

LAPORAN PRAKTIKUM
MODUL III
SINGLE DAN DOUBLE LINKED LIST



Disusun Oleh:

Muhammad Raafi Al Hafiidh

2311102070

Dosen:

Wahyu Andi Saputra, S.Pd., M.Eng.

PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024

BAB I

TUJUAN PRAKTIKUM

- Mahasiswa memahami perbedaan konsep Single dan Double Linked list.
- Mahasiswa mampu menerapkan Single dan Double Linked list ke dalam pemrogramman.

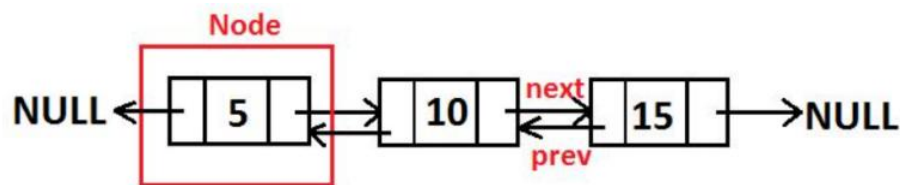
BAB II

DASAR TEORI

A. Double Linked list

Dalam pembahasan artikel sebelumnya telah diperkenalkan Single Linked List, yakni linked list dengan sebuah pointer penghubung. Dalam artikel ini, dibahas pula varian linked list dengan 2 pointer penunjuk, yakni Doubly linked list yang memiliki pointer penunjuk 2 arah, yakni ke arah node sebelum (previous/prev) dan node sesudah (next).

Representasi sebuah doubly linked list dapat dilihat pada gambar berikut ini :



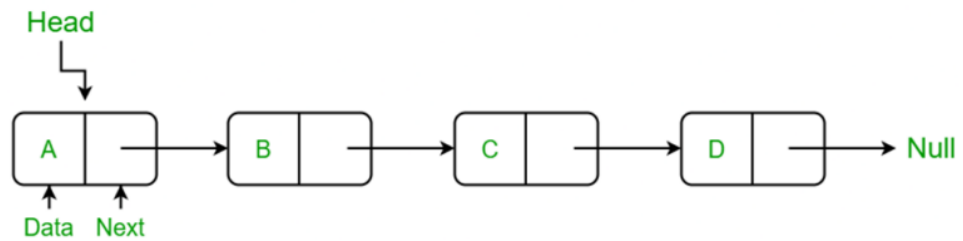
Di dalam sebuah linked list, ada 2 pointer yang menjadi penunjuk utama, yakni pointer HEAD yang menunjuk pada node pertama di dalam linked list itu sendiri dan pointer TAIL yang menunjuk pada node paling akhir di dalam linked list. Sebuah linked list dikatakan kosong apabila isi pointer head adalah NULL. Selain itu, nilai pointer prev dari HEAD selalu NULL, karena merupakan data pertama. Begitu pula dengan pointer next dari TAIL yang selalu bernilai NULL sebagai penanda data terakhir.

B. Single Linked list

Linked List merupakan suatu bentuk struktur data yang berisi kumpulan data yang disebut sebagai node yang tersusun secara sekuensial, saling sambung menyambung, dinamis, dan terbatas. Untuk menghubungkan satu node dengan node yang lainnya Linked List menggunakan pointer sebagai penunjuk node selanjutnya. Node sendiri merupakan sebuah struct yang terdiri dari beberapa field, minimal ada 2 buah field yaitu field untuk isi dari struct datanya sendiri, dan 1 field arbitrary bertipe pointer sebagai penunjuk node selanjutnya.

Linked List merupakan suatu bentuk struktur data yang berisi kumpulan data yang disebut sebagai node yang tersusun secara sekuensial, saling sambung menyambung, dinamis, dan terbatas. Untuk menghubungkan satu node dengan node yang lainnya Linked List menggunakan pointer sebagai penunjuk node selanjutnya. Node sendiri merupakan sebuah struct yang

terdiri dari beberapa field, minimal ada 2 buah field yaitu field untuk isi dari struct datanya sendiri, dan 1 field arbitrary bertipe pointer sebagai penunjuk node selanjutnya.



Single linked list yang kedua adalah circular linked list. Perbedaan circular linked list dan non circular linked adalah penunjuk next pada node terakhir pada circular linked list akan selalu merujuk ke node pertama.

