MODUL 8

DOUBLE LINKED LIST

DASAR TEORI

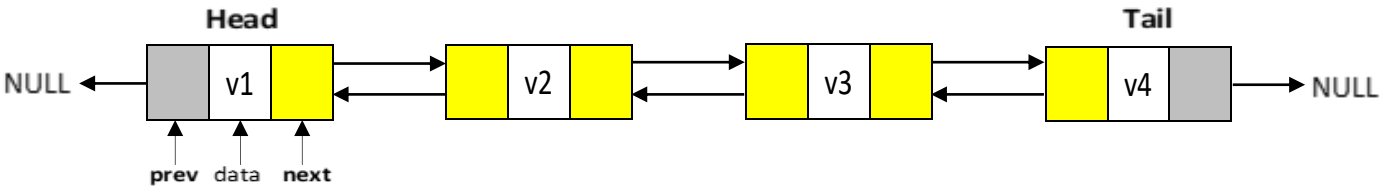
Double Linked List

Double Linked List (Senarai Berantai Ganda) merupakan jenis linked list yang hampir sama dengan Single Linked List. Yang membedakan adalah penggunaan pointernya. Double linked list menggunakan dua buah pointer yaitu **next** dan **prev**. Pointer next digunakan untuk menunjuk node/simpul **selanjutnya** dan pointer prev digunakan untuk menunjuk node/simpul **sebelumnya**. Keberadaan dua pointer menjadikan double linked list menjadi lebih fleksibel dibandingkan single linked list.

Seperti halnya single list, double linked list juga memiliki dua jenis yaitu Non-Circular dan Circular.

Double Linked List Non-Circular (DLLNC)

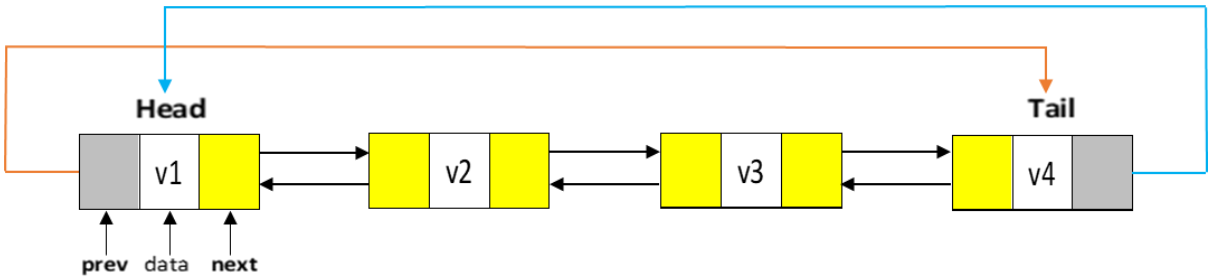
- Pointer prev pada node **head** menunjuk ke NULL, menandakan bahwa node head sebagai **node awal**.
- Pointer next pada node **tail** menunjuk ke NULL, menandakan bahwa node tail sebagai **node akhir**.



Gambar 1 Ilustrasi Double Linked List Non-Circular

Double Linked List Circular (DLLC)

- Pointer prev pada node **head** menunjuk ke node **tail**.
- Pointer next pada node **tail** menunjuk ke node **head**.



Gambar 2 Ilustrasi Double Linked List Circular

Operasi Dasar Double Linked List

- **Deklarasi**

  Pendeklarasian DLL menggunakan tipe data struct yang terdiri atas 3 buah field: **data**, pointer **next** dan **prev**.

