

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

MODUL 3

“SINGLE AND DOUBLE LINKED LIST”



Disusun Oleh :
NAMA : Ibnu Rizal Mutaqim
NIM : 2311102067

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

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

A. Dasar Teori

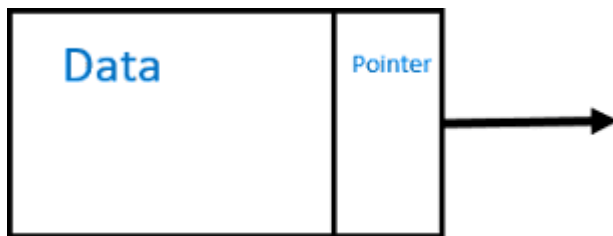
LINKED LIST

Linked list adalah suatu cara untuk menyimpan suatu data dengan struktur sehingga program dapat secara otomatis menciptakan suatu tempat baru untuk menyimpan suatu data kapan saja diperlukan. Secara rinci program dapat menuliskan struct atau definisi kelas yang berisi variabel yang memegang informasi yang ada didalamnya, dan mempunyai suatu pointer yang menunjukan ke suatu struct sesuai dengan tipe datanya.

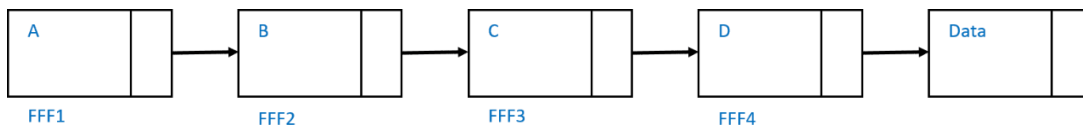
Single Linked List

Single artinya field pointernya hanya satu buah saja dan satu arahnya hanya satu buah saja dan satu arah serta pada akhir node, pointernya menunjuk NULL serta pada akhir node, pointer NULL.

Linked list artinya node linked list artinya node node tersebut saling terhubung satu sama node tersebut saling terhubung satu sama lain.



Setiap node-node pada linked list mempunyai field yang berisi pointer ke node berikutnya, dan juga memiliki field yang berisi data. Node terakhir akan menunjuk ke NULL akan digunakan sebagai kondisi berhenti pada saat pembacaan isi linked list.



Struktur dinamis ini mempunyai banyak keuntungan dibanding struktur array yang bersifat statis. Struktur ini lebih dinamis, karena banyak elemen dengan mudah ditambah atau dikurangi, berbeda dengan array yang ukurannya bersifat tetap. Disamping itu manipulasi terhadap setiap elemen seperti menyisipkan, menghapus, menambah dapat dilakukan dengan mudah.

Definisi link list dapat dituliskan secara sederhana dengan struktur berikut:

```
Struct Tnode {
```

```

    Int data;

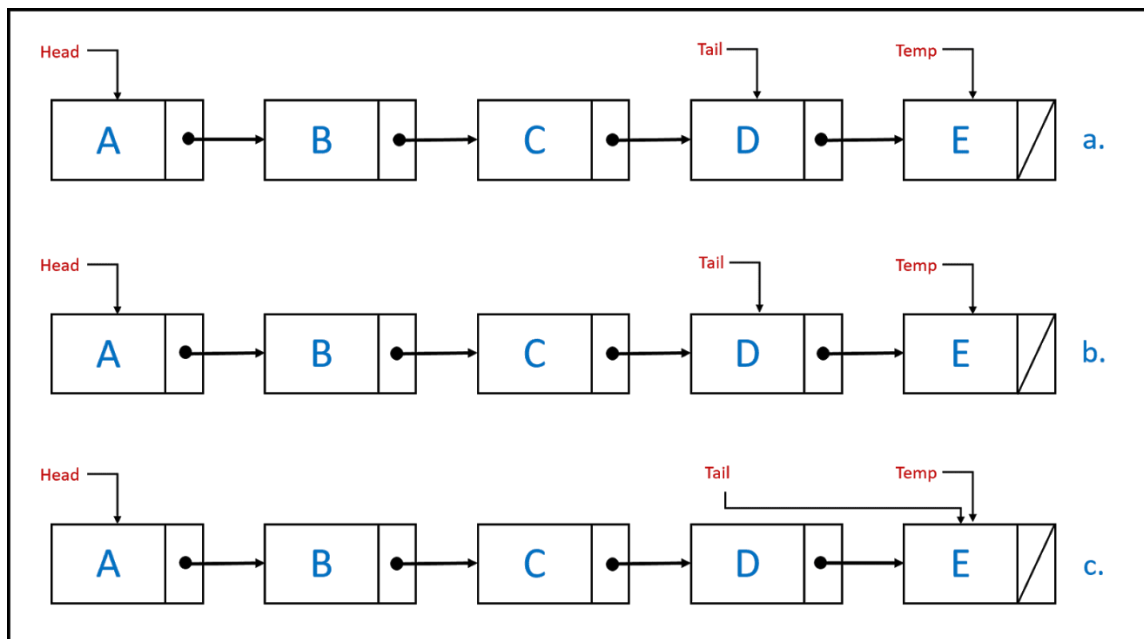
    Struct Tnode *next;

}

```

Pembuatan struct bernama Tnode yang berisi 2 field, yaitu field data bertipe integer dan field next yang bertipe pointer dari Tnode. Setelah pembuatan struct, buat variabel head yang bertipe pointer dari Tnode yang berguna sebagai kepala linked list.

