LAPORAN PRAKTIKUM

MODUL 3
SINGLE AND DOUBLE LINKED LIST

Disusun oleh:
Nofita Fitriyani
NIM: 2311102001

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

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

BAB I

TUJUAN PRAKTIKUM

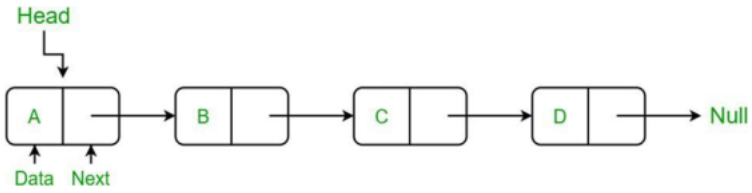
1. Mahasiswa memahami perbedaan konsep Single dan Double Linked List
2. Mahasiswa mampu menerapkan Single dan Double Linked List ke dalam pemrograman

BAB II

DASAR TEORI

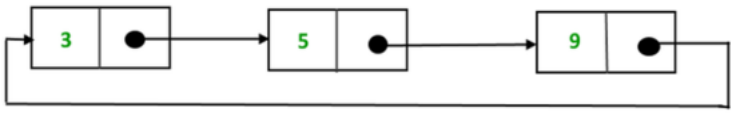
1. Single Linked List

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.

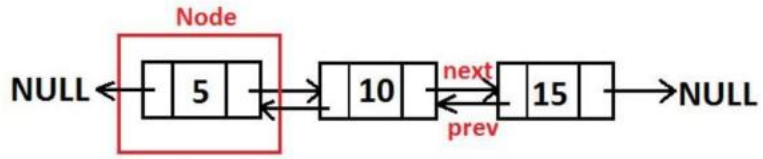


2. Double Linked List

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.

BAB III

GUIDED

1. Guided 1

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->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;
    }
```

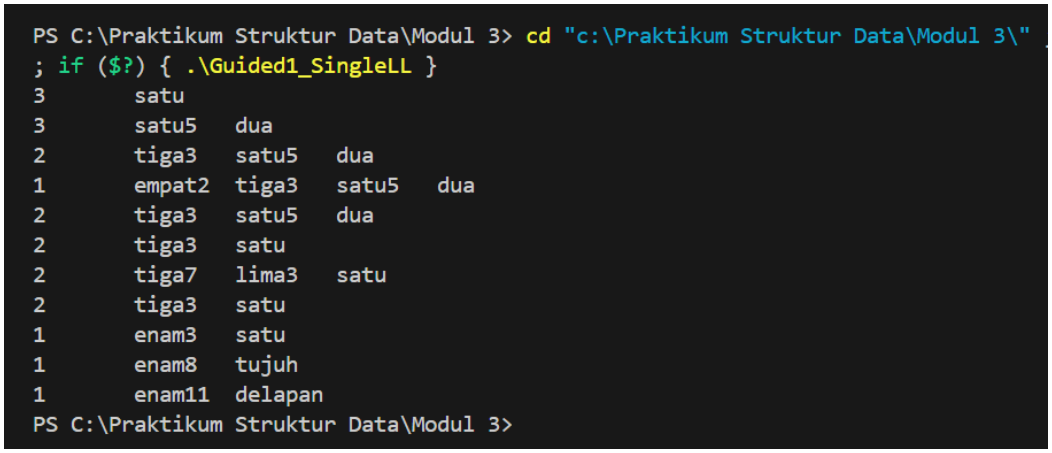
```
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
// Ubah Tengah
void ubahTengah(int data, string kata, int posisi)
{
    Node *bantu;
    if (isEmpty() == false)
    {
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "Posisi di luar jangkauan" <<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



Deskripsi program

Dalam program ini, setiap data disimpan dalam bentuk node yang terhubung satu sama lain dengan menggunakan pointer. Setiap node memiliki dua bagian, yaitu data yang bisa berupa angka atau kata, dan pointer yang menunjuk ke node berikutnya dalam list.

