

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA
MODUL 3
“SINGLE AND DOUBLE LINKED LIST”**



**DISUSUN OLEH:
TSAQIF KANZ AHMAD
2311102075
IF-11-B**

DOSEN:

WAHYU ANDI SAPUTRA S.Pd., M.Eng.

**PROGRAM STUDI S1 TEKNIK
INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2024**

TUJUAN PRAKTIKUM

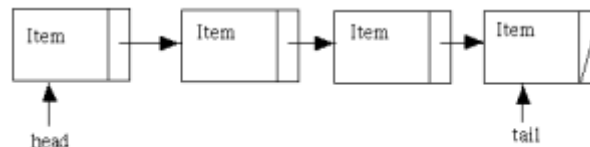
1. Memahami perbedaan konsep Single dan Double Linked List.
2. Mampu menerapkan Single dan Double Linked List ke dalam pemrograman.

DASAR TEORI

Linked list merupakan struktur data untuk menyimpan data dengan struktur sehingga programmer dapat secara otomatis menciptakan suatu tempat baru untuk menyimpan data kapan saja diperlukan. Linked list dikenal juga dengan sebutan senarai berantai adalah struktur data yang terdiri dari urutan record data dimana setiap record memiliki field yang menyimpan alamat/referensi dari record selanjutnya (dalam urutan). Elemen data yang dihubungkan dengan link pada linked list disebut Node. Dalam suatu link list, terdapat istilah head dan tail yang merupakan urutan pada suatu elemen yang terletak dari awal(Head) hingga akhir(Tail) dalam suatu linked list. Pada jenis Linked list yang akan dipelajari :

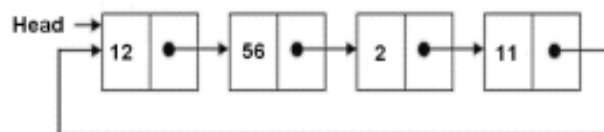
1). SINGLE LINKED LIST

Single linked list merupakan suatu linked list yang hanya memiliki satu variabel pointer saja. Dimana pointer tersebut menunjuk ke node selanjutnya. Biasanya field pada tail menunjuk ke NULL.



Gambar . Ilustrasi Single Linked List

Dalam operasi Single Linked list umumnya menggunakan operasi penambahan dan penghapusan simpul pada awal atau akhir daftar. Karena struktur data ini hanya memerlukan satu pointer untuk setiap simpul, maka Single Linked List akan lebih efisien dalam penggunaan memori dibandingkan dengan jenis Linked List lainnya. Single linked list kedua adalah Circular Single Linked list yaitu suatu linked list dimana tail (node akhir) menunjuk ke Head (node pertama). Jadi tidak ada pointer yang menunjuk NULL. Perbedaan keduanya terletak pada penunjuk next node terakhir pada circular linked list akan selalu ke node pertama.

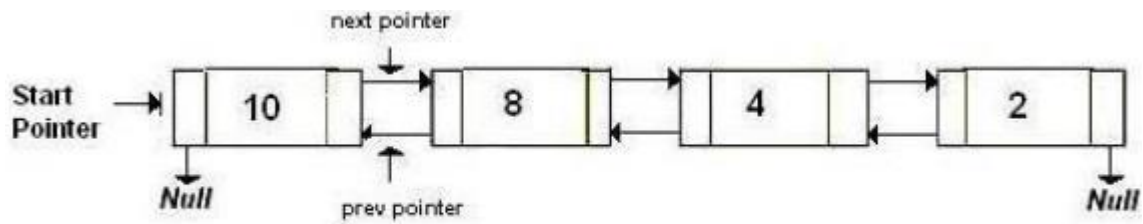


Gambar . Ilustrasi Circular Single Linked List

2). DOUBLE LINKED LIST

Double linked list merupakan suatu linked list yang memiliki dua variabel pointer yaitu pointer yang menunjuk ke node selanjutnya dan pointer yang menunjuk ke node sebelumnya. Double linked list memiliki hampir kemiripan dengan Single Linked List namun pada Double Linked List terdapat tambahan satu pointer pada setiap simpul yaitu pointer prev. Dengan adanya pointer prev, Double Linked List dapat 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. Untuk Double Linked List kedua yaitu bersifat Circular yang memiliki kemiripan namun perbedaannya terletak pada simpul terakhirnya yang menunjuk pada

simpul kepala. Jenis ini disebut bidirectional. Berikut representasi pada sebuah linked list dapat dilihat pada gambar berikut :



Gambar . Ilustrasi Double Linked List

Kelebihan pada Double Linked List terletak pada bagian pencarian data dan penghapusan data yang lebih cepat dibandingkan Single Linked List karena dapat dilakukan dari dua arah sehingga memudahkan dalam mengimplementasikan operasi beberapa algoritma. Selain itu, Double Linked List dapat digunakan untuk melakukan traversal pada list baik dari depan (head) maupun dari belakang (tail). Namun kekurangan pada Double Linked List yaitu membutuhkan memori dua kali lebih banyak dibandingkan dengan Single Linked List karena setiap node memiliki dua pointer selain itu juga membutuhkan waktu eksekusi yang lebih lama dalam operasi penambahan dan penghapusan jika dibandingkan dengan Single Linked List.

