LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA

MODUL III
PENGENALAN SINGLE AND DOUBLE LINKED LIST

Disusun Oleh :

NAMA : Pandia Arya Brata

NIM : 2311102076

Dosen

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

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA INSTITUT TEKNOLOGI TELKOM PURWOKERTO 2024

MODUL 3

SINGLE AND DOUBLE LINKED LIST

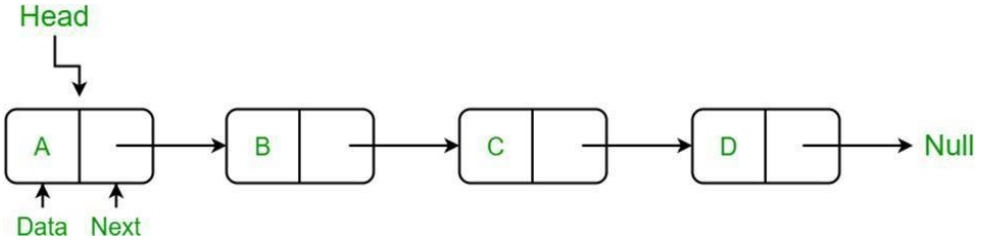
A. Tujuan

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

B. Dasar Teori

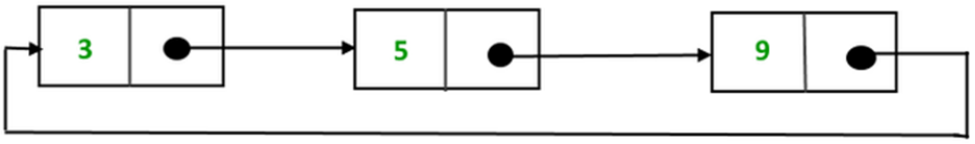
a) Single Linked List

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. Setiap elemen dalam linked list dihubungkan ke elemen lain melalui pointer. Masing-masing komponen sering disebut dengan simpul atau node atau verteks. Pointer adalah alamat elemen. Setiap simpul pada dasarnya dibagi atas dua bagian pertama disebut bagian isi atau informasi atau data yang berisi nilai yang disimpan oleh simpul. Bagian kedua disebut bagian pointer yang berisi alamat dari node berikutnya atau sebelumnya. Dengan menggunakan struktur seperti ini, linked list dibentuk dengan cara menunjuk pointer next suatu elemen ke elemen yang mengikutinya. Pointer next pada elemen terakhir merupakan NULL, yang menunjukkan akhir dari suatu list. Elemen pada awal suatu list disebut head dan elemen terakhir dari suatu list disebut tail.



Dalam operasi Single Linked List, umumnya dilakukan operasi penambahan dan penghapusan simpul pada awal atau akhir daftar, serta pencarian dan pengambilan nilai pada simpul tertentu dalam daftar. Karena struktur data ini hanya memerlukan satu pointer untuk setiap simpul, maka Single Linked List umumnya lebih efisien dalam penggunaan memori dibandingkan dengan jenis Linked List lainnya, seperti Double Linked List dan Circular Linked List.

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.

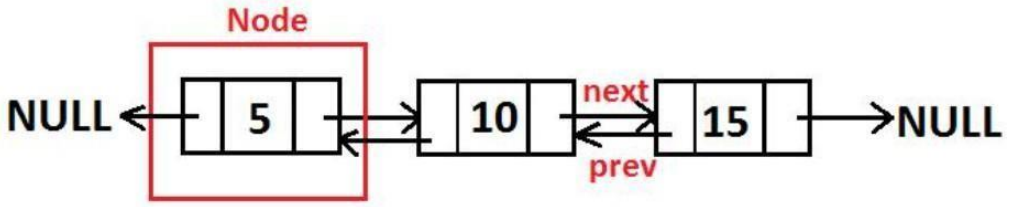


b) Double Linked List

Double Linked List adalah struktur data Linked List yang mirip dengan Single Linked List, namun dengan tambahan satu pointer tambahan pada setiap simpul yaitu pointer prev yang menunjuk ke simpul sebelumnya. 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.

