

LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA

MODUL III SINGLE AND DOUBLE LINKED LIST



Disusun Oleh :
NAUFAL THORIQ MUZHAFAR
2311102078

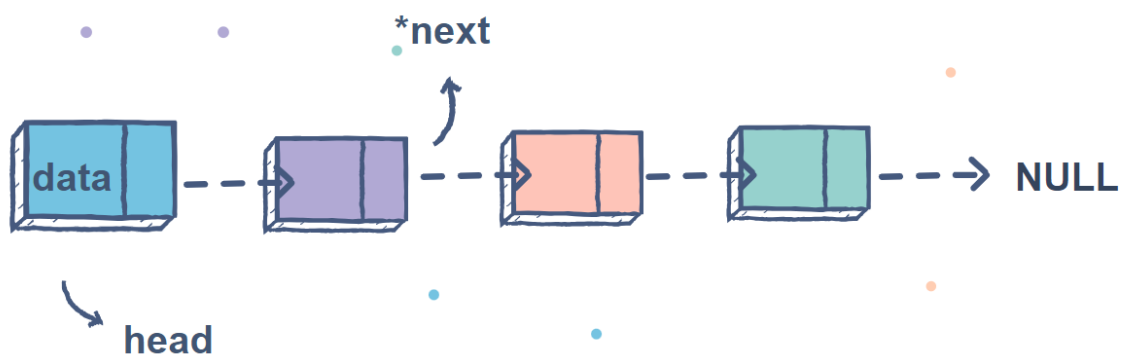
Dosen
WAHYU ANDI SAPUTRA, S.Pd., M.Eng

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

A. Dasar Teori

Single Linked list

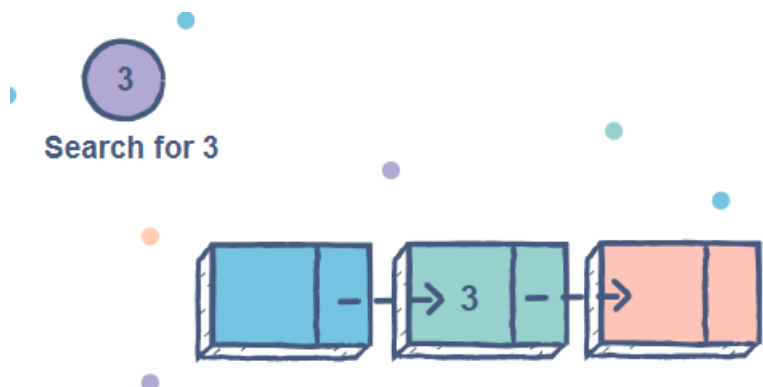
Adalah jenis linked list yang bersifat searah, yaitu dapat dilewati hanya dalam satu arah dari head hingga node terakhir. Setiap elemen dalam linked list disebut node. Sebuah node berisi data dan pointer ke node berikutnya, yang membantu menjaga struktur dalam list. Head adalah penunjuk yang menunjuk ke node pertama dari list agar membantu data dalam melewati list. Node terakhir menunjuk ke *nullptr*, yang membantu kita menentukan kapan list berakhir.



Beberapa operasi umum linked list

1. Cari node di dalam list

Kamu dapat menentukan dan mengambil node tertentu dari depan, akhir, atau di mana saja dalam daftar. Kasus terburuk untuk mengambil node dari mana pun dalam daftar adalah $O(n)$.

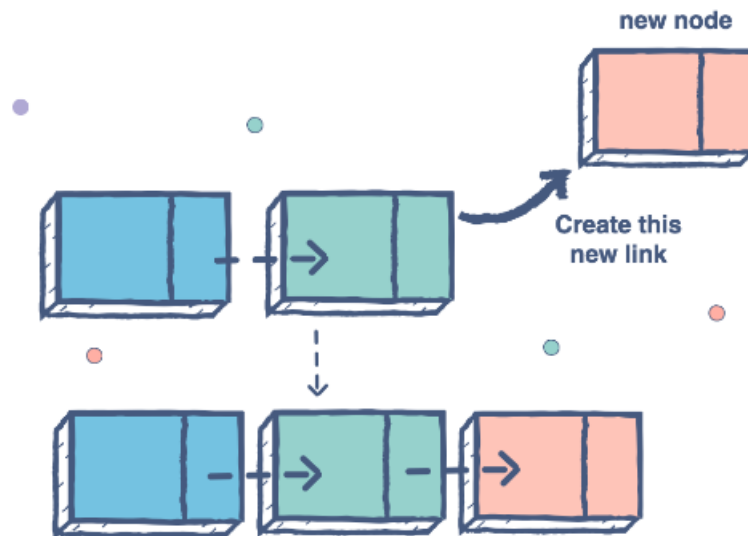


2. Menambah node ke dalam list

Kamu dapat menambahkan simpul di depan, akhir, atau di mana pun dalam linked list. Kasus terburuk untuk melakukan operasi ini adalah:

- Tambahkan item ke depan list: $O(1)$

- Tambahkan item ke akhir list: $O(n)$
- Tambahkan item di mana saja dalam list: $O(n)$



3. Menghapus node dari list

Kamu dapat menghapus node dari depan, akhir, atau di mana pun dalam list.