BAB III

LATIHAN DAN TUGAS

A. 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 *hapus, *bantu, *bantu2;
    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)
            {
                bantu2 = bantu;
            }
            if (nomor == posisi)
            {
                hapus = bantu;
            }
            bantu = bantu->next;
            nomor++;
        }
        bantu2->next = bantu;
        delete hapus;
    }
}
// Ubah Depan
void ubahDepan(int data)
{

```



```

        if (isEmpty() == false)
        {
            head->data = data;
        }
        else
        {
            cout << "List masih kosong!" << endl;
        }
    }
    // Ubah Tengah
    void ubahTengah(int data, int posisi)
    {
        Node *bantu;
        if (isEmpty() == false)
        {
            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() == false)
        {
            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();
}

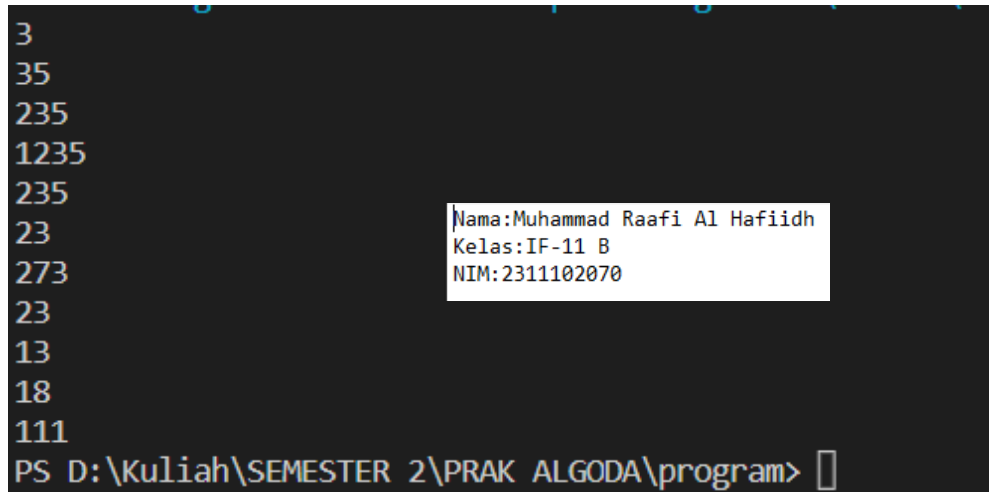
```

```

        hapusBelakang();
        tampil();
        insertTengah(7, 2);
        tampil();
        hapusTengah(2);
        tampil();
        ubahDepan(1);
        tampil();
        ubahBelakang(8);
        tampil();
        ubahTengah(11, 2);
        tampil();
        return 0;
    }

```

Screenshoot Program



```

3
35
235
1235
235
23
273
23
13
18
111
PS D:\Kuliah\SEMESTER 2\PRAK ALGODA\program>

```

Nama:Muhammad Raafi Al Hafiidh
 Kelas:IF-11 B
 NIM:2311102070

Deskripsi Program

Program di atas merupakan sebuah program yang menggunakan fungsi Single Linked list. Program tersebut menjalankan beberapa fungsi diantaranya insertDepan(3) yang artinya program diperintahkan untuk melakukan fungsi insertDepan() dengan memasukkan angka 3 sebagai inputan di depan single linked list, lalu ada insertBelakang(5) yang artinya program diperintahkan untuk melakukan fungsi insertBelakang() dengan memasukkan angka 5 sebagai inputan di belakang single linked list. Lalu ada insertTengah(7, 2) artinya program akan diperintahkan memasukkan angka 7 sebagai inputan di posisi ke 2 didalam single linked list. Selanjutnya ada hapusDepan() dan hapusBelakang() artinya program akan diperintahkan untuk menghapus angka yang ada di depan dan di belakang single linked list. Selanjutnya ada fungsi ubahDepan(1) dimana program akan diperintah untuk mengubah data single linked list paling depan dengan angka 1, berikutnya ubahBelakang(8) yang

sama seperti fungsi ubahDepan() tetapi perbedaannya fungsi tersebut akan merubah data paling belakang atau akhir dari single linked list. ubahTengah(11, 2) berarti program akan merubah data pada single linked list di posisi ke 2 menjadi angka 11. Lalu yang terakhir setiap fungsi dijalankan maka hasil dari fungsi tersebut akan ditampilkan menggunakan fungsi tampil().

2. Guided 2

Source Code

