LAPORAN PRAKTIKUM
STURKTUR DATA DAN ALGORITMA

MODUL IV
CIRCULAR AND NON CIRCULAR

Disusun Oleh :
Muhammad Dani Ayubi
2311102003

Dosen Pengmapu:
Wahyu Andi Syahputra, S.pd.,M.Eng

LABORATORIUM MULTIMEDIA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO

PURWOKERTO
2024

BAB I
TUJUAN PRAKTIKUM

a. Mahasiswa memahami perbedaan konsep Linked list circular dan Non Circular
b. Mahasiswa mampu menerapkan Linked list circular dan Non circular ke dalam pemrograman

BAB II
DASAR TEORI

Linked list adalah salah satu struktur data yang terdiri dari serangkaian simpul atau node yang terhubung satu sama lain. Setiap simpul memiliki dua komponen yaitu data dan pointer yang menunjukkan simpul berikutnya dalam struktur. Linked list dapat dibagi menjadi dua jenis, yaitu:

1. Linked List Non-Circular
   dapat ditemukan dalam dua bentuk, yaitu Singly Linked List dan Doubly Linked List.
   - Singly Linked List adalah linked list yang setiap node hanya memiliki satu pointer yang menunjuk ke node selanjutnya
   - Doubly Linked List adalah linked list yang setiap node memiliki dua pointer, yaitu pointer ke node sebelumnya dan pointer ke node selanjutnya. Linked List Circular adalah linked list yang pointer terakhir menunjuk ke node pertama

2. Linked List Circular
   dapat ditemukan dalam dua bentuk, yaitu
   - Singly Linked List
     Singly Linked List Circular adalah linked list yang setiap node hanya memiliki satu pointer yang menunjuk ke node selanjutnya, dan pointer terakhir menunjuk ke node pertama.
   - Doubly Linked List Circular
     adalah linked list yang setiap node memiliki dua pointer, yaitu pointer ke node sebelumnya dan pointer ke node selanjutnya, dan pointer terakhir menunjuk ke node pertama.

Linked List Circular memiliki beberapa keuntungan dibandingkan dengan Linked List Non-Circular, yaitu:
(1) Lebih mudah untuk mengakses node-node yang ada di linked list.
(2) Lebih mudah untuk menghapus node-node yang ada di linked list.
(3) Lebih mudah untuk mengganti urutan node-node yang ada di linked list.

Namun, Linked List Circular juga memiliki beberapa kelemahan, yaitu:
(1) Lebih sulit untuk menghitung jumlah node yang ada di linked list.
(2) Lebih sulit untuk mengakses node-node yang ada di tengah linked list.
(3) Lebih sulit untuk mengganti urutan node-node yang ada di tengah linked
list.

