

# **LAPORAN PRAKTIKUM**

## **MODUL III SINGLE AND DOUBLE LINKED LIST**



**Disusun oleh:**  
**Geranada Saputra Priambudi**  
**NIM: 2311102008**

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

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

## **BAB I**

### **TUJUAN PRAKTIKUM**

- a. Mahasiswa memahami perbedaan konsep Single dan Double Linked List.
- b. Mahasiswa mampu menerapkan Single dan Double Linked List ke dalam pemrograman.

## **BAB II**

### **DASAR TEORI**

Linked List adalah sekumpulan list yang saling terhubung dan list disini dapat menampung banyak nilai seperti halnya array. Wah hampir benar, jadi Linked list adalah struktur data yang menyimpan elemen secara berurutan yang dihubungkan dengan pointer. Berikut ada beberapa poin poin yang menjelaskan mengenai linked list :

- Tidak seperti array yang menampung elemen berutan dengan lokasi memori yang berutan pula, pada linked list elemen yang terhubung penyimpanan memorinya tidak harus berurutan atau berdekatan.
- Berbeda dengan array yang memiliki ukuran yang statis, Linked List memiliki ukuran yang dinamis dengan batas maksimal memori yang ada.
- Dalam linked list dikenal istilah node, yaitu setiap elemen yang berada pada linked list.
- Setiap node ini biasanya merupakan struct, namun apa itu struct? struct singkatnya adalah data yang dibentuk oleh beberapa data.
- Pada setiap node atau elemen linked list, pada data struct nya akan memiliki 1 variabel pointer yang biasanya dinamakan \*next.
- Node tail atau node terakhir tidak akan menunjuk kemana pun. Jadi nilai pointer nextnya akan NULL.

Operasi dalam Linked List :

- Insert yaitu menambahkan node ke bagian awal atau bagian akhir dari Linked List.
- Update yaitu merubah nilai salah satu node.
- Delete yaitu menghapus node yang berada pada bagian awal (head) atau akhir (tail).
- Display yaitu menampilkan setiap node yang ada pada Linked List.
- Travers yaitu melintasi setiap node pada linked list, biasa digunakan pada operasi display.

#### Jenis Linked List :

- Single Linked List merupakan jenis linked list yang hanya memiliki 1 pointer saja. Pointer digunakan untuk menunjuk node selanjutnya (next), kecuali pada node tail atau node terakhir yang pointernya menunjuk ke NULL.
- Double Linked List merupakan jenis linked list yang memiliki 2 pointer. 1 pointer menunjuk ke node selanjutnya (next) dan 1 lagi menunjuk ke node sebelumnya (prev). Pada node head, pointer prev akan bernilai NULL karena node Head adalah node pertama. Pada node Tail, pointer next akan menunjuk ke NULL.
- Circular Linked List merupakan jenis linked list yang melingkar, jadi node tail (node terakhir) akan menunjuk ke node head (node pertama).
- Multiple Linked List merupakan jenis linked list yang memiliki lebih dari 2 variabel pointer.

## BAB III

### GUIDED

#### 1. Guided 1

##### Source code

```
#include <iostream>

using namespace std;

/// PROGRAM SINGLE LINKED LIST NON-CIRCULAR
// Deklarasi Struct Node
struct Node
{
    // komponen/member
    int data;
    string kata;
    Node *next;
};

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

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

```
// Tambah Depan
void insertDepan(int nilai, string kata)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->kata = kata;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}

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

```

```

        tail->next = baru;
        tail = baru;
    }
}
// Hitung Jumlah List
int hitungList()
{
    Node *hitung;
    hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}
// Tambah Tengah
void insertTengah(int data, string kata, int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        Node *baru, *bantu;
        baru = new Node();
        baru->data = data;
        baru->kata = kata;
    }
}

```

