

LAPORAN PRAKTIKUM

MODUL II ARRAY



**Disusun oleh:
Arjun Werdho Kumoro NIM: 2311102009**

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

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

MODUL 3

SINGLE AND DOUBLE LINKED LIST

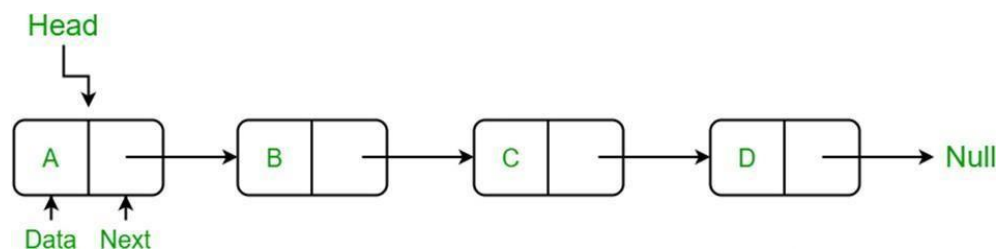
A. Tujuan

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

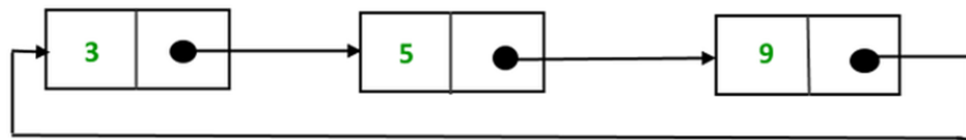
a) Single Linked List

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.

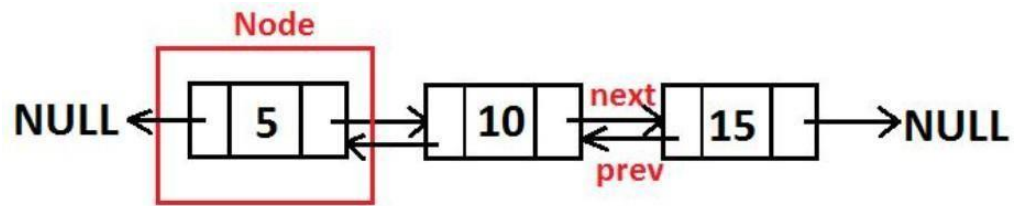


b) Double Linked List

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
{
    // 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->kata = kata;
    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, string kata)
{
    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)
{
    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 << "\t";
            cout << bantu->kata;
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

int main()
{
    init();
    insertDepan(3, "satu");
    tampil();
    insertBelakang(5 , "tiga");
    tampil();
    insertDepan(2 , "lima");
    tampil();
    insertDepan(1, "dua");
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, 2 , "tujuh");
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1);
    tampil();
    ubahBelakang(8);
    tampil();
    ubahTengah(11, 2);
}

```

