LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA

MODUL III
SINGLE DAN DOUBLE LINKED LIST

Disusun Oleh :
Muhammad Arsyad Zaidan (2311102058)

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

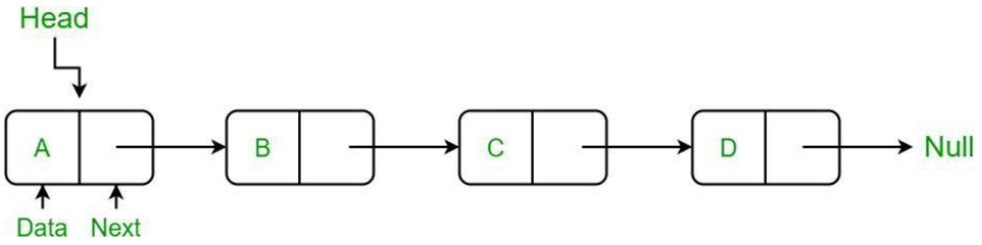
PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

A. Dasar Teori

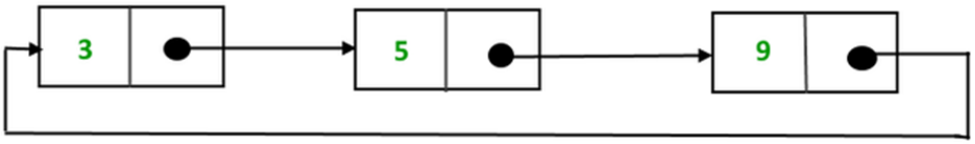
a) Single Linked List

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.



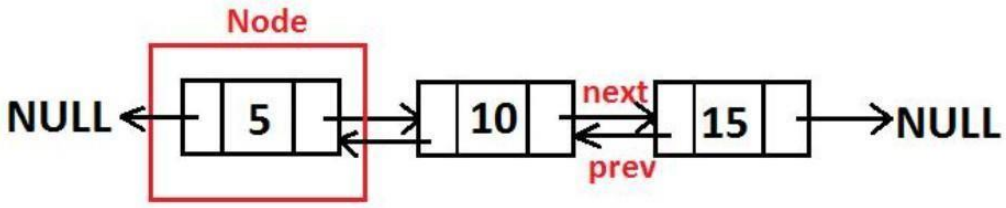
b) Double Linked List

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.

B. Latihan Guided dan Unguided

1) Guided

Guided 1

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;
}
```

Output Code

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Belajar coding\Strukdat\Learning Code\STRUKDAT (PRAKTEK)\LL>  & 'c:\Users\ASUS\.vscod
e\extensions\ms-vscode.cpptools-1.19.9-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe
' '--stdin=Microsoft-MIEngine-In-xbiuxoip.uhy' '--stdout=Microsoft-MIEngine-Out-o1rx0vui.dm
q' '--stderr=Microsoft-MIEngine-Error-dbxriw0y.4sf' '--pid=Microsoft-MIEngine-Pid-4nckkenx.
l0o' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
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 C:\Belajar coding\Strukdat\Learning Code\STRUKDAT (PRAKTEK)\LL>
```

Deskripsi Code

Single linked list pada pemrograman tersebut pada tiap node memiliki data pointer yang menunjukkan ke node berikutnya. Interaksi tersebut pada fungsi main yang memiliki beberapa fungsi untuk menambah, menghapus, mengubah dan menampilkan data pada pemrograman tersebut.

