LAPORAN PRAKTIKUM STRUKTUR DATA

MODUL IV
SINGLY LINKED LIST (BAGIAN PERTAMA)

Disusun Oleh :

NAMA : ADIKA AUNURFIKRI NOVIYANTO

NIM : 103112400195

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

C++ adalah pengembangan dari dari bahasa c yang dibuat oleh Bjarne Stroustrup sekitar tahun 1980-an. C++ disebut bahasa multi-paradigma, artinya bisa dipakai dengan gaya prosedural (pakai fungsi biasa), beriorientasi objek (pakai class dan object), atau bahkan gabungan keduanya. C++ punya dasar-dasar seperti variabel, operator percabangan (if, switch), perulangan (for, while), dan bisa memakai class untuk membuat objek.

B. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1

Playlist.h

```
#ifndef PLAYLIST_H
#define PLAYLIST_H

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

struct Lagu {
    string judul;
    string penyanyi;
    float durasi;
    Lagu* next;
};

class Playlist {
private:
    Lagu* head;
public:
    Playlist();
    void tambahAwal(string judul, string penyanyi, float durasi);
    void tambahAkhir(string judul, string penyanyi, float durasi);
    void tambahSetelahKe3(string judul, string penyanyi, float durasi);
    void hapusLagu(string judul);
    void tampilkan();
};

#endif
```

Playlist.cpp

```
#include "Playlist.h"

Playlist::Playlist() {
    head = nullptr;
}

void Playlist::tambahAwal(string judul, string penyanyi, float durasi) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, head};
    head = baru;
}

void Playlist::tambahAkhir(string judul, string penyanyi, float durasi) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, nullptr};
    if (!head) {
        head = baru;
        return;
    }
    Lagu* temp = head;
    while (temp->next) temp = temp->next;
    temp->next = baru;
}

void Playlist::tambahSetelahKe3(string judul, string penyanyi, float durasi) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, nullptr};
    Lagu* temp = head;
    int count = 1;
    while (temp && count < 3) {
        temp = temp->next;
        count++;
    }
    if (!temp) {
        cout << "Playlist kurang dari 3 lagu.\n";
        delete baru;
        return;
```

```
    }
    baru->next = temp->next;
    temp->next = baru;
}

void Playlist::hapusLagu(string judul) {
    if (!head) {
        cout << "Playlist kosong.\n";
        return;
    }
    if (head->judul == judul) {
        Lagu* hapus = head;
        head = head->next;
        delete hapus;
        cout << "Lagu \"" << judul << "\" dihapus.\n";
        return;
    }
    Lagu* temp = head;
    while (temp->next && temp->next->judul != judul) {
        temp = temp->next;
    }
    if (!temp->next) {
        cout << "Lagu \"" << judul << "\" tidak ditemukan.\n";
        return;
    }
    Lagu* hapus = temp->next;
    temp->next = hapus->next;
    delete hapus;
    cout << "Lagu \"" << judul << "\" dihapus.\n";
}

void Playlist::tampilkan() {
    if (!head) {
        cout << "Playlist kosong.\n";
        return;
```

```
    }
    Lagu* temp = head;
    int i = 1;
    cout << "\n=== Daftar Lagu ===\n";
    while (temp) {
        cout << i++ << ". Judul: " << temp->judul
             << " | Penyanyi: " << temp->penyanyi
             << " | Durasi: " << temp->durasi << " menit\n";
        temp = temp->next;
    }
}
```

