

LAPORAN PRAKTIKUM

MODUL III SINGLE AND DOUBLE LINKED LIST



**Disusun oleh:
Ilhan Sahal Mansiz
2311102029**

Dosen Pengampu:
Wahyu Andi Saputra, S.Pd., M.Eng

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2023**

BAB I

TUJUAN PRAKTIKUM

1. Mahasiswa memahami perbedaan konsep Single dan Double Linked List
2. Mahasiswa mampu menerapkan Single dan Double Linked List ke dalam pemrograman

BAB II

DASAR TEORI

A. 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.

B. 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.

BAB III

GUIDED

1. Guided 1

Source code

```
#include <iostream>
using namespace std;
/// PROGRAM SINGLE LINKED LIST NON-CIRCULAR
// Deklarasi Struct Node
struct Node
{
    // komponen/member
    int data;
    string kata;
    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, string kata)
```

```

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

// Tambah Belakang
void insertBelakang(int nilai, string kata)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->kata = kata;
    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, string kata, 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;
        baru->kata = kata;
        // 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, string kata)
{
    if (isEmpty() == false)
    {
        head->data = data;
        head->kata = kata;
    }
}

```

```

else
{
    cout << "List masih kosong!" << endl;
}
}
// Ubah Tengah
void ubahTengah(int data, string kata, 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;
            bantu->kata = kata;
        }
    }
    else
    {

```

```

        cout << "List masih kosong!" << endl;
    }
}
// Ubah Belakang
void ubahBelakang(int data, string kata)
{
    if (isEmpty() == false)
    {
        tail->data = data;
        tail->kata = kata;
    }
    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 << "\t";
            cout << bantu->kata;
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

int main()
{
    init();
    insertDepan(3, "satu");
    tampil();
    insertBelakang(5, "dua");
    tampil();
    insertDepan(2, "tiga");
    tampil();
    insertDepan(1, "empat");
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, "lima", 2);
    tampil();
    hapusTengah(2);
}

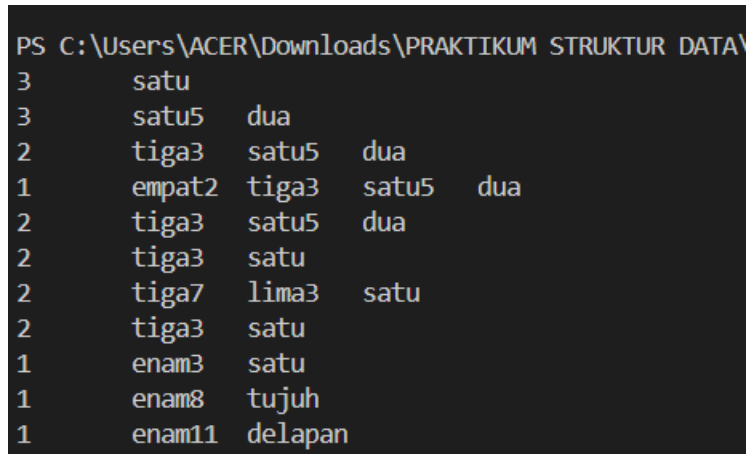
```

```

    tampil();
    ubahDepan(1, "enam");
    tampil();
    ubahBelakang(8, "tujuh");
    tampil();
    ubahTengah(11, "delapan", 2);
    tampil();
    return 0;
}

```

Screenshoot program



```

PS C:\Users\ACER\Downloads\PRAKTIKUM STRUKTUR DATA\
3      satu
3      satu5  dua
2      tiga3  satu5  dua
1      empat2 tiga3  satu5  dua
2      tiga3  satu5  dua
2      tiga3  satu
2      tiga7  lima3  satu
2      tiga3  satu
1      enam3  satu
1      enam8  tujuh
1      enam11 delapan

```

Deskripsi program

Program yang diberikan adalah implementasi dari sebuah linked list non-circular menggunakan bahasa pemrograman C++. Program ini menyediakan fungsi-fungsi dasar untuk manipulasi linked list seperti menambahkan node di depan, di belakang, atau di tengah, menghapus node dari depan, belakang, atau tengah, mengubah nilai node, menghitung jumlah node, serta membersihkan seluruh isi linked list. Setiap node dalam linked list memiliki dua data, yaitu sebuah bilangan bulat (int) dan sebuah string. Program juga menyediakan fungsi untuk menampilkan isi dari linked list. Di dalam fungsi main(), program melakukan serangkaian operasi pada linked list seperti menambahkan node baru, menghapus

node, mengubah nilai node, dan menampilkan isi linked list setelah setiap operasi dilakukan.