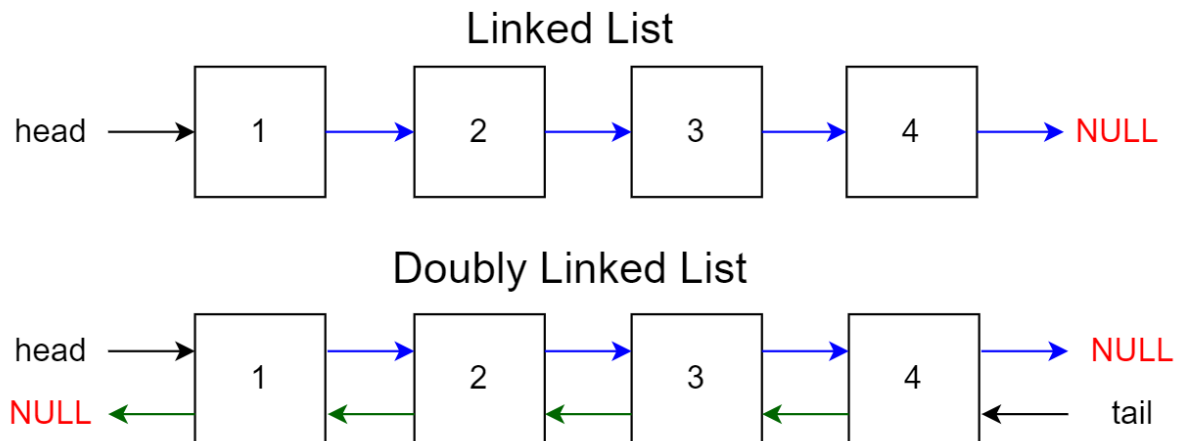
Kasus terburuk untuk melakukan operasi ini adalah:

- Hapus item dari depan list: $O(1)$
- Hapus item dari akhir list: $O(n)$
- Hapus item dari mana saja dalam list: $O(n)$

Double Linked List

Double linked list pada dasarnya sama dengan Single linked list. Satu-satunya perbedaan adalah bahwa node memiliki sebuah pointer, biasa disebut prev, yang menunjuk ke node sebelumnya dalam list. Selain itu, double linked list sering kali memiliki penunjuk tail. Dengan adanya pointer prev, Double Linked List memungkinkan untuk melakukan operasi penghapusan dan penambahan pada simpul mana saja secara efisien. Setiap simpul pada Double Linked List memiliki tiga elemen penting, yaitu elemen data (biasanya berupa nilai), pointer next yang menunjuk ke simpul berikutnya, dan pointer prev yang menunjuk ke simpul sebelumnya.

Blue arrows are next
Green arrows are prev



B. Guided

Guided 1 (Single Linked List)

Source Code

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

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

Node* head;
Node* tail;

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

// Pengecekan apakah list kosong
bool isEmpty() {
    return head == NULL;
}

// Tambah Node di depan
void insertDepan(int nilai) {
    Node* baru = new Node;
```

```

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

// Tambah Node di belakang
void insertBelakang(int nilai) {
    Node* baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty()) {
        head = tail = baru;
    } else {
        tail->next = baru;
        tail = baru;
    }
}

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

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

```

```

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

// Hapus Node di depan
void hapusDepan() {
    if (!isEmpty()) {
        Node* hapus = head;
        if (head->next != NULL) {
            head = head->next;
            delete hapus;
        } else {
            head = tail = NULL;
            delete hapus;
        }
    } else {
        cout << "List kosong!" << endl;
    }
}

// Hapus Node di belakang
void hapusBelakang() {
    if (!isEmpty()) {
        if (head != tail) {
            Node* hapus = tail;
            Node* 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 Node di posisi tengah
void hapusTengah(int posisi) {
    if (posisi < 1 || posisi > hitungList()) {
        cout << "Posisi diluar jangkauan" << endl;
    } else if (posisi == 1) {
        cout << "Posisi bukan posisi tengah" << endl;
    }
}

```

```

    } else {
        Node* hapus;
        Node* bantu = head;
        for (int nomor = 1; nomor < posisi - 1; nomor++) {
            bantu = bantu->next;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    }
}

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

// Ubah data Node di posisi tengah
void ubahTengah(int data, int posisi) {
    if (!isEmpty()) {
        if (posisi < 1 || posisi > hitungList()) {
            cout << "Posisi di luar jangkauan" << endl;
        } else if (posisi == 1) {
            cout << "Posisi bukan posisi tengah" << endl;
        } else {
            Node* bantu = head;
            for (int nomor = 1; nomor < posisi; nomor++) {
                bantu = bantu->next;
            }
            bantu->data = data;
        }
    } else {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah data Node di belakang
void ubahBelakang(int data) {
    if (!isEmpty()) {
        tail->data = data;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

```

