

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL 3
“SINGLE AND DOUBLE LIST”**



**DISUSUN OLEH :
RIZKY PERLINTA SEMBIRNG
2311102061**

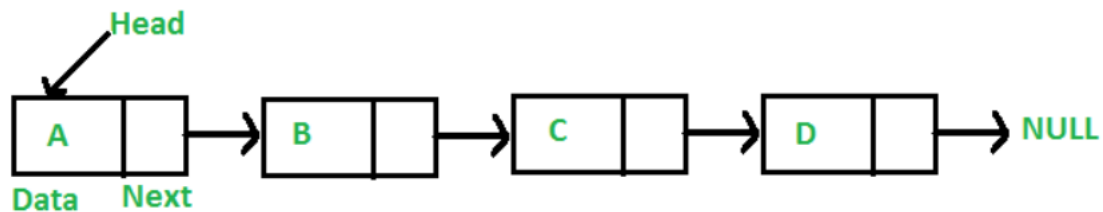
**DOSEN
WAHYU ANDI SAPUTRA, S.PD., M.PD.**

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

A. DASAR TEORI

Pengertian Linked List

Linked list adalah struktur data linier berbentuk rantai simpul di mana setiap simpul menyimpan 2 item, yaitu nilai data dan pointer ke simpul elemen berikutnya. Berbeda dengan array, elemen linked list tidak ditempatkan dalam alamat memori yang berdekatan melainkan elemen ditautkan menggunakan pointer.



Simpul pertama dari linked list disebut sebagai head atau simpul kepala. Apabila linked list berisi elemen kosong, maka nilai pointer dari head menunjuk ke NULL. Begitu juga untuk pointer berikutnya dari simpul terakhir atau simpul ekor akan menunjuk ke NULL.

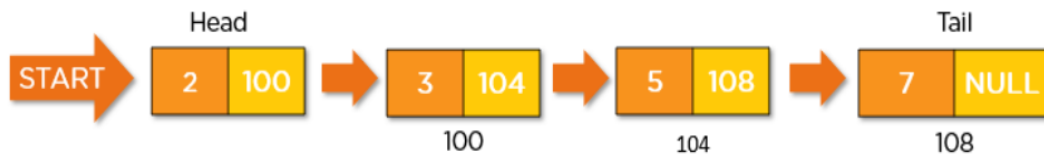
Ukuran elemen dari linked list dapat bertambah secara dinamis dan mudah untuk menyisipkan dan menghapus elemen karena tidak seperti array, kita hanya perlu mengubah pointer elemen sebelumnya dan elemen berikutnya untuk menyisipkan atau menghapus elemen.

Jenis Jenis Linked List

Secara umum, linked list dapat dibagi ke dalam 4 jenis, yakni: Singly linked list, Doubly linked list, Circular linked list, dan Circular doubly linked list. Tetapi di dalam dasar teori ini kita hanya membahas Single dan Double list.

1. Singly linked list

Single Linked List adalah salah satu struktur data yang paling mendasar dan penting dalam ilmu komputer dan pemrograman. Ini adalah kumpulan simpul yang terhubung, di mana setiap simpul memiliki dua bagian: data dan pointer ke simpul berikutnya dalam urutan. Ini adalah struktur data linier yang secara dinamis dapat tumbuh dan menyusut saat elemen-elemennya ditambahkan atau dihapus. Jadi, kita hanya dapat melintasinya dalam satu arah, yaitu dari simpul kepala ke simpul ekor.



Keuntungan dan Kekurangan:

Keuntungan:

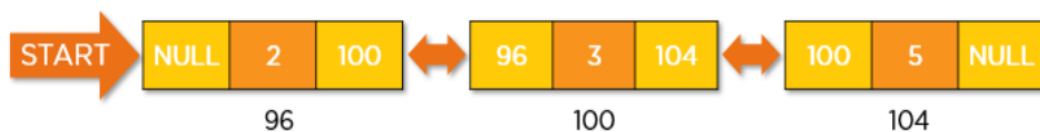
- Penyisipan dan penghapusan elemen dapat dilakukan dengan cepat di awal atau di tengah daftar.
- Struktur data yang fleksibel dan dinamis, memungkinkan penyesuaian ukuran saat berjalan.

Kekurangan:

- Tidak mendukung akses acak ke elemen (perlu dilakukan penelusuran dari awal untuk mencapai elemen tertentu).
- Memerlukan lebih banyak ruang dalam memori daripada array karena setiap elemen memiliki overhead pointer tambahan.