Pada suatu Linked List terdapat 2 pointer utama yaitu pointer Head yang menunjuk pada node pertama pada Linked List itu sendiri dan pointer Tail yang menunjuk pada node terakhir pada linked list. Apabila sebuah Linked List dikatakan kosong jika nilai pointer Head adalah NULL, karena ini adalah pertama. Demikian pula nilai pointer berikutnya dari Tail selalu NULL sebagai indikator data terakhir.

PENJELASAN GUIDED

1). Program Single Linked List.

SOURCE CODE

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

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

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

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

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

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

```

    }
    else{
        tail->next = baru;
        tail = baru;
    }
}

//Hitung Jumlah List
int hitungList(){
    Node *hitung;
    hitung = head;
    int jumlah = 0;
    while( hitung != NULL ){
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

//Tambah Tengah
void insertTengah(int data, int posisi){
    if( posisi < 1 || posisi > hitungList() ){
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if( posisi == 1){
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else{
        Node *baru, *bantu;
        baru = new Node();
        baru->data = data;
        // tranversing
        bantu = head;
        int nomor = 1;
        while( nomor < posisi - 1 ){
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

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

```

```

    }
    else{
        cout << "List kosong!" << endl;
    }
}

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

//Hapus Tengah
void hapusTengah(int posisi){
    Node *hapus, *bantu, *bantu2;
    if( posisi < 1 || posisi > hitungList() ){
        cout << "Posisi di luar jangkauan" << endl;
    }
    else if( posisi == 1){
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else{
        int nomor = 1;
        bantu = head;
        while( nomor <= posisi ){
            if( nomor == posisi-1 ){
                bantu2 = bantu;
            }
            if( nomor == posisi ){
                hapus = bantu;
            }
            bantu = bantu->next;
            nomor++;
        }
        bantu2->next = bantu;
        delete hapus;
    }
}

```

```

    }
}
//Ubah Depan
void ubahDepan(int data){
    if (isEmpty() == false){
        head->data = data;
    }
    else{
        cout << "List masih kosong!" << endl;
    }
}
//Ubah Tengah
void ubahTengah(int data, int posisi){
    Node *bantu;
    if (isEmpty() == false){
        if( posisi < 1 || posisi > hitungList() ){
            cout << "Posisi di luar jangkauan" << endl;
        }
        else if( posisi == 1){
            cout << "Posisi bukan posisi tengah" << endl;
        }
        else{
            bantu = head;
            int nomor = 1;
            while (nomor < posisi){
                bantu = bantu->next; nomor++;
            }
            bantu->data = data;
        }
    }
    else{
        cout << "List masih kosong!" << endl;
    }
}
//Ubah Belakang
void ubahBelakang(int data){
    if (isEmpty() == false){
        tail->data = data;
    }
    else{
        cout << "List masih kosong!" << endl;
    }
}
//Hapus List
void clearList(){
    Node *bantu, *hapus;
    bantu = head;
    while (bantu != NULL){
        hapus = bantu; bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
}

```



```

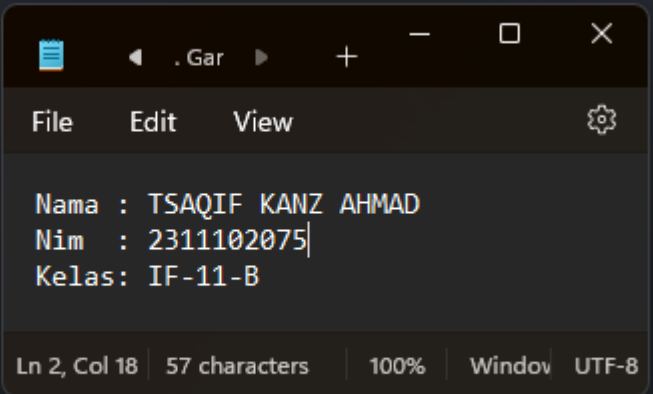
        cout << "List berhasil terhapus!" << endl;
    }
    //Tampilkan List
    void tampil(){
        Node *bantu;
        bantu = head;
        if (isEmpty() == false){
            while (bantu != NULL){
                cout << bantu->data << ends;
                bantu = bantu->next;
            }
            cout << endl;
        }
        else{
            cout << "List masih kosong!" << endl;
        }
    }
}

int main(){
    init();
    insertDepan(3);tampil();
    insertBelakang(5);
    tampil();
    insertDepan(2);
    tampil();
    insertDepan(1);
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7,2);
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1);
    tampil();
    ubahBelakang(8);
    tampil();
    ubahTengah(11, 2);
    tampil();
    return 0;
}

```

SCREENSHOT OUTPUT

```
PS C:\Users\ACER> cd "C:\Users\ACER\AppData\Local\Temp\" ; if ($?) {
mpCodeRunnerFile }
3
35
235
1235
235
23
273
23
13
18
111
PS C:\Users\ACER\AppData\Local\Temp>
```