Operasi yang dapat dilakukan pada sebuah link list adalah penambahan dan penghapusan elemen link list. Penambahan dan penghapusan dapat dilakukan pada posisi depan, posisi belakang atau posisi tertentu pada link list. Berikut ilustrasi penambahan data di belakang link list



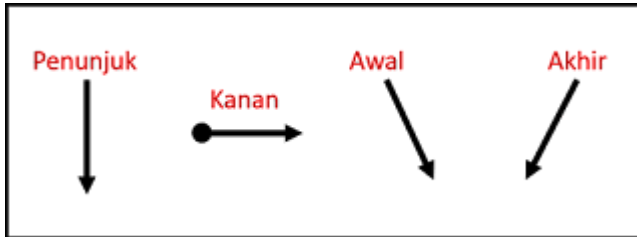
Double Linked List

Double linked list atau disebut juga Senarai Berantai Ganda hampir sama dengan Single Linked List yang telah dibahas pada bab sebelumnya, yaitu pengalokasian memori secara dinamis yang digunakan untuk menyimpan data. Bedanya Double Linked list lebih flexibel dibanding Single linked list karena pada Double Linked List semua simpul-simpulnya yang ada didalamnya memiliki 2 buah penunjuk yang digunakan untuk mengkaitkan diri dengan simpul lain yang ada disebelah kanannya dan di sebelah kirinya.

Didalam konsep Double Linked List ada 3 unsur pendukung yang penting, yaitu :

1. Penunjuk (Biasa disebut pointer)

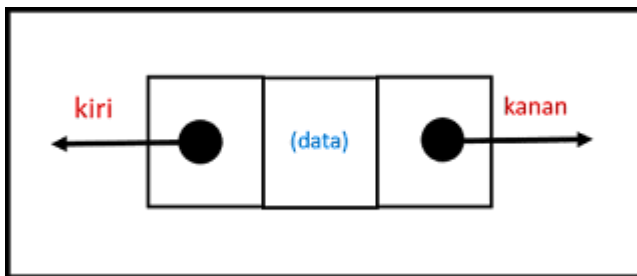
Penunjuk atau pointer yang dimaksud disini alat yang digunakan untuk menunjuk sebuah simpul, seperti pada gambar berikut.



Penunjuk dapat menunjuk ke sebuah simpul, ataupun ke sebuah tempat kosong (null).

2. Simpul Ganda (biasa disebut node)

Simpul yang digunakan dalam bab ini adalah simpul ganda yang digambarkan sebagai berikut.



Simpul yang demikian disebut 'simpul ganda' karena simpul ini mempunyai dua buah pointer, yaitu 'kiri' dan 'kanan'. Artinya pointer kanan menunjuk ke elemen disebelah kanannya dan pointer kiri menunjuk ke elemen disebelah kirinya. Dalam gambar diatas ini diilustrasikan sebuah simpul dalam Double Linked List.

Sedangkan (data) adalah data yang digunakan dalam simpul, kiri adalah pointer yang menunjuk pada simpul sebelumnya dan kanan adalah pointer yang menunjuk pada simpul sesudahnya. Untuk pembuatan simpul ganda dapat dideklarasikan dengan membuat struct strSimpul yang memiliki 2 anggota bertipe strSimpul, yaitu : kanan dan kiri yang berfungsi sebagai penunjuk. Kemudian, struct strSimpul tersebut dibuat aliasnya dengan nama simpul, seperti contoh dibawah ini.

```
typedef struct strSimpul
```

```

{

    data masukkan;

    struct strSimpul *kanan;

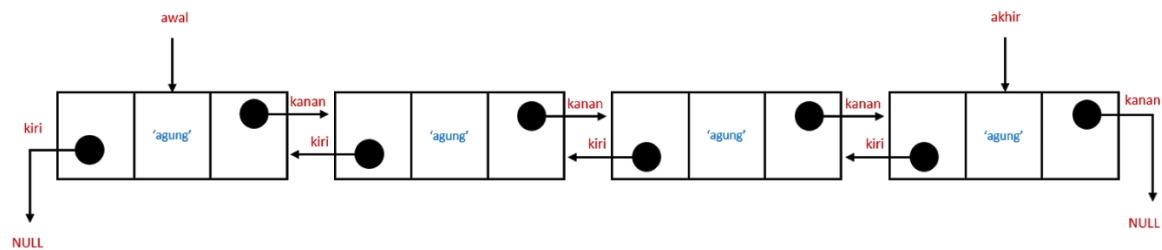
    struct strSimpul *kiri;

} Simpul;

```

3. Senarai berantai ganda atau Double Linked List itu sendiri

Seperti senari berantai tunggal, senarai berantai ganda merupakan kumpulan simpul-simpul yang terhubung satu dengan yang lain. Hanya bedanya pada senarai ini setiap simpulnya mempunyai 2 penghubung, kiri dan kanan, seperti Gambar berikut.