```
#include <iostream>
using namespace std;
class Node {
public:
int data;
Node* prev;
Node* next;
};
class DoublyLinkedList {
public:
Node* head;
Node* tail;
DoublyLinkedList() {
head = nullptr;
tail = nullptr;
}
void push(int data) {
Node* newNode = new Node;
newNode->data = data;
newNode->prev = nullptr;
newNode->next = head;
if (head != nullptr) {
head->prev = newNode;
} else {
tail = newNode;
}
head = newNode;
}
void pop() {
if (head == nullptr) {
return;
}
Node* temp = head;
head = head->next;
if (head != nullptr) {
head->prev = nullptr;
} else {
tail = nullptr;
}
}
```

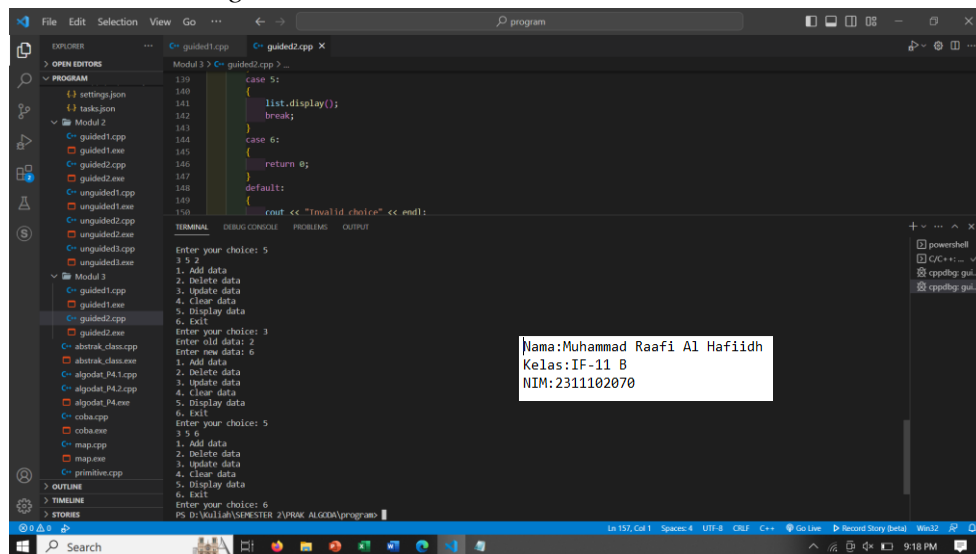
```

delete temp;
}
bool update(int oldData, int newData) {
Node* current = head;
while (current != nullptr) {
if (current->data == oldData) {
current->data = newData;
return true;
}
current = current->next;
}
return false;
}
void deleteAll() {
Node* current = head;
while (current != nullptr) {
Node* temp = current;
current = current->next;
delete temp;
}
head = nullptr;
tail = nullptr;
}
void display() {
Node* current = head;
while (current != nullptr) {
cout << current->data << " ";
current = current->next;
}
cout << endl;
}
};
int main() {
DoublyLinkedList list;
while (true) {
cout << "1. Add data" << endl;
cout << "2. Delete data" << endl;
cout << "3. Update data" << endl;
cout << "4. Clear data" << endl;
cout << "5. Display data" << endl;
cout << "6. Exit" << endl;
int choice;
cout << "Enter your choice: ";
cin >> choice;
switch (choice) {
case 1: {
int data;
cout << "Enter data to add: ";
cin >> data;

```