DESKRIPSI PROGRAM

Program diatas adalah program implementasi Single Linked List Non-Circular yang memiliki beberapa fungsi. Pada deklarasi dan inisialisasi terdapat beberapa tipe data yaitu 'struct node' yang berperan menyimpan data dan pointer ke node selanjutnya, Pada tipe data head dan tail terdapat fungsi yang berperan sebagai pointer pada node pertama hingga node terakhir. Pada tipe data 'init()' berperan mengatur head dan tail ke NULL untuk menandakan list kosong. Pada bagian 'isEmpty()' merupakan bagian pengecekan yang berperan mengembalikan true jika list kosong, jika false maka sebaliknya. Terdapat manipulasi data yang terdiri dari 3 bagian untuk menambah, menghapus dan mengubah data pada setiap node dari posisi list tersebut. Pada operasi lainnya terdapat 'hitungList' dan 'clearList' yang berfungsi menghitung jumlah node dan menghapus node dalam list tersebut. Dan terdapat tipe data 'tampil()' untuk menampilkan data dari semua node dalam list tersebut.

2). Program Doubel Linked List.
SOURCE CODE

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

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

class DoublyLinkedList {
public:
    Node* head;
    Node* tail;
    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }
    void push(int data) {
        Node* newNode = new Node;
        newNode->data = data;
        newNode->prev = nullptr;
        newNode->next = head;
        if (head != nullptr) {
            head->prev = newNode;
        }
        else {
            tail = newNode;
        }
        head = newNode;
    }
    void pop() {
        if (head == nullptr) {
            return;
        }
        Node* temp = head;
        head = head->next;
        if (head != nullptr) {
            head->prev = nullptr;
        }
        else {
            tail = nullptr;
        }
        delete temp;
    }
    bool update(int oldData, int newData) {
        Node* current = head;
        while (current != nullptr)
        {
            if (current->data == oldData) {
                current->data = newData;
                return true;
            }
        }
    }
}
```

```

        current = current->next;
    }
    return false;
}

void deleteAll() {
    Node* current = head;
    while (current != nullptr) {
        Node* temp = current;
        current = current->next;
        delete temp;
    }
    head = nullptr;
    tail = nullptr;
}

void display() {
    Node* current = head;
    while (current != nullptr) {
        cout << current->data << " ";
        current = current->next;
    }
    cout << endl;
}

};

int main() {
    DoublyLinkedList list;
    while (true) {
        cout << "1. Add data" << endl;
        cout << "2. Delete data" << endl;
        cout << "3. Update data" << endl;
        cout << "4. Clear data" << endl;
        cout << "5. Display data" << endl;
        cout << "6. Exit" << endl;
        int choice;
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1: {
                int data;
                cout << "Enter data to add: ";
                cin >> data;
                list.push(data);
                break;
            }
            case 2: {
                list.pop();
                break;
            }
            case 3: {
                int oldData, newData;
                cout << "Enter old data: ";
                cin >> oldData;
                cout << "Enter new data: ";
                cin >> newData;
            }
        }
    }
}

```

```

        bool updated = list.update(oldData, newData);
        if (!updated) {
            cout << "Data not found" << endl;
        }
        break;
    }
    case 4: {
        list.deleteAll();
        break;
    }
    case 5: {
        list.display();
        break;
    }
    case 6: {
        return 0;
    }
    default: {
        cout << "Invalid choice" << endl;
        break;
    }
}

return 0;
}

```

SCREENSHOT OUTPUT :

```

mpCodeRunnerFile }
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: 3
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 4
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 2
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 

```

DESKRIPSI PROGRAM

Program diatas adalah program implementasi Doubel Linked List yang menggunakan kelas node untuk mempresentasikan setiap elemen dalam Doubel Linked List. Setiap Node memiliki tiga atribut data untuk menyimpan nilai data, pada nilai 'prev' untuk menunjukkan ke node sebelumnya, dan next untuk menunjukkan ke node berikutnya. Kelas Doubel Linked List memiliki dua atribut yaitu head untuk menunjukkan ke node pertama dalam linked list, dan tail untuk menunjukkan ke node terakhir. Ada beberapa metode untuk melakukan operasi pada Doubel Linked List, seperti 'push' untuk menambahkan data di awal, 'pop' untuk menghapus data di awal, 'update' untuk memperbarui data yang ada, 'deleteAll' untuk menghapus semua data, dan 'display' untuk menampilkan semua data dalam linked list. Pada fungsi main menggunakan Loop while untuk membuat menu interaktif yang memungkinkan user memilih operasi yang ingin dilakukan pada Doubel Linked List. User dapat memilih untuk menambahkan data, menghapus data, memperbarui data, menghapus semua data, menampilkan data atau keluar dari program

PENJELASAN 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. masukan data sesuai urutan berikut. (Gunakan insert depan, belakang atau tengah). Data pertama yang dimasukan 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 antara john dan jane : Futaba 18

d. Tambahkan data berikut diawal : Igor 20

e. Ubah data Michael menjadi : Reyn 18

f. Tampilkan seluruh data

SOURCE CODE

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

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

Node *head;
Node *tail;

void init()
{
    head = NULL;
    tail = NULL;
}

bool isEmpty()
{
    return head == NULL;
}

void insertDepan(string nama, int usia)
{
    Node *baru = new Node;
    baru->nama = nama;
    baru->usia = usia;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}

void insertBelakang(string nama, int usia)
{
    Node *baru = new Node;
    baru->nama = nama;
    baru->usia = usia;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
}
```



```

        else
        {
            tail->next = baru;
            tail = baru;
        }
    }

int hitungList()
{
    Node *hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

void insertTengah(string nama, int usia, int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        Node *baru = new Node();
        baru->nama = nama;
        baru->usia = usia;
        Node *bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

void hapusDepan()
{
    if (!isEmpty())
    {
        Node *hapus = head;
        if (head->next != NULL)
    }

