{
		curr->next = head;
		head = curr;
	}
	tail->next = NULL;
}
```

b. Push tail adalah memasukkan data dari belakang. Push tail berkebalikan dengan push head contoh:

1 2 3 4 5 di push tail menjadi 1 2 3 4 5

Code untuk push tail adalah sebagai berikut:

```
void pushBelakang(int x){
	curr = (struct data*)malloc(sizeof(struct data));
	curr->x = x;
	if(head==NULL){
		head = tail = curr;
	}else{
		tail->next = curr;
		tail = curr;
	}
	tail->next = NULL;
}
```

**c.** Push mid adalah memasukkan data dari bagian tengah. Kelebihan dari push mid adalah push mid juga menyortir data yang berbentuk angka, sehingga sudah terurut secara ascendng atau descending.

Contoh:

1 3 4 2 5  dengan push mid menjadi 1 2 3 4 5 atau 5 4 3 2 1

Berikut contoh code untuk push mid:

```
void pushTengah(int id){
	curr = (struct data*)malloc(sizeof(struct data));
	curr->id = id;
	curr->next = NULL;

	if(head == NULL){
		head = tail = curr;
	}
	else if(id < head->id){
		curr->next = head;
		head = curr;
	}
	else if(id >= tail->id){
		tail->next = curr;
		tail = curr;
	}
	else{
		struct data *temp = head;
		while(temp->next != NULL && temp->next->id < id){
			temp = temp->next;
		}
		curr->next = temp->next;
		temp->next = curr;
	}
	tail->next = NULL;
}
```

2. Pop

Berkebalikan dengan push, pop adalah cara kita untuk melakukan penghapusan data di dalam linked list.

Mirip dengan push, pop juga terbagi menjadi 3, yaitu:

a. Pop head

Pop head, mirip dengan push head adalah cara melakukan penghapusan data dari depan

Contoh

5 4 3 2 1 di pop head menjadi  4 3 2 1

Code untuk Pop head adalah sebagai berikut

```
void popDepan(){
	if(head!=NULL){
		if(head==tail){
			free(head);
			head=NULL;
		}else{
			curr = head;
			head = head->next;
			free(curr);
		}
	}
}
```

b. Pop Tail

Pop tail, kebalikan dari pop head adalah cara melakukan penghapusan data dari belakang, contohnya adalah

5 4 3 2 1 di pop tail menjadi 5 4 3 2

Berikut adalah code untuk pop tail:

```
void popBelakang(){
	if(head!=NULL){
		if(head==tail){
			free(head);
			head=NULL;
		}else{
			curr = head;
			while(curr->next!=tail){
				curr = curr->next;
			}
			free(tail);
			tail = curr;
			tail->next = NULL;
		}
	}
}
```

c. Pop all

Pop all adalah cara kita menghapus seluruh data di linked list

Contoh: 5 4 2 6 7 dilakukan popAll) menjadi Null

Berikut adalah code untuk pop All:

```
void popAll(){
	curr = head;
	while(curr != NULL){
		head = head->next;
		curr->next = NULL;
		free(curr);
		curr = head;
	}
}
```

d. Pop Mid

Pop Mid adalah cara kita menghapus seluruh data di linked list

Contoh: 5 4 2 6 7 dilakukan popmid(2) menjadi 5 4 6 7

Berikut adalah code untuk pop Mid:

```
struct node* popMid(int value){
	if(head == NULL){
		return NULL;
	}
	else if(head == tail){
		struct node* curr = head;
		head = tail = NULL;
		return curr;
	}
	else {
		if(head->value == value){
			popHead();
		}
		else if(tail->value == value){
			popTail();
		}
		else{
			struct node* curr = head;
			while(curr->next != NULL && curr->next->value != value){
				curr = curr->next;
			}
			struct node* temp = curr->next;
			curr->next = temp->next;
			return temp;
		}	}	}
```

Himpunan Mahasiswa
HIMTI (Teknik Informatika)
BINUS
UNIVERSITY

# III. Circular Single Linked List

Circular linked list adalah link list dimana head dan tail terhubung satu sama lain sehingga membentuk lingkaran.

Catatan dalam operasi push:

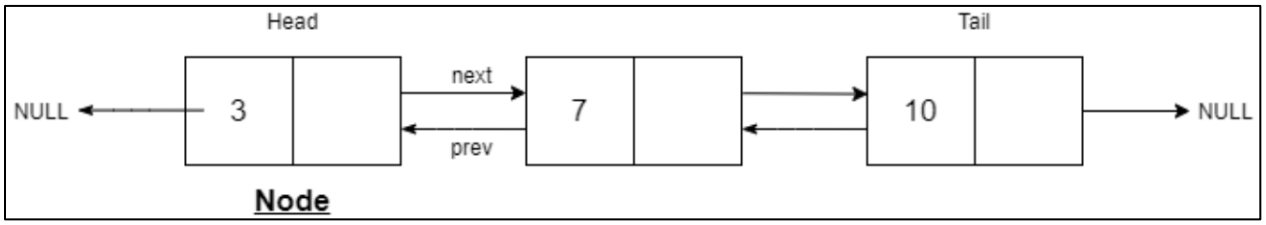
Untuk Push tail, pada bagian tail->next= Null, diubah menjadi tail->next=head

Hal ini diakarenakan kita ingin membuat tail memiliki pointer yang menunjuk kepada head

# III. Doubly Linked List

### A. Pengertian Doubly Linked List

Doubly Linked List adalah linked list dengan pointer penunjuk 2 arah, yakni ke arah node sebelum (previous/prev) dan node sesudah (next).




### B. Perbedaan Doubly Linked List dengan Single Linked List :

Pada Single Linked List, masing-masing node hanya memiliki satu pointer saja yaitu pointer next sehingga node hanya dapat bergerak satu arah saja, yaitu maju atau kekanan. Untuk mengatasi kelemahan ini maka dibuat dengan Doubly Linked List yang mempunyai dua pointer penujuk 2 arah ,yakni ke arah node sebelum (previous/prev) dan node sesudah (next).

### C. Operasi – Operasi pada Doubly Linked List:

Beberapaoperasi yang biasanya ada di dalam sebuah doubly linked list pada dasarnya sama dengan yang ada di dalam *single linked list*, yakni:

**1. Push**

Push merupakan sebuahoperasi insert node. Dalam linked list terdapat 3 kemungkinan insert data, yaitu insert node paling depan (pushHead), insert node paling belakang (pushTail), dan insert node bagiantengah (pushMid).

➔Operasi push Head berarti data atau node yang paling baruakandimasukkan pada bagian head.
➔Operasi push Tail berarti data atau node yang paling baru akan dimasukkan pada bagian tail.
➔Operasi push Mid berarti data atau node yang paling baru akan dimasukkan pada bagian tengah sesuai dengan kondisi yang diberikan.

Representasinyaadalahsebagaiberikut:

➔pushHead:3, 5, 10 maka hasilnya adalah10, 5, 3
➔pushTail:3, 5, 10 maka hasilnya adalah3, 5, 10
➔pushMid : 5, 3, 2, 6, 4 maka hasilnya adalah 2, 3, 4, 5, 6 dengan kondisi yang diberikan adalah push mid data dengan sorting ascending.

Pada linked list dapat ditulis seperti ini:

1. Buat struct node yang berisi sebuah value dan dua pointer next dan prev. Inisialisasinilaidari head dan tail adalah NULL.

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

struct node{
    int value;
    struct node* next, *prev;
}*head = NULL, *tail = NULL;
```

2. Buat function create New Node untuk menginisialiasi setiap node baru yang akan dibuat.

```c
struct node* createNewNode(int value){
    struct node* newNode = (struct node*)malloc(sizeof(struct node));
    newNode->value = value;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}
```