```

        // tranversing
        bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

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

```



```

// Hapus Belakang
void hapusBelakang()
{
    Node *hapus;
    Node *bantu;
    if (isEmpty() == false)
    {
        if (head != tail)
        {
            hapus = tail;
            bantu = head;
            while (bantu->next != tail)
            {
                bantu = bantu->next;
            }
            tail = bantu;
            tail->next = NULL;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}

// Hapus Tengah
void hapusTengah(int posisi)
{
    Node *hapus, *bantu, *bantu2;
    if (posisi < 1 || posisi > hitungList())

```

```

    {
        cout << "Posisi di luar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        int nomor = 1;
        bantu = head;
        while (nomor <= posisi)
        {
            if (nomor == posisi - 1)
            {
                bantu2 = bantu;
            }
            if (nomor == posisi)
            {
                hapus = bantu;
            }
            bantu = bantu->next;
            nomor++;
        }
        bantu2->next = bantu;
        delete hapus;
    }
}

// Ubah Depan
void ubahDepan(int data, string kata)
{
    if (isEmpty() == false)
    {
        head->data = data;
    }
}

```

```

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

```

```

        else
        {
            cout << "List masih kosong!" << endl;
        }
    }
    // Ubah Belakang
    void ubahBelakang(int data, string kata)
    {
        if (isEmpty() == false)
        {
            tail->data = data;
            tail->kata = kata;
        }
        else
        {
            cout << "List masih kosong!" << endl;
        }
    }
    // Hapus List
    void clearList()
    {
        Node *bantu, *hapus;
        bantu = head;
        while (bantu != NULL)
        {
            hapus = bantu;
            bantu = bantu->next;
            delete hapus;
        }
        head = tail = NULL;
        cout << "List berhasil terhapus!" << endl;
    }
    // Tampilkan List
    void tampil()

```

```

{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << "\t";
            cout << bantu->kata<<"\t";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

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

```

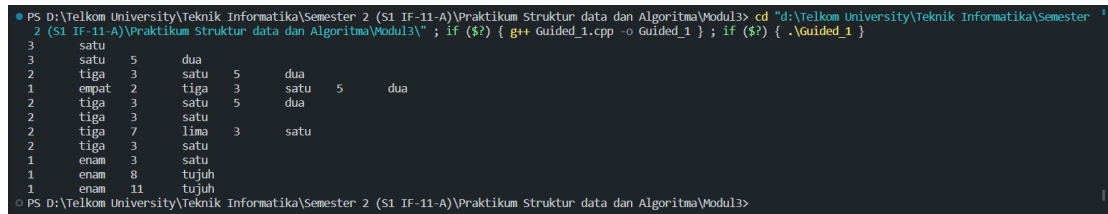
```

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

    return 0;
}

```

## Screenshoot program



```

PS D:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul3> cd "d:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul3\" ; if ($?) { g++ Guided_1.cpp -o Guided_1 } ; if ($?) { .\Guided_1 }
3      satu 5      dua
2      tiga 3      satu 5      dua      5      dua
1      empat 2      tiga 3      satu 5      dua
2      tiga 3      satu 5      dua
2      tiga 7      lima 3      satu
2      tiga 3      satu
1      enam 3      satu
1      enam 8      tujuh
1      enam 11     tujuh
PS D:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul3>

```

## Deskripsi program

Program diatas merupakan program untuk mendemonstrasikan operasi dasar pada struktur data Single Linked List Non-Circular. Program ini memungkinkan pengguna untuk mengelola data yang terdiri dari pasangan bilangan integer dan string.

## 2. Guided 2

### Source Code

```
#include <iostream>

using namespace std;

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

class DoublyLinkedList
{
public:
    Node *head;
    Node *tail;
    DoublyLinkedList()
    {
        head = nullptr;
        tail = nullptr;
    }
    void push(int data, string kata)
    {
        Node *newNode = new Node;
        newNode->data = data;
        newNode->kata = kata;
        newNode->prev = nullptr;
        newNode->next = head;
        if (head != nullptr)
        {
            head->prev = newNode;
        }
        else
        {
            tail = newNode;
        }
        head = newNode;
    }
    void pop()
    {
        if (head == nullptr)
        {
            return;
        }
        Node *temp = head;
        head = head->next;
        if (head != nullptr)
        {
            temp->next = nullptr;
        }
        delete temp;
    }
};
```