```

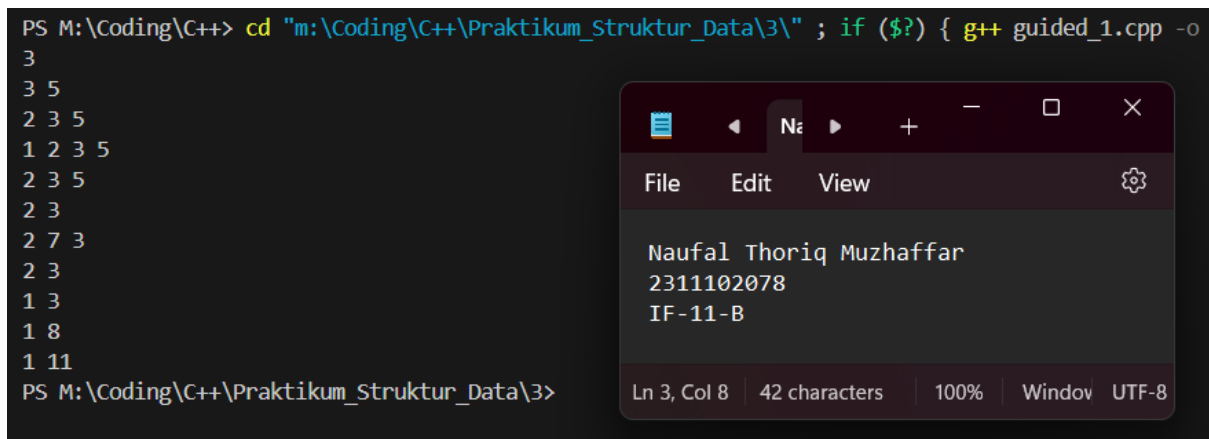
// Hapus semua Node di list
void clearList() {
    Node* bantu = head;
    while (bantu != NULL) {
        Node* hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}

// Tampilkan semua data Node di list
void tampil() {
    if (!isEmpty()) {
        Node* bantu = head;
        while (bantu != NULL) {
            cout << bantu->data << " ";
            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;
}

```


Screenshots Output



```
PS M:\Coding\C++> cd "m:\Coding\C++\Praktikum_Struktur_Data\3\" ; if ($?) { g++ guided_1.cpp -o 3
3
3 5
2 3 5
1 2 3 5
2 3 5
2 3
2 7 3
2 3
1 3
1 8
1 11
PS M:\Coding\C++\Praktikum_Struktur_Data\3>
```

Deskripsi Program

Program ini merupakan implementasi dari linked list sederhana yang memungkinkan operasi dasar seperti penambahan, penghapusan, dan perubahan data pada linked list. Program juga menyediakan fungsi untuk mengecek apakah linked list kosong, menghitung jumlah node, dan menampilkan seluruh data dalam linked list.

Guided 2

Source Code (Double Linked List)

```
#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 = temp;
        }
    }
};
```

```

        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;
        }
    }
}

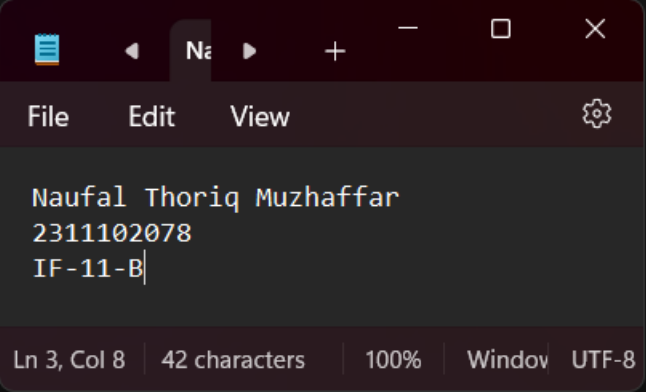
return 0;

```

```
}
```

Screenshots Output

```
PS M:\Coding\C++> cd "m:\Coding\C++\Praktikum_Struktur_Data\3\" ; if ($?) { g++
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: 21
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: 5
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
5 21
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 4
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
1. Add data
2. Delete data
3. Update data
```



Deskripsi Program

Program ini merupakan implementasi dari Doubly Linked List dalam bahasa pemrograman C++. Doubly Linked List adalah struktur data berantai di mana setiap node memiliki dua pointer, yaitu pointer ke node sebelumnya (prev) dan pointer ke node selanjutnya (next). Program ini dilengkapi dengan fitur-fitur untuk menambah data ke awal daftar (push), menghapus data dari awal daftar (pop), mengupdate data, menghapus semua data, dan menampilkan semua data yang tersedia dalam daftar. Untuk memilih operasi yang ingin dilakukan pada doubly linked list, program ini menggunakan menu sederhana.

C. Unguided

Unguided 1

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>
using namespace std;

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

Node *head;
Node *tail;

void init()
{
```

```

    head = NULL;
    tail = NULL;
}

bool isEmpty()
{
    return head == NULL;
}

void insertDepan(string nama, int usia)
{
    Node *baru = new Node;
    baru->nama = nama;
    baru->usia = usia;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}

void insertBelakang(string nama, int usia)
{
    Node *baru = new Node;
    baru->nama = nama;
    baru->usia = usia;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}

int hitungList()
{
    Node *hitung = head;
    int jumlah = 0;
    while (hitung != NULL)

```



```

    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

void insertTengah(string nama, int usia, 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 = new Node();
        baru->nama = nama;
        baru->usia = usia;
        Node *bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

void hapusDepan()
{
    if (!isEmpty())
    {
        Node *hapus = head;
        if (head->next != NULL)
        {
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
            delete hapus;
        }
    }
}

```

```

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

void hapusBelakang()
{
    if (!isEmpty())
    {
        if (head != tail)
        {
            Node *hapus = tail;
            Node *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;
    }
}

void hapusTengah(int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        Node *hapus;