Keuntungan dari Double Linked List adalah memungkinkan untuk melakukan operasi penghapusan dan penambahan pada simpul dimana saja dengan efisien, sehingga sangat berguna dalam implementasi beberapa algoritma yang membutuhkan operasi tersebut. Selain itu, Double Linked List juga memungkinkan kita untuk melakukan traversal pada list baik dari depan (head) maupun dari belakang (tail) dengan mudah. Namun, kekurangan dari Double Linked List adalah penggunaan memori yang lebih besar dibandingkan dengan Single Linked List, karena setiap simpul membutuhkan satu pointer tambahan. Selain itu, Double Linked List juga membutuhkan waktu eksekusi yang lebih lama dalam operasi penambahan dan penghapusan jika dibandingkan dengan Single Linked List.

Representasi sebuah double 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.

C. Guided

a) Latihan Single Linked List

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
```

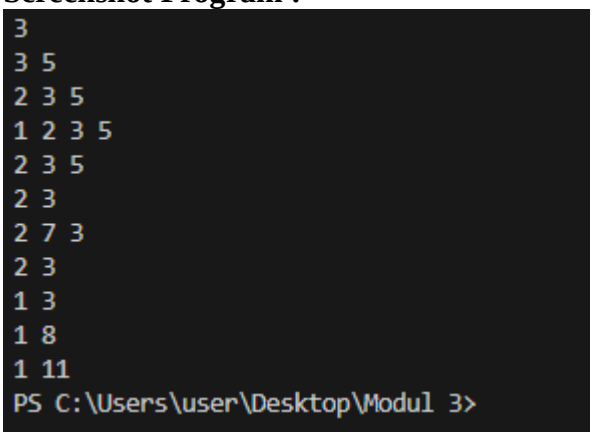
```
void hapusBelakang() {
    Node *hapus;
    Node *bantu;
    if (isEmpty() == false){
        if (head != tail){
            hapus = tail;
            bantu = head;
            while (bantu->next != tail){
                bantu = bantu->next;
            }
            tail = bantu;
            tail->next = 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;
}
```

Screenshot Program :



Deskripsi Program :

Single Linked List Non-Circular adalah salah satu struktur data yang digunakan untuk menyimpan kumpulan data dalam bentuk urutan satu arah (unidirectional). Setiap elemen atau simpul pada linked list ini, disebut sebagai node, terdiri dari dua bagian yaitu data dan pointer yang menunjuk ke simpul berikutnya.Dalam Single Linked List Non-Circular, simpul terakhir tidak menunjuk ke simpul pertama, sehingga tidak membentuk sebuah lingkaran. Karena itu, linked list ini hanya memiliki satu pointer, yaitu pointer yang menunjuk ke simpul pertama, yang biasanya disebut dengan head.

b) Latihan Double Linked List
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;
}
```

Screenshot Progrgam :

```
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: 12345
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 3
Enter old data: 12345
Enter new data: 54321
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
54321
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 2
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
```

Deskripsi Program :

Double linked list adalah salah satu struktur data yang digunakan untuk menyimpan kumpulan data dalam bentuk urutan dua arah (bidirectional). Dalam double linked list, setiap node memiliki dua pointer yaitu pointer yang menunjuk ke simpul sebelumnya (previous node) dan pointer yang menunjuk ke simpul berikutnya (next node). Dalam sebuah program double linked list, biasanya terdapat operasi-operasi dasar seperti :

- Insert: operasi untuk menambahkan sebuah elemen ke dalam double linked list. Elemen baru akan ditempatkan pada posisi yang ditentukan, misalnya pada posisi awal, akhir, atau di tengah-tengah linked list.
- Delete: operasi untuk menghapus sebuah elemen dari double linked list. Elemen yang dihapus biasanya diidentifikasi dengan menggunakan kunci tertentu seperti ID atau nomor indeks.
- Traverse: operasi untuk mengunjungi semua elemen yang terdapat pada double linked list. Dalam operasi ini, kita bisa menggunakan loop untuk mengunjungi setiap elemen secara berurutan.
- Search: operasi untuk mencari sebuah elemen pada double linked list berdasarkan kunci yang telah ditentukan.