```

```

        {
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
            delete hapus;
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}

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

void hapusTengah(int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
}

```

```

    }
    else
    {
        Node *hapus;
        Node *bantu = head;
        for (int nomor = 1; nomor < posisi - 1; nomor++)
        {
            bantu = bantu->next;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    }
}

void ubahDepan(string nama, int usia)
{
    if (!isEmpty())
    {
        head->nama = nama;
        head->usia = usia;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

void ubahTengah(string nama, int usia, int posisi)
{
    if (!isEmpty())
    {
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "Posisi di luar jangkauan" << endl;
        }
        else if (posisi == 1)
        {
            cout << "Posisi bukan posisi tengah" << endl;
        }
        else
        {
            Node *bantu = head;
            for (int nomor = 1; nomor < posisi; nomor++)
            {
                bantu = bantu->next;
            }
            bantu->nama = nama;
            bantu->usia = usia;
        }
    }
    else

```

```

    {
        cout << "List masih kosong!" << endl;
    }
}

void ubahBelakang(string nama, int usia)
{
    if (!isEmpty())
    {
        tail->nama = nama;
        tail->usia = usia;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

void clearList()
{
    Node *bantu = head;
    while (bantu != NULL)
    {
        Node *hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}

void tampil()
{
    if (!isEmpty())
    {
        Node *bantu = head;
        while (bantu != NULL)
        {
            cout << bantu->nama << " ";
            cout << bantu->usia << " , ";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

int main()
{

```

```

init();
int menu, usia, posisi;
string nama;
cout << "\n# Menu Linked List Mahasiswa #" << endl;
do
{
    cout << "\n 1. Insert Depan"
        << "\n 2. Insert Belakang"
        << "\n 3. Insert Tengah"
        << "\n 4. Hapus Depan"
        << "\n 5. Hapus Belakang"
        << "\n 6. Hapus Tengah"
        << "\n 7. Ubah Depan"
        << "\n 8. Ubah Belakang"
        << "\n 9. Ubah Tengah"
        << "\n 10. Tampilkan"
        << "\n 0. Keluar Program"
        << "\n Pilihan : ";

    cin >> menu;
    switch (menu)
    {
        case 1:
            cout << "Masukkan Nama : ";
            cin >> nama;
            cout << "Masukkan Usia : ";
            cin >> usia;
            insertDepan(nama, usia);
            cout << endl;
            tampil();
            break;
        case 2:
            cout << "Masukkan Nama : ";
            cin >> nama;
            cout << "Masukkan Usia : ";
            cin >> usia;
            insertBelakang(nama, usia);
            cout << endl;
            tampil();
            break;
        case 3:
            cout << "Masukkan Posisi : ";
            cin >> posisi;
            cout << "Masukkan Nama : ";
            cin >> nama;
            cout << "Masukkan Usia : ";
            cin >> usia;
            insertTengah(nama, usia, posisi);
            cout << endl;
            tampil();
            break;
        case 4:
            hapusDepan();

```

```
        cout << endl;
        tampil();
        break;
    case 5:
        hapusBelakang();
        cout << endl;
        tampil();
        break;
    case 6:
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        hapusTengah(posisi);
        cout << endl;
        tampil();
        break;
    case 7:
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan Usia : ";
        cin >> usia;
        ubahDepan(nama, usia);
        cout << endl;
        tampil();
        break;
    case 8:
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan Usia : ";
        cin >> usia;
        ubahBelakang(nama, usia);
        cout << endl;
        tampil();
        break;
    case 9:
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan Usia : ";
        cin >> usia;
        ubahTengah(nama, usia, posisi);
        cout << endl;
        tampil();
        break;
    case 10:
        tampil();
        break;

    default:
        cout << "Pilihan Salah" << endl;
        break;
}
```

```

    } while (menu != 0);
    return 0;
}

```

SCREENSHOT OUTPUT

```

# Menu Linked List Mahasiswa #

1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 1
Masukkan Nama : Tsaqif
Masukkan Usia : 19

Tsaqif 19 ,

1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 2
Masukkan Nama : John
Masukkan Usia : 19

Tsaqif 19 , John 19 ,

1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 2
Masukkan Nama : Jane
Masukkan Usia : 20

Tsaqif 19 , John 19 , Jane 20 ,

1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 2
Masukkan Nama : Michael
Masukkan Usia : 18

Tsaqif 19 , John 19 , Jane 20 , Michael 18 ,

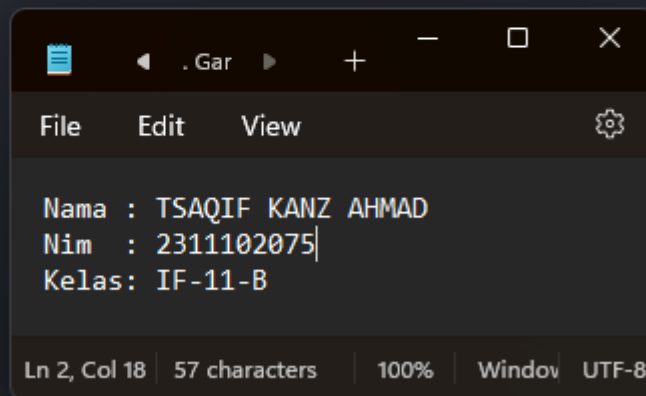
```

```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 2
Masukkan Nama : Yusuke
Masukkan Usia : 19
```