```
list.push(data);
break;
}
case 2: {
list.pop();
break;
}
case 3: {
int oldData, newData;
cout << "Enter old data: ";
cin >> oldData;
cout << "Enter new data: ";
cin >> newData;
bool updated = list.update(oldData, newData);
if (!updated) {
cout << "Data not found" << endl;
}
break;
}
case 4: {
list.deleteAll();
break;
}
case 5: {
list.display();
break;
}
case 6: {
return 0;
}
default: {
cout << "Invalid choice" << endl;
break;
}
}
return 0;
}
```

Screenshoot Program



Deskripsi Program

Program tersebut merupakan program yang menggunakan double linked list dalam proses nya. Pertama program akan menampilkan 6 menu dalam mengelola double linked list nantinya. Ada Add data, Delete data, Update data, Clear data, Display data, dan Exit. Untuk menambahkan data user harus memilih 1 untuk masuk ke menu Add data lalu memasukkan angka berapa yang ingin dimasukkan, setelah itu user akan kembali lagi ke menu awal. Apabila user ingin menghapus data maka user harus memilih 2 untuk ke menu Delete data dan memilih angka mana yang ingin dihapus, apabila user ingin mengubah data yang sebelumnya sudah dimasukkan maka user bisa memilih Update data dengan memasukkan 3 sebagai pilihan, nantinya user akan diminta memasukkan angka sebelumnya yang ingin diubah dan memasukkan angka baru. Lalu apabila user ingin menghapus semua data yang ada, user bisa memasukkan angka 4 untuk menghapus semua data yang ada sekaligus. Apabila user ingin melihat data yang sudah diinputkan user bisa memasukkan angka 5 untuk ke menu Display data, nantinya semua data yang ada akan ditampilkan secara berderet. Terakhir apabila user ingin keluar atau menyelesaikan program user harus memasukkan angka 6 dan menuju ke menu Exit.

B. UNGUIDED

1. Unguided 1

Buatlah program menu Single Linked List non-circular untuk menyimpan Nama dan usia mahasiswa, dengan menggunakan inputan dari user. Lakukan operasi berikut :

- a. Masukkan data sesuai urutan berikut. (Gunakan insert depan, belakang, atau tengah). Data pertama yang dimasukkan adalah nama dan usia anda.

[Nama_anda]	[Usia_anda]
Budi	19
Carol	20
Ann	18
Yusuke	19
Akechi	20
Hoshino	18
Karin	18

- b. Hapus data Akechi
- c. Tambahkan data berikut diantara Carol dan Ann : Futaba 18
- d. Tambahkan data berikut diawal : Igor 20
- e. Ubah data Carol menjadi : Reyn 18
- f. Tampilkan seluruh data

Source Code

```
#include <iostream>
using namespace std;
/// PROGRAM SINGLE LINKED LIST NON-CIRCULAR
// Deklarasi Struct Node

struct Node
{
    string nama;
    int umur;
    Node *next;
};

Node *head = NULL;
Node *tail = NULL;

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



```

        else
            return false;
    }

    // Tambah Depan
    void insertDepan(string nama, int umur)
    {
        // Buat Node baru
        Node *baru = new Node;
        baru->nama = nama;
        baru->umur = umur;
        baru->next = NULL;

        if (isEmpty() == true)
        {
            head = tail = baru;
            tail->next = NULL;
        }
        else
        {
            baru->next = head;
            head = baru;
        }
    }

    // Tambah Belakang
    void insertBelakang(string nama, int umur)
    {
        // Buat Node baru
        Node *baru = new Node;
        baru->nama = nama;
        baru->umur = umur;
        baru->next = NULL;

        if (head == NULL)
        {
            head = baru;
            tail = baru;
        }
        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(string nama, int umur, int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        // buat node baru
        Node *baru;
        baru = new Node();
        baru->nama = nama;
        baru->umur = umur;

        int i = 1;
        Node *current = head;
        while (i < posisi - 1 && current->next != NULL)
        {
            current = current->next;
            i++;
        }
        baru->next = current->next;
        current->next = baru;
    }
}

void remove(string nama)
{
    if (head == NULL)
    {

```

```

        cout << "List kosong!" << endl;
        return;
    }
    if (head->nama == nama)
    {
        Node *temp = head;
        head = head->next;
        delete temp;
        cout << "Data berhasil dihapus!" << endl;
        return;
    }
    Node *current = head;
    while (current->next != NULL && current->next->nama !=
nama)
    {
        current = current->next;
    }
    if (current->next == NULL)
    {
        cout << "Data tidak ditemukan!" << endl;
        return;
    }
    Node *temp = current->next;
    current->next = temp->next;
    delete temp;
    cout << "Data berhasil dihapus!" << endl;
}

void update(string oldnama, string newnama, int newumur)
{
    if (head == NULL)
    {
        cout << "List kosong!" << endl;
        return;
    }
    Node *current = head;
    while (current != NULL)
    {
        if (current->nama == oldnama)
        {
            current->nama = newnama;
            current->umur = newumur;
            cout << "Data berhasil diupdate!" << endl;
            return;
        }
        current = current->next;
    }
    cout << "Data tidak ditemukan!" << endl;
}