Untuk pembuatan Senarai Berantai Ganda dapat dideklarasika dengan membuat struct strSenarai Ganda yang memiliki 2 anggota bertipe simpul yaitu : awal dan akhir yang berfungsi sebagai penunjuk. Kemudian, struct strSenaraiGanda tersebut dibuat aliasnya dengan nama SenaraiGanda. Selain itu diperlukan juga fungsi untuk membuat Senarai Berantai ganda dengan nama buat dan parameter berupa pointer *q, yang memiliki anggota awal dan akhir dengan nilai awal masih kosong (null), seperti contoh dibawah ini:

```

typedef struct strSeneraiGanda

{

    Simpul *awal;

    Simpul *akhir;

} SeneraiGanda;

```

```
void buat (SeneraiGanda *q)
```

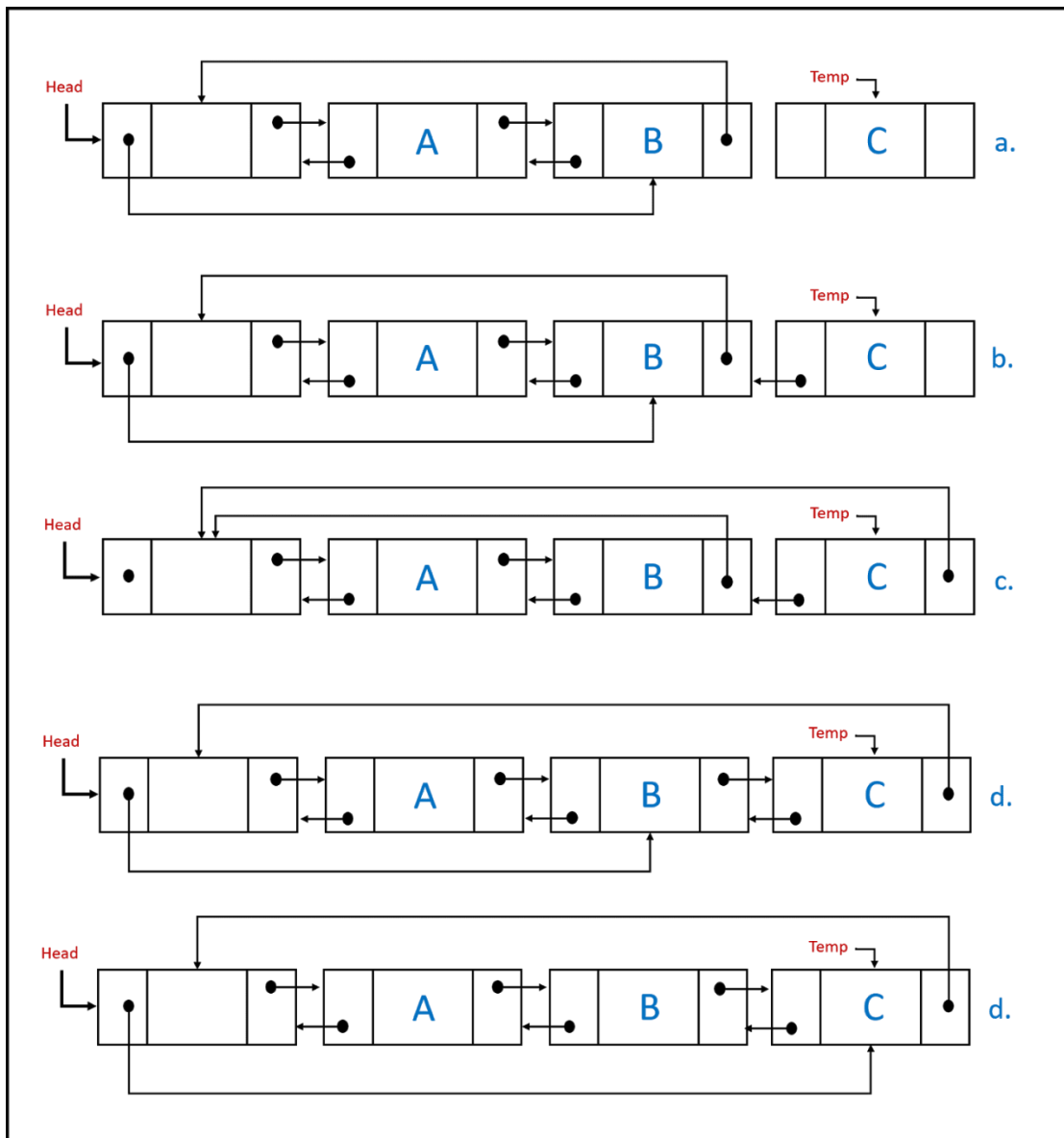
```
{
```

```
    q -> awal = NULL;
```

```
    q -> akhir = NULL;
```

```
}
```

Gambar berikut merupakan ilustrasi penambahan data dibelakang pada double link list.



B. Guided

Guided 1: Single Linked List

Source code:

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

// Deklarasi Struct Node
struct Node {
    int data;
    Node* next;
};

Node* head;
Node* tail;

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

// Pengecekan apakah list kosong
bool isEmpty() {
    return head == NULL;
}

// Tambah Node di depan
void insertDepan(int nilai) {
    Node* baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty()) {
```

```

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

// Tambah Node di belakang
void insertBelakang(int nilai) {
    Node* baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty()) {
        head = tail = baru;
    } else {
        tail->next = baru;
        tail = baru;
    }
}

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

// Tambah Node di posisi tengah
void insertTengah(int data, int posisi) {

```



```

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

```

// Hapus Node di depan

```

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

```

// Hapus Node di belakang

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

// Hapus Node di posisi tengah