Main.cpp

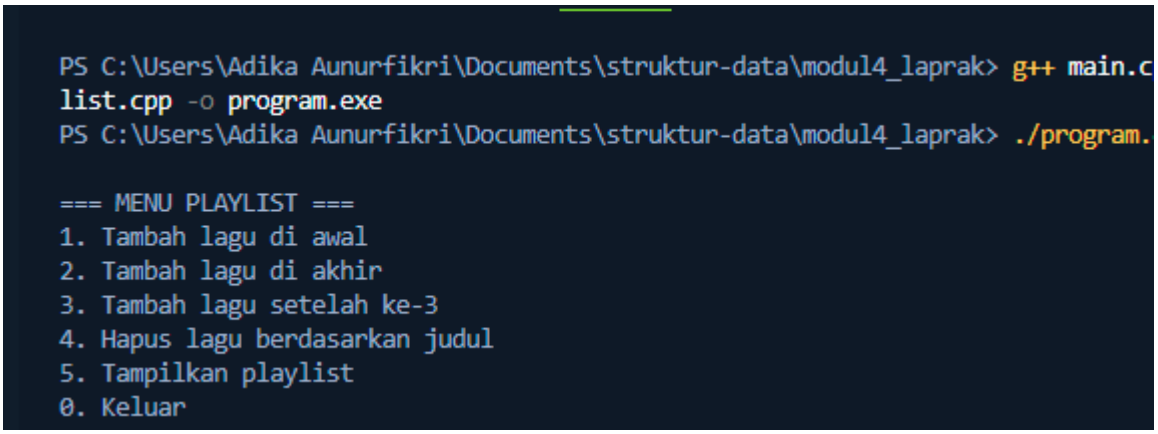
```
#include "Playlist.h"

int main() {
    Playlist p;
    int pilih;
    string judul, penyanyi;
    float durasi;

    do {
        cout << "\n=== MENU PLAYLIST ===\n";
        cout << "1. Tambah lagu di awal\n";
        cout << "2. Tambah lagu di akhir\n";
        cout << "3. Tambah lagu setelah ke-3\n";
        cout << "4. Hapus lagu berdasarkan judul\n";
        cout << "5. Tampilkan playlist\n";
        cout << "0. Keluar\n";
        cout << "Pilih: ";
        cin >> pilih;
        cin.ignore();

        switch (pilih) {
            case 1:
```

```
    cout << "Judul lagu: "; getline(cin, judul);
    cout << "Penyanyi: "; getline(cin, penyanyi);
    cout << "Durasi (menit): "; cin >> durasi;
    p.tambahAwal(judul, penyanyi, durasi);
    break;

case 2:
    cout << "Judul lagu: "; getline(cin, judul);
    cout << "Penyanyi: "; getline(cin, penyanyi);
    cout << "Durasi (menit): "; cin >> durasi;
    p.tambahAkhir(judul, penyanyi, durasi);
    break;

case 3:
    cout << "Judul lagu: "; getline(cin, judul);
    cout << "Penyanyi: "; getline(cin, penyanyi);
    cout << "Durasi (menit): "; cin >> durasi;
    p.tambahSetelahKe3(judul, penyanyi, durasi);
    break;

case 4:
    cout << "Masukkan judul lagu yang ingin dihapus: ";
    getline(cin, judul);
    p.hapusLagu(judul);
    break;

case 5:
    p.tampilkan();
    break;

case 0:
    cout << "Keluar dari program.\n";
    break;

default:
```

```
            cout << "Pilihan tidak valid.\n";
        }
    } while (pilih != 0);

    return 0;
}
```

Screenshots Output:

Menu Playlist



Tambah Lagu di Awal

Tambah Lagu dan Akhir

Tambah Lagu Setelah ke-3

Tampilkan Playlist

```
=== Daftar Lagu ===
1. Judul: Laskar Pelangi | Penyanyi: Nidji | Durasi: 4.2 menit
2. Judul: hahaha | Penyanyi: juicy luicy | Durasi: 3.14 menit
3. Judul: Halu | Penyanyi: Feby Putri | Durasi: 3.5 menit
4. Judul: hujan | Penyanyi: utopia | Durasi: 3.56 menit
```

Hapus Lagu Berdasarkan Judul

```
Pilih: 4
Masukkan judul lagu yang ingin dihapus: Halu
Lagu "Halu" dihapus.
```

Deskripsi:

Program Single Linked List pengelola playlist lagu ini merupakan program berbasis C++ yang menggunakan struktur data *linked list* untuk menyimpan dan mengelola daftar lagu secara dinamis. Setiap lagu direpresentasikan sebagai node yang berisi judul, penyanyi, dan durasi lagu, serta penunjuk ke lagu berikutnya. Program ini menyediakan menu interaktif bagi pengguna untuk menambahkan lagu di awal, di akhir, atau setelah lagu ke-3 dalam playlist, menghapus lagu berdasarkan judul, serta menampilkan seluruh isi playlist secara berurutan. Dengan memanfaatkan pointer dan operasi dasar pada *linked list*, program ini memudahkan pengelolaan data tanpa perlu menentukan ukuran tetap pada awalnya, sehingga efisien dan fleksibel untuk menambah atau menghapus lagu kapan saja.

C. Kesimpulan

Kesimpulannya, program Single Linked List pengelola playlist lagu ini menunjukkan bagaimana struktur data *linked list* dapat digunakan untuk mengelola data secara dinamis dan efisien. Melalui penerapan pointer, setiap elemen data (lagu) dapat ditambahkan atau dihapus tanpa perlu menggeser elemen lain seperti pada array. Konsep ini memperlihatkan pentingnya pemahaman tentang alokasi memori dinamis dan penghubung antar-node dalam pemrograman. Dengan memahami cara kerja *linked list*, kita dapat mengembangkan berbagai aplikasi yang membutuhkan pengelolaan data fleksibel, seperti sistem antrian, daftar tugas, maupun manajemen playlist seperti pada program ini.

D. Referensi

- W3Resource. (2020). *C++ String Exercises: Convert digit/number to words*.
- GeeksforGeeks. (2020). *Loops in C++ (for, while, do-while)*.