BAB III

GUIDED

1. Guided 1

Source code

```
#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;
```

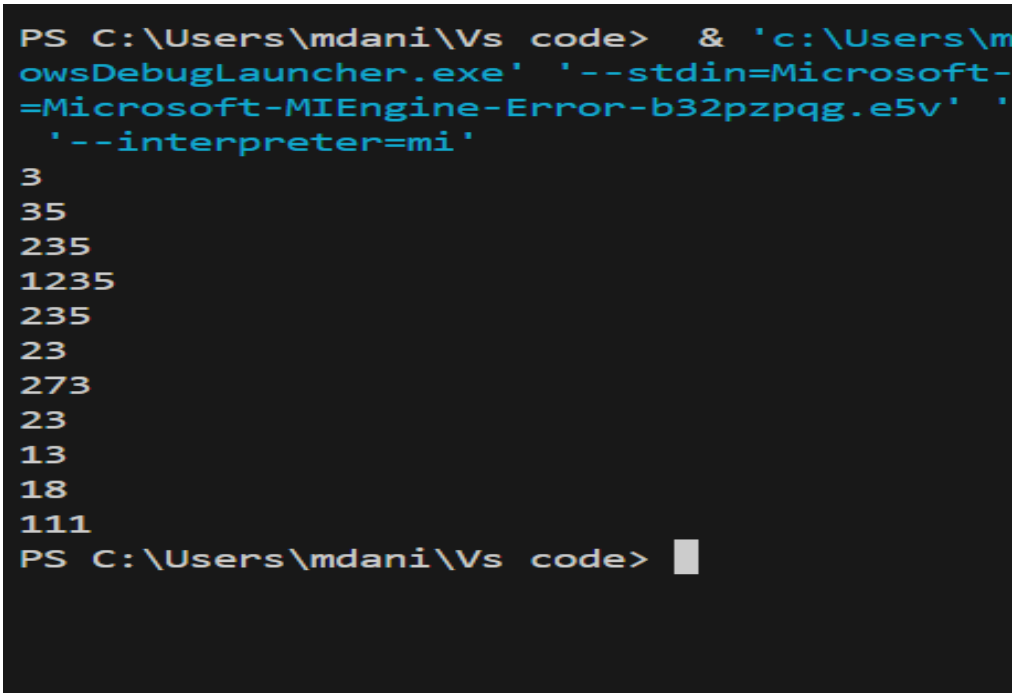
```
if (posisi < 1 || posisi > hitungList())
{
cout << "Posisi di luar jangkauan" << endl;
}
else if (posisi == 1)
{
cout << "Posisi bukan posisi tengah" << endl;
}
else
{
int nomor = 1;
bantu = head;
while (nomor <= posisi)
{
if (nomor == posisi - 1)
{

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;
}
```

Screenshoot program



Deskripsi program

Program ini memiliki beberapa variabel global, yaitu head dan tail. Head digunakan sebagai pointer ke node pertama di linked list. Tail digunakan sebagai pointer ke node terakhir di linked list. Fitur inisialisasi linked list akan membuat head dan tail bernilai NULL. Fitur pengecekan apakah linked list kosong atau tidak akan mengembalikan true jika linked list kosong dan false jika linked list tidak kosong. Fitur menambah data di depan, belakang, dan tengah linked list akan membuat node baru dengan data yang diinginkan. Fitur menambah data di depan akan menambahkan node baru di depan linked list. Fitur menambah data di belakang akan menambahkan node baru di belakang linked list. Fitur

menambah data di tengah linked list akan melakukan traversing di linked list sampai pointer next menjadi NULL atau sampai posisi yang diinginkan, lalu menambahkan node baru di posisi tersebut. Fitur menghapus data di depan, belakang, dan tengah linked list akan melakukan traversing di linked list sampai pointer next menjadi NULL atau sampai posisi yang diinginkan, lalu menghapus node tersebut. Fitur menghapus semua data di linked list akan melakukan traversing di linked list sampai pointer next menjadi NULL, lalu menghapus semua node. Fitur menampilkan data di linked list akan melakukan traversing di linked list sampai pointer next menjadi NULL dan menampilkan data di setiap node. Program ini akan menambah beberapa data ke linked list, menghapus data di linked list, mengubah data di linked list, dan menampilkan data di linked list. Program akan menampilkan hasil dari setiap operasi yang dilakukan.

2. Guided 2

Source code

```
#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;
```

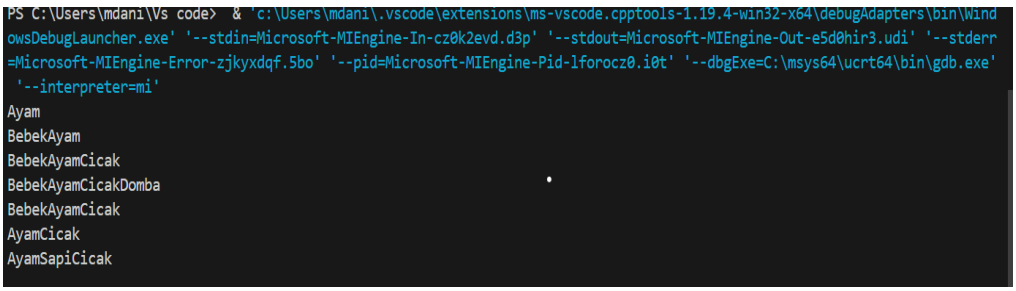
```
        tail = head;
        baru->next = head;
    }
    else
    {
        baru->data = data;
        // transversing
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        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;
}
```