```
void hapusTengah(int posisi) {  
    if (posisi < 1 || posisi > hitungList()) {  
        cout << "Posisi diluar jangkauan" << endl;  
    } else if (posisi == 1) {  
        cout << "Posisi bukan posisi tengah" << endl;  
    } else {  
        Node* hapus;  
        Node* bantu = head;  
        for (int nomor = 1; nomor < posisi - 1; nomor++) {  
            bantu = bantu->next;  
        }  
    }  
}
```

```

        hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    }
}

// Ubah data Node di depan
void ubahDepan(int data) {
    if (!isEmpty()) {
        head->data = data;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

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

```

```

// Ubah data Node di belakang
void ubahBelakang(int data) {
    if (!isEmpty()) {
        tail->data = data;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

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

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

```

```

    }
}

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

```

Screenshots Output

```

PS D:\KULIAH\KULIAH SEMESTER 2\PRAKTIKUM STRUKTUR DATA DAN ALGORITMA\PRAKRIKUM WEEK 3> cd
st.cpp -o SingleLinkedList } ; if ($?) { .\SingleLinkedList }
3
3 5
2 3 5
1 2 3 5
2 3 5
2 3
2 7 3
2 3
1 3
1 8
1 11
PS D:\KULIAH\KULIAH SEMESTER 2\PRAKTIKUM STRUKTUR DATA DAN ALGORITMA\PRAKRIKUM WEEK 3>

```

Nama : Ibnu Rizal Mutaqimpp
 NIM : 2311102067
 Kelas : IF-11-B

Ln 1, Col 2 57 characters | 100% | Windo UTF-8

Deskripsi:

Dalam program ini, setiap elemen dalam linked list direpresentasikan oleh struktur data Node, dengan atribut data untuk menyimpan nilai integer dan next untuk menunjukkan elemen berikutnya. Terdapat fungsi-fungsi untuk operasi dasar seperti penambahan, penghapusan, dan pengubahan data node, serta fungsi utilitas untuk memeriksa

kekosongan linked list dan menampilkan isi linked list. Fungsi main() digunakan untuk melakukan pengujian terhadap fungsi-fungsi tersebut dengan melakukan serangkaian operasi pada linked list dan memverifikasi hasilnya.

Guided 2: Double 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;
            }
        }
    }
}

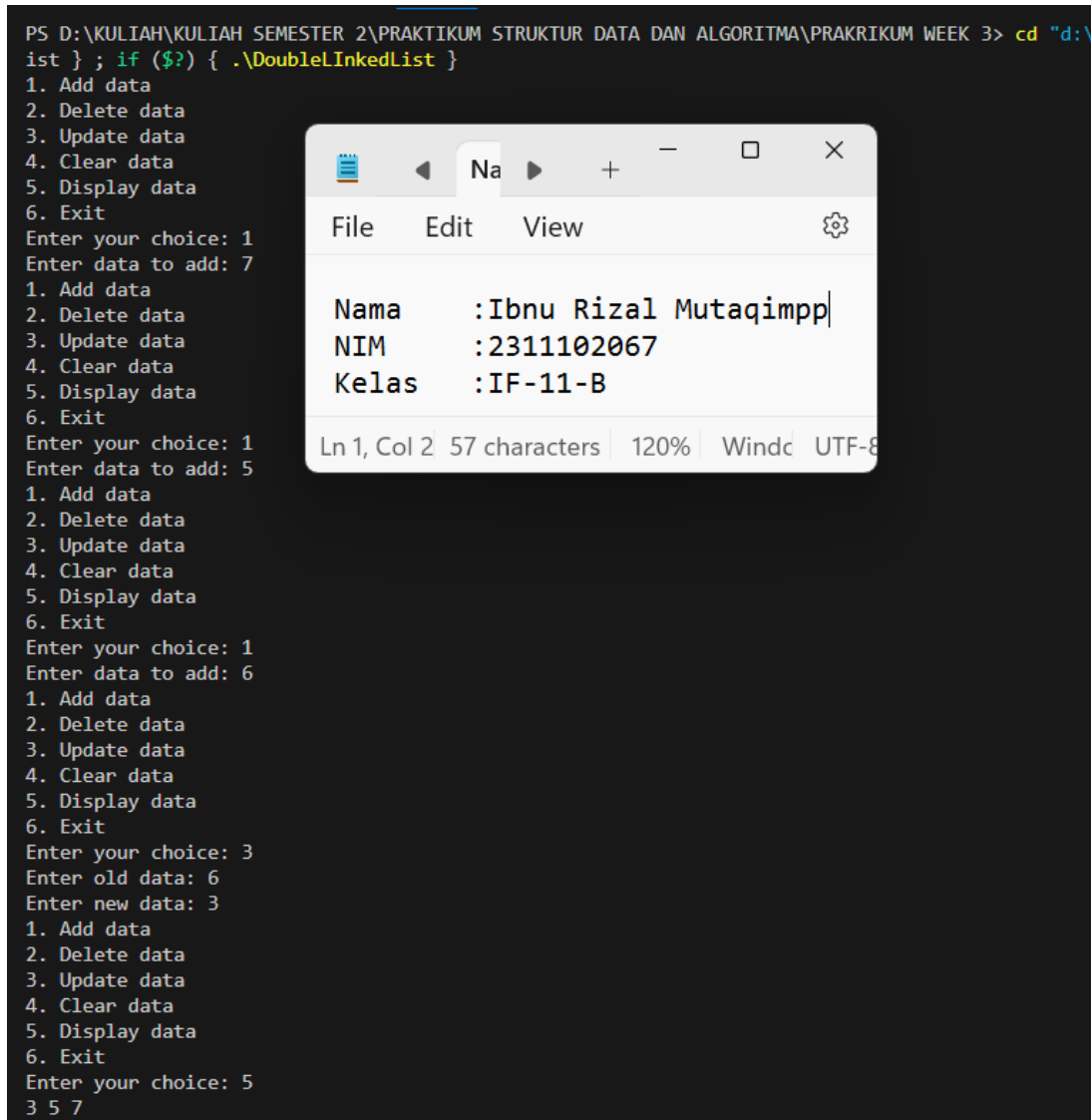
```



```
}  
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;  
}
```