```

```

    }

    // Tampilkan List
    void tampil()
    {
        if (head == NULL)
        {
            cout << "List Kosong!" << endl;
            return;
        }
        Node *current = head;
        int o = 1;
        while (current != NULL)
        {
            cout << o++ << ". "
                << "Nama:" << current->nama << ", Usia:" <<
current->umur << endl;
            current = current->next;
        }
    }

    int main()
    {
        int choice;
        int choice_data;
        string nama;
        int umur;
        int posisi;
        loop_menu:

        cout << "\nMenu" << endl;
        cout << "1. Tambah data" << endl;
        cout << "2. Hapus data" << endl;
        cout << "3. Ubah data" << endl;
        cout << "4. Tampilkan data" << endl;
        cout << "5. keluar" << endl;
        cout << "Masukkan pilihan dari menu (1-5): ";
        cin >> choice;
        switch (choice)
        {

        case 1:
        {
            loop_choice_data:
            cout << endl;
            cout << "Tambahkan Data\n";
            cout << "Masukan posisi Data :\n";
            cout << "1. Posisi di Depan\n";
            cout << "2. Posisi di Belakang\n";

```

```

        cout << "3. Posisi Di Tengah\n";
        cout << "4. Keluar\n";
        cout << "Masukkan posisi pilihan (1-4): ";
        cin >> choice_data;
        if (choice_data == 1)
        {
            cout << "--Posisi Di Depan--\n";
            cout << "Masukkan nama: ";
            cin >> nama;
            cout << "Masukkan umur: ";
            cin >> umur;
            insertDepan(nama, umur);
        }
        else if (choice_data == 2)
        {
            cout << "--Posisi Di Belakang--\n";
            cout << "Masukkan nama: ";
            cin >> nama;
            cout << "Masukkan umur: ";
            cin >> umur;
            insertBelakang(nama, umur);
        }
        else if (choice_data == 3)
        {
            cout << "--Posisi Di Tengah--\n";
            cout << "Masukkan nama: ";
            cin >> nama;
            cout << "Masukkan umur: ";
            cin >> umur;
            cout << "Masukkan posisi: ";
            cin >> posisi;
            insertTengah(nama, umur, posisi);
        }
        else if (choice_data == 4)
        {
            cout << "Keluar dari Menu\n";
            goto loop_menu;
        }
        else
        {
            cout << "Pilihan Anda Tidak Valid\n";
        }
        goto loop_choice_data;
        break;
    }

    case 2:
    {
        cout << "---- Hapus Data ----\n";

```

```

        cout << "Masukkan nama yang ingin dihapus :";
        cin >> nama;
        remove(nama);
        break;
    }
    case 3:
    {
        string oldnama, newnama;
        cout << "---- Ubah Data ----\n";
        cout << "Masukkan nama yang ingin diubah: ";
        cin >> oldnama;
        cout << "Masukkan nama yang baru: ";
        cin >> newnama;
        cout << "Masukkan usia baru: ";
        cin >> umur;
        update(oldnama, newnama, umur);
        break;
    }

    case 4:
    {
        cout << "---- Tampilkan Data ----\n";
        tampil();
        break;
    }

    case 5:
    {
        cout << "Terima Kasih" << endl;
        return 0;
    }
    default:
    {
        cout << "Pilihan tidak valid" << endl;
    }
    }

    goto loop_menu;
    return 0;
}

```

Screenshoot Program

```
Tambahkan Data
Masukan posisi Data :
1. Posisi di Depan
2. Posisi di Belakang
3. Posisi di Tengah
4. Keluar
Masukkan posisi pilihan (1-4): 1
--Posisi Di Depan--
Masukkan nama: fauzi
Masukkan umur: 19
```

```
Menu
1. Tambah data
2. Hapus data
3. Ubah data
4. Tampilkan data
5. keluar
Masukkan pilihan dari menu (1-5): 2
---- Hapus Data ----
Masukkan nama yang ingin dihapus :Akechi
Data berhasil dihapus!
```

```
Tambahkan Data
Masukan posisi Data :
1. Posisi di Depan
2. Posisi di Belakang
3. Posisi di Tengah
4. Keluar
Masukkan posisi pilihan (1-4): 3
--Posisi Di Tengah--
Masukkan nama: Futaba
Masukkan umur: 18
Masukkan posisi: 4
```

```
Tambahkan Data
Masukan posisi Data :
1. Posisi di Depan
2. Posisi di Belakang
3. Posisi di Tengah
4. Keluar
Masukkan posisi pilihan (1-4): 1
--Posisi Di Depan--
Masukkan nama: Igor
Masukkan umur: 20
```

```
Menu
1. Tambah data
2. Hapus data
3. Ubah data
4. Tampilkan data
5. keluar
Masukkan pilihan dari menu (1-5): 3
---- Ubah Data ----
Masukkan nama yang ingin diubah: Carol
Masukkan nama yang baru: Reyn
Masukkan usia baru: 18
Data berhasil diupdate!
```

```
Masukkan pilihan dari menu (1-5): 4
---- Tampilkan Data ----
1. Nama:Igor, Usia:20
2. Nama:fauzi, Usia:19
3. Nama:Budi, Usia:19
4. Nama:Reyn, Usia:18
5. Nama:Futaba, Usia:18
6. Nama:Ann, Usia:18
7. Nama:Yusuke, Usia:19
8. Nama:Hoshino, Usia:18
9. Nama:Karin, Usia:18
```