```

```

        Node *bantu = head;
        for (int nomor = 1; nomor < posisi - 1; nomor++)
        {
            bantu = bantu->next;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    }
}

void ubahDepan(string nama, int usia)
{
    if (!isEmpty())
    {
        head->nama = nama;
        head->usia = usia;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

void ubahTengah(string nama, int usia, int posisi)
{
    if (!isEmpty())
    {
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "Posisi di luar jangkauan" << endl;
        }
        else if (posisi == 1)
        {
            cout << "Posisi bukan posisi tengah" << endl;
        }
        else
        {
            Node *bantu = head;
            for (int nomor = 1; nomor < posisi; nomor++)
            {
                bantu = bantu->next;
            }
            bantu->nama = nama;
            bantu->usia = usia;
        }
    }
    else

```

```

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

void ubahBelakang(string nama, int usia)
{
    if (!isEmpty())
    {
        tail->nama = nama;
        tail->usia = usia;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

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

void tampil()
{
    if (!isEmpty())
    {
        Node *bantu = head;
        while (bantu != NULL)
        {
            cout << bantu->nama << " ";
            cout << bantu->usia << " , ";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

```

```

}

int main()
{
    init();
    int menu, usia, posisi;
    string nama;
    cout << "\n# Menu Linked List Mahasiswa #" << endl;
    do
    {
        cout << "\n 1. Insert Depan"
            << "\n 2. Insert Belakang"
            << "\n 3. Insert Tengah"
            << "\n 4. Hapus Depan"
            << "\n 5. Hapus Belakang"
            << "\n 6. Hapus Tengah"
            << "\n 7. Ubah Depan"
            << "\n 8. Ubah Belakang"
            << "\n 9. Ubah Tengah"
            << "\n 10. Tampilkan"
            << "\n 0. Keluar Program"
            << "\n Pilihan : ";

        cin >> menu;
        switch (menu)
        {
            case 1:
                cout << "Masukkan Nama : ";
                cin >> nama;
                cout << "Masukkan Usia : ";
                cin >> usia;
                insertDepan(nama, usia);
                cout << endl;
                tampil();
                break;
            case 2:
                cout << "Masukkan Nama : ";
                cin >> nama;
                cout << "Masukkan Usia : ";
                cin >> usia;
                insertBelakang(nama, usia);
                cout << endl;
                tampil();
                break;
            case 3:
                cout << "Masukkan Posisi : ";
                cin >> posisi;
                cout << "Masukkan Nama : ";
                cin >> nama;

```

```
        cout << "Masukkan Usia : ";
        cin >> usia;
        insertTengah(nama, usia, posisi);
        cout << endl;
        tampil();
        break;
    case 4:
        hapusDepan();
        cout << endl;
        tampil();
        break;
    case 5:
        hapusBelakang();
        cout << endl;
        tampil();
        break;
    case 6:
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        hapusTengah(posisi);
        cout << endl;
        tampil();
        break;
    case 7:
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan Usia : ";
        cin >> usia;
        ubahDepan(nama, usia);
        cout << endl;
        tampil();
        break;
    case 8:
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan Usia : ";
        cin >> usia;
        ubahBelakang(nama, usia);
        cout << endl;
        tampil();
        break;
    case 9:
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan Usia : ";
        cin >> usia;
```

```

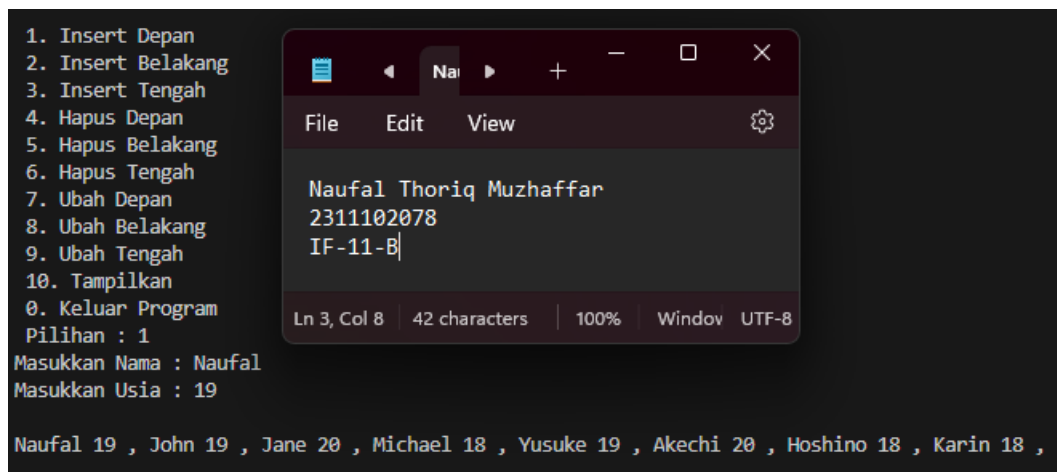
        ubahTengah(nama, usia, posisi);
        cout << endl;
        tampil();
        break;
    case 10:
        tampil();
        break;

    default:
        cout << "Pilihan Salah" << endl;
        break;
    }
} while (menu != 0);
return 0;
}

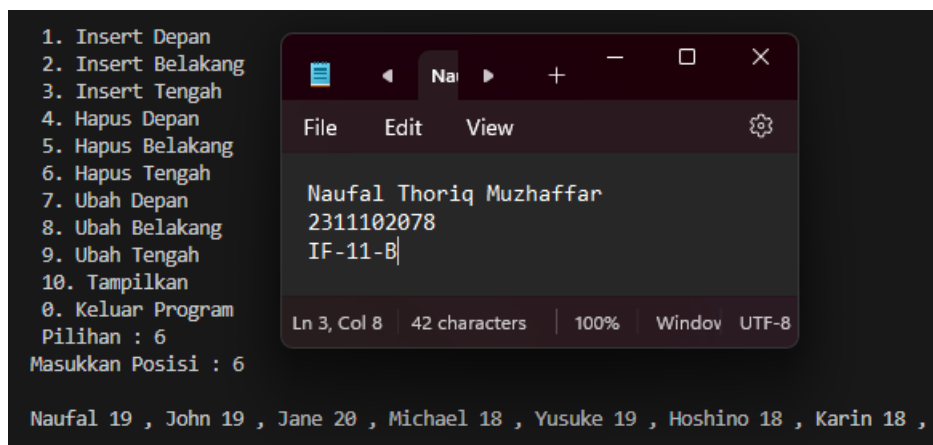
```