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

- **Inisialisasi**

  Diperlukan untuk memberikan nilai awal node pada list kosong dengan memberikan nilai NULL pada node head dan tail.

```
void init()
{
    head = NULL;
```

```
    tail = NULL;
}
```

- **Create**

  Operasi untuk membuat node baru.

```
void create(int input)
{
    baru = new Node;
    baru->data = input;
    baru->next = NULL;
    baru->prev = NULL;
}
```

- **isEmpty**

  Operasi untuk memeriksa apakah suatu linked list masih kosong.

```
int isEmpty()
{
    if (head == NULL && tail == NULL)
        return 1;   //true
    else
        return 0;   //false
}
```

- **Insert**

  Operasi untuk menambahkan satu node ke dalam linked list.

```
void insertDepan(int input)
{
    create(input);

    if (isEmpty() == 1){
        head = baru;
        tail = baru;
```

```
    }
    else{
        baru->next = head;
        head->prev = baru;
        head = baru;
    }
}
```

- **Delete**

  Operasi untuk menghapus node pada linked list.

```
void hapusDepan()
{
    if (isEmpty() == 0){
        hapus = head;
        head = head->next;
        head->prev = NULL;
        delete hapus;
    }
    else{
        cout << " List masih kosong!"
<< endl;
    }
}
```

- **Clear**

  Operasi untuk menghapus atau mengosongkan seluruh data pada linked list.

```
void clearList()
{
    bantu = head;

    while (bantu != NULL){
        hapus = bantu;
```

```
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << " List berhasil terhapus!"
<< endl;
}
```

- **Display**

  Operasi untuk menampilkan seluruh data linked list.

```
void tampil()
{
    bantu = head;

    if (isEmpty() == 0){
        while (bantu != NULL){
            cout << ends << bantu->data << ends;
            bantu = bantu->next;
        }
        cout << endl;
    }
    else{
        cout << " List masih kosong!" << endl;
    }
}
```