Screenshots Output



```
PS D:\KULIAH\KULIAH SEMESTER 2\PRAKTIKUM STRUKTUR DATA DAN ALGORITMA\PRAKTIKUM WEEK 3> cd "d:\  
ist } ; if ($?) { .\DoubleLinkedList }  
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: 7  
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: 5  
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: 6  
1. Add data  
2. Delete data  
3. Update data  
4. Clear data  
5. Display data  
6. Exit  
Enter your choice: 3  
Enter old data: 6  
Enter new data: 3  
1. Add data  
2. Delete data  
3. Update data  
4. Clear data  
5. Display data  
6. Exit  
Enter your choice: 5  
3 5 7
```

Na

File Edit View

Nama : Ibnu Rizal Mutaqimpp
NIM : 2311102067
Kelas : IF-11-B

Ln 1, Col 2 57 characters 120% Windc UTF-8

Deskripsi:

Program ini implementasi dari Doubly Linked List. Doubly Linked List adalah struktur data berantai di mana setiap node memiliki dua pointer: satu untuk node sebelumnya dan satu lagi untuk node berikutnya. Program ini memiliki dua kelas utama: Node yang merepresentasikan node dalam Doubly Linked List, dan DoublyLinkedList yang merepresentasikan struktur data Doubly Linked List itu sendiri. Kelas DoublyLinkedList menyediakan fungsi untuk menambah, menghapus, mengubah, dan menampilkan data

dalam Doubly Linked List. Program memberikan opsi kepada pengguna untuk melakukan operasi tersebut melalui antarmuka terminal.

C. Unguided

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
-------	----

- Hapus data Akechi
- Tambahkan data berikut diantara John dan Jane: Futaba 18
- Tambahkan data berikut diawal: Igor 20
- Ubah data Michael menjadi: Reyn 18
- Tampilkan seluruh data

Source code:

```
#include <iostream>

using namespace std;
```

```
// deklarasi single linked list
struct Data{

    // komponen / member
    string nama;
    int umur;

    Data *next;

};

Data *head, *tail, *cur, *newNode, *del, *before;

// create single linked list
void createSingleLinkedList(string nama, int umur){
    head = new Data;
    head->nama = nama;
    head->umur = umur;
    head->next = NULL;
    tail = head;
}

// print single linked list
int countSingleLinkedList(){
    cur = head;
    int jumlah = 0;
    while( cur != NULL ){
        jumlah++;
        cur = cur->next;
    }
    return jumlah;
}
```

```

}

// tambahAwal Single linked list
void addFirst(string nama, int umur){
    newNode = new Data;
    newNode->nama = nama;

    newNode->umur = umur;
    newNode->next = NULL;
    if(head == NULL) {
        head = tail = newNode;
    } else {
        newNode->next = head;
        head = newNode;
    }
}

// tambahAkhir Single linked list
void addLast(string nama, int umur){
    newNode = new Data;
    newNode->nama = nama;

    newNode->umur = umur;
    newNode->next = NULL;
    if(head == NULL) {
        head = tail = newNode;
    } else {
        tail->next = newNode;
        tail = newNode;
    }
}

```

```

// tambah tengah single linked list
void addMiddle(string nama, int umur, int posisi){
    if( posisi < 1 || posisi > countSingleLinkedList() ){
        cout << "Posisi diluar jangkauan" << endl;
    }else if( posisi == 1){
        cout << "Posisi bukan posisi tengah" << endl;
    }else{
        newNode = new Data;
        newNode->nama = nama;

        newNode->umur = umur;

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

```

```

// Remove First
void removeFirst(){
    del = head;
    head = head->next;
    delete del;
}

```

```

// Remove Last

```

```

void removeLast(){
    del = tail;
    cur = head;
    while( cur->next != tail ){
        cur = cur->next;
    }
    tail = cur;
    tail->next = NULL;
    delete del;
}

// remove middle
void removeMiddle(int posisi){
    if( posisi < 1 || posisi > countSingleLinkedList() ){
        cout << "Posisi diluar jangkauan" << endl;
    }else if( posisi == 1 ){
        cout << "Posisi bukan posisi tengah" << endl;
    }else{
        int nomor = 1;
        cur = head;
        while( nomor <= posisi ){
            if( nomor == posisi-1 ){
                before = cur;
            }
            if( nomor == posisi ){
                del = cur;
            }
            cur = cur->next;
            nomor++;
        }
        before->next = cur;
        delete del;
    }
}