Screenshots Output

1. Data pertama yang dimasukkan adalah nama dan usia anda.



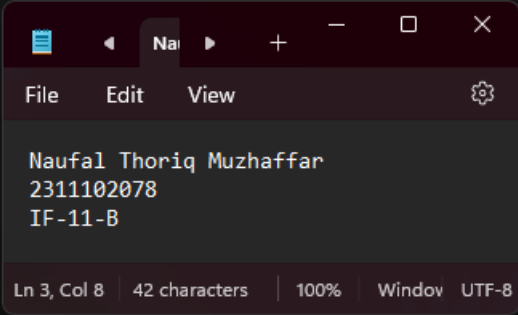
2. Hapus data akechi.



3. Tambah data Futaba 18 diantara John dan Jane.

```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 3
Masukkan Posisi : 3
Masukkan Nama : Futaba
Masukkan Usia : 18

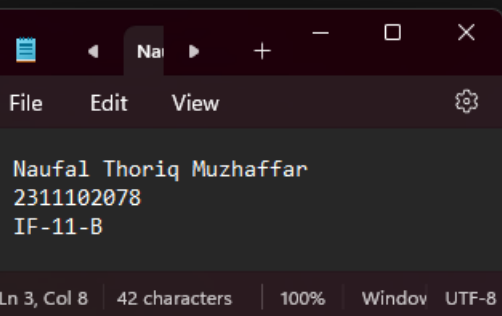
Naufal 19 , John 19 , Futaba 18 , Jane 20 , Michael 18 , Yusuke 19 , Hoshino 18 , Karin 18 ,
```



4. Tambahkan data Igor 20 diawal.

```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 1
Masukkan Nama : Igor
Masukkan Usia : 20

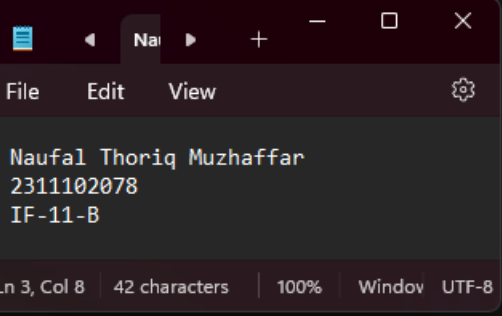
Igor 20 , Naufal 19 , John 19 , Futaba 18 , Jane 20 , Michael 18 , Yusuke 19 , Hoshino 18 , Karin 18 ,
```



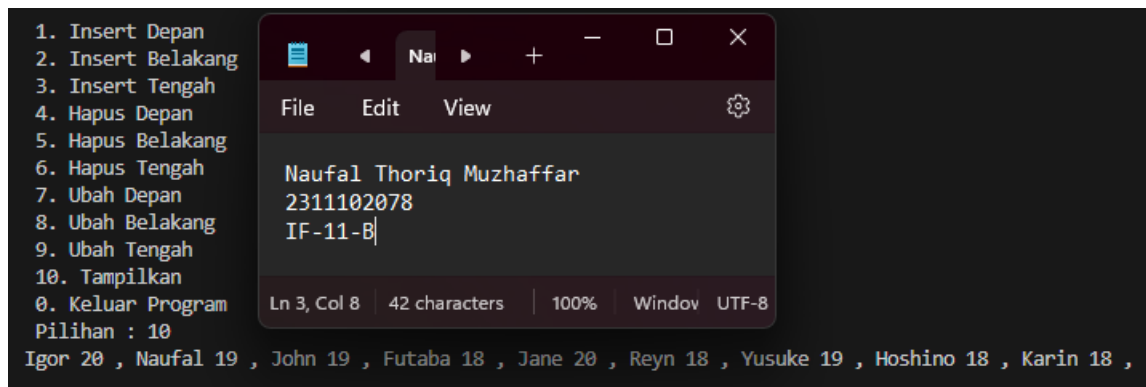
5. Ubah data Michael menjadi Reyn 18.

```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 9
Masukkan Posisi : 6
Masukkan Nama : Reyn
Masukkan Usia : 18

Igor 20 , Naufal 19 , John 19 , Futaba 18 , Jane 20 , Reyn 18 , Yusuke 19 , Hoshino 18 , Karin 18 ,
```



6. Tampilkan seluruh data.



```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 10
Igor 20 , Naufal 19 , John 19 , Futaba 18 , Jane 20 , Reyn 18 , Yusuke 19 , Hoshino 18 , Karin 18 ,
Naufal Thoriq Muzhaffar
2311102078
IF-11-B
```

Deskripsi Program

Program tersebut adalah sebuah implementasi sederhana dari linked list yang dirancang untuk menyimpan informasi tentang mahasiswa. Fungsionalitas dasar program ini mencakup penambahan, penghapusan, dan pengubahan data mahasiswa di berbagai posisi dalam linked list, serta kemampuan untuk menghitung total jumlah data dalam linked list dan menampilkan seluruh data yang tersimpan.