```

        head->prev = nullptr;
    }
    else
    {
        tail = nullptr;
    }
    delete temp;
}
bool update(int oldData, int newData, string oldKata, string
newKata)
{
    Node *current = head;
    while (current != nullptr)
    {
        if (current->data == oldData)
        {
            current->data = newData;
            current->kata = newKata;
            return true;
        }
        current = current->next;
    }
    return false;
}
void deleteAll()
{
    Node *current = head;
    while (current != nullptr)
    {
        Node *temp = current;
        current = current->next;
        delete temp;
    }
    head = nullptr;
    tail = nullptr;
}
void display()
{
    Node *current = head;
    while (current != nullptr)
    {
        cout << current->data << " " << current->kata << endl;
        current = current->next;
    }
    cout << endl;
}
};
int main()
{
    DoublyLinkedList list;
    while (true)
    {

```



```

cout << "1. Add data" << endl;
cout << "2. Delete data" << endl;
cout << "3. Update data" << endl;
cout << "4. Clear data" << endl;
cout << "5. Display data" << endl;
cout << "6. Exit" << endl;
int choice;
cout << "Enter your choice: ";
cin >> choice;
switch (choice)
{
case 1:
{
    int data;
    string kata;
    cout << "Enter data to add: ";
    cin >> data;
    cout << "Enter kata to add: ";
    cin >> kata;
    list.push(data, kata);
    break;
}
case 2:
{
    list.pop();
    break;
}
case 3:
{
    int oldData, newData;
    string oldKata, newKata;
    cout << "Enter old data: ";
    cin >> oldData;
    cout << "Enter new data: ";
    cin >> newData;
    cout << "Enter old Kata: ";
    cin >> oldKata;
    cout << "Enter new kata: ";
    cin >> newKata;
    bool updated = list.update (oldData, newData,
oldKata, newKata);
    if (!updated)
    {
        cout << "Data not found" << endl;
    }
    break;
}
case 4:
{
    list.deleteAll();
    break;
}
}

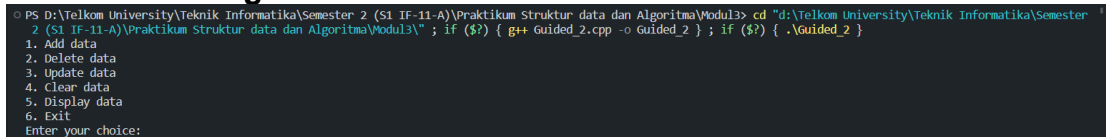
```

```

        case 5:
        {
            list.display();
            break;
        }
        case 6:
        {
            return 0;
        }
        default:
        {
            cout << "Invalid choice" << endl;
            break;
        }
    }
    return 0;
}

```

## Screenshot Program



```

PS D:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul3> cd "d:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul3\" ; if ($?) { g++ Guided_2.cpp -o Guided_2 } ; if ($?) { .\Guided_2 }
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice:

```

## Deskripsi program

Program diatas merupakan program untuk mengelola data yang terdiri dari pasangan bilangan integer string menggunakan Double Linked List. Program ini merupakan contoh penggunaan Double Linked List untuk mengelola data berpasangan integer dan string. program ini memungkinkan pengguna untuk menambah, menghapus, mengubah dan menampilkan data yang tersimpan.

## LATIHAN KELAS - UNGUIDED

### 1. Unguided 1

#### Source code

```
#include <iostream>
#include <string>

using namespace std;

struct Node {
    string name;
    int age;
    Node* next;
};

class LinkedList {
private:
    Node* head;

public:
    LinkedList() {
        head = nullptr;
    }

    // Masukkan node diawal
    void insertFirst(string name, int age) {
        Node* newNode = new Node;
        newNode->name = name;
        newNode->age = age;
        newNode->next = head;
        head = newNode;
    }
}
```