D. 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;

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

Mahasiswa *head = nullptr;
```

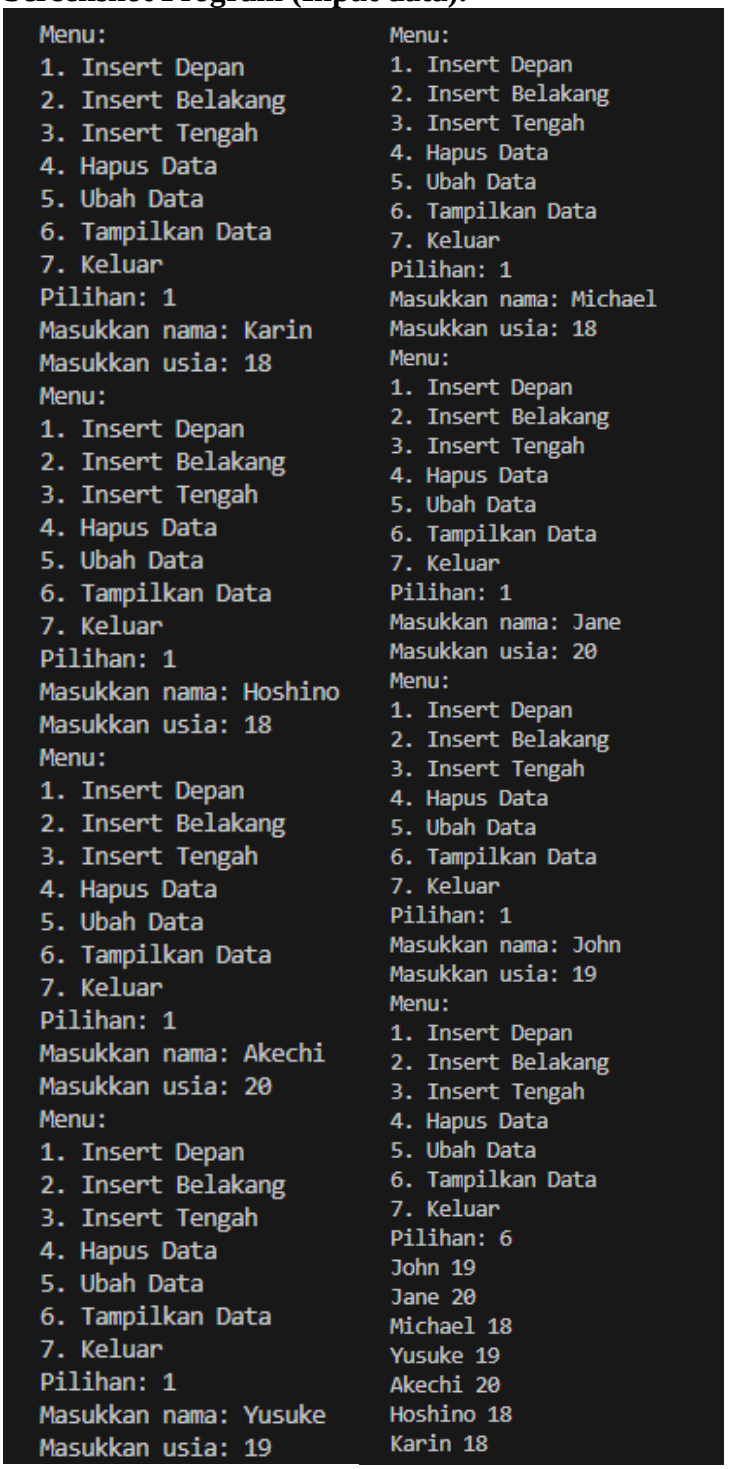
```
void insertDepan(string nama, int usia)
{
    Mahasiswa *node = new Mahasiswa;
    node->nama = nama;
    node->usia = usia;
    node->next = head;
    head = node;
}

void insertBelakang(string nama, int usia)
{
    Mahasiswa *node = new Mahasiswa;
    node->nama = nama;
    node->usia = usia;
    node->next = nullptr;

    if (head == nullptr)
    {
        head = node;
        return;
    }

    Mahasiswa *curr = head;
    while (curr->next != nullptr)
    {
        curr = curr->next;
    }

    curr->next = node;
}
void insertTengah(string nama, int usia, int pos)
{
    Mahasiswa *node = new Mahasiswa;
    node->nama = nama;
    node->usia = usia;

    Mahasiswa *curr = head;
    for (int i = 0; i < pos - 1; i++)
    {
        curr = curr->next;
    }

    node->next = curr->next;
    curr->next = node;
}

void hapusData(string nama)
{
    if (head == nullptr)
```

```
    {
        return;
    }

    if (head->nama == nama)
    {
        Mahasiswa *node = head;
        head = head->next;
        delete node;
        return;
    }