Program dimulai dengan membuat struct Node yang memiliki beberapa bagian seperti data untuk nilai integer dan kata untuk nilai string, serta pointer next yang menunjuk ke node berikutnya. Ada juga variabel global head dan tail yang digunakan untuk menunjukkan awal (head) dan akhir (tail) dari list.

Ada beberapa fungsi utama dalam program ini. Fungsi init() digunakan untuk menginisialisasi list, isEmpty() digunakan untuk mengecek apakah list kosong, insertDepan() dan insertBelakang() digunakan untuk menambahkan node di awal

dan akhir list, hitungList() digunakan untuk menghitung jumlah node dalam list, dan masih banyak fungsi lainnya seperti hapusDepan(), hapusBelakang(), hapusTengah(), ubahDepan(), ubahTengah(), ubahBelakang(), clearList(), dan tampil() untuk operasi-operasi lainnya seperti menghapus, mengubah nilai, membersihkan list, dan menampilkan isi list.

Dalam fungsi main(), program melakukan inisialisasi list, kemudian melakukan operasi-operasi seperti menambahkan node, menghapus node, mengubah nilai node, dan menampilkan isi list.

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;
```

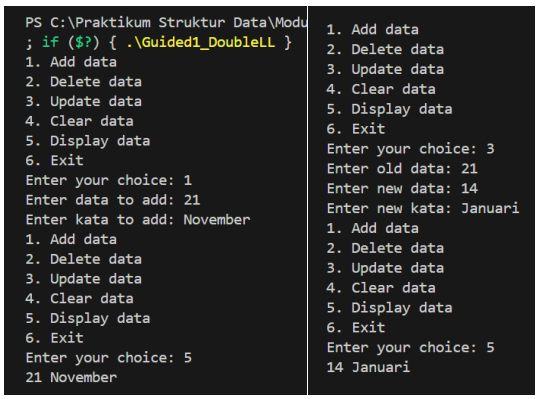
```
    newNode->data = data;
    newNode->kata = kata;
    newNode->prev = nullptr;
    newNode->next = head;
    if (head != nullptr) {
        head->prev = newNode;
    } else {
        tail = newNode;
    }
    head = newNode;
}

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;
```

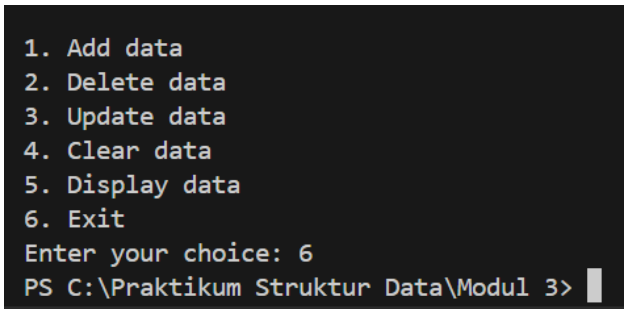
```
        }
        return false;
    }

