LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL IV
SINGLY LINKED LIST
(BAGIAN PERTAMA)

Disusun Oleh :
NAMA : KHOIRUL ADDIFA
NIM : 103112400172

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Singly linked list adalah struktur data dinamis yang efisien dan fleksibel untuk menyimpan dan memanipulasi data secara berantai. Operasi dasar seperti insert, delete, view, dan update menjadikannya pondasi penting dalam pemrograman berorientasi data. Konsep pointer menjadi inti utama dalam linked list karena berfungsi menghubungkan antar node secara logis di dalam memori.

Guided 1

Playlist.h

```h
#ifndef PLAYLIST_H
#define PLAYLIST_H

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

// Struktur Node untuk lagu
struct Lagu {
    string judul;
    string penyanyi;
    float durasi;
    Lagu* next;
};

// Kelas Playlist untuk mengelola linked list
class Playlist {
private:
    Lagu* head;

public:
    Playlist();
    void tambahDepan(string judul, string penyanyi, float durasi);
    void tambahBelakang(string judul, string penyanyi, float durasi);
    void tambahSetelahKe3(string judul, string penyanyi, float durasi);
    void hapusLagu(string judul);
    void tampilkan();
};

#endif
```

Playlist.cpp

```cpp
#include "Playlist.h"

// Konstruktor
Playlist::Playlist() {
```

```
    head = nullptr;
}

// Tambah lagu di awal playlist
void Playlist::tambahDepan(string judul, string penyanyi, float durasi) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, head};
    head = baru;
    cout << "Lagu \"" << judul << "\" berhasil ditambahkan di awal.\n";
}

// Tambah lagu di akhir playlist
void Playlist::tambahBelakang(string judul, string penyanyi, float
durasi) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, nullptr};
    if (head == nullptr) {
        head = baru;
    } else {
        Lagu* temp = head;
        while (temp->next != nullptr)
            temp = temp->next;
        temp->next = baru;
    }
    cout << "Lagu \"" << judul << "\" berhasil ditambahkan di akhir.\n";
}

// Tambah lagu setelah lagu ke-3
void Playlist::tambahSetelahKe3(string judul, string penyanyi, float
durasi) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, nullptr};
    Lagu* temp = head;
    int posisi = 1;

    while (temp != nullptr && posisi < 3) {
        temp = temp->next;
        posisi++;
    }

    if (temp == nullptr) {
        cout << "Playlist kurang dari 3 lagu. Lagu ditambahkan di
akhir.\n";
        tambahBelakang(judul, penyanyi, durasi);
        delete baru; // sudah ditambahkan lewat tambahBelakang
        return;
    }

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

```
    cout << "Lagu \"" << judul << "\" berhasil ditambahkan setelah lagu
ke-3.\n";
}

// Hapus lagu berdasarkan judul
void Playlist::hapusLagu(string judul) {
    if (head == nullptr) {
        cout << "Playlist kosong.\n";
        return;
    }

    Lagu* temp = head;
    Lagu* prev = nullptr;

    while (temp != nullptr && temp->judul != judul) {
        prev = temp;
        temp = temp->next;
    }

    if (temp == nullptr) {
        cout << "Lagu \"" << judul << "\" tidak ditemukan.\n";
        return;
    }

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

    delete temp;
    cout << "Lagu \"" << judul << "\" berhasil dihapus.\n";
}

// Tampilkan semua lagu
void Playlist::tampilkan() {
    if (head == nullptr) {
        cout << "Playlist kosong.\n";
        return;
    }

    cout << "\n=== Daftar Lagu Dalam Playlist ===\n";
    Lagu* temp = head;
    int no = 1;
    while (temp != nullptr) {
        cout << no++ << ". " << temp->judul << " - " << temp->penyanyi
             << " (" << temp->durasi << " menit)\n";
        temp = temp->next;
```

```
    }
    cout << "==================================\n";
}

main.cpp

#include "Playlist.h"

int main() {
    Playlist playlist;
    int pilihan;
    string judul, penyanyi;
    float durasi;