    Mahasiswa *curr = head;
    while (curr->next != nullptr && curr->next->nama !=
                                        nama)
    {
        curr = curr->next;
    }

    if (curr->next != nullptr)
    {
        Mahasiswa *node = curr->next;
        curr->next = curr->next->next;
        delete node;
    }
}

void ubahData(string nama, int usia)
{
    if (head == nullptr)
    {
        cout << "Tidak ada data yang bisa diubah" << endl;
        return;
    }
    Mahasiswa *node = head;
    string key;
    cout << "Masukkan Nama yang ingin diubah: ";
    cin >> key;
    while (node != nullptr)
    {
        if (node->nama == key)
        {
            cout << "Masukan nama baru: ";
            cin >> node->nama;
            cout << "Masukan umur baru: ";
            cin >> node->usia;
            return;
        }
```

```
        node = node->next;
    }

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

void tampilkanData()
{
    Mahasiswa *curr = head;
    while (curr != nullptr)
    {
        cout << curr->nama << " " << curr->usia << endl;
        curr = curr->next;
    }
}

int main()
{
    int pilihan;
    string nama;
    int usia;
    int posisi;

    do
    {
        // Tampilkan menu
        cout << "Menu:" << endl;
        cout << "1. Insert Depan" << endl;
        cout << "2. Insert Belakang" << endl;
        cout << "3. Insert Tengah" << endl;
        cout << "4. Hapus Data" << endl;
        cout << "5. Ubah Data" << endl;
        cout << "6. Tampilkan Data" << endl;
        cout << "7. Keluar" << endl;
        cout << "Pilihan: ";
        cin >> pilihan;

        switch (pilihan)
        {
        case 1:
            cout << "Masukkan nama: ";
            cin >> nama;
            cout << "Masukkan usia: ";
            cin >> usia;
            insertDepan(nama, usia);
            break;
        case 2:
            cout << "Masukkan nama: ";
            cin >> nama;
```

```
            cout << "Masukkan usia: ";
            cin >> usia;
            insertBelakang(nama, usia);
            break;
        case 3:
            cout << "Masukkan nama: ";
            cin >> nama;
            cout << "Masukkan umur: ";
            cin >> usia;
            cout << "Masukkan posisi: ";
            cin >> posisi;
            insertTengah(nama, usia, posisi);
            break;
        case 4:
            cout << "Masukkan nama: ";
            cin >> nama;
            hapusData(nama);
            break;
        case 5:
            ubahData(nama, usia);
            break;
        case 6:
            tampilkanData();
            break;
        case 7:
            cout << "Terima kasih." << endl;
            break;
        default:
            cout << "Pilihan tidak valid." << endl;
        }
    } while (pilihan != 7);

    return 0;
}
```

Screenshot Program (Input data):



```
Menu:
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Data
5. Ubah Data
6. Tampilkan Data
7. Keluar
Pilihan: 1
Masukkan nama: Karin
Masukkan usia: 18
Menu:
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Data
5. Ubah Data
6. Tampilkan Data
7. Keluar
Pilihan: 1
Masukkan nama: Hoshino
Masukkan usia: 18
Menu:
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Data
5. Ubah Data
6. Tampilkan Data
7. Keluar
Pilihan: 1
Masukkan nama: Akechi
Masukkan usia: 20
Menu:
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Data
5. Ubah Data
6. Tampilkan Data
7. Keluar
Pilihan: 1
Masukkan nama: Yusuke
Masukkan usia: 19
Menu:
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Data
5. Ubah Data
6. Tampilkan Data
7. Keluar
Pilihan: 1
Masukkan nama: Michael
Masukkan usia: 18
Menu:
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Data
5. Ubah Data
6. Tampilkan Data
7. Keluar
Pilihan: 1
Masukkan nama: Jane
Masukkan usia: 20
Menu:
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Data
5. Ubah Data
6. Tampilkan Data
7. Keluar
Pilihan: 1
Masukkan nama: John
Masukkan usia: 19
Menu:
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Data
5. Ubah Data
6. Tampilkan Data
7. Keluar
Pilihan: 6
John 19
Jane 20
Michael 18
Yusuke 19
Akechi 20
Hoshino 18
Karin 18
```

Screenshot program (Hapus data Akechi) :