    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;
}
```

Screenshoot program





Deskripsi program

Dalam program ini, terdapat dua kelas utama yaitu Node dan DoublyLinkedList. Kelas Node memiliki atribut data, kata, prev, dan next untuk merepresentasikan node dalam DLL. Kelas DoublyLinkedList memiliki atribut head dan tail yang merupakan pointer ke node pertama (head) dan terakhir (tail) dalam DLL. Constructor DoublyLinkedList digunakan untuk menginisialisasi head dan tail dengan nilai null.

Program ini memiliki beberapa fungsi utama. Fungsi push() digunakan untuk menambahkan node baru di awal DLL. Fungsi pop() digunakan untuk menghapus node pertama dalam DLL. Fungsi update() digunakan untuk mengubah nilai data dan kata pada node yang sudah ada dalam DLL. Fungsi deleteAll() digunakan untuk menghapus semua node dalam DLL. Fungsi display() digunakan untuk menampilkan isi DLL.

Dalam fungsi main(), program meminta pengguna untuk memilih operasi yang ingin dilakukan seperti menambahkan data baru, menghapus data, mengubah data, menghapus semua data, menampilkan data, atau keluar dari program. Program ini memberikan kemudahan dalam manajemen data dengan menggunakan Doubly Linked List sehingga pengguna dapat dengan mudah menambah, menghapus, mengubah, dan menampilkan data sesuai kebutuhan.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

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;

// NOFITA FITRIYANI_2311102001_IF11A
class Node
{
public:
    string name;
    int age;
    Node *next;
};
class LinkedList
```

```
{
public:
    Node *head;
    LinkedList()
    {
        head = NULL;
    }
    void insertAtFront(string name, int age)
    {
        Node *newNode = new Node();
        newNode->name = name;
        newNode->age = age;
        newNode->next = head;
        head = newNode;
    }
    void insertAtEnd(string name, int age)
    {
        Node *newNode = new Node();
        newNode->name = name;
        newNode->age = age;
        newNode->next = NULL;
        if (head == NULL)
        {
            head = newNode;
            return;
        }
        Node *temp = head;
        while (temp->next != NULL)
        {
            temp = temp->next;
        }
        temp->next = newNode;
    }
    void insertAfter(string name, int age, string keyName)
```

```
    {
        Node *temp = head;
        while (temp != NULL)
        {
            if (temp->name == keyName)
            {
                Node *newNode = new Node();
                newNode->name = name;
                newNode->age = age;
                newNode->next = temp->next;
                temp->next = newNode;
                return;
            }
            temp = temp->next;
        }
        cout << keyName << " not found in the list." << endl;
    }
    void updateNode(string name, int age)
    {
        Node *temp = head;
        while (temp != NULL)
        {
            if (temp->name == name)
            {
                temp->age = age;
                return;
            }
            temp = temp->next;
        }
        cout << name << " not found in the list." << endl;
    }
    void deleteNode(string name)
    {
        Node *temp = head;
```

```
        Node *prev = NULL;
        while (temp != NULL)
        {
            if (temp->name == name)
            {
                if (prev == NULL)
                {
                    head = temp->next;
                }
                else
                {
                    prev->next = temp->next;
                }
                delete temp;
                return;
            }
            prev = temp;
            temp = temp->next;
        }
        cout << name << " not found in the list." << endl;
    }
    void clearAll()
    {
        Node *temp = head;
        while (temp != NULL)
        {
            Node *next = temp->next;
            delete temp;
            temp = next;
        }
        head = NULL;
    }
    // Display all nodes
    void display()
```

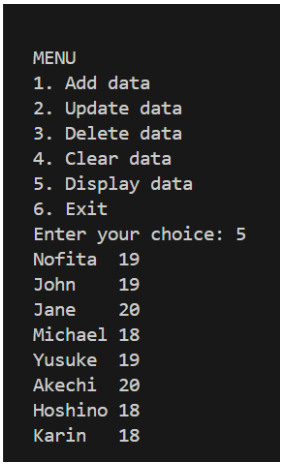
```
    {
        Node *temp = head;
        while (temp != NULL)
        {
            cout << temp->name << "\t" << temp->age << endl;
            temp = temp->next;
        }
    }
};
// Main function
int main()
{
    LinkedList list;
    int choice;
    string name, keyName;
    int age;
    do
    {
        cout << endl;
        cout << "MENU" << endl;
        cout << "1. Add data" << endl;
        cout << "2. Update data" << endl;
        cout << "3. Delete data" << endl;
        cout << "4. Clear data" << endl;
        cout << "5. Display data" << endl;
        cout << "6. Tambah Data diawal" << endl;
        cout << "7. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice)
        {
        case 1:
            for (int i = 0; i < 8; ++i)
            {
```

```
                cout << "Enter name " << i + 1 << ": ";
                cin >> name;
                cout << "Enter age " << i + 1 << ": ";
                cin >> age;
                list.insertAtEnd(name, age);
            }
            cout << "Data sudah ditambahkan" << endl;
            break;
        case 2:
            cout << "Enter name to update: ";
            cin >> name;
            if (name == "Michael")
            {
                cout << "Enter new name: ";
                cin >> name;
                cout << "Enter new age: ";
                cin >> age;
                list.updateNode(name, age);
            }
            else
            {
                cout << "Name not found in the list." << endl;
            }
            break;
        case 3:
            cout << "Enter name to delete: ";
            cin >> name;
            list.deleteNode(name);
            break;
        case 4:
            list.clearAll();
            break;
        case 5:
            list.display();
```

```
            break;
        case 6:
            cout << "Enter additional data to add at the
beginning:" << endl;
            cout << "Enter name: ";
            cin >> name;
            cout << "Enter age: ";
            cin >> age;
            list.insertAtFront(name, age);
            break;
        case 7:
            cout << "Exiting program..." << endl;
            break;
        default:
            cout << "Invalid choice." << endl;
        }
    } while (choice != 7);
    return 0;
}
```