Tsaqif 19 , John 19 , Jane 20 , Michael 18 , Yusuke 19 ,

```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 2
Masukkan Nama : Akechi
Masukkan Usia : 20
```

Tsaqif 19 , John 19 , Jane 20 , Michael 18 , Yusuke 19 , Akechi 20 ,

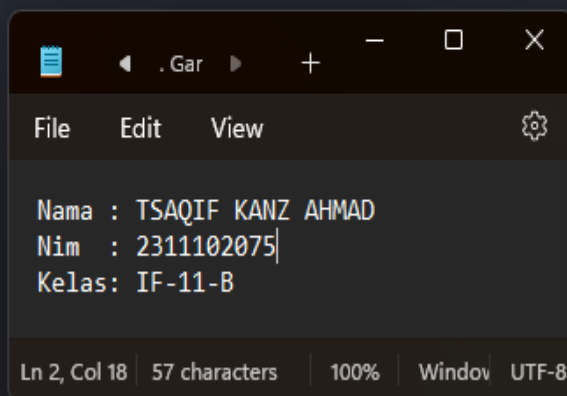


```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 2
Masukkan Nama : Hoshino
Masukkan Usia : 18
```

Tsaqif 19 , John 19 , Jane 20 , Michael 18 , Yusuke 19 , Akechi 20 , Hoshino 18 ,

```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 2
Masukkan Nama : Karin
Masukkan Usia : 18
```

Tsaqif 19 , John 19 , Jane 20 , Michael 18 , Yusuke 19 , Akechi 20 , Hoshino 18 , Karin 18 ,




```

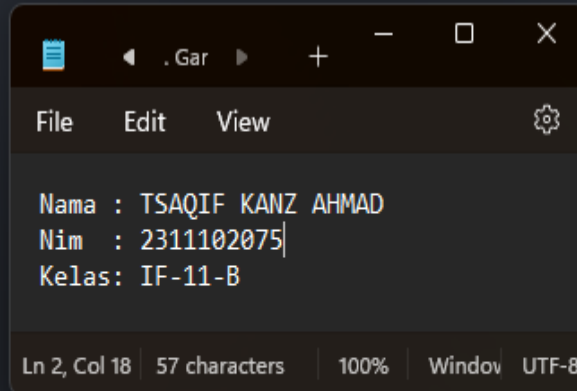
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 6
Masukkan Posisi : 6

Tsaqif 19 , John 19 , Jane 20 , Michael 18 , Yusuke 19 , Hoshino 18 , Karin 18 ,

1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 3
Masukkan Posisi : 3
Masukkan Nama : Futaba
Masukkan Usia : 18

Tsaqif 19 , John 19 , Futaba 18 , Jane 20 , Michael 18 , Yusuke 19 , Hoshino 18 , Karin 18 ,

```



```

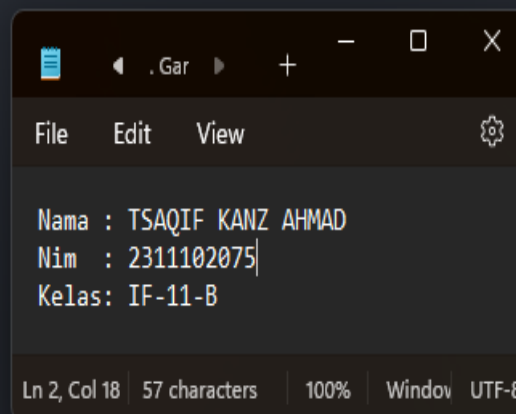
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
0. Keluar Program
Pilihan : 1
Masukkan Nama : Igor
Masukkan Usia : 20

Igor 20 , Tsaqif 19 , John 19 , Futaba 18 , Jane 20 , Michael 18 , Yusuke 19 , Hoshino 18 , Karin 18 ,

1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 9
Masukkan Posisi : 6
Masukkan Nama : Reyn
Masukkan Usia : 18

Igor 20 , Tsaqif 19 , John 19 , Futaba 18 , Jane 20 , Reyn 18 , Yusuke 19 , Hoshino 18 , Karin 18 ,

```



DESKRIPSI PROGRAM

Program diatas adalah program menu Single Linked List Non-Circular untuk menyimpan Nama dan usia mahasiswa, dengan menggunakan inputan dari user berupa node untuk menyimpan informasi mahasiswa (nama dan usia) dan pointer ke node selanjutnya dengan menggunakan variabel Head dan Tail dan menggunakan berbagai fungsi berupa inisialisasi 'init()' untuk mengatur Head dan Tail ke NULL dan memanipulasi data berupa menambah data, menghapus data serta mengubah data pada posisi node dari list. Dan ada beberapa operasi lainnya. Pada program kerja tersebut pengguna memilih menu yang diinginkan, program akan menjalankan fungsi yang sesuai dengan pilihan pengguna. Fungsi-fungsi tersebut akan memanipulasi data dalam Linked List berdasarkan input dari pengguna.