```
tampil();  
return 0;  
}
```

OUTPUT

```
nterpreter=mi'  
3      satu  
3      satu5   tiga  
2      lima3   satu5   tiga  
1      dua2    lima3   satu5   tiga  
2      lima3   satu5   tiga  
2      lima3   satu  
2      lima7   tujuh3   satu  
2      lima3   satu  
1      lima3   satu  
1      lima8   satu  
1      lima11  satu  
PS C:\Users\lenovo>
```

Deskripsi program:

- Program pertama kali menambahkan node dengan nilai 3 dan kata "satu" ke depan list, sehingga list menjadi "3 satu".
- Kemudian program menambahkan node dengan nilai 5 dan kata "tiga" ke belakang list, sehingga list menjadi "3 satu 5 tiga".
- Selanjutnya, program menambahkan node dengan nilai 2 dan kata "lima" ke depan list, sehingga list menjadi "2 lima 3 satu 5 tiga".
- Setelah itu, program menambahkan node dengan nilai 1 dan kata "dua" ke depan list, sehingga list menjadi "1 dua 2 lima 3 satu 5 tiga".
- Program menghapus node pertama dari list, sehingga list menjadi "2 lima 3 satu 5 tiga".
- Kemudian, program menghapus node terakhir dari list, sehingga list menjadi "2 lima 3 satu".
- Selanjutnya, program menambahkan node dengan nilai 7 dan kata "tujuh" di posisi kedua, sehingga list menjadi "2 tujuh lima 3 satu".
- Program menghapus node kedua dari list, sehingga list menjadi "2 lima 3 satu".
- Kemudian, program mengubah nilai node pertama menjadi 1, sehingga list menjadi "1 lima 3 satu".
- Program mengubah nilai node terakhir menjadi 8, sehingga list menjadi "1 lima 3 delapan".
- Terakhir, program mengubah nilai node kedua menjadi 11, sehingga list menjadi "1 sebelas 3 delapan".

b) Latihan Double Linked List

Source Code

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

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

class DoublyLinkedList {
public:
    Node* head;
    Node* tail;
    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    void push(int data, string kata) {
        Node* newNode = new Node;
        newNode->data = data;
        newNode->kata = kata;
        newNode->prev = nullptr;
        newNode->next = head;
        if (head != nullptr) {
            head->prev = newNode;
        } else {
            tail = newNode;
        }
        head = newNode;
    }

    void pop() {
        if (head == nullptr) {
            return;
        }
        Node* temp = head;
        head = head->next;
        if (head != nullptr) {
            head->prev = nullptr;
        } else {
            tail = nullptr;
        }
        delete temp;
    }

    bool update(int oldData, int newData, string 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 << " ";
            cout << 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 newKata;
                cout << "Enter old data: ";
                cin >> oldData;
                cout << "Enter new data: ";
                cin >> newData;
                cout << "Enter new kata: ";
                cin >> newKata;
                bool updated = list.update(oldData,
                    newData, 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;
}
```

OUTPUT:

```
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: 4
Enter kata to add: uyeee
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
4 uyeee
```

Deskripsi program: Dari hasil output dapat di simpulkan program menampilkan menu Add data, Delete data, Update data, Clear data, Display data, Exit kemudian user di suruh menginputkan data melalui menu 1 jika menghapus data menu 2 dan selanjutnya

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>
using namespace std;
struct Node
{
    string nama;
    int usia;
    Node *next;
};
class LinkedList
{
private:
    Node *head;
public:
    LinkedList()
```

```

{
    head = nullptr;
}
void insertAwal(string nama, int usia)
{
    Node *newNode = new Node;
    newNode->nama = nama;
    newNode->usia = usia;
    newNode->next = head;
    head = newNode;
}
void insertAkhir(string nama, int usia)
{
    Node *newNode = new Node;
    newNode->nama = nama;
    newNode->usia = usia;
    newNode->next = nullptr;
    if (head == nullptr)
    {
        head = newNode;
        return;
    }
    Node *temp = head;
    while (temp->next != nullptr)
    {
        temp = temp->next;
    }
    temp->next = newNode;
}
void insertSetelah(string nama, int usia, string namaSebelum)
{
    Node *newNode = new Node;
    newNode->nama = nama;
    newNode->usia = usia;
    Node *temp = head;
    while (temp != nullptr && temp->nama != namaSebelum)
    {
        temp = temp->next;
    }
    if (temp == nullptr)
    {
        cout << "Node dengan nama " << namaSebelum << " tidak ditemukan."
<< endl;
        return;
    }
    newNode->next = temp->next;
    temp->next = newNode;
}
void hapus(string nama)
{
    if (head == nullptr)
    {
        cout << "Linked list kosong." << endl;
        return;
    }
    if (head->nama == nama)
    {
        Node *temp = head;

```