2. Double List

Double Linked List adalah struktur data yang mirip dengan Single Linked List, namun dengan tambahan pointer tambahan yang menunjuk ke simpul sebelumnya, sehingga memungkinkan navigasi maju dan mundur. Dalam Double Linked List, setiap simpul memiliki tiga bagian: data, pointer ke simpul sebelumnya, dan pointer ke simpul berikutnya.



Keuntungan dan Kekurangan:

Keuntungan:

- Mendukung navigasi maju dan mundur, memungkinkan operasi seperti penghapusan di tengah daftar menjadi lebih efisien.
- Penyisipan dan penghapusan elemen dapat dilakukan dengan cepat di awal atau di tengah daftar.

Kekurangan:

- Memerlukan lebih banyak ruang dalam memori daripada Single Linked List karena setiap elemen memiliki overhead pointer tambahan.
- Kompleksitas implementasi yang sedikit lebih tinggi dibandingkan dengan Single Linked List.

B. Guided

Guided 1 : Program Input Array Tiga Dimensi

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;
    }
}

```

```

}

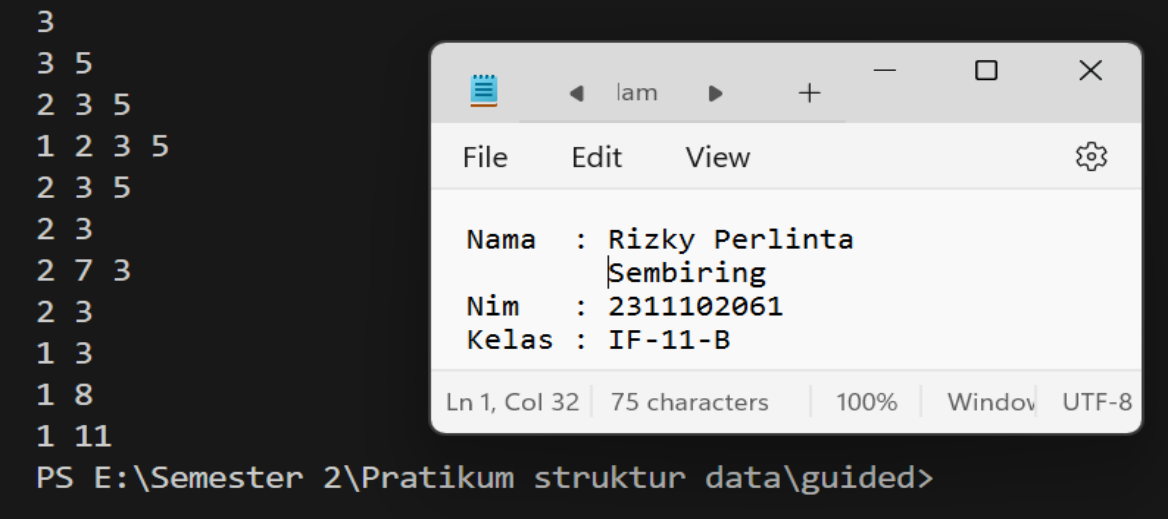
// Hapus semua Node di list
void clearList() {
    Node* bantu = head;
    while (bantu != NULL) {
        Node* hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}

// 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;
}

```


Screenshot Output



The screenshot shows a terminal window on the left and a Notepad++ window on the right. The terminal displays the output of a program, which is a linked list containing the numbers 3, 5, 2, 3, 5, 1, 2, 3, 5, 2, 3, 2, 7, 3, 2, 3, 1, 3, 1, 8, and 1, 11. The Notepad++ window shows a text file named 'lam' with the following content:

```
Nama : Rizky Perlinta  
Sembiring  
Nim : 2311102061  
Kelas : IF-11-B
```

The terminal prompt is `PS E:\Semester 2\Pratikum struktur data\guided>`.

Deskripsi Program : Program ini adalah program yang mengimplementasikan struktur data linked list dengan menggunakan struct Node. Program ini menggunakan fungsi untuk mengelola linked list, seperti inisialisasi list, pengecekan list kosong, tambah data di depan dan belakang, hitung jumlah data di list, tambah data di posisi tengah, hapus data di depan, belakang, dan posisi tengah, ubah data di depan, belakang, dan posisi tengah, serta hapus semua data di list dan tampilkan semua data di list.

Guided 2 : Latihan 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;
}


```

Screenshot Output :

```

1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 1
Enter data to add: 10
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit

```


lam

File

Edit

View

⚙️

Nama : Rizky Perlinta

Sembiring

Nim : 2311102061

Kelas : IF-11-B

Ln 1, Col 32

75 characters

100%

Window

UTF-8

```
Enter your choice: 3
Enter old data: 10
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
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 
```

File Edit View

Nama : Rizky Perlinta
Sembiring
Nim : 2311102061
Kelas : IF-11-B

Ln 1, Col 32 | 75 characters | 100% | Window UTF-8

Deskripsi Program : Program ini adalah contoh penggunaan linked list dua arah (doubly linked list) dalam bahasa pemrograman C++. Program ini menggunakan class DoublyLinkedList yang memiliki beberapa fungsi untuk mengelola linked list, seperti push, pop, update, deleteAll, dan display. push(int data): Fungsi ini digunakan untuk menambahkan data baru ke dalam linked list. Data baru akan ditambahkan di depan linked list. pop(): Fungsi ini digunakan untuk menghapus data terakhir dari linked list. update(int oldData, int newData): Fungsi ini digunakan untuk mengubah data dengan nilai oldData menjadi newData dalam linked list. deleteAll(): Fungsi ini digunakan untuk menghapus semua data dalam linked list. display(): Fungsi ini digunakan untuk menampilkan semua data dalam linked list.

C. 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

d. Tambahkan data berikut diawal : Igor

e. Ubah data Michael menjadi : Reyn

f. Tampilkan seluruh data

Source Code :

```
// UNGUIDED 1
#include <iostream>
using namespace std;

// Deklarasi Struct Node
struct Node
{
    string nama;
    int usia;
    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(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;
    }
}

// Tambah Node di belakang
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;
    }
}

// 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(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;
    }
}

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

// Ubah data Node di posisi tengah
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;
}
}

// Ubah data Node di belakang
void ubahBelakang(string nama, int usia)
{
    if (!isEmpty())
    {
        tail->nama = nama;
        tail->usia = usia;
    }
    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->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;

```

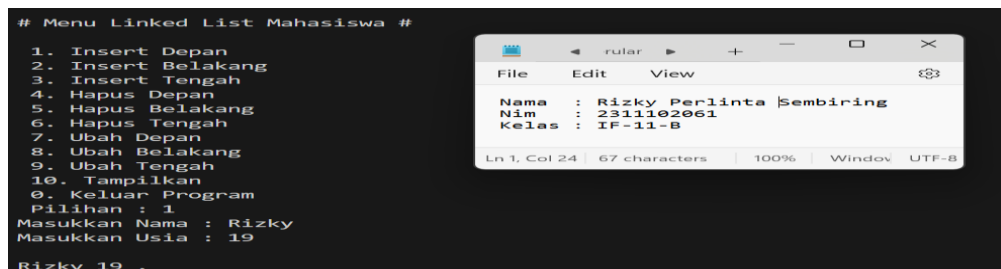
```
case 3:
    cout << "Masukkan Posisi : ";
    cin >> posisi;
    cout << "Masukkan Nama : ";
    cin >> nama;
    cout << "Masukkan Usia : ";
    cin >> usia;
    insertTengah(nama, usia, posisi);
    cout << endl;
    tampil();
    break;
case 4:
    hapusDepan();
    cout << endl;
    tampil();
    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;
}
```

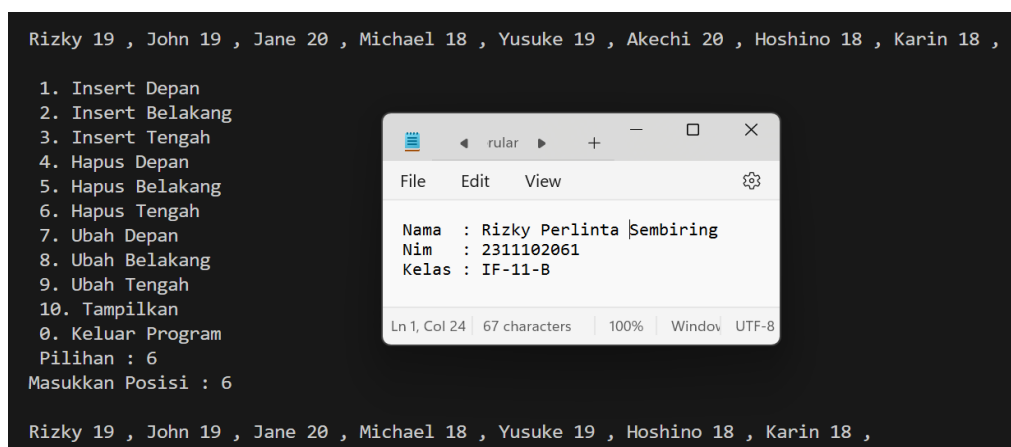
Screenshot Output :

- a. Data pertama yang dimasukkan adalah nama dan usia anda.