```
Menu:
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Data
5. Ubah Data
6. Tampilkan Data
7. Keluar
Pilihan: 4
Masukkan nama: Akechi
Menu:
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Data
5. Ubah Data
6. Tampilkan Data
7. Keluar
Pilihan: 6
John 19
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18
```

Screenshot program (Data Futaba) :

```
Menu:
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Data
5. Ubah Data
6. Tampilkan Data
7. Keluar
Pilihan: 3
Masukkan nama: Futaba
Masukkan umur: 18
Masukkan posisi: 1
Menu:
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Data
5. Ubah Data
6. Tampilkan Data
7. Keluar
Pilihan: 6
John 19
Futaba 18
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18
```

Screenshot program (Data awal Igor) :

```
Menu:
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Data
5. Ubah Data
6. Tampilkan Data
7. Keluar
Pilihan: 1
Masukkan nama: Igor
Masukkan usia: 20
Menu:
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Data
5. Ubah Data
6. Tampilkan Data
7. Keluar
Pilihan: 6
Igor 20
John 19
Futaba 18
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18
```

Screenshot program (Ubah data Michael):

```
Menu:
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Data
5. Ubah Data
6. Tampilkan Data
7. Keluar
Pilihan: 5
Masukkan Nama yang ingin diubah: Michael
Masukan nama baru: Reyn
Masukan umur baru: 18
Menu:
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Data
5. Ubah Data
6. Tampilkan Data
7. Keluar
Pilihan: 6
Igor 20
John 19
Futaba 18
Jane 20
Reyn 18
Yusuke 19
Hoshino 18
Karin 18
```

Screenshot program (Tampilkan data):

Deskripsi program :

Dalam program di atas, terdapat fungsi-fungsi untuk menambahkan data di depan dan belakang linked list, menghapus data di depan dan belakang linked list, mengubah data di depan dan belakang linked list, serta menampilkan semua data dalam linked list.

Program juga menggunakan menu untuk memudahkan pengguna dalam memilih operasi yang ingin dilakukan. Setiap kali pengguna memilih salah satu operasi, program akan menampilkan pesan untuk meminta input yang diperlukan dan kemudian memanggil fungsi yang sesuai. Program akan berhenti saat pengguna memilih opsi keluar.

Program di atas akan memunculkan menu dengan 7 pilihan, dimulai dari menambah data di depan, menambah data di belakang,menambah data di tengah, menghapus data , mengubah data, menampilkan semua data, dan keluar dari program. Setiap pilihan akan memanggil fungsi yang sesuai untuk mengeksekusi operasi yang dipilih oleh pengguna.

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>

using namespace std;

struct Node
{
    string nama_produk;
    int harga;
```

```
    Node *prev;
    Node *next;
};

class LinkedList
{
private:
    Node *head;
    Node *tail;

public:
    LinkedList()
    {
        head = NULL;
        tail = NULL;
    }
    void tambahData(string nama_produk, int harga)
    {
        Node *newNode = new Node;
        newNode->nama_produk = nama_produk;
        newNode->harga = harga;
        newNode->prev = NULL;
        newNode->next = NULL;
        if (head == NULL)
        {
            head = newNode;
            tail = newNode;
        }
        else
        {
            tail->next = newNode;
            newNode->prev = tail;
            tail = newNode;
        }
    }
    void hapusData(string nama_produk)
    {
        Node *currentNode = head;
        while (currentNode != NULL)
        {
            if (currentNode->nama_produk == nama_produk)
            {
                if (currentNode == head)
                {
                    head = head->next;
                    head->prev = NULL;
                }
                else if (currentNode == tail)
                {
                    tail = tail->prev;
                    tail->next = NULL;
                }
                else
```

```
                {
                    currentNode->prev->next = currentNode ->
next;
                    currentNode->next->prev = currentNode ->
prev;
                }
                delete currentNode;
                break;
            }
            currentNode = currentNode->next;
        }
    }
    void updateData(string nama_produk, string
new_nama_produk, int new_harga)
    {
        Node *currentNode = head;
        while (currentNode != NULL)
        {
            if (currentNode->nama_produk == nama_produk)
            {
                currentNode->nama_produk = new_nama_produk;
                currentNode->harga = new_harga;
                break;
            }
            currentNode = currentNode->next;
        }
    }
    void tambahDataUrutan(string nama_produk, int harga, int
urutan)
    {