2. Guided 2

Source Code

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

class Node {
public:
    int data;
    string kata;
    Node *prev;
    Node *next;
};

class DoublyLinkedList {
public:
    Node *head;
    Node *tail;
    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }
    void push(int data, string kata) {
        Node *newNode = new Node;
        newNode->data = data;
        newNode->kata = kata;
        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, string oldKata, string newKata) {
        Node *current = head;
        while (current != nullptr) {
            if (current->data == oldData && current->kata == oldKata) {
                current->data = newData;
                current->kata = newKata;
                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->kata << " ";
            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;
        string kata;
        cout << "Enter data to add: ";
        cin >> data;
        cout << "Enter kata to add: ";
        cin >> kata;
        list.push(data, kata);
        break;
    }
    case 2: {
        list.pop();
        break;
    }
    case 3: {
        int oldData, newData;
        string oldKata, newKata;
        cout << "Enter old data: ";
        cin >> oldData;
        cout << "Enter new data: ";
        cin >> newData;
        cout << "Enter old kata: ";
        cin >> oldKata;
        cout << "Enter new kata: ";
        cin >> newKata;
        bool updated = list.update(oldData, newData, oldKata, newKata);
        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;
}

```

Screenshot Program

```

Enter your choice: 1
Enter data to add: 8
Enter kata to add: awe
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 1
Enter data to add: 9
Enter kata to add: kucing
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
9 kucing 8 awe

```

Deskripsi Program

Program yang diberikan adalah implementasi dari sebuah doubly linked list menggunakan bahasa pemrograman C++. Program ini menyediakan kelas `Node` yang merepresentasikan sebuah node dalam linked list, dengan setiap node memiliki data integer (`int`) dan string (`string`) serta dua pointer yang menunjukkan ke node sebelumnya dan node berikutnya. Kelas `DoublyLinkedList` memiliki pointer kepala (`head`) dan pointer ekor (`tail`) yang menunjukkan ke node pertama dan terakhir dalam linked list. Program ini menyediakan fungsi untuk menambahkan data ke depan linked list (`push()`), menghapus data dari depan linked list (`pop()`), mengubah data

dalam linked list (`update()`), menghapus seluruh data dalam linked list (`deleteAll()`), dan menampilkan seluruh data dalam linked list (`display()`). Di dalam fungsi `main()`, program menampilkan menu pilihan kepada pengguna untuk menambah, menghapus, mengubah, menghapus seluruh, atau menampilkan data dalam linked list, dan kemudian menjalankan operasi sesuai dengan pilihan pengguna tersebut.

LATIHAN KELAS - UNGUIDED

1. Soal Mengenai Single Linked List

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]
John	19
Jane	20
Michael	18
Yusuke	19
Akechi	20
Hoshino	18
Karin	18

- b) Hapus data Akechi
- c) Tambahkan data berikut diantara John dan Jane : Futaba 18
- d) Tambahkan data berikut diawal : Igor 20
- e) Ubah data Michael menjadi : Reyn 18
- f) Tampilkan Seluruh data

Source code

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

class dataMahasiswa
{
public:
    string nama;
    int usia;

    dataMahasiswa *next;

    dataMahasiswa(string nama, int usia)
    {
        this->nama = nama;
        this->usia = usia;
        next = nullptr;
    }
};

class linkedlist
{
private:
    dataMahasiswa *head;
    dataMahasiswa *tail;

public:
    linkedlist()
    {
        head = nullptr;
        tail = nullptr;
    }
}
```

```

void datamasuk(string nama, int usia)
{
    dataMahasiswa *newnode = new dataMahasiswa(nama, usia);
    if (head == nullptr)
    {
        head = newnode;
        tail = newnode;
        return;
    }
    newnode->next = nullptr;
    tail->next = newnode;
    tail = tail->next;
}

void updatenode(string oldname, string newname, int newage)
{
    if (head == nullptr)
    {
        cout << "list kosong\n";
        return;
    }
    dataMahasiswa *current = head;
    while (current != nullptr && current->nama != oldname)
    {
        current = current->next;
    }
    if (current == nullptr)
    {
        cout << "data tidak ditemukan\n";
        return;
    }
    current->nama = newname;
    current->usia = newage;
    cout << "data telah diupdate\n";
}