GUIDED

Program Double Linked List Non-Circular (DLLNC)

```
#include <iostream>

using namespace std;

///PROGRAM DOUBLE LINKED LIST NON-CIRCULAR (DLLNC)

//Deklarasi DLLNC
struct Node
{
    int data;
    Node *next;
    Node *prev;
};
Node *head, *tail, *baru, *bantu, *bantu2, *hapus;

//Inisialisasi Node
void init()
{
    head = NULL;
    tail = NULL;
}

//IsEmpty
int isEmpty()
{
    if (head == NULL && tail == NULL)
        return 1;   //true
    else
        return 0;   //false
}

//Buat Node Baru
void create(int input)
{
    baru = new Node;
    baru->data = input;
    baru->next = NULL;
    baru->prev = NULL;
}

//Tambah Depan
void insertDepan(int input)
{
    create(input);
```

```
    if (isEmpty() == 1){
        head = baru;
        tail = baru;
    }
    else{
        baru->next = head;
        head->prev = baru;
        head = baru;
    }
}

//Tambah Belakang
void insertBelakang(int input)
{
    create(input);

    if (isEmpty() == 1){
        head = baru;
        tail = baru;
    }
    else{
        tail->next = baru;
        baru->prev = tail;
        tail = baru;
    }
}

//Hitung List
int countList()
{
    int counter = 0;
    bantu = head;

    while(bantu != NULL){
        counter++;
        bantu = bantu->next;
    }

    return counter;
}

//Tambah Tengah
void insertTengah(int input, int posisi)
{
    create(input);
```

```
    if (posisi < 1 || posisi > countList()){
            cout << " Posisi di luar jangkauan!" << endl;
    }
    else if ( posisi == 1){
        cout << " Posisi bukan posisi tengah!" << endl;
    }
    else{
        bantu = head;
        int counter = 1;

        while (counter < posisi-1){
            bantu = bantu->next;
            counter++;
        }

        bantu2 = bantu->next;
        baru->prev = bantu;
        baru->next = bantu2;
        bantu->next = baru;
        bantu2->prev = baru;
    }
}

//Hapus Depan
void hapusDepan()
{
    if (isEmpty() == 0){
        hapus = head;
        head = head->next;
        head->prev = NULL;
        delete hapus;
    }
    else{
        cout << " List masih kosong!" << endl;
    }
}

//Hapus Belakang
void hapusBelakang()
{
    if (isEmpty() == 0){
        hapus = tail;
        tail = tail->prev;
        tail->next = NULL;
        delete hapus;
    }
    else{
```

```
        cout << " List masih kosong!" << endl;
    }
}

//Hapus Tengah
void hapusTengah(int posisi)
{
    if (isEmpty() == 0){
        if (posisi < 1 || posisi > countList()){
            cout << " Posisi di luar jangkauan!" << endl;
        }
        else if (posisi == 1 || posisi == countList()){
            cout << " Posisi bukan posisi tengah!" << endl;
        }
        else{
            int counter = 1;
            bantu = head;

            while (counter < posisi-1){
                bantu = bantu->next;
                counter++;
            }

            hapus = bantu->next;
            bantu2 = hapus->next;
            bantu->next = bantu2;
            bantu2->prev = bantu;
            delete hapus;
        }
    }
    else{
        cout << " List masih kosong!" << endl;
    }
}

//Ubah Depan
void ubahDepan(int input)
{
    if (isEmpty() == 0){
        head->data = input;
    }
    else{
        cout << " List masih kosong!" << endl;
    }
}

//Ubah Belakang
```

```
void ubahBelakang(int input)
{
    if (isEmpty() == 0){
        tail->data = input;
    }
    else{
        cout << " List masih kosong!" << endl;
    }
}

//Ubah Tengah
void ubahTengah(int input, int posisi)
{
    if (isEmpty() == 0){
        if (posisi < 1 || posisi > countList()){
            cout << " Posisi di luar jangkauan!" << endl;
        }
        else if (posisi == 1 || posisi == countList()){
            cout << " Posisi bukan posisi tengah!" << endl;
        }
        else{
            bantu = head;
            int counter = 1;
            while (counter < posisi){
                bantu = bantu->next;
                counter++;
            }
            bantu->data = input;
        }
    }
    else{
        cout << " List masih kosong!" << endl;
    }
}

//Hapus List
void clearList()
{
    bantu = head;

    while (bantu != NULL){
        hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << " List berhasil terhapus!" << endl;
```

```
}

//Tampilkan List
void tampil()
{
    bantu = head;

    if (isEmpty() == 0){
        while (bantu != NULL){
            cout << ends << bantu->data << ends;
            bantu = bantu->next;
        }
        cout << endl;
    }
    else{
        cout << " List masih kosong!" << endl;
    }
}

int main()
{
    init();
    insertDepan(3);
    tampil();
    insertBelakang(5);
    tampil();
    insertDepan(2);
    tampil();
    insertDepan(1);
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, 2);
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1);
    tampil();
    ubahBelakang(8);
    tampil();
    ubahTengah(11, 2);
    tampil();

    return 0;
}
```