Dalam struktur linked list ini, setiap simpul (node) memiliki dua atribut utama, yaitu nama dan usia mahasiswa. Selain itu, setiap simpul juga memiliki pointer yang mengarah ke simpul berikutnya dalam linked list.

Program menggunakan menu interaktif untuk memilih operasi yang ingin dilakukan terhadap linked list, dan akan berlanjut hingga pengguna memilih untuk keluar dengan memasukkan angka 0.

Secara keseluruhan, program ini memiliki beberapa keunggulan, antara lain:

- Kemampuan untuk menyimpan data mahasiswa secara efisien menggunakan struktur linked list.
- Fasilitas dasar yang mencakup penambahan, penghapusan, pengubahan, dan penampilan data.
- Penggunaan konsep linked list meningkatkan fleksibilitas dalam manajemen data dibandingkan dengan menggunakan struktur data array.

Unguided 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 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>
#include <iomanip>
using namespace std;

class Node
{
public:
    string namaProduk;
    int harga;
    Node *prev;
    Node *next;
};

class DoublyLinkedList
{
public:
    Node *head;
    Node *tail;

    DoublyLinkedList()
    {
        head = nullptr;
        tail = nullptr;
    }

    void push(string namaProduk, int harga)
    {
        Node *newNode = new Node;
        newNode->namaProduk = namaProduk;
        newNode->harga = harga;
        newNode->prev = nullptr;
        newNode->next = head;

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

        head = newNode;
    }

    void pushCenter(string namaProduk, int harga, int posisi)
    {

```

```

if (posisi < 0)
{
    cout << "Posisi harus bernilai non-negatif." << endl;
    return;
}

Node *newNode = new Node;
newNode->namaProduk = namaProduk;
newNode->harga = harga;

if (posisi == 0 || head == nullptr)
{
    newNode->prev = nullptr;
    newNode->next = head;

    if (head != nullptr)
    {
        head->prev = newNode;
    }
    else
    {
        tail = newNode;
    }
    head = newNode;
}
else
{
    Node *temp = head;
    int count = 0;
    while (temp != nullptr && count < posisi)
    {
        temp = temp->next;
        count++;
    }

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

```

```

    }
}

void pop()
{
    if (head == nullptr)
    {
        return;
    }
    Node *temp = head;
    head = head->next;

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

    delete temp;
}

void popCenter(int posisi)
{
    if (head == nullptr)
    {
        cout << "List kosong. Tidak ada yang bisa dihapus." << endl;
        return;
    }

    if (posisi < 0)
    {
        cout << "Posisi harus bernilai non-negatif." << endl;
        return;
    }

    if (posisi == 0)
    {
        Node *temp = head;
        head = head->next;

        if (head != nullptr)
        {
            head->prev = nullptr;
        }
    }
}

```

```

        else
        {
            tail = nullptr;
        }

        delete temp;
    }
    else
    {
        Node *temp = head;
        int count = 0;
        while (temp != nullptr && count < posisi)
        {
            temp = temp->next;
            count++;
        }

        if (temp == nullptr)
        {
            cout << "Posisi melebihi ukuran list. Tidak ada yang
dihapus." << endl;
            return;
        }

        if (temp == tail)
        {
            tail = tail->prev;
            tail->next = nullptr;
            delete temp;
        }
        else
        {
            temp->prev->next = temp->next;
            temp->next->prev = temp->prev;
            delete temp;
        }
    }
}

bool update(string oldNamaProduk, string newNamaProduk, int newHarga)
{
    Node *current = head;

    while (current != nullptr)
    {
        if (current->namaProduk == oldNamaProduk)
        {
            current->namaProduk = newNamaProduk;

```

```

        current->harga = newHarga;
        return true;
    }
    current = current->next;
}
return false;
}

bool updateCenter(string newNamaProduk, int newHarga, int posisi)
{
    if (head == nullptr)
    {
        cout << "List kosong. Tidak ada yang dapat diperbarui." << endl;
        return false;
    }

    if (posisi < 0)
    {
        cout << "Posisi harus bernilai non-negatif." << endl;
        return false;
    }

    Node *current = head;
    int count = 0;

    while (current != nullptr && count < posisi)
    {
        current = current->next;
        count++;
    }

    if (current == nullptr)
    {
        cout << "Posisi melebihi ukuran list. Tidak ada yang
diperbarui." << endl;
        return false;
    }

    current->namaProduk = newNamaProduk;
    current->harga = newHarga;
    return true;
}

void deleteAll()
{
    Node *current = head;
    while (current != nullptr)
    {

```