```

        head = head->next;
        delete temp;
        return;
    }
    Node *prev = head;
    Node *temp = head->next;
    while (temp != nullptr && temp->nama != nama)
    {
        prev = temp;
        temp = temp->next;
    }
    if (temp == nullptr)
    {
        cout << "Node dengan nama " << nama << " tidak ditemukan." <<
endl;
        return;
    }
    prev->next = temp->next;
    delete temp;
}
void ubah(string nama, string namaBaru, int usiaBaru)
{
    Node *temp = head;
    while (temp != nullptr && temp->nama != nama)
    {
        temp = temp->next;
    }
    if (temp == nullptr)
    {
        cout << "Node dengan nama " << nama << " tidak ditemukan." <<
endl;
        return;
    }
    temp->nama = namaBaru;
    temp->usia = usiaBaru;
}
void tampilkan()
{
    Node *temp = head;
    while (temp != nullptr)
    {
        cout << temp->nama << " " << temp->usia << endl;
        temp = temp->next;
    }
}
};
int main()
{
    LinkedList myList;
    myList.insertAwal("Karin", 18);
    myList.insertAwal("Hoshino", 18);
    myList.insertAwal("Akechi", 20);
    myList.insertAwal("Yusuke", 19);
    myList.insertAwal("Michael", 18);
    myList.insertAwal("Jane", 20);
    myList.insertAwal("John", 19);
    myList.insertAwal("Ajun werdho kumoro", 19);
    cout << "=====LANGKAH A=====:" << endl;

```

```

myList.tampilkan();
cout << endl;

myList.hapus("Akechi");
cout << "=====LANGKAH B=====:" << endl;
cout << "=====MENGHAPUS AKECHI:=====:" << endl;
myList.tampilkan();
cout << endl;

myList.insertSetelah("Futaba", 18, "Jane");
cout << "=====LANGKAH C=====:" << endl;
cout << "==MENAMBAHKAN DATA FUTABA DIANTARA JOHN DAN JANE==" << endl;
myList.tampilkan();
cout << endl;

myList.insertAwal("Igor", 20);
cout << "=====LANGKAH D=====:" << endl;
cout << "=====MENAMBAHKAN DATA IGOR KE DATA AWAL=====:" << endl;
myList.tampilkan();
cout << endl;

myList.ubah("Michael", "Reyn", 18);
cout << "=====LANGKAH E=====:" << endl;
cout << "=====MENGUBAH DATA MICHAEL DENGAN DATA REYN=====:" << endl;
myList.tampilkan();
cout << endl;

cout << "=====LANGKAH F=====:" << endl;
cout << "=====MENAMPILKAN SELURUH DATA=====:" << endl;
myList.tampilkan();
cout << endl;

return 0;
}

```

OUTPUT

```

=====LANGKAH A=====:
Ajun werdho kumoro 19
John 19
Jane 20
Michael 18
Yusuke 19
Akechi 20
Hoshino 18
Karin 18

=====LANGKAH B=====:
=====MENGHAPUS AKECHI:=====
Ajun werdho kumoro 19
John 19
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18

=====LANGKAH C=====:
==MENAMBAHKAN DATA FUTABA DIANTARA JOHN DAN JANE==
Ajun werdho kumoro 19
John 19
Jane 20
Futaba 18
Michael 18
Yusuke 19
Hoshino 18
Karin 18

```

```

=====LANGKAH D=====:
=====MENAMBAHKAN DATA IGOR KE DATA AWAL=====
Igor 20
Ajun werdho kumoro 19
John 19
Jane 20
Futaba 18
Michael 18
Yusuke 19
Hoshino 18
Karin 18

=====LANGKAH E=====:
=====MENGUBAH DATA MICHAEL DENGAN DATA REYN=====
Igor 20
Ajun werdho kumoro 19
John 19
Jane 20
Futaba 18
Reyn 18
Yusuke 19
Hoshino 18
Karin 18