Program Double Linked List Circular (DLLC)

```
#include <iostream>

using namespace std;

///PROGRAM DOUBLE LINKED LIST CIRCULAR

//Deklarasi DLLC
struct Node
{
    string data;
    Node *next;
    Node *prev;
};

//Deklarasi Node
Node *head, *tail, *baru, *bantu, *bantu2, *hapus;

//Inisialisasi Node
void init()
{
    head = NULL;
    tail = head;
}

//isEmpty
int isEmpty()
{
    if (head == NULL && tail == NULL)
        return 1; //true
    else
        return 0; //false
}

//Buat Node Baru
void create(string input)
{
    baru = new Node;
    baru->data = input;
    baru->next = NULL;
    baru->prev = NULL;
}

//Tambah Depan
void insertDepan(string input)
```

```
{
    create(input);

    if (isEmpty() == 1){
        head = baru;
        tail = head;
        head->prev = tail;
        tail->next = head;
    }
    else{
        baru->next = head;
        head->prev = baru;
        head = baru;
        head->prev = tail;
        tail->next = head;
    }
}

//Tambah Belakang
void insertBelakang(string input)
{
    create(input);

    if (isEmpty() == 1){
        head = baru;
        tail = head;
        head->prev = tail;
        tail->next = head;
    }
    else{
        baru->prev = tail;
        tail->next = baru;
        tail = baru;
        tail->next = head;
        head->prev = tail;
    }
}

//Tambah Tengah
void insertTengah(string input, int posisi)
{
    create(input);

    if (isEmpty() == 1){
        cout << " List masih kosong!" << endl;
    }
    else{
```

```
        if (posisi == 1){
            cout << " Posisi bukan posisi tengah!" << endl;
        }
        else if (posisi < 1){
            cout << " Posisi di luar jangkauan!" << endl;
        }
        else{
            int counter = 1;
            bantu = head;

            while (counter < posisi-1){
                bantu = bantu->next;
                counter++;
            }

            bantu2 = bantu->next;
            bantu->next = baru;
            bantu2->prev = baru;
            baru->prev = bantu;
            baru->next = bantu2;
        }
    }
}

//Hapus Depan
void hapusDepan()
{
    if (isEmpty() == 0){
        hapus = head;
        tail = head;
        if (hapus->next == head){
            head = NULL;
            tail = NULL;
            delete hapus;
        }
        else{
            while (tail->next != hapus){
                tail = tail->next;
            }
            head = head->next;
            tail->next = head;
            hapus->next = NULL;
            delete hapus;
        }
    }
    else{
        cout << "List masih kosong!" << endl;
```

```
    }
}

//Hapus Belakang
void hapusBelakang()
{
    if (isEmpty() == 0){
        hapus = head;
        tail = head;

        if (hapus->next == head){
            head = NULL;
            tail = NULL;
            delete hapus;
        }
        else{
            while (hapus->next != head){
                hapus = hapus->next;
            }
            while (tail->next != hapus){
                tail = tail->next;
            }
            tail->next = head;
            hapus->next = NULL;
            delete hapus;
        }
    }
    else{
        cout << "List masih kosong!" << endl;
    }
}

//Hapus Tengah
void hapusTengah(int posisi)
{
    if (isEmpty() == 1){
        cout << " List masih kosong!" << endl;
    }
    else{
        if (posisi == 1){
            cout << " Posisi bukan posisi tengah!" << endl;
        }
        else if (posisi < 1){
            cout << " Posisi di luar jangkauan!" << endl;
        }
        else{
            int counter = 1;
```

```
            bantu = head;

            while (counter < posisi-1){
                bantu = bantu->next;
                counter++;
            }

            hapus = bantu->next;
            bantu2 = hapus->next;
            bantu->next = bantu2;
            bantu2->prev = bantu;
            delete hapus;
        }
    }
}

//Hapus List
void clearList()
{   if (isEmpty() == 0){
        hapus = head;

        while (hapus->next != head){
            hapus = bantu;
            bantu = bantu->next;
            delete hapus;
        }
        head = NULL;
        tail = NULL;
    }
    cout << "List berhasil terhapus!" << endl;
}

//Tampil
void tampil()
{
    if (isEmpty() == 0){
        tail = head;
        do{
            cout << tail->data << ends;
            tail = tail->next;
        }while (tail != head);
        cout << endl;
    }
    else{
        cout << "List masih kosong!" << endl;
    }
}
```

```
int main()
{
    init();
    insertDepan("Ayam");
    tampil();
    insertDepan("Bebek");
    tampil();
    insertBelakang("Cicak");
    tampil();
    insertBelakang("Domba");
    tampil();
    hapusBelakang();
    tampil();
    insertDepan("Babi");
    tampil();
    hapusDepan();
    tampil();
    insertTengah("Sapi", 2);
    tampil();
    hapusTengah(2);
    tampil();

    return 0;
}
```

TUGAS

1. Buatlah program menu Double Linked List **Circular** dengan minimal 2 data input user. Tambahkan operasi **search**/**find** untuk mencari data pada linked list! **[Bobot 55]**

2. Buatlah program Double Linked List dengan input dari user dimana data yang ditampilkan secara terbalik (dari belakang ke depan)! **[Bobot 45]**

~ SELAMAT MENGERJAKAN 😊 ~