```

        Node *temp = current;
        current = current->next;
        delete temp;
    }
    head = nullptr;
    tail = nullptr;
}

void display()
{
    if (head == nullptr)
    {
        cout << "List kosong." << endl;
        return;
    }

    Node *current = head;

    cout << setw(37) << setfill('-') << "-" << setfill(' ') << endl;
    cout << "| " << setw(20) << left << "Nama Produk"
        << " | " << setw(10) << "Harga"
        << " |" << endl;
    cout << setw(37) << setfill('-') << "-" << setfill(' ') << endl;

    while (current != nullptr)
    {
        cout << "| " << setw(20) << left << current->namaProduk << " | "
        << setw(10) << current->harga << " |" << endl;
        current = current->next;
    }
    cout << setw(37) << setfill('-') << "-" << setfill(' ') << endl;
}

};

int main()
{
    DoublyLinkedList list;
    int choice;
    cout << endl
        << "Toko Skincare Purwokerto" << endl;
    do
    {
        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 >> choice;

switch (choice)
{
case 1:
{
    string namaProduk;
    int harga;
    cout << "Masukkan nama produk: ";
    cin.ignore();
    getline(cin, namaProduk);
    cout << "Masukkan harga produk: ";
    cin >> harga;
    list.push(namaProduk, harga);
    break;
}
case 2:
{
    list.pop();
    break;
}
case 3:
{
    string newNamaProduk;
    int newHarga, posisi;
    cout << "Masukkan posisi produk: ";
    cin >> posisi;
    cout << "Masukkan nama baru produk: ";
    cin >> newNamaProduk;
    cout << "Masukkan harga baru produk: ";
    cin >> newHarga;
    bool updatedCenter = list.updateCenter(newNamaProduk, newHarga,
posisi);

    if (!updatedCenter)
    {
        cout << "Data not found" << endl;
    }
    break;
}
case 4:
{
    string namaProduk;
    int harga, posisi;
    cout << "Masukkan posisi data produk: ";

```

```

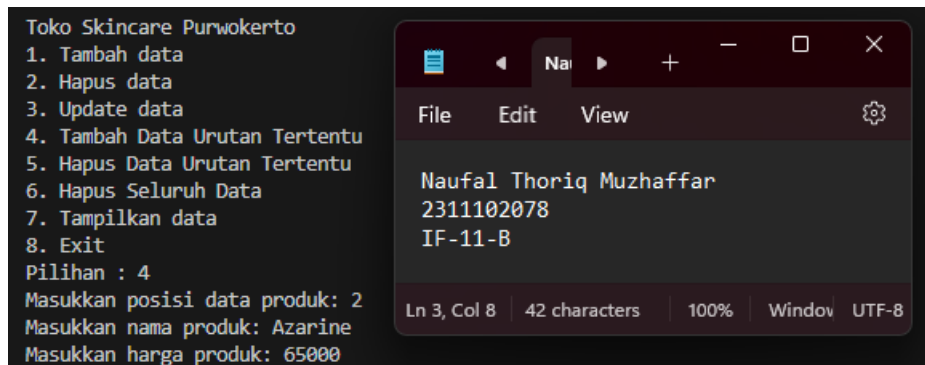
        cin >> posisi;
        cout << "Masukkan nama produk: ";
        cin.ignore();
        getline(cin, namaProduk);
        cout << "Masukkan harga produk: ";
        cin >> harga;
        list.pushCenter(namaProduk, harga, posisi);
        break;
    }
    case 5:
    {
        int posisi;
        cout << "Masukkan posisi data produk: ";
        cin >> posisi;
        list.popCenter(posisi);
        break;
    }
    case 6:
    {
        list.deleteAll();
        break;
    }
    case 7:
    {
        list.display();
        break;
    }
    case 8:
    {
        return 0;
    }
    default:
    {
        cout << "Invalid choice" << endl;
        break;
    }
}
} while (choice != 8);