```

```

    }

    void tampilandata()
    {
        if (head == nullptr)
        {
            cout << "list kosong\n";
            return;
        }
        dataMahasiswa *current = head;
        cout << endl;
        while (current != nullptr)
        {
            cout << current->nama << "[" << current->usia << "]"
<< endl;
            current = current->next;
        }
        cout << endl;
    }

    void hapusdata(string nama)
    {
        if (head == nullptr)
        {
            cout << "list kosong\n";
            return;
        }
        dataMahasiswa *current = head;
        dataMahasiswa *previous = nullptr;
        while (current != nullptr && current->nama != nama)
        {
            previous = current;
            current = current->next;
        }
        if (current == nullptr)

```

```

    {
        cout << "data tidak ditemukan\n";
        return;
    }
    if (current == head)
    {
        head = head->next;
    }
    else
    {
        previous->next = current->next;
    }
    if (current == tail)
    {
        tail = previous;
    }
    delete current;
}

void tambahnode(string nama, int usia, int pos)
{
    dataMahasiswa *newnode = new dataMahasiswa(nama, usia);
    newnode->next = nullptr;
    dataMahasiswa *temp = head;
    int count = 1;
    while (temp != nullptr && count < pos)
    {
        temp = temp->next;
        count++;
    }
    if (temp == nullptr)
    {
        cout << "posisi yang diminta tidak valid" << endl;
        return;
    }
}

```

```

        newnode->next = temp->next;
        temp->next = newnode;
    }
};

int main()
{
    linkedlist list;
    int pilihan, usia, usiabar, pos;
    string nama, namabar;

    do
    {
        cout << "pilih" << endl;
        cout << "1. masukan data" << endl;
        cout << "2. tambahkan data" << endl;
        cout << "3. tampilkan data" << endl;
        cout << "4. ubah data" << endl;
        cout << "5. menghapus data" << endl;
        cout << "0. keluar" << endl;
        cout << "masukan pilihan :";
        cin >> pilihan;
        cin.ignore();
        cout << endl;
        switch (pilihan)
        {
            case 1:
                cout << "masukan nama :";
                cin >> nama;
                cout << "masukan usia :";
                cin >> usia;
                list.datamasuk(nama, usia);
                cout << endl;
                break;
            case 2:

```



```

        cout << "menambahkan data baru :";
        cout << "masukan nama :";
        cin >> namabaru;
        cout << "masukan usia: ";
        cin >> usia;
        cout << "masukan data ke posisi ke-";
        cin >> pos;
        if (pos < 1)
        {
            cout << "posisi yang diminta tidak valid" <<
endl;
        }
        else
        {
            list.tambahnode(namabaru, usia, pos);
        }
        break;
case 3:
    list.tampilandata();
    cout << endl;
    break;
case 4:
    cout << "masukan nama lama :";
    cin >> nama;
    cout << "masukan nama baru :";
    cin >> namabaru;
    cout << "masukan usia baru :";
    cin >> usiabaru;
    list.updatenode(nama, namabaru, usiabaru);
    cout << endl;
    break;
case 5:
    cout << "masukan nama yang ingin dihapus :";
    cin >> nama;

```

```
        list.hapusdata(nama);  
        cout << endl;  
        break;  
    default:  
        break;  
    }  
    } while (pilihan != 0);  
    return 0;  
}
```

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

```
pilih  
1. masukan data  
2. tambahkan data  
3. tampilkan data  
4. ubah data  
5. menghapus data  
0. keluar  
masukan pilihan :3
```

```
Ilhan[19]  
John[19]  
Jane[20]  
Michael[18]  
Yusuke[19]  
Akechi[20]  
Hoshino[18]  
Karin[18]
```

b) Hapus Data Akechi

```
pilih
1. masukan data
2. tambahkan data
3. tampilkan data
4. ubah data
5. menghapus data
0. keluar
masukan pilihan :5

masukan nama yang ingin dihapus :Akechi

pilih
1. masukan data
2. tambahkan data
3. tampilkan data
4. ubah data
5. menghapus data
0. keluar
masukan pilihan :3

Ilhan[19]
John[19]
Jane[20]
Michael[18]
Yusuke[19]
Hoshino[18]
Karin[18]
```