Nama:Muhammad Raafi Al Hafiidh
Kelas:IF-11 B
NIM:2311102070

Deskripsi Program

Program di atas adalah program menu Single Linked List non-circular untuk menyimpan nama dan usia mahasiswa, program diawali dengan beberapa menu yang bisa dipilih oleh user dengan menginputkan angka 1-5 untuk memilih menu mana yang akan digunakan, untuk contoh diatas pertama-tama user akan memilih untuk menambah data beberapa mahasiswa dengan menu Tambah data, lalu user memilih akan ada dimana posisi data tersebut. Setelah selesai menambahkan data user diminta untuk mengubah data Carol menjadi Reyn dengan usia 18 di menu Ubah data. Setelah itu user diminta untuk menghapus data Igor dengan menggunakan menu Hapus data dan terakhir user diminta untuk menampilkan semua data secara keseluruhan.

2. Unguided 2

Modifikasi Guided Double Linked List ditambahkan fungsi menambahkan data, menghapus, dan update di tengah / di urutan tertentu yang diminta. Selain itu, buatlah agar tampilannya menampilkan Nama produk dan harga.

Nama Produk	Harga
Originote	60.000
Somethinc	150.000
Skintific	100.00
Wardah	50.000
Hanasui	30.000

Case :

1. Tambahkan produk Azarine dengan harga 65000 diantara Somethinc dan Skintific
2. Hapus produk wardah
3. Update produk Hanasui menjadi Cleora dengan harga 55.000
4. Tampilkan menu seperti dibawah ini
Toko Skincare Purwokerto
 1. Tambah data

2. Hapus data
3. Update data
4. Tambah data urutan tertentu
5. Hapus data urutan tertentu
6. Hapus seluruh data
7. Tampilkan data
8. Exit

Pada menu 7, tampilan akhirnya akan menjadi seperti dibawah ini :

Nama Produk	Harga
Originote	60.000
Somethinc	150.000
Azarine	65.000
Skintific	100.000
Cleora	55.000

Source Code

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

class skincare
{
public:
    string nama, oldnama, newnama;
    int harga;
    skincare *prev;
    skincare *next;
    skincare(string _nama = "", int _harga = 0, skincare *_prev
= nullptr, skincare *_next = nullptr)
    {
        nama = _nama;
        harga = _harga;
        prev = _prev;
        next = _next;
    }
};

class tokoskincare
{
private:
    skincare *head;
    skincare *tail;

public:
    tokoskincare()
```

```

{
    head = nullptr;
    tail = nullptr;
}

void tambahdata(string _nama, int _harga)
{
    skincare *newNode = new skincare(_nama, _harga);

    if (head == nullptr)
    {
        head = newNode;
        tail = newNode;
    }
    else
    {
        tail->next = newNode;
        newNode->prev = tail;
        tail = newNode;
    }

    cout << "Data berhasil ditambahkan" << endl;
}

void hapusData(string nama)
{
    if (head == nullptr)
    {
        cout << "Tidak ada data yang dapat dihapus!" <<
endl;
        return;
    }

    skincare *temp = head;
    while (temp != nullptr)
    {
        if (temp->nama == nama)
        {
            if (temp == head)
            {
                head = temp->next;
                if (head != nullptr)
                    head->prev = nullptr;
            }
            else if (temp == tail)
            {
                tail = temp->prev;
                if (tail != nullptr)
                    tail->next = nullptr;
            }
        }
    }
}

```

