MODUL 4
LINKED LIST CIRCULAR DAN NON CIRCULAR

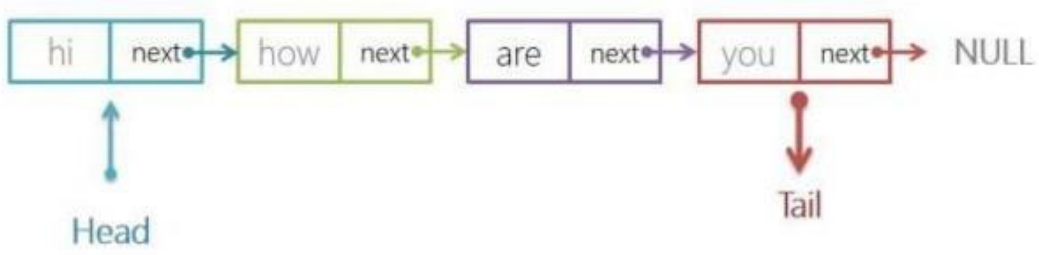
A. TUJUAN PRAKTIKUM

a. Praktikan dapat mengetahui dan memahami linked list circular dan non circular.
b. Praktikan dapat membuat linked list circular dan non circular.
c. Praktikan dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat.

B. DASAR TEORI

1. Linked List Non Circular

Linked list non circular merupakan *linked list* dengan node pertama (head) dan node terakhir (tail) yang tidak saling terhubung. Pointer terakhir (tail) pada *Linked List* ini selalu bernilai '***NULL***' sebagai pertanda data terakhir dalam *list*-nya. *Linked list non circular* dapat digambarkan sebagai berikut.



Gambar 1 *Single Linked List Non Circular*

OPERASI PADA LINKED LIST NON CIRCULAR

1. Deklarasi Simpul (Node)

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

2. Membuat dan Menginisialisasi Pointer Head dan Tail

```
node *head, *tail;
void init()
{
    head = NULL;
    tail = NULL;
};
```

3. Pengecekan Kondisi Linked List

```
bool isEmpty()
{
    if (head == NULL && tail == NULL)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

4. Penambahan Simpul (Node)

```
void insertBelakang(string data User)
{
      if (isEmpty() == true)
      {
            node *baru = new node;
            baru->data = data User;
            head = baru;
            tail = baru;
            baru->next = NULL;
      }

      else
      {
            node *baru = new node;
            baru->data = data User;
            baru->next = NULL;
            tail->next = baru;
            tail = baru;
      }
};
```

5. Penghapusan Simpul (Node)

```
void hapusDepan()
{
    if (isEmpty() == true)
    {
        cout << "List kosong!" << endl;
    }
    else
    {
        node *helper;
        helper = head;
        if (head == tail)
        {
            head = NULL;
            tail = NULL;
            delete helper;
        }
        else
            head = head->next;
        helper->next = NULL;
        delete helper;
      }
   }
}
```

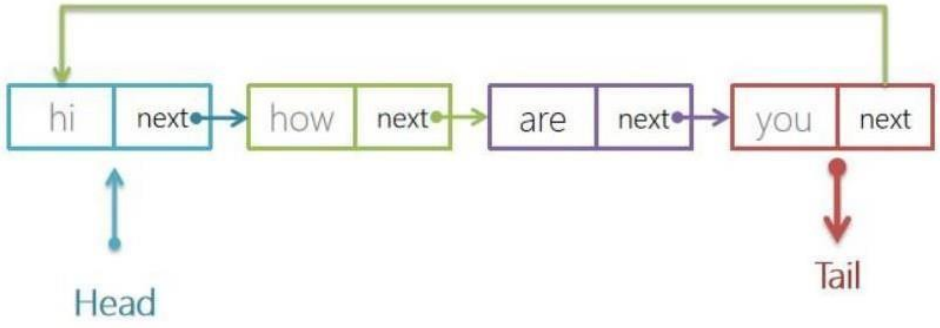
6. Tampil Data Linked List

```
void tampil()
{
    if (isEmpty() == true)
    {
        cout << "List kosong!" << endl;
    }
    else
    {
        node *helper;
        helper = head;
        while (helper != NULL)
        {
            cout << helper->data << ends;
            helper = helper->next;
        }
    }
}
```

2. Linked List Circular

Linked list circular merupakan *linked list* yang tidak memiliki akhir karena node terakhir (tail) tidak bernilai '***NULL***', tetapi terhubung dengan node pertama (head). Saat menggunakan *linked list circular* kita membutuhkan *dummy node* atau node pengecoh yang biasanya dinamakan dengan node *current* supaya program dapat berhenti menghitung data ketika node *current* mencapai node pertama (head).

Linked list circular dapat digunakan untuk menyimpan data yang perlu diakses secara berulang, seperti daftar putar lagu, daftar pesan dalam antrian, atau penggunaan memori berulang dalam suatu aplikasi. *Linked list circular* dapat digambarkan sebagai berikut.



Gambar 2 *Single Linked List Circular*

OPERASI PADA LINKED LIST CIRCULAR

1. Deklarasi Simpul (Node)