Screenshoot program



Deskripsi program

Program di atas merupakan implementasi dari single linked list circular menggunakan bahasa pemrograman C++. Single linked list circular adalah salah satu struktur data yang terdiri dari sejumlah simpul atau node yang saling terhubung secara sekuensial satu sama lain dengan arah satu arah. Setiap simpul berisi data dan pointer yang menunjuk ke simpul berikutnya dalam urutan. Struktur data ini juga bersifat melingkar, artinya simpul terakhir terhubung dengan simpul pertama sehingga membentuk suatu lingkaran.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

Source code

```
#include <iostream>
using namespace std;

struct Node {
    string nama;
    string nim;
    Node* next;
};

class LinkedList {
private:
    Node* head;

public:
    LinkedList() {
        head = nullptr;
    }

    void tambahDepan(string nama, string nim) {
        Node* newNode = new Node;
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = head;
        head = newNode;
        cout << "Data telah ditambahkan" << endl;
    }

    void tambahBelakang(string nama, string nim) {
        Node* newNode = new Node;
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = nullptr;
        if (head == nullptr) {
            head = newNode;
            return;
        }
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
        cout << "Data telah ditambahkan" << endl;
```

```
}

void tambahTengah(string nama, string nim, int posisi) {
    if (posisi <= 0) {
        cout << "Posisi tidak valid" << endl;
        return;
    }
    Node* newNode = new Node;
    newNode->nama = nama;
    newNode->nim = nim;
    Node* temp = head;
    for (int i = 0; i < posisi - 1; i++) {
        if (temp == nullptr) {
            cout << "Posisi tidak valid" << endl;
            return;
        }
        temp = temp->next;
    }
    if (temp == nullptr) {
        cout << "Posisi tidak valid" << endl;
        return;
    }
    newNode->next = temp->next;
    temp->next = newNode;
    cout << "Data telah ditambahkan" << endl;
}

void hapusDepan() {
    if (head == nullptr) {
        cout << "Linked list kosong" << endl;
        return;
    }
    Node* temp = head;
    head = head->next;
    delete temp;
    cout << "Data berhasil dihapus" << endl;
}

void hapusBelakang() {
    if (head == nullptr) {
        cout << "Linked list kosong" << endl;
        return;
    }
    if (head->next == nullptr) {
        delete head;
        head = nullptr;
        cout << "Data berhasil dihapus" << endl;
```

```
            return;
        }
        Node* temp = head;
        while (temp->next->next != nullptr) {
            temp = temp->next;
        }
        delete temp->next;
        temp->next = nullptr;
        cout << "Data berhasil dihapus" << endl;
    }

    void hapusTengah(int posisi) {
        if (posisi <= 0 || head == nullptr) {
            cout << "Linked list kosong atau posisi tidak
valid" << endl;
            return;
        }
        if (posisi == 1) {
            hapusDepan();
            return;
        }
        Node* temp = head;
        for (int i = 0; i < posisi - 2; i++) {
            if (temp->next == nullptr) {
                cout << "Posisi tidak valid" << endl;
                return;
            }
            temp = temp->next;
        }
        if (temp->next == nullptr) {
            cout << "Posisi tidak valid" << endl;
            return;
        }
        Node* nodeToDelete = temp->next;
        temp->next = temp->next->next;
        delete nodeToDelete;
        cout << "Data berhasil dihapus" << endl;
    }

     void ubahDepan(string namaBaru, string nimBaru) {
        if (head == nullptr) {
            cout << "Linked list kosong" << endl;
            return;
        }
        head->nama = namaBaru;
        head->nim = nimBaru;
        cout << "Data berhasil diubah" << endl;
```

```
    }

    void ubahBelakang(string namaBaru, string nimBaru) {
        if (head == nullptr) {
            cout << "Linked list kosong" << endl;
            return;
        }
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->nama = namaBaru;
        temp->nim = nimBaru;
        cout << "Data berhasil diubah" << endl;
    }

    void ubahTengah(string namaBaru, string nimBaru, int
posisi) {
        if (posisi <= 0 || head == nullptr) {
            cout << "Linked list kosong atau posisi tidak
valid" << endl;
            return;