Screenshoot program

a. Masukkan data sesuai urutan berikut. (Gunakan insert depan, belakang atau tengah). Data pertama yang dimasukkan adalah nama dan usia anda.



b. Hapus data Akechi

```
MENU
1. Add data
2. Update data
3. Delete data
4. Clear data
5. Display data
6. Exit
Enter your choice: 3
Enter name to delete: Akechi

MENU
1. Add data
2. Update data
3. Delete data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
Nofita  19
John    19
Jane    20
Michael 18
Yusuke  19
Hoshino 18
Karin   18
```

c. Tambahkan data berikut diantara John dan Jane : Futaba 18

```
MENU
1. Add data
2. Update data
3. Delete data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
Nofita  19
John    19
Futaba  18
Jane    20
Michael 18
Yusuke  19
Hoshino 18
Karin   18
```

d. Tambahkan data berikut diawal : Igor 20

```
MENU
1. Add data
2. Update data
3. Delete data
4. Clear data
5. Display data
6. Tambah Data diawal
7. Exit
Enter your choice: 6
Enter additional data to add at the beginning:
Enter name: Igor
Enter age: 20

MENU
1. Add data
2. Update data
3. Delete data
4. Clear data
5. Display data
6. Tambah Data diawal
7. Exit
Enter your choice: 5
Igor    20
Nofita  19
John    19
Futaba  18
Jane    20
Michael 18
Yusuke  19
Hoshino 18
Karin   18
```

e. Ubah data Michael menjadi : Reyn 18

```
MENU
1. Add data
2. Update data
3. Delete data
4. Clear data
5. Display data
6. Tambah Data diawal
7. Exit
Enter your choice: 2
Enter name to update: Michael
Enter new name: Reyn
Enter new age: 18
Reyn not found in the list.

MENU
1. Add data
2. Update data
3. Delete data
4. Clear data
5. Display data
6. Tambah Data diawal
7. Exit
Enter your choice: 5
Igor    20
Nofita  19
John    19
Futaba  18
Jane    20
Michael 18
Yusuke  19
Hoshino 18
karin   18
```

f. Tampilkan seluruh data

```
MENU
1. Add data
2. Update data
3. Delete data
4. Clear data
5. Display data
6. Tambah Data diawal
7. Exit
Enter your choice: 5
Igor    20
Nofita  19
John    19
Futaba  18
Jane    20
Michael 18
Yusuke  19
Hoshino 18
karin   18
```

Deskripsi program

Program tersebut merupakan program sederhana yang menggunakan struktur data linked list untuk mengelola data yang terdiri dari nama dan usia. Program menyediakan beberapa menu untuk pengguna, seperti menambah data, mengupdate data, menghapus data, membersihkan data, dan menampilkan data. Setiap menu memiliki fungsi yang sesuai dengan namanya, misalnya menambah data akan menanyakan nama dan usia untuk ditambahkan ke linked list, mengupdate data akan meminta nama baru dan usia baru untuk mengupdate data yang sesuai, dan sebagainya.