➔Operasi pushHead dapat dilakukan denganpotongan code berikut ini.

```
void pushHead(int value){
    struct node* newNode = createNewNode(value);
    if(head == NULL){
        head = tail = newNode;
    }
    else{
        head->prev = newNode;
        newNode->next = head;
        newNode->prev = NULL;
        head = newNode;
    }
}
```

➔Operasi pushTail dapat dilakukan dengan potongan code berikut ini.

```
void pushTail(int value){
    struct node* newNode = createNewNode(value);
    if(head == NULL){
        head = tail = newNode;
    }
    else{
        tail->next = newNode;
        newNode->next = NULL;
        newNode->prev = tail;
        tail = newNode;
    }
}
```

➔Operasi pushMid dapat dilakukan dengan potongan code berikut ini.

```
void pushMid(int value){
    struct node* newNode = createNewNode(value);
    if(head == NULL){
        head = tail = newNode;
    }
    else if(head->value > value){
        pushHead(value);
    }
    else if(tail->value < value){
        pushTail(value);
    }
    else{
        struct node *curr = head;
        while(curr->next != NULL && curr->next->value < value){
            curr = curr->next;
        }
        newNode->next = curr->next;
        curr->next = newNode;
        newNode->prev = curr;
    }
}
```

**2. Pop**

Pop merupakan sebuahoperasi delete node. Dalam linked list terdapat 3 kemungkinandelete data, yaitu delete node paling depan (popHead), delete node paling belakang(popTail), dan delete node bagian tengah (popMid).

➔Operasi popHead berarti data atau node pada head akan di delete.
➔Operasi pushTail berarti data atau node pada tail akan di delete.
➔Operasi pushMid berarti data atau node dengan key value yang diinput akan di delete.

➔Operasi popHead dapat dilakukan dengan potongan code berikut ini.

```
void popHead(){
    if(head == NULL){
        return;
    }
    else{
        struct node *curr = head;
        if(head == tail){
            head = tail = NULL;
        }
        else{
            head = head->next;
            head->next->prev = NULL;
        }
        free(curr);
    }
}
```

➔Operasi pop Tail dapat dilakukan dengan potongan code berikutini.

```
void popTail(){
    if(head == NULL){
        return;
    }
    else{
        struct node *curr = head;
        while(curr->next != tail){
            curr = curr->next;
        }
        struct node *temp = curr->next;
        curr->next = NULL;
        tail = curr;
        free(temp);
    }
}
```

➔Operasi popMid dapat dilakukan dengan potongan code berikut ini.

```
void popMid(int value){
    if(head == NULL){
        return;
    }
    else if(head->value == value){
        popHead();
    }
    else if(tail->value == value){
        popTail();
    }
    else{
        struct node *temp = head;
        while((temp) && temp->value != value){
            temp = temp->next;
        }
        if(!temp) return;
        else{
            struct node *curr = temp->next;
            temp->prev->next = curr;
            curr->prev = temp->prev;
            free(temp);
        }
    }
}
```

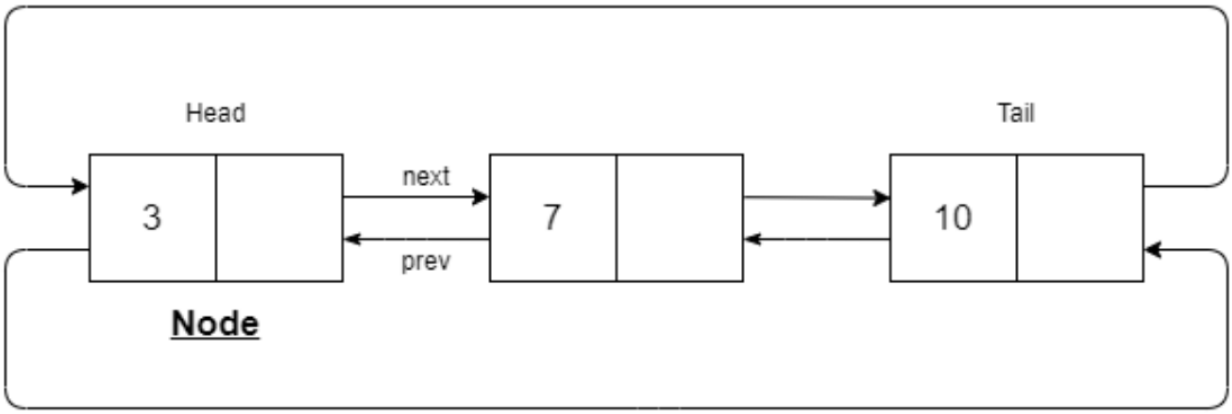
Untuk melihat data atau print data *linked list*, dapat dilakukan denganpotongan code berikut ini:

```
void printAll(){
    if(head == NULL){
        return;
    }
    struct node *curr = head;
    while(curr != NULL){
        if(curr == head && curr == tail){
            printf("%d (h)(t)", curr->value);
        }
        else if(curr == head){
            printf("%d(h) -> ", curr->value);
        }
        else if(curr == tail){
            printf("%d(t)", curr->value);
        }
        else{
            printf("%d -> ", curr->value);
        }
        curr = curr->next;
    }
    printf("\n");
}
```