2. Soal mengenai Doubel Linked List

Modifikasi Guided Doubel Linked List dilakukan dengan penambahan operasi untuk menambah data, menghapus, dan update di tengah / di urutan tertentu yang diminta. Selain itu, buatlah agar tampilannya menampilkan Nama produk dan harga.

Nama Produk	Harga
Originote	60.000
Somethinc	150.000
Skintific	100.000
Wardah	50.000
Hanasui	30.000

Case :

1. Tambahkan produk Azarine dengan harga 65000 diantara Somethinc dan Skintific
2. Hapus produk wardah
3. Update produk Hanasui menjadi Cleora dengan harga 55.000
4. Tampilkan menu seperti dibawah ini

Toko Skincare Purwokerto

1. **Tambah Data**
2. **Hapus Data**
3. **Update Data**
4. **Tambah Data Urutan Tertentu**
5. **Hapus Data Urutan Tertentu**
6. **Hapus Seluruh Data**
7. **Tampilkan Data**
8. **Exit**

Pada menu 7, tampilan akhirnya akan menjadi seperti dibawah ini :

Nama Produk	Harga
Originote	60.000
Somethinc	150.000
Azarine	65.000
Skintific	100.000
Cleora	55.000

SOURCE CODE

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

class Node
{
public:
    string namaProduk;
    int harga;
    Node *prev;
    Node *next;
};

class DoublyLinkedList
{
public:
    Node *head;
    Node *tail;

    DoublyLinkedList()
    {
        head = nullptr;
        tail = nullptr;
    }

    void push(string namaProduk, int harga)
    {
        Node *newNode = new Node;
        newNode->namaProduk = namaProduk;
        newNode->harga = harga;
        newNode->prev = nullptr;
        newNode->next = head;

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

        head = newNode;
    }

    void pushCenter(string namaProduk, int harga, int posisi)
    {
        if (posisi < 0)
```

```

    {
        cout << "Posisi harus bernilai non-negatif." << endl;
        return;
    }

    Node *newNode = new Node;
    newNode->namaProduk = namaProduk;
    newNode->harga = harga;

    if (posisi == 0 || head == nullptr)
    {
        newNode->prev = nullptr;
        newNode->next = head;

        if (head != nullptr)
        {
            head->prev = newNode;
        }
        else
        {
            tail = newNode;
        }
        head = newNode;
    }
    else
    {
        Node *temp = head;
        int count = 0;
        while (temp != nullptr && count < posisi)
        {
            temp = temp->next;
            count++;
        }

        if (temp == nullptr)
        {
            newNode->prev = tail;
            newNode->next = nullptr;
            tail->next = newNode;
            tail = newNode;
        }
        else
        {
            newNode->prev = temp->prev;
            newNode->next = temp;
            temp->prev->next = newNode;
            temp->prev = newNode;
        }
    }
}

void pop()

```

```

{
    if (head == nullptr)
    {
        return;
    }
    Node *temp = head;
    head = head->next;

    if (head != nullptr)
    {
        head->prev = nullptr;
    }
    else
    {
        tail = nullptr;
    }

    delete temp;
}

void popCenter(int posisi)
{
    if (head == nullptr)
    {
        cout << "List kosong. Tidak ada yang bisa dihapus." << endl;
        return;
    }

    if (posisi < 0)
    {
        cout << "Posisi harus bernilai non-negatif." << endl;
        return;
    }

    if (posisi == 0)
    {
        Node *temp = head;
        head = head->next;

        if (head != nullptr)
        {
            head->prev = nullptr;
        }
        else
        {
            tail = nullptr;
        }

        delete temp;
    }
    else
    {

```

```

        Node *temp = head;
        int count = 0;
        while (temp != nullptr && count < posisi)
        {
            temp = temp->next;
            count++;
        }

        if (temp == nullptr)
        {
            cout << "Posisi melebihi ukuran list. Tidak ada yang
dihapus." << endl;
            return;
        }

        if (temp == tail)
        {
            tail = tail->prev;
            tail->next = nullptr;
            delete temp;
        }
        else
        {
            temp->prev->next = temp->next;
            temp->next->prev = temp->prev;
            delete temp;
        }
    }
}

bool update(string oldNamaProduk, string newNamaProduk, int newHarga)
{
    Node *current = head;

    while (current != nullptr)
    {
        if (current->namaProduk == oldNamaProduk)
        {
            current->namaProduk = newNamaProduk;
            current->harga = newHarga;
            return true;
        }
        current = current->next;
    }
    return false;
}

bool updateCenter(string newNamaProduk, int newHarga, int posisi)
{
    if (head == nullptr)
    {
        cout << "List kosong. Tidak ada yang dapat diperbarui." <<

```