        Node *newNode = new Node;
        newNode->nama_produk = nama_produk;
        newNode->harga = harga;
        newNode->prev = NULL;
        newNode->next = NULL;
        if (urutan == 1)
        {
            newNode->next = head;
            head->prev = newNode;
            head = newNode;
        }
        else
        {
            Node *currentNode = head;
            int i = 1;
            while (i < urutan - 1)
            {
                currentNode = currentNode->next;
                i++;
            }
            newNode->prev = currentNode;
            newNode->next = currentNode->next;
            currentNode->next->prev = newNode;
```

```
            currentNode->next = newNode;
        }
    }
    void hapusDataUrutan(int urutan)
    {
        if (urutan == 1)
        {
            head = head->next;
            head->prev = NULL;
        }
        else
        {
            Node *currentNode = head;
            int i = 1;
            while (i < urutan)
            {
                currentNode = currentNode->next;
                i++;
            }
            if (currentNode == tail)
            {
                tail = tail->prev;
                tail->next = NULL;
            }
            else
            {
                currentNode->prev->next = currentNode->next;
                currentNode->next->prev = currentNode->prev;
            }
        }
    }
    void hapusSeluruhData()
    {
        while (head != NULL)
        {
            Node *currentNode = head;
            head = head->next;
            delete currentNode;
        }
        tail = NULL;
    } // <-- add closing bracket here

    void tampilkanData()
    {
        Node *currentNode = head;
        cout << "Nama Produk\tHarga" << endl;
        while (currentNode != NULL)
        {
            cout << currentNode->nama_produk << "\t\t" <<
currentNode->harga << endl;
            currentNode = currentNode->next;
        }
    }
```

```
    void tampilkanMenu()
    {
        int pilihan, harga, urutan;
        string nama_produk, new_nama_produk;
        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;
        do
        {
            cout << "Pilih menu: ";
            cin >> pilihan;
            switch (pilihan)
            {
            case 1:
                cout << "Masukkan Nama Produk: ";
                cin >> nama_produk;
                cout << "Masukkan Harga: ";
                cin >> harga;
                tambahData(nama_produk, harga);
                break;
            case 2:
                cout << "Masukkan Nama Produk yang akan
dihapus : ";
                        cin >>nama_produk;
                hapusData(nama_produk);
                break;
            case 3:
                cout << "Masukkan Nama Produk yang akan
diupdate : ";
                        cin >>nama_produk;
                cout << "Masukkan Nama Produk baru: ";
                cin >> new_nama_produk;
                cout << "Masukkan Harga baru: ";
                cin >> harga;
                updateData(nama_produk, new_nama_produk,
harga);
                break;
            case 4:
                cout << "Masukkan Nama Produk: ";
                cin >> nama_produk;
                cout << "Masukkan Harga: ";
                cin >> harga;
                cout << "Masukkan Urutan: ";
                cin >> urutan;
                tambahDataUrutan(nama_produk, harga, urutan);
                break;
            case 5:
```

```
                cout << "Masukkan Urutan: ";
                cin >> urutan;
                hapusDataUrutan(urutan);
                break;
            case 6:
                hapusSeluruhData();
                break;
            case 7:
                tampilkanData();
                break;
            case 8:
                cout << "Terima kasih." << endl;
                break;
            default:
                cout << "Pilihan tidak tersedia." << endl;
            }
        } while (pilihan != 8);
    }
};

int main()
{
    LinkedList list;
    list.tampilkanMenu();
    return 0;
}
```

Screenshot program (input) :

```
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
Pilih menu: 1
Masukkan Nama Produk: Originote
Masukkan Harga: 60000
Pilih menu: 1
Masukkan Nama Produk: Somethinc
Masukkan Harga: 150000
Pilih menu: 1
Masukkan Nama Produk: Skintific
Masukkan Harga: 100000
Pilih menu: 1
Masukkan Nama Produk: Wardah
Masukkan Harga: 50000
Pilih menu: 1
Masukkan Nama Produk: Hanasui
Masukkan Harga: 30000
Pilih menu: 7
Nama Produk     Harga
Originote               60000
Somethinc               150000
Skintific               100000
Wardah          50000
Hanasui         30000
```