The screenshot shows a terminal window with a menu titled "# Menu Linked List Mahasiswa #". The menu lists 10 options: 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, and 0. Keluar Program. The user has selected option 1. The program prompts for "Masukkan Nama : Rizky" and "Masukkan Usia : 19". The output shows "Rizky 19 ,". An overlaid window shows the input data: Nama : Rizky Perlinta Sembiring, Nim : 2311102061, and Kelas : IF-11-B.

- b. Hapus data Akechi



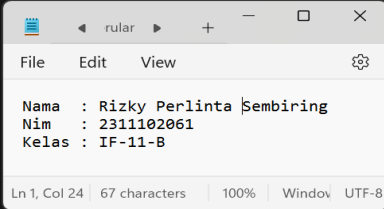
The screenshot shows the same menu as in the previous screenshot. The user has selected option 6. The program prompts for "Masukkan Posisi : 6". The output shows "Rizky 19 , John 19 , Jane 20 , Michael 18 , Yusuke 19 , Akechi 20 , Hoshino 18 , Karin 18 ,". An overlaid window shows the input data: Nama : Rizky Perlinta Sembiring, Nim : 2311102061, and Kelas : IF-11-B.

c. Tambah data Futaba 18 diantara John dan Jane

```
Rizky 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

Rizky 19 , John 19 , Futaba 18 , Jane 20 , Michael 18 , Yusuke 19 , Hoshino 18 , Karin 18 ,
```

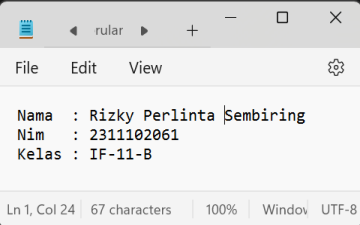


d. Tambah data Igor 20 diawal

```
Rizky 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 : 1
Masukkan Nama : Igor
Masukkan Usia : 20

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

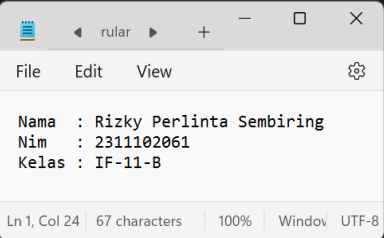


e. Ubah data Michael menjadi Reyn 18

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

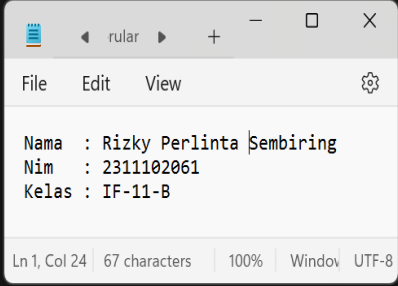
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 9
Masukkan Posisi : 6
Masukkan Nama : Reyn
Masukkan Usia : 18

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



f. Tampilkan seluruh data

```
,
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 : 10
Igor 20 , Rizky 19 , John 19 , Futaba 18 , Jane 20 , Reyn 18 , Yusuke 19 , Hoshino 18 , Karin 18
,
```



The screenshot shows a window titled 'rular' with a menu bar containing 'File', 'Edit', and 'View'. The window displays the output of a C++ program. It shows a menu with 11 options (1-10 and 0 for exit). Option 10, 'Tampilkan', has been selected. Below the menu, the program has printed a list of names and ages: 'Igor 20 , Rizky 19 , John 19 , Futaba 18 , Jane 20 , Reyn 18 , Yusuke 19 , Hoshino 18 , Karin 18'. The status bar at the bottom indicates 'Ln 1, Col 24 | 67 characters | 100% | Window | UTF-8'.

Dekripsi Program : Program di atas merupakan implementasi dari struktur data Linked List menggunakan bahasa pemrograman C++. Program ini memungkinkan pengguna untuk melakukan operasi-operasi dasar pada Linked List, seperti menambahkan node di depan, di belakang, atau di tengah, menghapus node di depan, di belakang, atau di tengah, serta mengubah data node. Di program diatas kita disuruh untuk menginput nama dan umur kita, menghapus data akechi menambah data futaba diantara john dan jane, menambah data igor di awal, mengubah data michael menjadi reyn dan menampilkan output.

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 Somethinc dan Skintific 2.
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: ";
            }
        }
    } while (choice != 8);
}