```

```

}

// ubahAwal Single linked list
void changeFirst(string nama, int umur){
    head->nama = nama;
    head->umur = umur;
}

// ubahAkhir Single linked list
void changeLast(string nama, int umur){
    tail->nama = nama;

    tail->umur = umur;
}

// ubah Tengah Single linked list
void changeMiddle(string nama, int umur, int posisi){
    if( posisi < 1 || posisi > countSingleLinkedList() ){
        cout << "Posisi diluar jangkauan" << endl;
    }else if( posisi == 1 || posisi == countSingleLinkedList() ){
        cout << "Posisi bukan posisi tengah" << endl;
    }else{
        cur = head;
        int nomor = 1;
        while( nomor < posisi ){
            cur = cur->next;
            nomor++;
        }
        cur->nama = nama;

        cur->umur = umur;
    }
}

```



```

// print single linked list
void printSingleLinkedList(){
    if(head != NULL) {
        cur = head;
        while( cur != NULL ){
            cout << cur->nama << "\t";

            cout << cur->umur << endl;

            cur = cur->next;
        }
        cout << endl;
    }
}

int main(){
    int choose, umur, posisi;
    string nama;
    cout << "Data Mahasiswa" << endl;
    do
    {
        cout << " 1. Tambah Depan" << endl;
        cout << " 2. Tambah Belakang" << endl;
        cout << " 3. Tambah Tengah" << endl;
        cout << " 4. Hapus Depan" << endl;
        cout << " 5. Hapus Belakang" << endl;
        cout << " 6. Hapus Tengah" << endl;
        cout << " 7. Ubah Depan" << endl;
        cout << " 8. Ubah Belakang" << endl;
        cout << " 9. Ubah Tengah" << endl;
        cout << " 10. Tampilkan" << endl;
        cout << " 0. Keluar Program" << endl;
    }
}

```

```
cout << " Pilihan : ";
cin >> choose;
switch (choose)
{
case 1:
    cout << "Masukkan Nama : ";
    cin >> nama;
    cout << "Masukkan umur : ";
    cin >> umur;
    addFirst(nama,umur);
    cout << endl;

    break;
case 2:
    cout << "Masukkan Nama : ";
    cin >> nama;
    cout << "Masukkan umur : ";
    cin >> umur;
    addLast(nama,umur);
    cout << endl;

    break;
case 3:
    cout << "Masukkan Posisi : ";
    cin >> posisi;
    cout << "Masukkan Nama : ";
    cin >> nama;
    cout << "Masukkan umur : ";
    cin >> umur;
    addMiddle(nama,umur,posisi);
    cout << endl;

    break;
```

case 4:

removeFirst();

cout << endl;

break;

case 5:

removeLast();

cout << endl;

break;

case 6:

cout << "Masukkan Posisi : ";

cin >> posisi;

removeMiddle(posisi);

cout << endl;

break;

case 7:

cout << "Masukkan Nama : ";

cin >> nama;

cout << "Masukkan umur : ";

cin >> umur;

changeFirst(nama,umur);

cout << endl;

break;

case 8:

cout << "Masukkan Nama : ";

cin >> nama;

cout << "Masukkan umur : ";

cin >> umur;

changeLast(nama,umur);

cout << endl;

```

        break;
    case 9:
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan umur : ";
        cin >> umur;
        changeMiddle(nama,umur,posisi);
        cout << endl;

        break;
    case 10:
        printSingleLinkedList();
        break;
    case 0:
        cout << "terimakasih";
        break;
    default:
        cout << "Pilihan Salah" << endl;
        break;
}
} while (choose != 0);
}

```

Screenshots Output

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

```
Ibnu 19
John 19
Jane 20
Michael 18
Yusuke 19
Akechi 20
Hoshino 18
Karin 18

1. Tambah Depan
2. Tambah Belakang
3. Tambah 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 : 
```

Na

File Edit View

Nama : Ibnu Rizal Mutaqimpp
NIM : 2311102067
Kelas : IF-11-B

Ln 1, Col 2 57 characters 100% Windc UTF-8

Hapus data Akechi

```
John 19
Jane 20
Michael 18
Yusuke 19
Akechi 20
Hoshino 18
Karin 18

1. Tambah Depan
2. Tambah Belakang
3. Tambah 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

1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 10
Ibnu 19
John 19
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18
```

Na

File Edit View

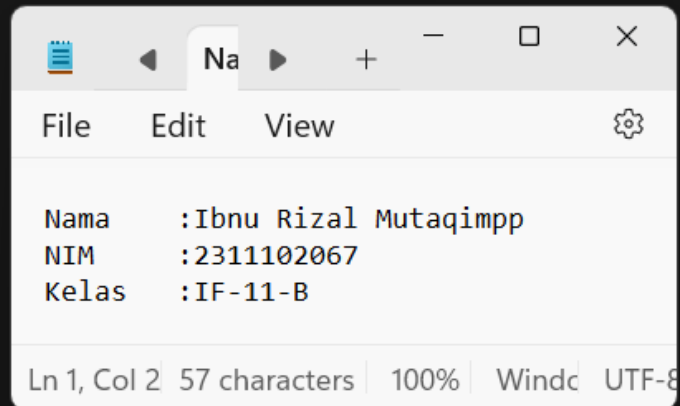
Nama : Ibnu Rizal Mutaqimpp
NIM : 2311102067
Kelas : IF-11-B

Ln 1, Col 2 57 characters 100% Windc UTF-8

Tambahkan data berikut diantara John dan Jane : Futaba 18

```
1. Tambah Depan
2. Tambah Belakang
3. Tambah 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 umur : 18
```

```
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 10
Ibnu    19
John    19
Futaba  18
Jane    20
Michael 18
Yusuke  19
Hoshino 18
Karin   18
```