# III. Doubly Linked List

**A.Pengertian Circular Doubly Linked List**

Circular Doubly Linked List adalah Doubly Linked List yang pointer next dan prev nya menunjuk pada dirinya sendiri. Jika sebuah Circular Doubly Linked List terdiri dari beberapa node, maka pointer next dan prev pada node terakhir akan menunjukke node terdepannya atau head. Dalam Circular Doubly Linked List, semua node berhubungan dan membentuk suatu bentuk lingkaran sehingga tidak ada NULL di akhir. Sedangkan pada Doubly Linked List, pointer next dan prev akan selalu bernilai NULL bila pointer next dan prev tidak menunjuk ke node manapun.




**B. Operasi – Operasi pada Circular Doubly Linked List:**

Beberapaoperasi yang biasanya ada di dalam sebuah circular doubly linked list pada dasarnya sama dengan yang ada di dalam *doubly linked list*, yakni:

**1. Push**

Push merupakan sebuah operasi insert node. Dalam linked list terdapat 3kemungkinan insert data, yaitu insert node paling depan (pushHead), insert node paling belakang (pushTail), dan insert node bagian tengah (pushMid).

➔Operasi pushHead berarti data atau node yang paling baru akan dimasukkan pada bagian head.
➔Operasi pushTail berarti data atau node yang paling baru akan dimasukkan pada bagian tail.
➔Operasi pushMid berarti data atau node yang paling baru akan dimasukkan pada bagian tengah sesuai dengan kondisi yang diberikan.

Representasinyaadalahsebagaiberikut:

➔pushHead:3, 5, 10maka hasilnya adalah10, 5, 3
➔pushTail:3, 5, 10maka hasilnya adalah3, 5, 10
➔pushMid : 5, 3, 2, 6, 4maka hasilnya adalah 2, 3, 4, 5, 6 dengan kondisi yang diberikan adalah push mid data dengan sorting ascending.

Pada linked list dapat ditulis seperti ini:

1. Buat struct node yang berisi sebuah value dan dua pointer next dan prev. Inisialisasi nilai dari head dan tail adalah NULL.

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

struct node{
    int value;
    struct node* next, *prev;
}*head = NULL, *tail = NULL;
```

2. Buat function create New Node untuk menginisialiasi setiap node baru yang akan dibuat.

```c
struct node* createNewNode(int value){
    struct node* newNode = (struct node*)malloc(sizeof(struct node));
    newNode->value = value;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}
```

3. Agar membentuk cycle, setiap push dan pop dapatditambahkan :

```c
head->prev = tail;
tail->next = head;
```

➔Operasi pushHead dapat dilakukan dengan potongan code berikut ini.

```
void pushHead(int value){
    struct order* newNode = createNewNode(value);
    if(head == NULL){
        head = tail = newNode;
    }
    else{
        newNode->next = head;
        head->prev = newNode;
        head = newNode;
    }
    head->prev = tail;
    tail->next = head;
}
```

➔Operasi pushTail dapat dilakukan dengan potongan code berikut ini.