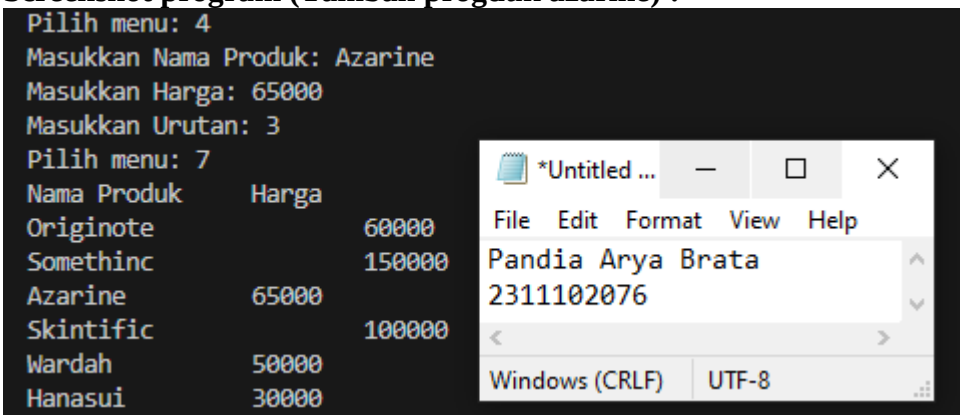
*Untitled ...

File Edit Format View Help

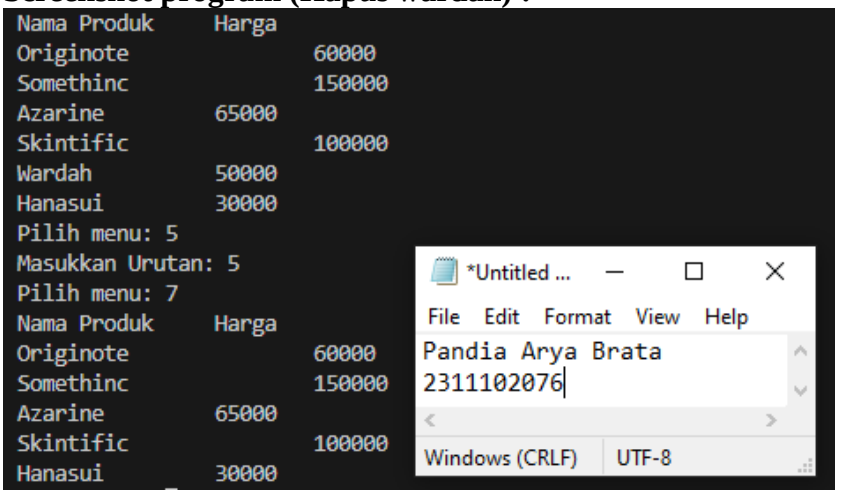
Pandia Arya Brata
2311102076

Windows (CRLF) UTF-8

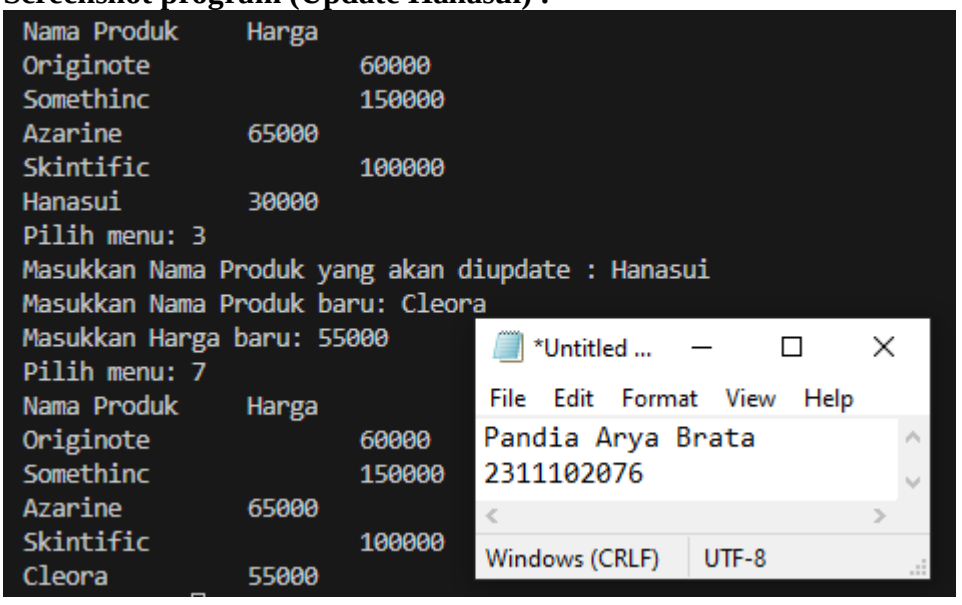
Screenshot program (Tambah progduk azarine) :