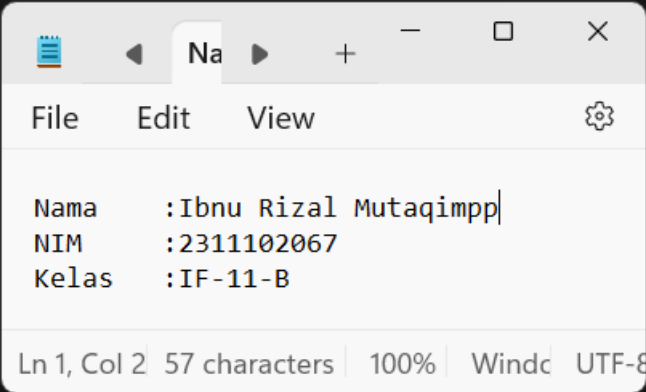


Tambahkan data berikut diawal: Igor

18

```
1. Tambah Depan
2. Tambah Belakang
3. Tambah 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 : Igor
Masukkan umur : 20

1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 10
Igor    20
Ibnu    19
John    19
Futaba  18
Jane    20
Michael 18
Yusuke  19
Hoshino 18
Karin   18
```



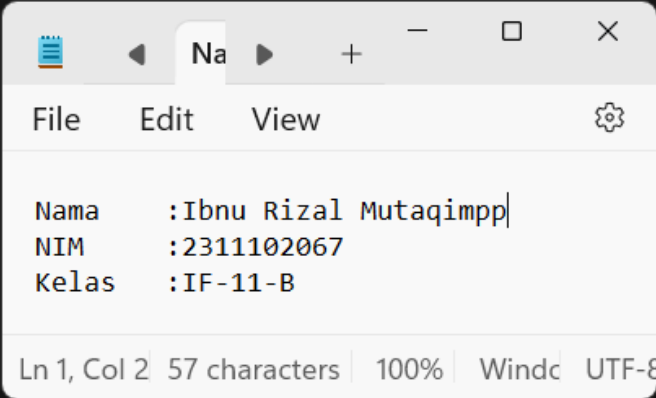
The image shows a Notepad++ window with a single tab titled 'Na'. The window has a menu bar with 'File', 'Edit', and 'View'. The text content of the window is as follows:

```
Nama      : Ibnu Rizal Mutaqimpp|
NIM       : 2311102067
Kelas    : IF-11-B
```

The status bar at the bottom of the window indicates 'Ln 1, Col 2 | 57 characters | 100% | Windo | UTF-8'.

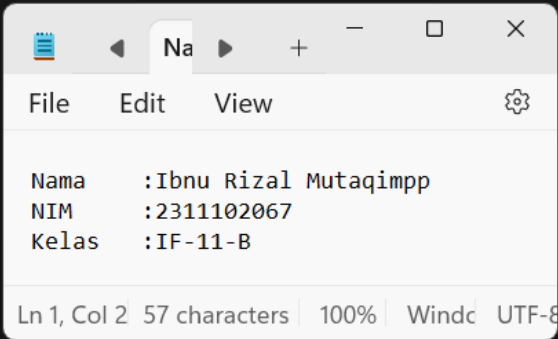
```
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 9
Masukkan Posisi : 6
Masukkan Nama : Reyn
Masukkan umur : 18

1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 10
Igor    20
Ibnu    19
John    19
Futaba  18
Jane    20
Reyn    18
Yusuke  19
Hoshino 18
Karin   18
```



Tampilkan seluruh Data:

```
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 10
Igor    20
Ibnu    19
John    19
Futaba  18
Jane    20
Reyn    18
Yusuke  19
Hoshino 18
Karin   18
```



Deskripsi:

Program ini menyajikan menu untuk pengguna untuk menambahkan data mahasiswa di awal, di akhir, atau di posisi tengah linked list, serta menghapus data mahasiswa dari awal, dari akhir, atau dari posisi tengah linked list. Pengguna juga dapat mengubah data mahasiswa yang ada di awal, di akhir, atau di posisi tengah linked list, serta menampilkan seluruh data mahasiswa yang tersimpan dalam linked list. Aplikasi ini menggunakan struktur data linked list sederhana yang terdiri dari elemen-elemen Data yang memiliki atribut nama dan umur, serta pointer next yang menunjukkan elemen berikutnya dalam linked list. Program berjalan dalam loop utama yang memungkinkan pengguna memilih berbagai operasi yang ingin dilakukan pada data mahasiswa, hingga pengguna memilih untuk keluar dari program.