=====LANGKAH F=====:
=====MENAMPILKAN SELURUH DATA=====
Igor 20
Ajun werdho kumoro 19
John 19
Jane 20
Futaba 18
Reyn 18
Yusuke 19
Hoshino 18
Karin 18

```

Deskripsi program:

1. LANGKAH A: Menampilkan semua data yang telah dimasukkan ke dalam linked list pada awal program. Data yang ditampilkan adalah:

- Ajun werdho kumoro 19
- John 19
- Jane 20
- Michael 18
- Yusuke 19
- Akechi 20
- Hoshino 18
- Karin 18

2. LANGKAH B: Menghapus data nama "Akechi" dari linked list. Setelah penghapusan, data yang ditampilkan adalah:

- Ajun werdho kumoro 19
- John 19
- Jane 20
- Michael 18
- Yusuke 19
- Hoshino 18
- Karin 18

3. LANGKAH C: Menambahkan data "Futaba" setelah data dengan nama "Jane" dalam linked list. Setelah penambahan, data yang ditampilkan adalah:

- Ajun werdho kumoro 19
- John 19
- Jane 20
- Futaba 18
- Michael 18
- Yusuke 19
- Hoshino 18
- Karin 18

4. LANGKAH D: Menambahkan data "Igor" ke awal linked list. Setelah penambahan, data yang ditampilkan adalah:

- Igor 20
- Ajun werdho kumoro 19
 - John 19

- Jane 20
 - Futaba 18
 - Michael 18
 - Yusuke 19
 - Hoshino 18
 - Karin 18
5. LANGKAH E: Mengubah data "Michael" menjadi "Reyn" dengan usia baru 18. Setelah perubahan, data yang ditampilkan adalah:
- Igor 20
- Ajun werdho kumoro 19
- John 19
 - Jane 20
 - Futaba 18
 - Reyn 18
 - Yusuke 19
 - Hoshino 18
 - Karin 18
6. LANGKAH F: Menampilkan seluruh data yang tersimpan dalam linked list. Outputnya sama dengan LANGKAH E.

7. 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 <string>
using namespace std;
struct Node
{
    string nama;
    int harga;
    Node *prev;
    Node *next;
};
class DoubleLinkedList
{
private:
    Node *head;
    Node *tail;
    int size;

public:
    DoubleLinkedList()
    {
        head = NULL;
        tail = NULL;
        size = 0;
    }
    void addData(string nama, int harga)
    {
        Node *node = new Node;
        node->nama = nama;
        node->harga = harga;
        node->prev = tail;
        node->next = NULL;
        if (head == NULL)
        {
            head = node;
            tail = node;
        }
        else
        {
            tail->next = node;
            tail = node;
        }
        size++;
    }
    void addDataAt(int index, string nama, int harga)
    {
        if (index < 0 || index > size)
        {
            cout << "Index out of bounds" << endl;
            return;
        }
    }
};
```



```

    }
    Node *node = new Node;
    node->nama = nama;
    node->harga = harga;
    if (index == 0)
    {
        node->prev = NULL;
        node->next = head;
        head->prev = node;
        head = node;
    }
    else if (index == size)
    {
        node->prev = tail;
        node->next = NULL;
        tail->next = node;
        tail = node;
    }
    else
    {
        Node *current = head;
        for (int i = 0; i < index - 1; i++)
        {
            current = current->next;
        }
        node->prev = current;
        node->next = current->next;
        current->next->prev = node;
        current->next = node;
    }
    size++;
}

void deleteDataAt(int index)
{
    if (index < 0 || index >= size)
    {
        cout << "Index out of bounds" << endl;
        return;
    }
    if (index == 0)
    {
        Node *temp = head;
        head = head->next;
        head->prev = NULL;
        delete temp;
    }
    else if (index == size - 1)
    {
        Node *temp = tail;
        tail = tail->prev;
        tail->next = NULL;
        delete temp;
    }
    else
    {
        Node *current = head;
        for (int i = 0; i < index; i++)
        {
            current = current->next;
        }
        current->prev->next = current->next;
        current->next->prev = current->prev;
        delete current;
    }
    size--;
}

void clearData()