Guided 2

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){
```

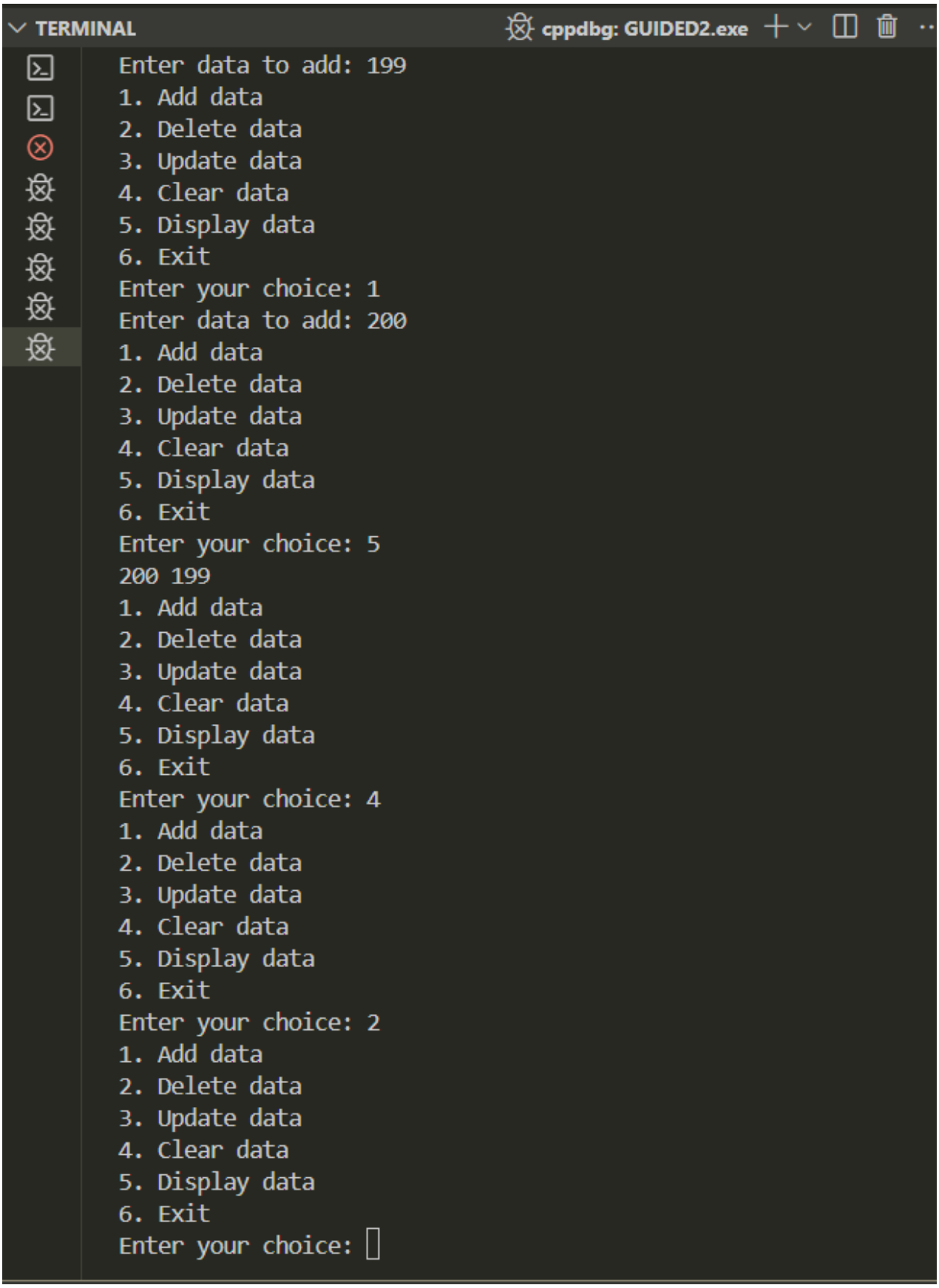
```
        head->data = data;
    }
    else{
        cout << "List masih kosong!" << endl;
    }
}
//Ubah Tengah
void ubahTengah(int data, int posisi){
    Node *bantu;
    if (isEmpty() == false){
        if( posisi < 1 || posisi > hitungList() ){
            cout << "Posisi di luar jangkauan" <<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

Output Code



Deskripsi Code

Pemrograman C++ diatas menggunakan double linked list yang dimana user dapat menghapus, menambah dan mengubah elemen pada linked list dan opsi tampilkan untuk menampilkan linked list saat terjadi perubahan. Elemen pada linked list disimpa seacara terpisah dan memiliki tautan ke elemen berikutnya.