```

        }
        else
        {
            temp->prev->next = temp->next;
            temp->next->prev = temp->prev;
        }
        delete temp;
        cout << "Data dengan nama " << nama << "
Berhasil dihapus!" << endl;
        return;
    }
    temp = temp->next;
}
cout << "data dengan nama " << nama << ", tidak
ditemukan" << endl;
}

bool updatedata(string _oldnama, string _newnama, int
_harga)
{
    skincare *current = head;
    while (current != nullptr)
    {
        if (current->nama == _oldnama)
        {
            current->nama = _newnama;
            current->harga = _harga;
            cout << "Data berhasil di update!" << endl;
            return true;
        }

        current = current->next;
    }
    cout << "Data tidak ditemukan!" << endl;
    return false;
}

void tambahdataurutan(string _nama, int _harga, int pos)
{
    skincare *current = head;
    skincare *prevNode = nullptr;
    skincare *newNode = new skincare(_nama, _harga);
    for (int i = 1; pos && current != nullptr; i++)
    {
        prevNode = current;
        current = current->next;
    }
}

```

```

        if (prevNode == nullptr)
        {
            head = newNode;
        }
        else
        {
            prevNode->next = newNode;
            prevNode->prev = prevNode;
        }
        newNode->next = current;
        if (current != nullptr)
        {
            current->prev = newNode;
        }
        else
        {
            tail = newNode;
        }
        cout << "Data berhasil ditambahkan pada posisi " <<
pos << endl;
    }

    void hapusdataurutan(int pos)
    {
        skincare *current = head;
        skincare *prevNode = nullptr;
        for (int i = 1; i < pos && current != nullptr; i++)
        {
            prevNode = current;
            current = current->next;
        }
        if (current == nullptr)
        {
            cout << "Posisi" << pos << "tidak valid" << endl;
            return;
        }
        if (prevNode == nullptr)
        {
            head = current->next;
        }
        else
        {
            prevNode->next = current->next;
        }
        if (current->next != nullptr)
        {
            current->next->prev = prevNode;
        }
        else
    }

```

```

        {
            tail = prevNode;
        }
        delete current;
        cout << "Data pada posisi " << pos << " Berhasil
dihapus!" << endl;
    }
    void hapusseluruhdata()
    {
        skincare *current = head;
        skincare *nextNode = nullptr;
        while (current != nullptr)
        {
            nextNode = current->next;
            delete current;
            current = nextNode;
        }
        head = nullptr;
        tail = nullptr;
        cout << "seluruh data berhasil dihapus!" << endl;
    }

    void tampilkandata()
    {
        if (head == nullptr)
        {
            cout << "tidak ada data yang dapat ditampilkan" <<
endl;
            return;
        }
        skincare *current = head;
        int i = 1;
        while (current != nullptr)
        {
            cout << i << ". Nama :" << current->nama << ",
Harga: Rp" << current->harga << endl;
            current = current->next;
            i++;
        }
    }
};

int main()
{
    tokoskincare toko;
    int pilihan, harga, pos;
    string nama;
    do
    {

```