c) Tambahkan data berikut diantara john dan jane : Futaba 18

```
pilih
1. masukan data
2. tambahakan data
3. tampilkan data
4. ubah data
5. menghapus data
0. keluar
masukan pilihan :2

menambahkan data baru :masukan nama :Futaba
masukan usia: 18
masukan data ke posisi ke-2
pilih
1. masukan data
2. tambahakan data
3. tampilkan data
4. ubah data
5. menghapus data
0. keluar
masukan pilihan :3

Ilhan[19]
John[19]
Futaba[18]
Jane[20]
Michael[18]
Yusuke[19]
Hoshino[18]
Karin[18]
```

d) Tambahkan data berikut diawal : Igor 20

```
pilih
1. masukan data
2. tambahkan data
3. tampilkan data
4. ubah data
5. menghapus data
0. keluar
masukan pilihan :2

menambahkan data baru :masukan nama :Igor
masukan usia: 20
masukan data ke posisi ke-1
pilih
1. masukan data
2. tambahkan data
3. tampilkan data
4. ubah data
5. menghapus data
0. keluar
masukan pilihan :3

Ilhan[19]
Igor[20]
John[19]
Futaba[18]
Jane[20]
Michael[18]
Yusuke[19]
Hoshino[18]
Karin[18]
```

e) Ubah data Michael menjadi : Reyn 18

```
1. masukan data
2. tambahakan data
3. tampilkan data
4. ubah data
5. menghapus data
0. keluar
masukan pilihan :4

masukan nama lama :Michael
masukan nama baru :Reyn
masukan usia baru :18
data telah diupdate
```

```
pilih
1. masukan data
2. tambahakan data
3. tampilkan data
4. ubah data
5. menghapus data
0. keluar
masukan pilihan :3
```

```
Ilhan[19]
Igor[20]
John[19]
Futaba[18]
Jane[20]
Reyn[18]
Yusuke[19]
Hoshino[18]
Karin[18]
```

f) Tampilkan Seluruh data

```
Ilhan[19]
Igor[20]
John[19]
Futaba[18]
Jane[20]
Reyn[18]
Yusuke[19]
Hoshino[18]
Karin[18]
```

Deskripsi Program

Program yang diberikan adalah implementasi dari sebuah linked list dalam bahasa pemrograman C++, yang digunakan untuk menyimpan data mahasiswa. Program ini memanfaatkan dua kelas, yaitu `dataMahasiswa` yang merepresentasikan data setiap mahasiswa dalam linked list, dengan atribut nama dan usia, serta pointer next yang menunjukkan ke node berikutnya. Kelas kedua adalah `linkedlist` yang memiliki fungsi-fungsi untuk menambahkan data mahasiswa ke dalam linked list (`datamasuk()`), menampilkan seluruh data mahasiswa (`tampilandata()`), mengubah data mahasiswa (`updatenode()`), menghapus data mahasiswa (`hapusdata()`), dan menambahkan data mahasiswa baru di posisi tertentu (`tambahnode()`). Di dalam fungsi `main()`, program menampilkan menu pilihan kepada pengguna untuk melakukan operasi-operasi tersebut, dan menjalankan operasi sesuai dengan pilihan pengguna hingga pengguna memilih untuk keluar dari program.

2. Soal mengenai Double Linked List

Modifikasi Guided Double Linked List dilakukan dengan penambahan operasi untuk menambah 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	100.000
Wardah	50.000
Hanasui	30.000

Case :

1. Tambahkan produk Azarine dengan harga 65.000 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>
#include <string>
using namespace std;

struct Node {
    string nama_produk;
    int harga;
    Node* prev;
    Node* next;
};

class DoubleLinkedList {
private:
    Node* head;
    Node* tail;
    int size;

public:
    DoubleLinkedList() {
        head = nullptr;
        tail = nullptr;
        size = 0;
    }

    void tambahData(string nama_produk, int harga, int posisi) {
        if (posisi < 0 || posisi > size) {
            cout << "posisi tidak valid." << endl;
            return;
        }
        Node* newNode = new Node;
        newNode->nama_produk = nama_produk;
        newNode->harga = harga;
```