        }
        Node* temp = head;
        for (int i = 0; i < posisi - 1; i++) {
            if (temp == nullptr) {
                cout << "Posisi tidak valid" << endl;
                return;
            }
            temp = temp->next;
        }
        if (temp == nullptr) {
            cout << "Posisi tidak valid" << endl;
            return;
        }
        temp->nama = namaBaru;
        temp->nim = nimBaru;
        cout << "Data berhasil diubah" << endl;
    }

    void hapusList() {
        Node* current = head;
        Node* next;
        while (current != nullptr) {
            next = current->next;
            delete current;
            current = next;
```

```
        }
        head = nullptr;
        cout << "Linked list berhasil dihapus" << endl;
    }

    void tampilkanData() {
        Node* temp = head;
        cout << "DATA MAHASISWA" << endl;
        cout << "NAMA\tNIM" << endl;
        while (temp != nullptr) {
            cout << temp->nama << "\t" << temp->nim << endl;
            temp = temp->next;
        }
    }
};

int main() {
    LinkedList linkedList;
    int choice;
    string nama, nim;
    int posisi;

    do {
        cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR" <<
endl;
        cout << "1. Tambah Depan" << endl;
        cout << "2. Tambah Belakang" << endl;
        cout << "3. Tambah Tengah" << endl;
        cout << "4. Ubah Depan" << endl;
        cout << "5. Ubah Belakang" << endl;
        cout << "6. Ubah Tengah" << endl;
        cout << "7. Hapus Depan" << endl;
        cout << "8. Hapus Belakang" << endl;
        cout << "9. Hapus Tengah" << endl;
        cout << "10. Hapus List" << endl; // Menu untuk hapus
list
        cout << "11. Tampilkan Data" << endl;
        cout << "0. Keluar" << endl;
        cout << "Pilih Operasi : ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "-Tambah Depan-" << endl;
                cout << "Masukkan Nama : ";
                cin >> nama;
                cout << "Masukkan NIM : ";
```

```
            cin >> nim;
            linkedList.tambahDepan(nama, nim);
            break;
        case 2:
            cout << "-Tambah Belakang-" << endl;
            cout << "Masukkan Nama : ";
            cin >> nama;
            cout << "Masukkan NIM : ";
            cin >> nim;
            linkedList.tambahBelakang(nama, nim);
            break;
        case 3:
            cout << "-Tambah Tengah-" << endl;
            cout << "Masukkan Nama : ";
            cin >> nama;
            cout << "Masukkan NIM : ";
            cin >> nim;
            cout << "Masukkan Posisi : ";
            cin >> posisi;
            linkedList.tambahTengah(nama, nim, posisi);
            break;
        case 4:
            cout << "-Ubah Depan-" << endl;
            cout << "Masukkan Nama Baru : ";
            cin >> nama;
            cout << "Masukkan NIM Baru : ";
            cin >> nim;
            linkedList.ubahDepan(nama, nim);
            break;
        case 5:
            cout << "-Ubah Belakang-" << endl;
            cout << "Masukkan Nama Baru : ";
            cin >> nama;
            cout << "Masukkan NIM Baru : ";
            cin >> nim;
            linkedList.ubahBelakang(nama, nim);
            break;
        case 6:
            cout << "-Ubah Tengah-" << endl;
            cout << "Masukkan Nama Baru : ";
            cin >> nama;
            cout << "Masukkan NIM Baru : ";
            cin >> nim;
            cout << "Masukkan Posisi : ";
            cin >> posisi;
            linkedList.ubahTengah(nama, nim, posisi);
            break;
```

```
            case 7:
                linkedList.hapusDepan();
                break;
            case 8:
                linkedList.hapusBelakang();
                break;
            case 9:
                cout << "-Hapus Tengah-" << endl;
                cout << "Masukkan Posisi : ";
                cin >> posisi;
                linkedList.hapusTengah(posisi);
                break;
            case 10:
                linkedList.hapusList(); // Hapus List
                break;
            case 11:
                linkedList.tampilkanData();
                break;
            case 12:
                cout << "Program selesai." << endl;
                break;
            default:
                cout << "Pilihan tidak valid." << endl;
        }
    } while (choice != 0);

    return 0;
}
```