```

cout << "\n=== Toko Skincare Purwokerto ===" << endl;
cout << "1. Tambah Data" << endl;
cout << "2. Hapus Data" << endl;
cout << "3. Update Data" << endl;
cout << "4. Tambah Data Urutan Tertentu" << endl;
cout << "5. Hapus Data Urutan Tertentu" << endl;
cout << "6. Hapus Seluruh Data" << endl;
cout << "7. Tampilkan Data" << endl;
cout << "8. Exit" << endl;
cout << "Pilihan : ";
cin >> pilihan;
switch (pilihan)
{
case 1:
{
    int jumlah;
    cout << "Masukan jumlah yang diinputkan : ";
    cin >> jumlah;
    for (int q = 1; q <= jumlah; q++)
    {
        cout << "Nama : ";
        cin.ignore();
        getline(cin, nama);
        cout << "harga : ";
        cin >> harga;
        toko.tambahdata(nama, harga);
    }
    break;
}
case 2:
{
    toko.hapusData(nama);
    break;
}
case 3:
{
    string oldnama, newnama;
    cout << "Nama produk lama :";
    cin.ignore();
    getline(cin, oldnama);
    cout << "Nama produk baru :";
    getline(cin, newnama);
    cout << "harga : ";
    cin >> harga;
    toko.updatedata(oldnama, newnama, harga);
    break;
}
case 4:
{

```

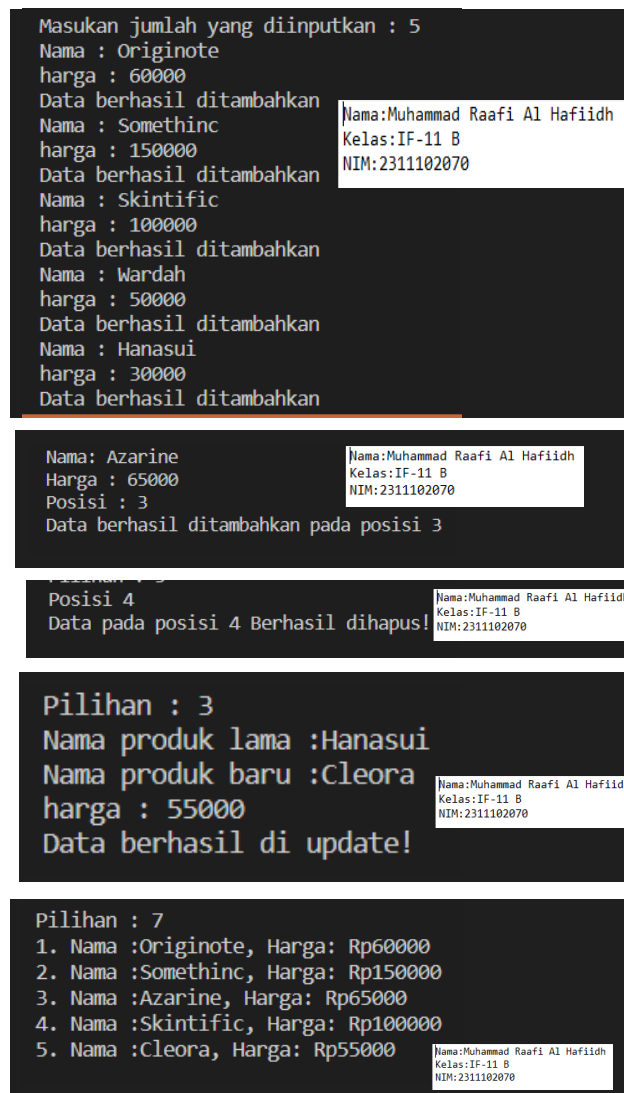
```

        cout << "Nama: ";
        cin.ignore();
        getline(cin, nama);
        cout << "Harga : ";
        cin >> harga;
        cout << "Posisi : ";
        cin >> pos;
        toko.tambahdataurutan(nama, harga, pos);
        break;
    }
    case 5:
    {
        cout << "Posisi ";
        cin >> pos;
        toko.hapusdataurutan(pos);
        break;
    }
    case 6:
    {
        toko.hapusseluruhdata();
        break;
    }
    case 7:
    {
        toko.tampil kandata();
        break;
    }
    case 8:
    {
        cout << "Terima kasih!" << endl;
        break;
    }

    default:
    {
        cout << "Pilihan tidak valid" << endl;
        break;
    }
    }
} while (pilihan != 8);
return 0;
}

```

Screenshoot Program



Deskripsi Program

Pada program di atas merupakan contoh dari penggunaan Double Linked List, program tersebut berfungsi untuk menambahkan data, menghapus, dan melakukan update ditengah/di urutan tertentu yang diminta oleh user. Seperti contoh diatas user melakukan inputan data sebanyak 5 kali di dalam menu tambah data, lalu melakukan penambahan data produk Azarine dengan harga 65.000 diantara Somethinc dan Skintific. Lalu user menggunakan menu hapus data untuk menghapus produk Wardah dengan memasukkan nama data yang ingin dihapus, selanjutnya user merubah data produk Hanasui menjadi Cleora dengan harga 55.000 di menu Update data. Terakhir user akan menampilkan semua data yang sudah diproses sebelumnya.

BAB IV

KESIMPULAN

Kesimpulan yang dapat diambil dari laporan praktikum dengan “*Modul 3 : Single dan Double Linked List*” adalah bahwa praktikum ini bertujuan untuk mempelajari fungsi dari penggunaan Single dan Double Linked List dalam pemrograman. Praktikum jadi bisa memahami perbedaan dari kedua konsep tersebut dan bagaimana kegunaan dari masing-masing konsep yang ada, serta setelah melakukan praktikum, mahasiswa mampu menerapkan kedua konsep tersebut didalam program yang mereka buat. Seperti contohnya praktikum membuat sebuah program untuk menambahkan, mengubah, dan menghapus data diposisi tertentu sesuai dengan apa yang diminta user serta bisa menampilkan data dari program yang menggunakan konsep Single dan Double Linked List.