```

        if (size == 0) {
            head = newNode;
            tail = newNode;
            newNode->prev = nullptr;
            newNode->next = nullptr;
        } else if (posisi == 0) {
            newNode->next = head;
            head->prev = newNode;
            head = newNode;
            newNode->prev = nullptr;
        } else if (posisi == size) {
            tail->next = newNode;
            newNode->prev = tail;
            tail = newNode;
            newNode->next = nullptr;
        } else {
            Node* temp = head;
            for (int i = 0; i < posisi - 1; i++) {
                temp = temp->next;
            }
            newNode->prev = temp;
            newNode->next = temp->next;
            temp->next->prev = newNode;
            temp->next = newNode;
        }
        size++;
        cout << "Data berhasil ditambahkan." << endl;
    }

    void hapusData(int posisi) {
        if (posisi < 0 || posisi >= size) {
            cout << "posisi tidak valid." << endl;
            return;
        }
    }

```

```

        if (size == 1) {
            delete head;
            head = nullptr;
            tail = nullptr;
        } else if (posisi == 0) {
            Node* temp = head;
            head = head->next;
            head->prev = nullptr;
            delete temp;
        } else if (posisi == size - 1) {
            Node* temp = tail;
            tail = tail->prev;
            tail->next = nullptr;
            delete temp;
        } else {
            Node* temp = head;
            for (int i = 0; i < posisi; i++) {
                temp = temp->next;
            }
            temp->prev->next = temp->next;
            temp->next->prev = temp->prev;
            delete temp;
        }
        size--;
        cout << "Data berhasil dihapus." << endl;
    }

```

```

    void updateData(string data_lama, string data_baru, int
harga_baru) {
        Node* temp = head;
        bool found = false;
        while (temp != nullptr) {
            if (temp->nama_produk == data_lama) {
                temp->nama_produk = data_baru;
            }
        }
    }

```

```

        temp->harga = harga_baru;
        found = true;
        cout << "Data berhasil diperbarui." << endl;
        break;
    }
    temp = temp->next;
}
if (!found) {
    cout << "Data tidak ditemukan." << endl;
}
}

void tampilkanData() {
    if (size == 0) {
        cout << "Tidak ada data yang tersedia." << endl;
        return;
    }

    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->nama_produk << " [ " << temp->harga << "]"
<< endl;
        temp = temp->next;
    }
}

void deleteAllData() {
    Node* temp = head;
    while (temp != nullptr) {
        Node* nextNode = temp->next;
        delete temp;
        temp = nextNode;
    }
    head = nullptr;
}

```

```

        tail = nullptr;
        size = 0;
        cout << "Seluruh data berhasil dihapus." << endl;
    }

    void deleteData(int posisi) {
        if (posisi <= 0 || posisi > size) {
            cout << "Posisi tidak valid." << endl;
            return;
        }
        Node* temp = head;
        if (posisi == 1) {
            head = head->next;
            if (head != nullptr) {
                head->prev = nullptr;
            } else {
                tail = nullptr;
            }
        } else if (posisi == size) {
            temp = tail;
            tail = tail->prev;
            tail->next = nullptr;
        } else {
            int i = 1;
            while (i < posisi) {
                temp = temp->next;
                i++;
            }
            temp->prev->next = temp->next;
            temp->next->prev = temp->prev;
        }
        delete temp;
        size--;
    }

```

```

        cout << "Data pada posisi " << posisi << " berhasil dihapus."
<< endl;
    };
};

int main() {
    DoubleLinkedList list;
    string nama_produk, data_lama, data_baru;
    int pilihan, posisi, harga, harga_baru;
    do {
        cout << endl;
        cout << "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 << "Masukan pilihan anda : ";
        cin >> pilihan;
        cin.ignore();
        cout << endl;
        switch (pilihan) {
            case 1:
                cout << "Masukan nama produk : ";
                cin >> nama_produk;
                cout << "Masukan harga produk : ";
                cin >> harga;
                cout << "Masukan posisi : ";
                cin >> posisi;
                list.tambahData(nama_produk, harga, posisi);
                break;

```