```

// Masukkan node setelah node spesifik
void insertAfter(string prevName, string name, int age) {
    Node* current = head;
    while (current != nullptr) {
        if (current->name == prevName) {
            Node* newNode = new Node;
            newNode->name = name;
            newNode->age = age;
            newNode->next = current->next;
            current->next = newNode;
            return;
        }
        current = current->next;
    }
    cout << "Node with name " << prevName << " not found!"
<< endl;
}

// Hapus node berdasarkan nama
void deleteNode(string name) {
    Node* temp = head;
    Node* prev = nullptr;

    // Jika head node memegang kunci yang akan dihapus
    if (temp != nullptr && temp->name == name) {
        head = temp->next;
        delete temp;
        return;
    }

    // Cari kunci yang akan dihapus, pantau node sebelumnya
    // karena perlu mengubah 'prev->next'
    while (temp != nullptr && temp->name != name) {
        prev = temp;

```

```

        temp = temp->next;
    }

    // Jika kunci tidak ada dalam linked list
    if (temp == nullptr) {
        cout << "Node with name " << name << " not found!"
<< endl;
        return;
    }

    // Putuskan node dari linked list
    prev->next = temp->next;
    delete temp;
}

// Ubah node berdasarkan nama
void modifyNode(string name, string newName, int newAge) {
    Node* current = head;
    while (current != nullptr) {
        if (current->name == name) {
            current->name = newName;
            current->age = newAge;
            return;
        }
        current = current->next;
    }
    cout << "Node with name " << name << " not found!" <<
endl;
}

// Tampilkan semua node
void display() {
    Node* current = head;
    while (current != nullptr) {

```

```

        cout << current->name << "\t" << current->age <<
endl;

        current = current->next;

    }

}

};

int main() {
    LinkedList list;

    // Masukkan data sesuai urutan
    list.insertFirst("John", 19);
    list.insertAfter("John", "Jane", 20);
    list.insertAfter("Jane", "Michael", 18);
    list.insertAfter("Michael", "Yusuke", 19);
    list.insertAfter("Yusuke", "Akechi", 20);
    list.insertAfter("Akechi", "Hoshino", 18);
    list.insertAfter("Hoshino", "Karin", 18);

    // Tampilkan data
    cout << "Data before modifications:" << endl;
    list.display();
    cout << endl;

    // Lakukan operasi
    list.deleteNode("Akechi"); // Delete Akechi
    list.insertAfter("John", "Futaba", 18); // Insert Futaba
between John and Jane
    list.insertFirst("Igor", 20); // Insert Igor at the beginning
    list.modifyNode("Michael", "Reyn", 18); // Modify Michael to
Reyn

    // Tampilan setelah di modifikasi
    cout << "Data after modifications:" << endl;

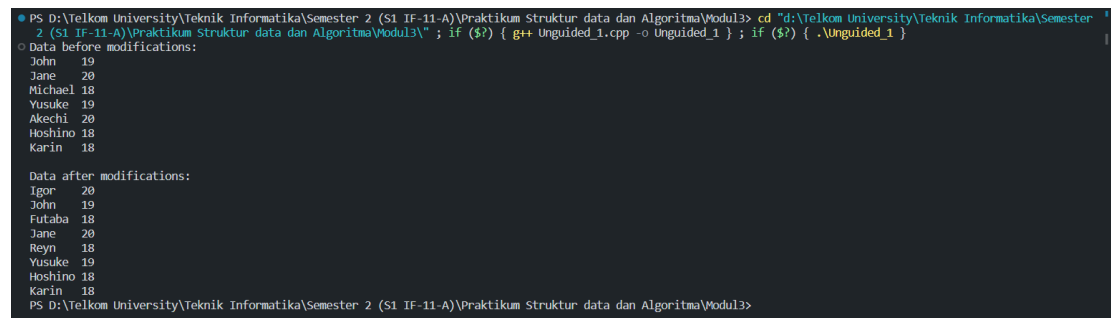
```

```
list.display();

return 0;