2) 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>
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 << " " << 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;
```

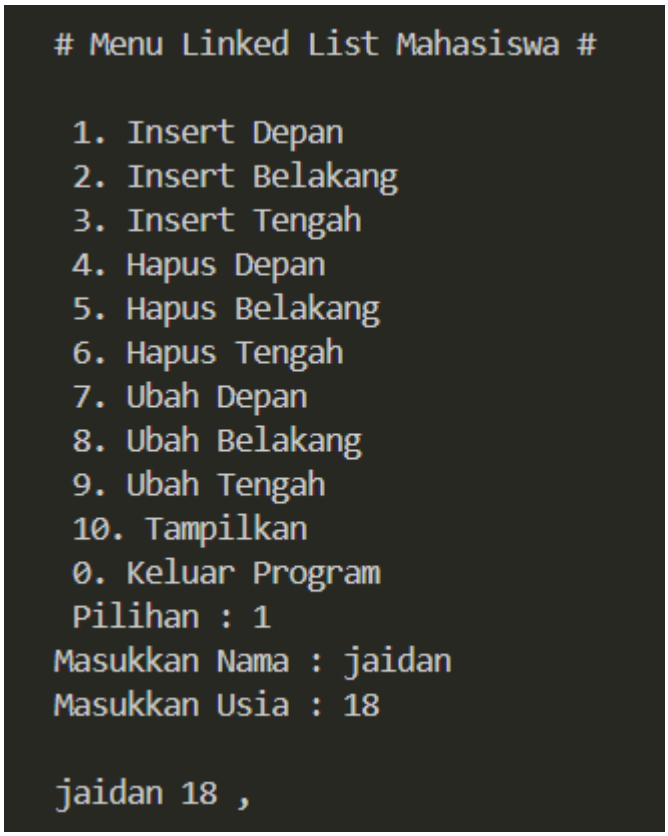
```
                tampil();
                break;
            case 3:
                cout << "Masukkan Posisi : ";
                cin >> posisi;
                cout << "Masukkan Nama : ";
                cin >> nama;
                cout << "Masukkan Usia : ";
                cin >> usia;
                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;
            case 0:
                cout << "Keluar Program" << endl;
                break;
            default:
                cout << "Pilihan Salah" << endl;
                break;
        }
    } while (menu != 0);

    return 0;
}
```

Output Code

1. Memasukan data pada insert depan



2. Memasukan data pada insert belakang

```
 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 : 2
Masukkan Nama : daffa
Masukkan Usia : 18

jaidan 18 , daffa 18 ,
```

3. Memasukan data pada insert tengah

```
jaidan 18 , daffa 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 : 3
Masukkan Posisi : 2
Masukkan Nama : hamzah
Masukkan Usia : 18

jaidan 18 , hamzah 18 , daffa 18 ,
```

4. Hapus data pada bagian depan

```
  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 : 4

 hamzah 18 , daffa 18 ,
```

5. Hapus data pada bagian tengah

```
  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 : 6
 Masukkan Posisi : 2

 jaidan 18 , daffa 18 ,
```

6. Hapus data pada bagian belakang

```
  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 : 5

 jaidan 18 , hamzah 18 ,
```

7. Ubah pada bagian depan

```
  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 : 7
 Masukkan Nama : rizki
 Masukkan Usia : 19

 rizki 19 , hamzah 18 , daffa 18 ,
```

8. Ubah pada bagian tengah

```
 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 : 2
Masukkan Nama : zaki
Masukkan Usia : 19

rizki 19 , zaki 19 , daffa 18 ,
```

9. Ubah pada bagian belakang

```
 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 : 8
Masukkan Nama : ahmad
Masukkan Usia : 19

rizki 19 , zaki 19 , ahmad 19 ,
```

10. Menampilkan data program

```
 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
rizki 19 , zaki 19 , ahmad 19 ,
```

11. Keluar dari program

```
 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 : 0
Keluar Program
PS C:\Belajar coding\Strukdat\Learning Code\STRUKDAT (P
RAKTEK)\LL>
```

Deskripsi Code

Pada pemrograman ini memakai single linked list yang memiliki satu pointer yang menunjuk node selanjutnya dalam urutan linear, pada kasus tersebut, pada setiap nodenya memiliki satu pointer yang menunjuk pada linear dari node ke node berikutnya yang mana akan mempermudah pengelolaan data yang lebih efesien walaupun pada operasi tersebut memiliki keterbatasan

seperti mencari node sebelumnya secara langsung.

Pemrogramanan berfungsi dapat digunakan untuk para user untuk menambah, menghapus dan mengubah pada bagian yang ingin diberi tindakan dari user dan program tereksekusi lewat perintah yang telah diprogram dari code. Jika sudah selesai maka user dapat memilih opsi paling terakhir 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 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;
}
```