Unguided 2: 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
2. Somethinc dan Skintific
3. Hapus produk wardah
4. Update produk Hanasui menjadi Cleora dengan harga 55.000
5. 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 pushMiddle(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 popMiddle(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 updateMiddle(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 bukan 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 print()
{
    if (head == nullptr)
    {
        cout << "List kosong." << endl;
        return;
    }

    Node *current = head;

    cout << setw(20) << left << "Nama Produk " << setw(10) << left << "Harga
Produk" << endl;

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

};

int main()
{
    DoublyLinkedList list;
    int choose;
    cout << endl
        << "Toko Ambatukam" << 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 >> choose;

    switch (choose)
    {
    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 p1osisi produk: ";
    cin >> posisi;
    cout << "Masukkan nama baru produk: ";
    cin >> newNamaProduk;
    cout << "Masukkan harga baru produk: ";
    cin >> newHarga;
    bool updatedMiddle = list.updateMiddle(newNamaProduk, newHarga, posisi);
    if (!updatedMiddle)
    {
        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.pushMiddle(namaProduk, harga, posisi);
    break;
}

case 5:
{

```

```

    int posisi;
    cout << "Masukkan posisi data produk: ";
    cin >> posisi;
    list.popMiddle(posisi);
    break;
}
case 6:
{
    list.deleteAll();
    break;
}
case 7:
{
    list.print();
    break;
}
case 8:
{
    return 0;
}
default:
{
    cout << "Invalid choice" << endl;
    break;
}
}
} while (choose != 8);

return 0;
}

```

Screenshots Output

Tambahkan Produk Azarine dengan harga 65000 diantara Somethinc dan Skintific

```
Pilihan : 4
Nama Produk      Harga Produk
Originate        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 Produk
Originate        60000
Somethinc         150000
Azarine          65000
Skintific         100000
Wardah           50000
Hanasui          30000
```

Na

File Edit View

Nama : Ibnu Rizal Mutaqimpp
NIM : 2311102067
Kelas : IF-11-B

Ln 1, Col 2 57 characters 100% Windc UTF-8

Hapus produk wardah

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

Na

File Edit View

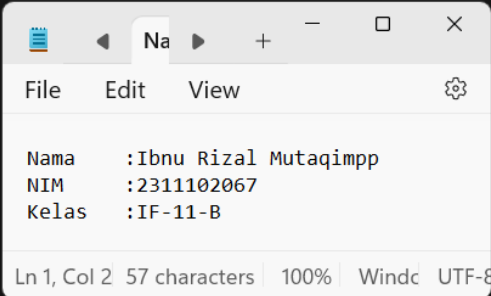
Nama : Ibnu Rizal Mutaqimpp
NIM : 2311102067
Kelas : IF-11-B

Ln 1, Col 2 57 characters 100% Windc UTF-8

Update produk Hanasui menjadi Cleora dengan harga 55000

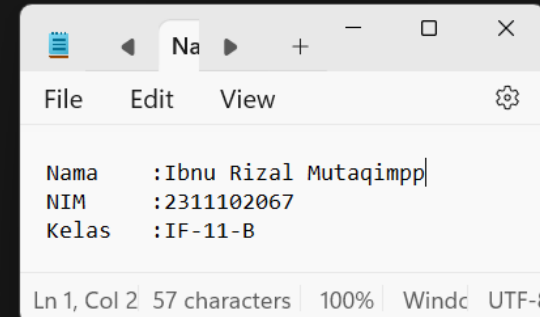
```
Nama Produk      Harga Produk
Originote        60000
Somethinc        150000
Azarine          65000
Skintific        100000
Hanasui          30000

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 : 3
Masukkan plosisi produk: 4
Masukkan nama baru produk: Cleora
Masukkan harga baru produk: 55000
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 Produk
Originote        60000
Somethinc        150000
Azarine          65000
Skintific        100000
Cleora           55000
```



Tampilkan Menu

```
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 Produk
Originote        60000
Somethinc        150000
Azarine          65000
Skintific        100000
Cleora           55000
```



Deskripsi:

Program ini memungkinkan pengguna untuk melakukan berbagai operasi, termasuk menambah, menghapus, mengubah, dan menampilkan data produk. Setiap produk direpresentasikan sebagai sebuah Node dalam linked list, yang memiliki atribut seperti nama produk dan harga, serta pointer yang menunjuk ke elemen sebelumnya dan berikutnya. Program berjalan dalam sebuah loop utama yang memungkinkan pengguna memilih operasi yang diinginkan dari menu yang disediakan. Ini memungkinkan interaksi yang fleksibel dengan data produk, dengan kemampuan untuk menambah, menghapus,

dan mengubah data pada posisi tertentu, serta menghapus semua data produk jika diperlukan.

D. Kesimpulan

Kesimpulan singkatnya adalah:

Single Linked List:

- Terdiri dari node-node yang terhubung secara satu arah.
- Hanya memiliki satu pointer (`next`) untuk menunjuk ke node berikutnya.
- Membutuhkan sedikit ruang memori dan lebih sederhana dalam implementasi.
- Tidak mendukung penelusuran mundur langsung.

Double Linked List:

- Terdiri dari node-node yang terhubung dua arah, yaitu ke node sebelumnya dan berikutnya.
- Memiliki dua pointer, prev dan next, untuk menunjuk ke node sebelumnya dan berikutnya.
- Mendukung penelusuran mundur langsung dan lebih fleksibel dalam operasi-operasi tertentu seperti penghapusan dan penambahan pada posisi tertentu.
- Memerlukan lebih banyak ruang memori karena menyimpan dua pointer untuk setiap node.

2. Referensi (APA)

[1] Asisten Pratikum “Modul 3 Single And Double Linked List ”, Learning Management System, 2024.

[2] Imam Ibnu Badri. (2019, maret). Single Linked List. Diakses pada 30 maret 2024.

<https://www.teachmesoft.com/2019/03/single-linked-list-c-disertai-contoh.html>

[3] Imam Ibnu Badri. (2019, maret). Double Linked List. Diakses pada 30 maret 2024.

<https://www.teachmesoft.com/2019/03/double-link-list-c-disertai-contoh.html>