```
struct Node
{
    string data;
    Node *next;
};
```

2. Membuat dan Menginisialisasi Pointer Head dan Tail

```
Node *head, *tail, *baru, *bantu, *hapus;

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

3. Pengecekan Kondisi Linked List

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

4. Pembuatan Simpul (Node)

```
void buatNode(string data)
{
    baru = new Node;
    baru->data = data;
    baru->next = NULL;
}
```

5. Penambahan Simpul (Node)

```
// Tambah Depan
void insertDepan(string data)
{
    // Buat Node baru
    buatNode(data);

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

6. Penghapusan Simpul (Node)

```
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;
        }
}
```

7. Menampilkan Data Linked List

```
void tampil()
{
    if (isEmpty() == 0)
    {
        tail = head;
        do
        {
            cout << tail->data << ends;
            tail = tail->next;
        } while (tail != head);
        cout << endl;
    }
}
```

C. GUIDED

1. Linked List Non Circular

```
#include <iostream>
using namespace std;
/// PROGRAM SINGLE LINKED LIST NON-CIRCULAR
// Deklarasi Struct Node
struct Node
{
    int data;
    Node *next;
};
Node *head;
Node *tail;

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

// Pengecekan
bool isEmpty()
{
    if (head == NULL)
        return true;
    else
        return false;
```

```
}

// Tambah Depan
void insertDepan(int nilai)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}

// Tambah Belakang
void insertBelakang(int nilai)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
```

```
    else
    {
        tail->next = baru;
        tail = baru;
    }
}

// Hitung Jumlah List
int hitungList()
{
    Node *hitung;
    hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

// Tambah Tengah
void insertTengah(int data, int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
```

```
    {
        Node *baru, *bantu;
        baru = new Node();
        baru->data = data;
        // tranversing
        bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }

        baru->next = bantu->next;

        bantu->next = baru;
    }
}

// Hapus Depan
void hapusDepan()
{
    Node *hapus;
    if (isEmpty() == false)
    {
        if (head->next != NULL)
        {
            hapus = head;
            head = head->next
```

```
            delete hapus;


        }
        else
        {   head = tail = NULL;


        }
    }

    else
    {
        cout << "List kosong!" << endl;
    }
}

// Hapus Belakang
void hapusBelakang()
{
    Node *hapus;
    Node *bantu;
    if (isEmpty() == false)
    {
        if (head != tail)
        {
            hapus = tail;

            bantu = head;

            while (bantu->next != tail)

            {
                bantu = bantu->next;
            }
            tail = bantu;
            tail->next = NULL;
```

```
            delete hapus;
        }
        else
        {
            head = tail = NULL;
        }
    }

    else
    {
        cout << "List kosong!" << endl;
    }
}

// Hapus Tengah
void hapusTengah(int posisi)
{
    Node *bantu, *hapus, *sebelum;
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi di luar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        int nomor = 1;
        bantu = head;
        while (nomor <= posisi)
        {
            if (nomor == posisi - 1)
            {
```

```
                sebelum = bantu;
            }
            if (nomor == posisi)
            {
                hapus = bantu;
            }
            bantu = bantu->next;
            nomor++;
        }
        sebelum->next = bantu;
        delete hapus;
    }
}

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

// Ubah Tengah
void ubahTengah(int data, int posisi)
{
    Node *bantu;
    if (isEmpty() == 0)
    {
```

```
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "Posisi di luar jangkauan" << endl;
        }
        else if (posisi == 1)
        {
            cout << "Posisi bukan posisi tengah" << endl;
        }
        else
        {
            bantu = head;
            int nomor = 1;
            while (nomor < posisi)
            {
                bantu = bantu->next;
                nomor++;
            }
            bantu->data = data;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah Belakang
void ubahBelakang(int data)
{
    if (isEmpty() == 0)
    {
        tail->data = data;
    }
```

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

// Hapus List
void clearList()
{
    Node *bantu, *hapus;
    bantu = head;
    while (bantu != NULL)
    {
        hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}

// Tampilkan List
void tampil()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << 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;

}
```

2. Linked List Circular

```
#include <iostream>

using namespace std;

/// PROGRAM SINGLE LINKED LIST CIRCULAR

// Deklarasi Struct Node
struct Node
{
    string data;
    Node *next;
};

Node *head, *tail, *baru, *bantu, *hapus;

void init()
{
    head = NULL;
    tail = head;
}

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