```

```

{
    while (head != NULL)
    {
        Node *temp = head;
        head = head->next;
        delete temp;
    }
    tail = NULL;
    size = 0;
}
void displayData()
{
    cout << "Nama Produk\tHarga" << endl;
    Node *current = head;
    while (current != NULL)
    {
        cout << current->nama << "\t\t" << current->harga
            << endl;
        current = current->next;
    }
}
void updateDataAt(int index, string nama, int harga)
{
    if (index < 0 || index >= size)
    {
        cout << "Index out of bounds" << endl;
        return;
    }
    Node *current = head;
    for (int i = 0; i < index; i++)
    {
        current = current->next;
    }
    current->nama = nama;
    current->harga = harga;
}
};
int main()
{
    DoubleLinkedList dll;
    int choice;
    string nama;
    int harga;
    int index;
    do
    {
        cout << "Toko Skincare Purwokerto" << endl;
        cout << "1. Tambah Data" << endl;
        cout << "2. Hapus Data" << endl;
        cout << "3. Update Data" << endl;
        cout << "4. Tambah Data pada Urutan Tertentu" << endl;
        cout << "5. Hapus Data pada Urutan Tertentu" << endl;
        cout << "6. Hapus Semua Data" << endl;
        cout << "7. Tampilkan Data" << endl;
        cout << "8. Keluar" << endl;
        cout << "Pilih: ";
        cin >> choice;
        switch (choice)
        {
            case 1:
                cout << "Nama Produk: ";
                cin >> nama;
                cout << "Harga: ";
                cin >> harga;
                dll.addData(nama, harga);
                break;
            case 2:

```

```
        cout << "Index: ";
        cin >> index;
        dll.deleteDataAt(index);
        break;
    case 3:
        cout << "Index: ";
        cin >> index;
        cout << "Nama Produk: ";
        cin >> nama;
        cout << "Harga: ";
        cin >> harga;
        dll.updateDataAt(index, nama, harga);
        break;
    case 4:
        cout << "Index: ";
        cin >> index;
        cout << "Nama Produk: ";
        cin >> nama;
        cout << "Harga: ";
        cin >> harga;
        dll.addDataAt(index, nama, harga);
        break;
    case 5:
        cout << "Index: ";
        cin >> index;
        dll.deleteDataAt(index);
        break;
    case 6:
        dll.clearData();
        break;
    case 7:
        dll.displayData();
        break;
    case 8:
        break;
    default:
        cout << "Pilihan tidak valid" << endl;
        break;
    }
    cout << endl;
} while (choice != 8);
return 0;
}
```

OUTPUT:

Case 1

Toko Skincare Purwokerto

1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 4
Index: 2
Nama Produk: Azrine
Harga: 65000

Toko Skincare Purwokerto

1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 7

Nama Produk	Harga
originote	60000
Somethinc	150000
Azrine	65000
Skintific	100000
Wardah	50000
Hanasui	30000

Case2

Toko Skincare Purwokerto

1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 5
Index: 4

Toko Skincare Purwokerto

1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 7

Nama Produk	Harga
originote	60000
Somethinc	150000
Azrine	65000
Skintific	100000
Hanasui	30000

Case3

Toko Skincare Purwokerto

1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 3
Index: 4
Nama Produk: Cleora
Harga: 55000

Toko Skincare Purwokerto

1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 7

Nama Produk	Harga
originote	60000
Somethinc	150000
Azrine	65000
Skintific	100000
Cleora	55000

Case4 & Tampil awal

Toko Skincare Purwokerto

1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 7

Nama Produk	Harga
originote	60000
Somethinc	150000
Skintific	100000
Wardah	50000
Hanasui	30000

DESKRIPSI PROGRAM: Program ini dibuat untuk sebuah toko skincare dengan menggunakan operasi dasar untuk mengelola Double Linked List, seperti penambahan data, penghapusan data, pembaruan data, dan penampilan data. Setiap kali fungsi-fungsi ini dipanggil, dikarenakan adanya loop program akan menampilkan menu dan menunggu masukan dari pengguna. Kemudian, program akan mengeksekusi tindakan yang dipilih berdasarkan masukan pengguna.

KESIMPULAN

- 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.
- 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.