2. Unguided 2

Source code

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

//NOFITAFITRIYANI_2311102001_IF11A
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;
    }
};
```

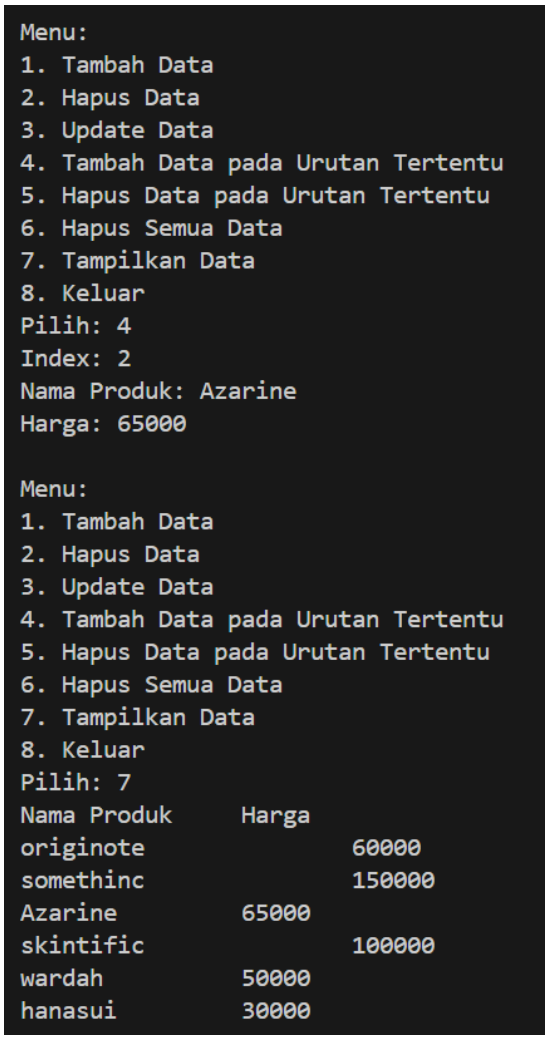
```
int main()
{
    DoubleLinkedList dll;
    int choice;
    string nama;
    int harga;
    int index;
    do
    {
        cout << "Menu:" << endl;
        cout << "1. Tambah Data" << endl;
        cout << "2. Hapus Data" << endl;
        cout << "3. Update Data" << endl;
        cout << "4. Tambah 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:
             for (int i = 0; i < 5; ++i)
           { cout << "Nama Produk " << i + 1 << ": ";
            cin >> nama;
            cout << "Harga " << i + 1 << ": ";
            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;
}
```

Screenshot program

Tambahkan produk Azarine dengan harga 65000 diantara Somethinc dan Skintific



Hapus produk wardah

```
Menu:
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

Menu:
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
Azarine         65000
skintific               100000
hanasui         30000
```

Update produk Hanasui menjadi Cleora dengan harga 55.000

```
Menu:
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

Menu:
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
Azarine         65000
skintific               100000
cleora          55000
```

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 :

```
Menu:
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
Azarine         65000
skintific               100000
cleora          55000
```

Deskripsi program

Pada program ini, terdapat kelas DoubleLinkedList yang memiliki beberapa fungsi utama seperti addData untuk menambah data pada akhir list, addDataAt untuk menambah data pada posisi tertentu, deleteDataAt untuk menghapus data pada posisi tertentu, clearData untuk menghapus semua data, dan displayData untuk menampilkan semua data yang telah dimasukkan.

Selain itu, program ini juga memiliki menu interaktif yang memungkinkan pengguna untuk memilih operasi yang ingin dilakukan, seperti menambah data, menghapus data, mengupdate data, menampilkan data, dan sebagainya. Pengguna dapat memasukkan nama produk dan harga sesuai dengan permintaan program.

BAB IV

KESIMPULAN

Dibandingkan dengan single linked list yang hanya memiliki satu pointer untuk menghubungkan node secara berurutan, double linked list memiliki dua pointer yang menghubungkan node secara berurutan dan dua arah. Meskipun fitur tambahan ini memungkinkan lebih fleksibilitas saat menjalankan operasi seperti menambah atau menghapus node dengan mudah, ini juga memerlukan memori tambahan untuk menyimpan kedua pointer tersebut.