Screenshoot program

```
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 Data
0. Keluar
Pilih Operasi : 1
-Tambah Depan-
Masukkan Nama : Amel
Masukkan NIM : 2311102001
Data telah ditambahkan
```

```
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 Data
0. Keluar
Pilih Operasi : 2
-Tambah Belakang-
Masukkan Nama : Dimas
Masukkan NIM : 2311102002
Data telah ditambahkan
```

```
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 Data
0. Keluar
Pilih Operasi : 11
DATA MAHASISWA
NAMA    NIM
Amel    2311102001
Dimas   2311102002
```

Deskripsi program

Program ini juga memiliki fitur untuk menghitung jumlah data mahasiswa di linked list. Selain itu, program ini juga memiliki fitur untuk mengecek apakah linked list kosong atau tidak. Program ini menggunakan struktur data linked list untuk menyimpan data mahasiswa. Setiap node linked list terdiri dari tiga bagian, yaitu nama, nim, dan pointer ke node selanjutnya. Program ini juga memiliki dua pointer, yaitu head dan tail, yang digunakan untuk menunjuk ke node pertama dan node terakhir di linked list. Program ini juga memiliki fitur untuk menginput data mahasiswa, mengubah data mahasiswa, menghapus data mahasiswa, dan menampilkan data mahasiswa. Program ini juga memiliki fitur untuk menghitung jumlah data mahasiswa di linked list.

BAB IV
KESIMPULAN

Berdasarkan praktikum yang telah dilakukan, dapat disimpulkan bahwa:

1. Linked list non-circular lebih sederhana dan mudah dipahami, sedangkan linked list circular membutuhkan sedikit pemahaman tambahan tentang bagaimana elemen terkait satu sama lain.

2. Pada linked list non-circular, elemen terakhir (tail) memiliki nilai pointer NULL, sedangkan pada linked list circular, elemen terakhir (tail) memiliki nilai pointer yang menunjuk pada elemen pertama (head).

3. Pada linked list non-circular, jika kita ingin mengunjungi elemen terakhir, kita harus memulai dari elemen pertama dan menelusuri seluruh linked list, sedangkan pada linked list circular, kita bisa langsung melompat dari elemen terakhir ke elemen pertama.

4. Operasi-operasi pada linked list circular dan non-circular hampir sama, tetapi implementasinya bisa sedikit berbeda karena adanya elemen yang terhubung secara siklik pada linked list circular.

5. Linked list circular sering digunakan pada aplikasi yang membutuhkan akses terus-menerus ke elemen awal dan akhir linked list secara berulang-ulang, seperti pengolahan antrian (queue) dan tabel hash (hash table).

6. Pemilihan jenis linked list yang akan digunakan tergantung pada kebutuhan aplikasi dan kompleksitas implementasi yang diinginkan.