return 0;
}

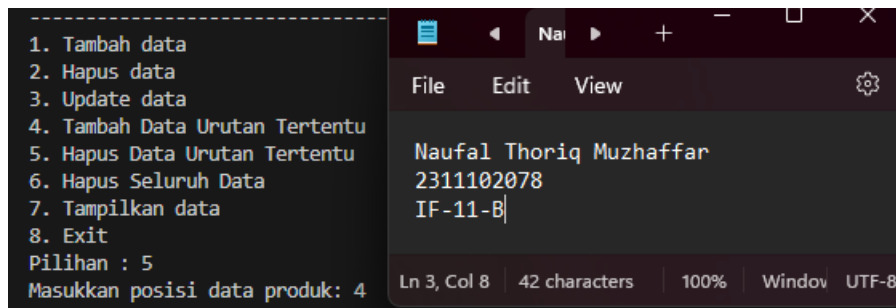
```

Screenshots Output

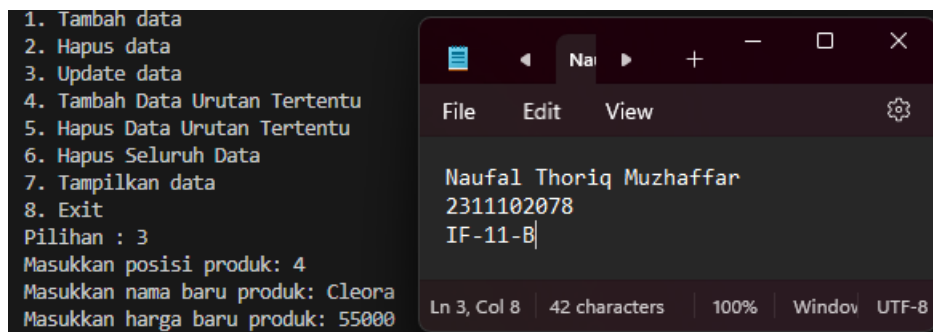
1. Tambahkan produk Azarine dengan harga 65.000 diantara Somethinc dan Skintific.



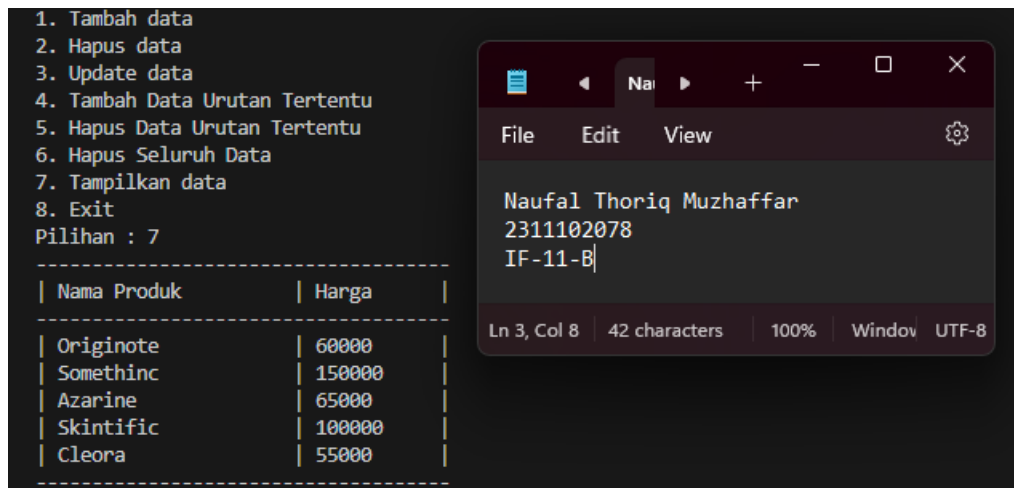
2. Hapus produk Wardah.



3. Update produk Hanasui menjadi Cleora dengan harga 55.000.



4. Tampilkan menu.



```
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
Pilihan : 7
-----
| Nama Produk      | Harga |
-----
| Originote        | 60000 |
| Somethinc        | 150000|
| Azarine          | 65000 |
| Skintific        | 100000|
| Cleora           | 55000 |
-----
```

The screenshot shows a terminal window with a menu of options. Option 7, 'Tampilkan data', has been selected. Below the menu, a table displays product information. The table has two columns: 'Nama Produk' and 'Harga'. The products listed are Originote, Somethinc, Azarine, Skintific, and Cleora, with their respective prices. To the right of the terminal window, a separate window titled 'Naufal Thoriq Muzhaffar' is visible, showing a text editor with the same name and ID '2311102078' and 'IF-11-B'.

Deskripsi Program

Program yang disebutkan adalah sebuah implementasi dari Doubly Linked List yang digunakan untuk menyimpan informasi tentang produk skincare. Fungsionalitas dasar dari program ini meliputi operasi-operasi seperti penambahan, penghapusan, dan pembaruan data, baik secara umum maupun pada posisi tertentu dalam linked list.

- Program ini menggunakan struktur data Doubly Linked List untuk mengatur dan mengelola data produk skincare.
- Setiap simpul (node) dalam Doubly Linked List memiliki dua penunjuk, yaitu penunjuk ke simpul sebelumnya (prev) dan penunjuk ke simpul berikutnya (next).
- Fungsi-fungsi yang tersedia dalam program mencakup operasi-operasi seperti menambah data di depan linked list (push), menambah data pada posisi tertentu (pushCenter), menghapus data di depan linked list (pop), menghapus data pada posisi tertentu (popCenter), mengubah data (update), mengubah data pada posisi tertentu (updateCenter), menghapus seluruh data (deleteAll), dan menampilkan data (display).
- Program ini menyediakan menu interaktif untuk memilih operasi yang ingin dilakukan pada Doubly Linked List, dan akan berlanjut hingga pengguna memilih untuk keluar dari program.

D. Kesimpulan

Linked list adalah struktur data yang terdiri dari simpul-simpul/node yang saling terhubung dalam urutan tertentu. Jika setiap simpul hanya memiliki satu penunjuk ke simpul berikutnya (berarah "next"), maka linked list tersebut disebut single linked list. Dalam praktikum ini, terdapat berbagai operasi yang dapat dilakukan pada linked list, termasuk penginisialisasian, penambahan, pengubahan, pengurangan, dan penampilan data.

E. Referensi

- [1] Asprak "Modul 3 Singel and Double Linked List". Learning Management System 2024.
- [2] Educative. Singly linked list in C++. Diakses pada 31 Maret 2024 , dari <https://www.educative.io/answers/singly-linked-list-in-cpp>
- [3] Educative, How to create a doubly linked list with helper methods in C++. Diakses pada 31 Maret 2024, dari <https://www.educative.io/answers/how-to-create-a-doubly-linked-list-with-helper-methods-in-cpp>