}
```

## Screenshoot program



```
PS D:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul3> cd "d:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul3\" ; if ($?) { g++ Unguided_1.cpp -o Unguided_1 } ; if ($?) { .\Unguided_1 }
Data before modifications:
John 19
Jane 20
Michael 18
Yusuke 19
Akechi 20
Hoshino 18
Karin 18

Data after modifications:
Igor 20
John 19
Futaba 18
Jane 20
Reyn 18
Yusuke 19
Hoshino 18
Karin 18
PS D:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul3>
```

## Deskripsi program

Program diatas merupakan program untuk mendemonstrasikan penggunaan Linked List dalam pengelolaan data. Program ini difokuskan pada pengelolaan data orang yang terdiri dari nama dan umur. Program ini merupakan contoh dasar penggunaan Linked List untuk mengelola data orang. Program ini berhasil menunjukkan kemampuan untuk menambah, menghapus, mengubah dan menampilkan data yang tersimpan dalam Linked List.

## 2. Unguided 2

### Source code

```
#include <iostream>
#include <vector>

using namespace std;

struct Produk
{
    string nama;
    int harga;
};

class TokoSkincare
{
private:
    vector<Produk> produkList;

public:
    void tambahData(string nama, int harga)
    {
        Produk newProduk;
        newProduk.nama = nama;
        newProduk.harga = harga;
        produkList.push_back(newProduk);
    }

    void hapusData(string nama)
    {
        for (int i = 0; i < produkList.size(); i++)
        {
            if (produkList[i].nama == nama)
            {
                produkList.erase(produkList.begin() + i);
            }
        }
    }
}
```



```

        return;
    }
}

cout << "Produk " << nama << " tidak ditemukan!" << endl;
}

void updateData(string nama, int harga)
{
    bool found = false;
    for (int i = 0; i < produkList.size(); i++)
    {
        if (produkList[i].nama == nama)
        {
            produkList[i].harga = harga;
            found = true;
            break;
        }
    }
    if (!found)
    {
        cout << "Produk " << nama << " tidak ditemukan!" <<
endl;
    }
}

void tambahDataUrutanTertentu(string nama, int harga, int
urutan)
{
    if (urutan < 0 || urutan > produkList.size())
    {
        cout << "Urutan tidak valid!" << endl;
        return;
    }
    Produk newProduk;

```

```

        newProduk.nama = nama;
        newProduk.harga = harga;
        produkList.insert(produkList.begin() + urutan,
newProduk);
    }

void hapusDataUrutanTertentu(int urutan)
{
    if (urutan < 0 || urutan >= produkList.size())
    {
        cout << "Urutan tidak valid!" << endl;
        return;
    }
    produkList.erase(produkList.begin() + urutan);
}

void tampilkanData()
{
    cout << "=====" << endl;
    cout << "Toko Skincare Purwokerto" << endl;
    cout << "=====" << endl;
    for (Produk produk : produkList)
    {
        cout << "Nama Produk: " << produk.nama << endl;
        cout << "Harga: " << produk.harga << endl
            << endl;
    }
}

void hapusSeluruhData()
{
    produkList.clear();
    cout << "Data berhasil dihapus!" << endl;
}

```

```

};

int main()
{
    TokoSkincare toko;