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

        switch (pilihan) {
        case 1:
            cout << "Judul lagu: "; getline(cin, judul);
            cout << "Penyanyi: "; getline(cin, penyanyi);
            cout << "Durasi (menit): "; cin >> durasi;
            playlist.tambahDepan(judul, penyanyi, durasi);
            break;
        case 2:
            cout << "Judul lagu: "; cin.ignore(); getline(cin, judul);
            cout << "Penyanyi: "; getline(cin, penyanyi);
            cout << "Durasi (menit): "; cin >> durasi;
            playlist.tambahBelakang(judul, penyanyi, durasi);
            break;
        case 3:
            cout << "Judul lagu: "; cin.ignore(); getline(cin, judul);
            cout << "Penyanyi: "; getline(cin, penyanyi);
            cout << "Durasi (menit): "; cin >> durasi;
            playlist.tambahSetelahKe3(judul, penyanyi, durasi);
            break;
        case 4:
            cout << "Masukkan judul lagu yang akan dihapus: ";
            cin.ignore();
```

```
            getline(cin, judul);
            playlist.hapusLagu(judul);
            break;
        case 5:
            playlist.tampilkan();
            break;
        case 0:
            cout << "Program selesai.\n";
            break;
        default:
            cout << "Pilihan tidak valid.\n";
        }
    } while (pilihan != 0);

    return 0;
}
```

Screenshots Output

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL
PS D:\Tel-U\semester 3\struktur data\laprak4> g++ main.cpp playlist.cpp
PS D:\Tel-U\semester 3\struktur data\laprak4> .\a.exe

=== MENU PLAYLIST LAGU ===
1. Tambah lagu di awal
2. Tambah lagu di akhir
3. Tambah lagu setelah lagu ke-3
4. Hapus lagu berdasarkan judul
5. Tampilkan seluruh lagu
0. Keluar
Pilih menu: 1
Judul lagu: happy
Penyanyi: Skinnyfabs
Durasi (menit): 2.53
Lagu "happy" berhasil ditambahkan di awal.
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL

=== MENU PLAYLIST LAGU ===
1. Tambah lagu di awal
2. Tambah lagu di akhir
3. Tambah lagu setelah lagu ke-3
4. Hapus lagu berdasarkan judul
5. Tampilkan seluruh lagu
0. Keluar
Pilih menu: 5

=== Daftar Lagu Dalam Playlist ===
1. Self Love (Spider-Man: Across the Spider-Verse) - Metro Boomin, Coi Leray (3.09 menit)
2. Sunflower - Spider-Man: Into the Spider-Verse - Post Malone, Swae Lee (2.38 menit)
3. happy - Skinnyfabs (2.53 menit)
================================
```

Deskripsi:

Program ini bertujuan menyimpan data maksimal 10 mahasiswa menggunakan array of struct dengan field nama, NIM, UTS, UAS, tugas, dan nilai akhir. Nilai akhir dihitung

otomatis melalui fungsi dengan rumus 0.3*UTS + 0.4*UAS + 0.3*Tugas. Program ini melatih mahasiswa dalam pengelolaan data terstruktur dan penerapan fungsi perhitungan.

Kesimpulan

Dalam praktikum Modul 4 ini, disimpulkan bahwa konsep Singly Linked List digunakan untuk mengelola data secara dinamis menggunakan pointer. Mahasiswa mempelajari cara membuat, menambah, menampilkan, menghapus, dan memperbarui elemen dalam list dengan operasi dasar seperti create list, insert, delete, dan view. Berbeda dengan array yang bersifat statis, linked list bersifat fleksibel karena ukurannya dapat bertambah atau berkurang sesuai kebutuhan. Secara keseluruhan, praktikum ini melatih pemahaman mahasiswa tentang penggunaan pointer dalam struktur data dinamis serta penerapan konsep Abstract Data Type (ADT) untuk membangun program yang efisien dan terstruktur.

B. Referensi

- Deitel, H. M., & Deitel, P. J. (2017). C++ How to Program (10th Edition). Pearson Education.
- Malik, D. S. (2018). Data Structures Using C++ (2nd Edition). Cengage Learning.
- Sahni, S. (2019). Data Structures, Algorithms, and Applications in C++. Universities Press.