```

endl;
        return false;
    }

    if (posisi < 0)
    {
        cout << "Posisi harus bernilai non-negatif." << endl;
        return false;
    }

    Node *current = head;
    int count = 0;

    while (current != nullptr && count < posisi)
    {
        current = current->next;
        count++;
    }

    if (current == nullptr)
    {
        cout << "Posisi melebihi ukuran list. Tidak ada yang
diperbarui." << endl;
        return false;
    }

    current->namaProduk = newNamaProduk;
    current->harga = newHarga;
    return true;
}

void deleteAll()
{
    Node *current = head;
    while (current != nullptr)
    {
        Node *temp = current;
        current = current->next;
        delete temp;
    }
    head = nullptr;
    tail = nullptr;
}

void display()
{
    if (head == nullptr)
    {
        cout << "List kosong." << endl;
        return;
    }

```



```

Node *current = head;

cout << setw(37) << setfill('-') << "-" << setfill(' ') << endl;
cout << " | " << setw(20) << left << "Nama Produk"
    << " | " << setw(10) << "Harga"
    << " | " << endl;
cout << setw(37) << setfill('-') << "-" << setfill(' ') << endl;

while (current != nullptr)
{
    cout << " | " << setw(20) << left << current->namaProduk << " | "
    << setw(10) << current->harga << " | " << endl;
    current = current->next;
}
cout << setw(37) << setfill('-') << "-" << setfill(' ') << endl;
}

};

int main()
{
    DoublyLinkedList list;
    int choice;
    cout << endl
        << "Toko Skincare Purwokerto" << endl;
    do
    {
        cout << "1. Tambah data" << endl;
        cout << "2. Hapus data" << endl;
        cout << "3. Update data" << endl;
        cout << "4. Tambah Data Urutan Tertentu" << endl;
        cout << "5. Hapus Data Urutan Tertentu" << endl;
        cout << "6. Hapus Seluruh Data" << endl;
        cout << "7. Tampilkan data" << endl;
        cout << "8. Exit" << endl;

        cout << "Pilihan : ";
        cin >> choice;

        switch (choice)
        {
            case 1:
            {
                string namaProduk;
                int harga;
                cout << "Masukkan nama produk: ";
                cin.ignore();
                getline(cin, namaProduk);
                cout << "Masukkan harga produk: ";
                cin >> harga;
                list.push(namaProduk, harga);
                break;
            }
        }
    }
}

```

```

case 2:
{
    list.pop();
    break;
}
case 3:
{
    string newNamaProduk;
    int newHarga, posisi;
    cout << "Masukkan posisi produk: ";
    cin >> posisi;
    cout << "Masukkan nama baru produk: ";
    cin >> newNamaProduk;
    cout << "Masukkan harga baru produk: ";
    cin >> newHarga;
    bool updatedCenter = list.updateCenter(newNamaProduk,
newHarga, posisi);
    if (!updatedCenter)
    {
        cout << "Data not found" << endl;
    }
    break;
}
case 4:
{
    string namaProduk;
    int harga, posisi;
    cout << "Masukkan posisi data produk: ";
    cin >> posisi;
    cout << "Masukkan nama produk: ";
    cin.ignore();
    getline(cin, namaProduk);
    cout << "Masukkan harga produk: ";
    cin >> harga;
    list.pushCenter(namaProduk, harga, posisi);
    break;
}
case 5:
{
    int posisi;
    cout << "Masukkan posisi data produk: ";
    cin >> posisi;
    list.popCenter(posisi);
    break;
}
case 6:
{
    list.deleteAll();
    break;
}
case 7:
{

```

```

        list.display();
        break;
    }
    case 8:
    {
        return 0;
    }
    default:
    {
        cout << "Invalid choice" << endl;
        break;
    }
}
} while (choice != 8);