Output Code

1. Menambah data pada program

```
Pilihan : 1
Masukkan nama produk: lacoco
Masukkan harga produk: 30000
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 : 1
Masukkan nama produk: bhumi
Masukkan harga produk: 40000
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      |
-------------------------------------
| bhumi                | 40000      |
| lacoco               | 30000      |
| mineral botanica     | 80000      |
| kahf                 | 60000      |
| scarlett             | 50000      |
-------------------------------------
```

2. Menghapus data pada program

```
-------------------------------------
| Nama Produk           | Harga     |
-------------------------------------
| jai                   | 10000     |
| scarlett              | 50000     |
| kahf                  | 60000     |
| olar                  | 25000     |
| mineral botanica      | 80000     |
| lacoco                | 30000     |
| bhumi                 | 40000     |
-------------------------------------
```

```
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 : 2
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     |
-------------------------------------
| scarlett              | 50000     |
| kahf                  | 60000     |
| olar                  | 25000     |
| mineral botanica      | 80000     |
| lacoco                | 30000     |
| bhumi                 | 40000     |
-------------------------------------
```

3. Mengubah/update data pada program

```
Pilihan : 4
Masukkan posisi data produk: 2
Masukkan nama produk: olar
Masukkan harga produk: 25000
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      |
-------------------------------------
| scarlett             | 50000      |
| kahf                 | 60000      |
| olar                 | 25000      |
| mineral botanica     | 80000      |
| lacoco               | 30000      |
| bhumi                | 40000      |
-------------------------------------
```

4. Hapus data tertentu pada program

```
-------------------------------------
| Nama Produk           | Harga     |
-------------------------------------
| scarlett              | 50000     |
| kahf                  | 60000     |
| olar                  | 25000     |
| mineral botanica      | 80000     |
| lacoco                | 30000     |
| bhumi                 | 40000     |
-------------------------------------
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 : 5
Masukkan posisi data produk: 2
-------------------------------------
| Nama Produk           | Harga     |
-------------------------------------
| scarlett              | 50000     |
| kahf                  | 60000     |
| mineral botanica      | 80000     |
| lacoco                | 30000     |
| bhumi                 | 40000     |
-------------------------------------
```

5. Menghapus semua data pada program

```
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 : 6
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
List kosong.
```

6. Menampilkan semua program

```
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      |
-------------------------------------
| scarlett             | 50000      |
| kahf                 | 60000      |
| olar                 | 25000      |
| mineral botanica     | 80000      |
| lacoco               | 30000      |
| bhumi                | 40000      |
-------------------------------------
```

Deskripsi Code

Double linked list pada pemrograman tersebut berfungsi untuk mengelola data produk dari skincare dari nama dan harga yang nodenya memiliki penunjuk dari node sebelumnya ke node selanjutnya. Dari program tersebut hampir sama dengan program sebelumnya pada unguided1 yang mana user dapat menambah, menghapus, mengubah dan menampilkan data. Tetapi, terdapat beberapa fitur tambahan seperti user dapat menambah atau menghapus pada baris data yang ingin diberi tindakan.

C. Kesimpulan

Mahasiswa dapat mengetahui bahwa linked list pada pemrograman C++ dapat digunakan sebagai struktur data yang dapat beroperasi seperti menambah data, menghapus data, mengubah data, hingga menampilkan data. Operasi tersebut dapat dijalankan dari node yang saling terhubung dalam urutan tertentu pada pemrograman.

D. Referensi

[1] Tim Asisten Asprak, “Single Dan Double Linked List”, Learning Management System, 2024.

[2] Trivusi, “Struktur Data Linked List: Pengertian, Karakteristik, dan Jenis-jenisnya”, Struktur Data Linked List: Pengertian, Karakteristik, dan Jenis-jenisnya - Trivusi, terakhir kali diakses pada tanggal 31 Maret 2024.