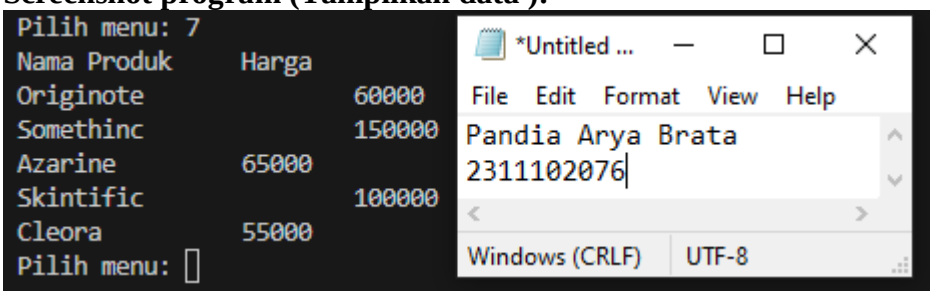
Screenshot program (Hapus wardah) :



Screenshot program (Update Hanasui) :



Screenshot program (Tampilkan data):



Deskripsi program :

Program ini adalah program untuk mengelola data produk skincare pada suatu toko. Program ini menggunakan struktur data linked list untuk menyimpan data produk dan menyediakan beberapa fitur seperti menambahkan data produk, menghapus data produk, mengupdate data produk, menambahkan data produk pada posisi tertentu, menghapus data produk pada posisi tertentu, menghapus semua data produk, dan menampilkan semua data produk.

Program ini memiliki kelas LinkedList yang berisi beberapa method untuk mengelola data produk skincare. Ada method untuk menambahkan data, menghapus data, mengupdate data, menambahkan data pada posisi tertentu, menghapus data pada posisi tertentu, menghapus semua data, dan menampilkan semua data. Setiap data produk direpresentasikan dengan struktur Node yang berisi nama produk dan harga, serta pointer untuk menghubungkan Node satu dengan yang lainnya.

Program ini memiliki menu dengan delapan pilihan, yaitu tambah data, hapus data, update data, tambah data urutan tertentu, hapus data urutan tertentu, hapus seluruh data, tampilkan data, dan exit. Pilihan-pilihan ini berfungsi untuk memudahkan pengguna dalam mengelola data produk skincare yang ada pada toko.

KESIMPULAN

Single Linked List adalah struktur data linear yang terdiri dari simpul-simpul yang terhubung satu sama lain dengan pointer ke simpul berikutnya dalam daftar. Setiap simpul memiliki data dan pointer ke simpul berikutnya dalam daftar. Single Linked List memiliki keuntungan dalam penggunaan memori yang lebih sedikit daripada Double Linked List, namun sulit dalam operasi penambahan dan penghapusan simpul di tengah-tengah daftar.

Double Linked List, seperti namanya, memiliki dua pointer di setiap simpul, yaitu pointer ke simpul sebelumnya dan pointer ke simpul berikutnya. Double Linked List memungkinkan operasi penambahan dan penghapusan simpul di tengah-tengah daftar lebih mudah daripada Single Linked List, namun membutuhkan lebih banyak memori.

Kesimpulannya, dalam pemrograman C++, pemilihan antara Single Linked List dan Double Linked List harus disesuaikan dengan kebutuhan dan keadaan penggunaannya. Single Linked List dapat digunakan jika memori adalah faktor yang penting dan penambahan/penghapusan simpul dilakukan terutama di akhir daftar. Sedangkan Double Linked List dapat digunakan jika penggunaannya lebih sering dalam penambahan/penghapusan simpul di tengah-tengah daftar dan alokasi memori lebih besar tidak menjadi masalah.