return 0;
}

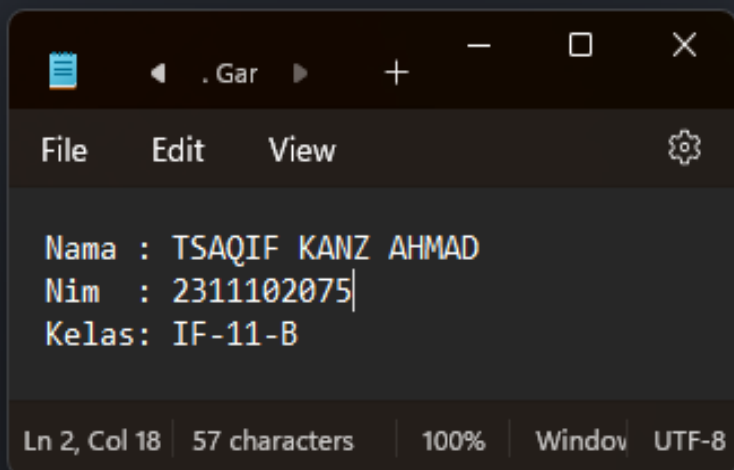
```

SCREENSHOT OUTPUT

```

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
Pilihan : 1
Masukkan nama produk: Hanasui
Masukkan harga produk: 30000
1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit
Pilihan : 1
Masukkan nama produk: Wardah
Masukkan harga produk: 50000
1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit
Pilihan : 1
Masukkan nama produk: Skintific
Masukkan harga produk: 100000

```



1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit

Pilihan : 1

Masukkan nama produk: Somethinc
Masukkan harga produk: 150000

1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit

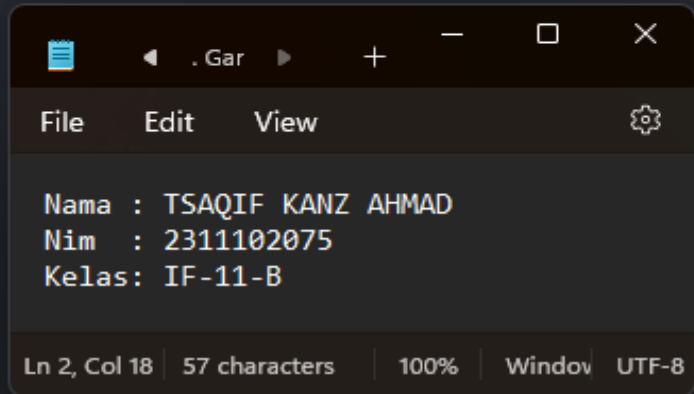
Pilihan : 1

Masukkan nama produk: Originote
Masukkan harga produk: 60000

1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit

Pilihan : 7

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



1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit

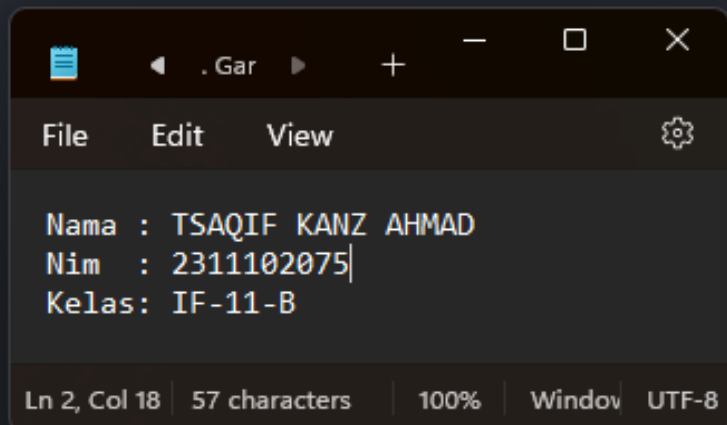
Pilihan : 4

Masukkan posisi data produk: 2
Masukkan nama produk: Azarine
Masukkan harga produk: 65000

1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit

Pilihan : 7

Nama Produk	Harga
Originote	60000
Somethinc	150000
Azarine	65000
Skintific	100000
Wardah	50000
Hanasui	30000



1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit

Pilihan : 5

Masukkan posisi data produk: 4

1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit

Pilihan : 7

Nama Produk	Harga
Originote	60000
Somethinc	150000
Azarine	65000
Skintific	100000
Hanasui	30000

.Gar

FileEditView

Nama : TSAQIF KANZ AHMAD
Nim : 2311102075
Kelas: IF-11-B

Ln 2, Col 1857 characters100%WindowUTF-8

1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit

Pilihan : 3

Masukkan posisi produk: 4

Masukkan nama baru produk: Cleora

Masukkan harga baru produk: 65000

1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit

Pilihan : 7

Nama Produk	Harga
Originote	60000
Somethinc	150000
Azarine	65000
Skintific	100000
Cleora	65000

.Gar

FileEditView

Nama : TSAQIF KANZ AHMAD
Nim : 2311102075
Kelas: IF-11-B

Ln 2, Col 1857 characters100%WindowUTF-8

DESKRIPSI PROGRAM

Pada program diatas adalah program mengimplementasikan Doubel Linked List untuk menyimpan data produk skincare disebuah toko. Program ini menyediakan berbagai fungsi untuk pengelolaan data produk secara efesien. Pada node menyimpan informasi produk dan pointer ke node sebelumnya ke node selanjutnya. Pada fungsi-fungsinya terdapat beberapa bagian untuk menambah data, menghapus data, update data dan pada operasi lainnya terdapat 'deleteAll' dan 'display' untuk menghapus semua node list dan menampilkan data semua node dalam list pada format tabel.

KESIMPULAN

Linked list merupakan struktur data yang sedikit berbeda jika dibandingkan dengan Array. Struktur data jenis ini termasuk cukup populer dan penting dipelajari. Karena sangat bermanfaat untuk meningkatkan pemahaman tentang struktur data, kemampuan pemrograman dan kemampuan problem solving. Masing – masing struktur data memiliki kelebihan dan kekurangannya sendiri, sehingga penting memilih untuk data yang tepat untuk menyelesaikan masalah tertentu.

DAFTAR PUSTAKA

[1] Asisten Praktikum, "Modul 3 Single dan Doubel Linked List", Learning ManagementSystem, 2024.

[2] Modul Kuliah Struktur data – Linked List – Universitas Esa Unggul :
https://lms-paralel.esaunggul.ac.id/pluginfile.php?file=%2F86227%2Fmod_resource%2Fcontent%2F1%2FModul%20Struktur%20Data-Linked%20List.pdf