    // Menambahkan beberapa produk awal
    toko.tambahData("Originote", 60000);
    toko.tambahData("Somethinc", 150000);
    toko.tambahData("Skintific", 100000);
    toko.tambahData("Wardah", 50000);
    toko.tambahData("Hanasui", 30000);

    int pilihan;
    string nama;
    int harga, urutan;

    do
    {
        cout << "\n===== " << endl;
        cout << "Menu Toko Skincare Purwokerto" << endl;
        cout << "===== " << 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
            << endl;

        cout << "Masukkan pilihan: ";
        cin >> pilihan;
    }
}

```

```

switch (pilihan)
{
case 1:
    cout << "Masukkan nama produk: ";
    cin >> nama;
    cout << "Masukkan harga produk: ";
    cin >> harga;
    toko.tambahData(nama, harga);
    break;
case 2:
    cout << "Masukkan nama produk yang ingin dihapus: ";
    cin >> nama;
    toko.hapusData(nama);
    break;
case 3:
    cout << "Masukkan nama produk yang ingin diupdate:
";

    cin >> nama;
    cout << "Masukkan harga baru: ";
    cin >> harga;
    toko.updateData(nama, harga);
    break;
case 4:
    cout << "Masukkan nama produk: ";
    cin >> nama;
    cout << "Masukkan harga produk: ";
    cin >> harga;
    cout << "Masukkan urutan penempatan (1-based): ";
    cin >> urutan;
    toko.tambahDataUrutanTertentu(nama, harga, urutan -
1);

    break;
case 5:

```

```

        cout << "Masukkan urutan data yang ingin dihapus (1-
based): ";

        cin >> urutan;
        toko.hapusDataUrutanTertentu(urutan - 1);
        break;

    case 6:
        toko.hapusSeluruhData();
        break;

    case 7:
        toko.tampilkanData();
        break;

    case 8:
        cout << "Terima kasih!" << endl;
        break;

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

return 0;
}

```

## Screenshot program

```

PS D:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul3> cd "d:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul3\" ; if ($?) { g++ Unguided_2.cpp -o Unguided_2 } ; if ($?) { .\Unguided_2 }

=====
Menu 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

Masukkan pilihan: 4
Masukkan nama produk: Azarine
Masukkan harga produk: 65000
Masukkan urutan penempatan (1-based): 3

=====
Menu 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

Masukkan pilihan: 2
Masukkan nama produk yang ingin dihapus: Wardah

```

```

=====
Menu 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

Masukkan pilihan: 2
Masukkan nama produk yang ingin dihapus: Hanasui

=====
Menu 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

Masukkan pilihan: 1
Masukkan nama produk: Cleora
Masukkan harga produk: 55000

```

```

=====
Menu 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

Masukkan pilihan: 7
=====
Toko Skincare Purwokerto
=====
Nama Produk: Originote
Harga: 60000

Nama Produk: Somethinc
Harga: 150000

Nama Produk: Azarine
Harga: 65000

Nama Produk: Skintific
Harga: 100000

Nama Produk: Cleora
Harga: 55000

```

```

=====
Menu 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

o Masukkan pilihan: 8
Terima kasih!
PS D:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul3>

```

## Deskripsi program

Program diatas merupakan sebuah program yang mensimulasikan pencatatan data produk di took skincare purwokerto. Program ini membantu took dalam mengelola data produk dengan lebih mudah dan efisien. Pengguna dapat dengan mudah menambahkan, menghapus dan memperbarui data produk. Selain itu, pengguna juga dapat melihat seluruh data produk yang terdaftar dan menghapus seluruh data sekaligus.

## **BAB IV**

### **KESIMPULAN**

Single dan Double Linked List adalah struktur data yang memiliki kelebihan dan kekurangan masing-masing. Pilihan struktur data yang tepat tergantung pada kebutuhan aplikasi. Single Linked List cocok untuk aplikasi yang membutuhkan struktur data sederhana dan efisien dalam penggunaan memori, sedangkan Double Linked List cocok untuk aplikasi yang membutuhkan performa yang lebih baik dalam operasi penghapusan dan akses node di Tengah list. Praktikum ini memberikan pemahaman yang mendalam tentang karakteristik, kelebihan dan keterbatasan dari Single dan Double Linked List, serta memperluas pemahaman tentang penerapan struktur data dalam pengembangan perangkat lunak.