```



```

        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:

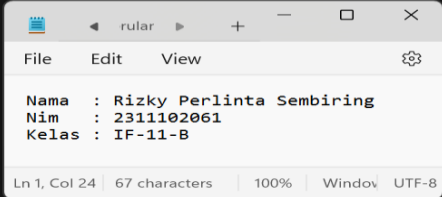
- a. Tambahkan produk Azarine dengan harga 65000 diantara Somethinc dan Skintific.

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



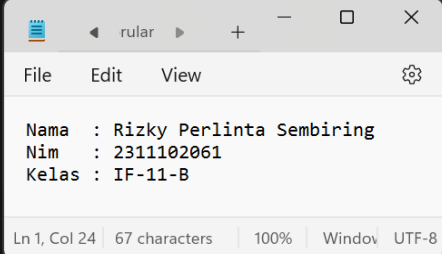
- b. Hapus produk wardah.

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

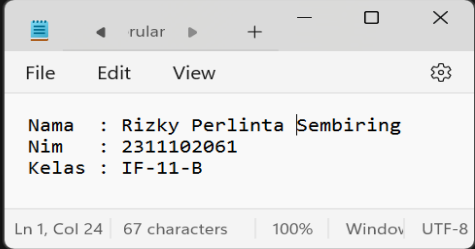
Nama Produk	Harga
Originote	60000
Somethinc	150000
Azarine	65000
Skintific	100000
Hanasui	30000



c. Update produk Hanasui menjadi Cleora dengan harga 55.000

```
Pilihan : 3
Masukkan posisi 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
```

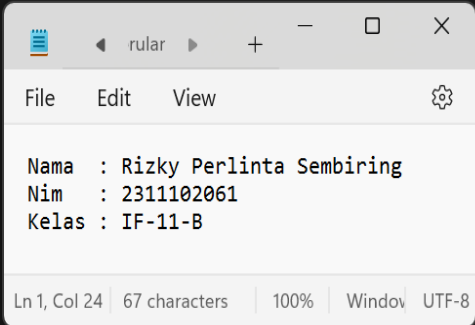
Nama Produk	Harga
Originote	60000
Somethinc	150000
Azarine	65000
Skintific	100000
Cleora	55000



d. Tampilkan menu

```
Pilihan : 7
```

Nama Produk	Harga
Originote	60000
Somethinc	150000
Azarine	65000
Skintific	100000
Cleora	55000



Deskripsi Program : Program di atas adalah implementasi dari struktur data Doubly Linked List dalam bahasa pemrograman C++. Doubly Linked List adalah jenis dari linked list dimana setiap node memiliki dua pointer, yaitu pointer yang menunjuk ke node sebelumnya (prev) dan pointer yang menunjuk ke node selanjutnya (next).

Dalam program ini, terdapat dua kelas utama: Node yang merepresentasikan node dalam linked list, dan DoublyLinkedList yang merepresentasikan linked list itu sendiri. Kelas Node memiliki empat atribut: namaProduk (sebuah string yang menyimpan nama produk), harga (sebuah integer yang menyimpan harga produk), prev (pointer yang menunjuk ke node sebelumnya), dan next (pointer yang menunjuk ke node selanjutnya). Kelas DoublyLinkedList memiliki dua pointer, head dan tail, yang menunjuk ke node pertama dan terakhir dalam linked list.

D. Kesimpulan

Penggunaan Single Linked List dan Double Linked List tergantung pada kebutuhan aplikasi dan tujuan penggunaan. Single Linked List lebih cocok untuk aplikasi yang membutuhkan operasi insertion dan deletion yang sederhana, sedangkan Double Linked List lebih cocok untuk aplikasi yang membutuhkan operasi insertion dan deletion yang lebih kompleks.

E. Referensi

[1] Asisten Pratikum “Modul 3 SINGLE AND DOUBLE LIST”, Learning Management System, 2024.

[2] Trivusi (2020, 16 September). Struktur Data Linked List: Pengertian, Karakteristik, dan Jenis-jenisnya.

Diakses Pada 29 Maret 2024, dari

<https://www.trivusi.web.id/2022/07/struktur-data-linked-list.html>