```
void pushTail(int value){
    struct order* newNode = createNewNode(value);
    if(head == NULL){
        head = tail = newNode;
    }
    else{
        tail->next = newNode;
        newNode->prev = tail;
        tail = newNode;
    }
    head->prev = tail;
    tail->next = head;
}
```

➔Operasi pushMid dapat dilakukandengan potongan code berikut ini.

```
void pushMid(int value){
    struct order* newNode = createNewNode(value);
    if(head == NULL){
        head = tail = newNode;
        head->prev = tail;
        tail->next = head;
    }
    else if(head->value < value){
        pushHead(value);
    }
    else if(tail->value > value){
        pushTail(value);
    }
    else{
        struct order* curr = head;
        while(value <= curr->value){
            curr = curr->next;
        }
        newNode->next = curr;
        newNode->prev = curr->prev;
        curr->prev->next = newNode;
        curr->prev = newNode;

        head->prev = tail;
        tail->next = head;
    }
}
```

**2. Pop**

Popmerupakansebuahoperasi delete node. Dalam linked list terdapat3kemungkinandelete data, yaitudeletenode paling depan (popHead), deletenode paling belakang(popTail), dan deletenode bagiantengah (popMid).

➔Operasi popHead berarti data atau node pada head akan di delete.
➔Operasi pushTail berarti data atau node pada tail akan di delete.
➔Operasi pushMid berarti data atau node dengan key value yang diinput akan di delete.

➔Operasi pop Head dapat dilakukan dengan potongan code berikut ini.

```
void popHead(){
    struct order* curr = head;
    if(head == NULL){
        return;
    }
    else if(head == tail){
        head = tail = NULL;
    }
    else{
        head = head->next;
        head->prev = NULL;

        head->prev = tail;
        tail->next = head;
    }
    free(curr);
}
```

➔Operasi popTail dapat dilakukan dengan potongan code berikut ini.

```
void popTail(){
    struct order* curr = tail;
    if(head == NULL){
        return;
    }
    else if(head == tail){
        head = tail = NULL;
    }
    else{
        tail = tail->prev;
        tail->next = NULL;

        head->prev = tail;
        tail->next = head;
    }
    free(curr);
}
```

➔Operasi popMid dapat dilakukan dengan potongan code berikut ini.

```
void popMid(int value){
    if(head == NULL){
        return;
    }
    else if(head->value == value){
        popHead();
    }
    else if(tail->value == value){
        popTail();
    }
    else{
        struct order *temp = head;
        while(temp != tail){
            if(temp->value != value){
                temp = temp->next;
            }
            else{
                break;
            }
        }
        if(temp == NULL){
            return;
        }
        temp->next->prev = temp->prev;
        temp->prev->next = temp->next;
        free(temp);
        head->prev = tail;
        tail->next = head;
    }
}
```

Untuk melihat data atau print data *linked list*, dapat dilakukan dengan potongan code berikut ini:

```
void printAll(){
    struct order* temp = head;
    if(head == NULL){
        printf("No data!\n");
        return;
    }
    while(temp != tail){
        if(temp == head && temp == tail){
            printf("%d (h)(t)\n", temp->value);
        }
        else if(temp == head){
            printf("%d (h)\n", temp->value);
        }
        else if(temp == tail){
            printf("%d (t)\n", temp->value);
        }
        else{
            printf("%d\n", temp->value);
        }
        temp = temp->next;
    }
```

```
    if(temp == head && temp == tail){
        printf("%d (h)(t)\n", temp->value);
    }
    else if(temp == head){
        printf("%d (h)\n", temp->value);
    }
    else if(temp == tail){
        printf("%d (t)\n", temp->value);
    }
    else{
        printf("%d\n", temp->value);
    }
    puts("");
}
```

Referensi: https://socs.binus.ac.id/2017/03/15/single-linked-list/