```
case 2:
    cout << "Posisi list produk yang akan dihapus : ";
    cin >> posisi;
    list.hapusData(posisi);
    break;
case 3:
    cout << "Nama produk lama : ";
    cin >> data_lama;
    cout << "Nama produk baru : ";
    cin >> data_baru;
    cout << "Harga baru : ";
    cin >> harga_baru;
    list.updateData(data_lama, data_baru, harga_baru);
    break;
case 4:
    cout << "Masukan nama produk : ";
    cin >> nama_produk;
    cout << "Masukan harga produk : ";
    cin >> harga;
    cout << "Masukan posisi : ";
    cin >> posisi;
    list.tambahData(nama_produk, harga, posisi);
    break;
case 5:
    cout << "Masukan posisi (mulai dari 1): ";
    cin >> posisi;
    list.deleteData(posisi);
    break;
case 6:
    list.deleteAllData();
    break;
case 7:
    list.tampilkanData();
    break;
```



```
    }  
    } while (pilihan != 8);  
    return 0;  
}
```

```
Masukan posisi : 4  
Data berhasil ditambahkan.  
  
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  
Masukan pilihan anda : 7  
  
Originote [ 60000]  
Somethinc [ 150000]  
Skintific [ 100000]  
Wardah [ 50000]  
Hanasui [ 30000]
```

1. Tambahkan produk Azarine dengan harga 65.000 diantara Somethinc dan Skintific

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

Masukan pilihan anda : 4

Masukan nama produk : Azarine

Masukan harga produk : 65000

Masukan posisi : 2

Data berhasil ditambahkan.

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
```

Masukan pilihan anda : 7

Originote [60000]

Somethinc [150000]

Azarine [65000]

Skintific [100000]

Wardah [50000]

Hanasui [30000]

2. Hapus produk wardah

```
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
Masukan pilihan anda : 2

Posisi list produk yang akan dihapus : 4
Data berhasil dihapus.

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
Masukan pilihan anda : 7

Originote [ 60000]
Somethinc [ 150000]
Azarine [ 65000]
Skintific [ 100000]
Hanasui [ 30000]
```

3. Update produk Hanasui menjadi Cleora dengan harga 55.000

```
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
```

Masukan pilihan anda : 3

```
Nama produk lama : Hanasui
Nama produk baru : Cleora
Harga baru : 55000
Data berhasil diperbarui.
```

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
```

Masukan pilihan anda : 7

```
Originote [ 60000]
Somethinc [ 150000]
Azarine [ 65000]
Skintific [ 100000]
Cleora [ 55000]
```

4. Pada menu 7, tampilan akhirnya akan menjadi 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
Masukan pilihan anda : 7

Originote [ 60000]
Somethinc [ 150000]
Azarine [ 65000]
Skintific [ 100000]
Cleora [ 55000]
```

Deskripsi Program

Program yang diberikan adalah sebuah implementasi dari doubly linked list dalam bahasa pemrograman C++, yang digunakan untuk menyimpan data tentang produk-produk skincare. Program ini menyediakan fungsi-fungsi untuk menambah data produk skincare dengan harga tertentu, menghapus data produk berdasarkan posisi tertentu, mengupdate data produk, menampilkan seluruh data produk, serta menghapus seluruh data produk yang tersimpan. Pengguna dapat memilih operasi yang ingin dilakukan melalui menu yang ditampilkan, dan program akan menjalankan operasi sesuai dengan pilihan pengguna tersebut hingga pengguna memilih untuk keluar dari program.

BAB IV

KESIMPULAN

Dari modul yang mencakup implementasi single linked list dan double linked list, dapat disimpulkan bahwa kedua struktur data tersebut merupakan metode yang berguna untuk menyimpan dan mengorganisir data secara terhubung. Single linked list memiliki simpul-simpul yang hanya menunjuk ke simpul berikutnya, sementara double linked list memiliki simpul-simpul yang menunjuk ke simpul sebelumnya dan berikutnya. Kelebihan double linked list terletak pada kemampuannya untuk melakukan traversing mundur, sementara single linked list memiliki overhead memori yang lebih kecil karena hanya membutuhkan satu pointer untuk setiap simpul. Pemilihan antara kedua jenis linked list tergantung pada kebutuhan spesifik dan kompleksitas operasi yang diinginkan dalam aplikasi yang sedang dikembangkan.

BAB V

REFERENSI

- <https://www.geeksforgeeks.org/introduction-and-insertion-in-a-doubly-linked-list/>
- <https://www.programiz.com/dsa/doubly-linked-list>
- <https://www.trivusi.web.id/2022/07/struktur-data-linked-list.html>