```
}

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

// Hitung List
int hitungList()
{
    bantu = head;
    int jumlah = 0;

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

    return jumlah;
}

// Tambah Depan
void insertDepan(string data)
{
    // Buat Node baru
    buatNode(data);

    if (isEmpty() == 1)
    {
```

```
        head = baru;
        tail = head;
        baru->next = head;
    }
    else
    {
        while (tail->next != head)
        {
            tail = tail->next;
        }
        baru->next = head;
        head = baru;
        tail->next = head;
    }
}

// Tambah Belakang
void insertBelakang(string data)
{
    // Buat Node baru
    buatNode(data);

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

```
        }

        tail->next = baru;
        baru->next = head;
    }
}

// Tambah Tengah
void insertTengah(string data, int posisi)
{
    if (isEmpty() == 1)
    {
        head = baru;
        tail = head;
        baru->next = head;
    }
    else
    {
        baru->data = data;
        // transversing
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

// 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() == 0)
    {
        // transversing
        int nomor = 1;
        bantu = head;

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

        delete hapus;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Hapus List
void clearList()
{
    if (head != NULL)
    {
```

```
        hapus = head->next;

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

// Tampilkan List
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();
    hapusDepan();
    tampil();
    insertTengah("Sapi", 2);
    tampil();
    hapusTengah(2);
    tampil();

    return 0;
}
```

D. UNGUIDED

Buatlah program menu Linked List Non Circular untuk menyimpan **Nama** dan **NIM mahasiswa**, dengan menggunakan *input* dari *user*.

1. Buatlah menu untuk menambahkan, mengubah, menghapus, dan melihat Nama dan NIM mahasiswa, berikut **contoh** tampilan output dari nomor 1:

   - Tampilan Menu:

```
PROGRAM SINGLE LINKED LIST NON-CIRCULAR

1.  Tambah Depan
2.  Tambah Belakang
3.  Tambah Tengah
4.  Ubah Depan
5.  Ubah Belakang
6.  Ubah Tengah
7.  Hapus Depan
8.  Hapus Belakang
9.  Hapus Tengah
10. Hapus List
11. TAMPILKAN
0.  KELUAR

Pilih Operasi :
```

   - Tampilan Operasi Tambah:

```
-Tambah Depan-

Masukkan Nama :
Masukkan NIM  :

Data telah ditambahkan
```

```
-Tambah Tengah-

Masukkan Nama   :
Masukkan NIM    :
Masukkan Posisi :

Data telah ditambahkan
```

- Tampilan Operasi Hapus:

```
-Hapus Belakang-

Data (nama mahasiswa yang dihapus) berhasil dihapus
```

```
-Hapus Tengah-

Masukkan posisi :

Data (nama mahasiswa yang dihapus) berhasil dihapus
```

- Tampilan Operasi Ubah:

```
-Ubah Belakang-

Masukkan nama :
Masukkan NIM  :

Data (nama lama) telah diganti dengan data (nama baru)
```

```
-Ubah Belakang-

Masukkan nama   :
Masukkan NIM    :
Masukkan posisi :

Data (nama lama) telah diganti dengan data (nama baru)
```

- Tampilan Operasi Tampil Data:

```
DATA MAHASISWA

NAMA                    NIM
Nama1                   NIM1
Nama2                   NIM2
```

***Buat tampilan output sebagus dan secantik mungkin sesuai kreatifitas anda masing-masing, jangan terpaku pada contoh output yang diberikan**

2. Setelah membuat menu tersebut, masukkan data sesuai urutan berikut, lalu tampilkan data yang telah dimasukkan. (Gunakan insert depan, belakang atau tengah)

Nama	NIM
Jawad	**23300001**
[Nama Anda]	**[NIM Anda]**
Farrel	**23300003**
Denis	**23300005**
Anis	**23300008**
Bowo	**23300015**
Gahar	**23300040**
Udin	**23300048**
Ucok	**23300050**
Budi	**23300099**

3. Lakukan perintah berikut:
   a) Tambahkan data berikut diantara Farrel dan Denis:
      Wati 2330004
   b) Hapus data Denis
   c) Tambahkan data berikut di awal:
      Owi 2330000
   d) Tambahkan data berikut di akhir:
      David 23300100
   e) Ubah data Udin menjadi data berikut:
      Idin 23300045
   f) Ubah data terkahir menjadi berikut:
      Lucy 23300101
   g) Hapus data awal
   h) Ubah data awal menjadi berikut:
      Bagas 2330002
   i) Hapus data akhir
   j) Tampilkan seluruh data

E. DAFTAR PUSTAKA

Karumanchi, N. (2016). *Data Structures and algorithms made easy: Concepts, problems, Interview Questions*. CareerMonk Publications.