LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 5
SINGLY LINKED LIST

Disusun Oleh :
NAMA : FANDIKA PRIMADANI
NIM : 103112400231

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Struktur data merupakan cara penyimpanan dan pengorganisasian data agar dapat digunakan secara efisien. Salah satu struktur data yang sering digunakan adalah *linked list*. *Linked list* adalah kumpulan elemen data yang disebut *node*, di mana setiap *node* berisi dua bagian utama: data itu sendiri dan pointer yang menunjuk ke *node* berikutnya. Salah satu jenis *linked list* adalah *single linked list*, yaitu daftar berantai di mana setiap *node* hanya memiliki satu pointer yang menunjuk ke *node* berikutnya. Berbeda dengan array yang menggunakan indeks untuk mengakses elemen, *linked list* memungkinkan penambahan dan penghapusan data secara dinamis tanpa harus menggeser elemen lainnya.

Dalam *single linked list*, terdapat beberapa operasi dasar seperti pembuatan list kosong, penambahan data di awal atau di akhir list, penyisipan data di posisi tertentu, penghapusan node, serta pencetakan isi list. Implementasi *single linked list* banyak digunakan dalam berbagai aplikasi, seperti manajemen antrian, sistem navigasi, dan juga pengelolaan playlist lagu. Pada program ini, *single linked list* digunakan untuk mengelola playlist lagu, di mana setiap lagu disimpan sebagai satu *node* dengan atribut berupa judul, penyanyi, dan durasi. Struktur ini memungkinkan pengguna untuk menambah, menghapus, dan menampilkan daftar lagu secara fleksibel dan efisien tanpa batasan ukuran tetap seperti pada array.

B. SOAL 1

main.cpp

```
#include "Playlist.h"
#include "Playlist.cpp"

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 Playlist\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: "; getline(cin, judul);
                cout << "Penyanyi: "; getline(cin, penyanyi);
                cout << "Durasi (menit): "; cin >> durasi;
                playlist.tambahBelakang(judul, penyanyi,
durasi);
                break;

            case 3:
                cout << "Judul lagu: "; 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 ingin
dihapus: ";
                getline(cin, judul);
                playlist.hapusLagu(judul);
                break;

            case 5:
                playlist.tampilkan();
                break;

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

```
            default:
                cout << "Pilihan tidak valid!\n";
        }

    } while (pilihan != 0);

    return 0;
}
```

Playlist.cpp

```
#include "Playlist.h"

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

Playlist::~Playlist() {
    Lagu* current = head;
    while (current != nullptr) {
        Lagu* temp = current;
        current = current->next;
        delete temp;
    }
}

void Playlist::tambahDepan(string judul, string penyanyi,
float durasi) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, nullptr};
    baru->next = head;
    head = baru;
    cout << "Lagu \"" << judul << "\" ditambahkan di awal
playlist.\n";
}

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 << "\" ditambahkan di akhir
playlist.\n";
}

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 != nullptr && count < 3) {
        temp = temp->next;
        count++;
    }

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

    baru->next = temp->next;
    temp->next = baru;
    cout << "Lagu \"" << judul << "\" ditambahkan setelah
lagu ke-3.\n";
}

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

```
    if (head->judul == judul) {
        Lagu* temp = head;
        head = head->next;
        delete temp;
        cout << "Lagu \"" << judul << "\" berhasil
dihapus.\n";
        return;
    }

    Lagu* temp = head;
    while (temp->next != nullptr && temp->next->judul !=
judul) {
        temp = temp->next;
    }

    if (temp->next == nullptr) {
        cout << "Lagu \"" << judul << "\" tidak
ditemukan.\n";
    } else {
        Lagu* hapus = temp->next;
        temp->next = hapus->next;
        delete hapus;
        cout << "Lagu \"" << judul << "\" berhasil
dihapus.\n";
    }
}

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";
```

```
}
```

Playlist.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
class Playlist {
private:
    Lagu* head;

public:
    Playlist();                    // Konstruktor
    ~Playlist();                   // Destruktor
    void tambahDepan(string, string, float);
    void tambahBelakang(string, string, float);
    void tambahSetelahKe3(string, string, float);
    void hapusLagu(string);
    void tampilkan();
};

#endif
```

Screenshots Output

1. Menu Tambah Lagu di awal

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS D:\SEMESTER 3\Pratikum Struktur Data\Pratikum M4 dan M5
5\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main

=== 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 Playlist
0. Keluar
Pilih menu: 1
Judul lagu: Kembali Pulang
Penyanyi: Feby Putri
Durasi (menit): 3.40
Lagu "Kembali Pulang" ditambahkan di awal playlist.
```

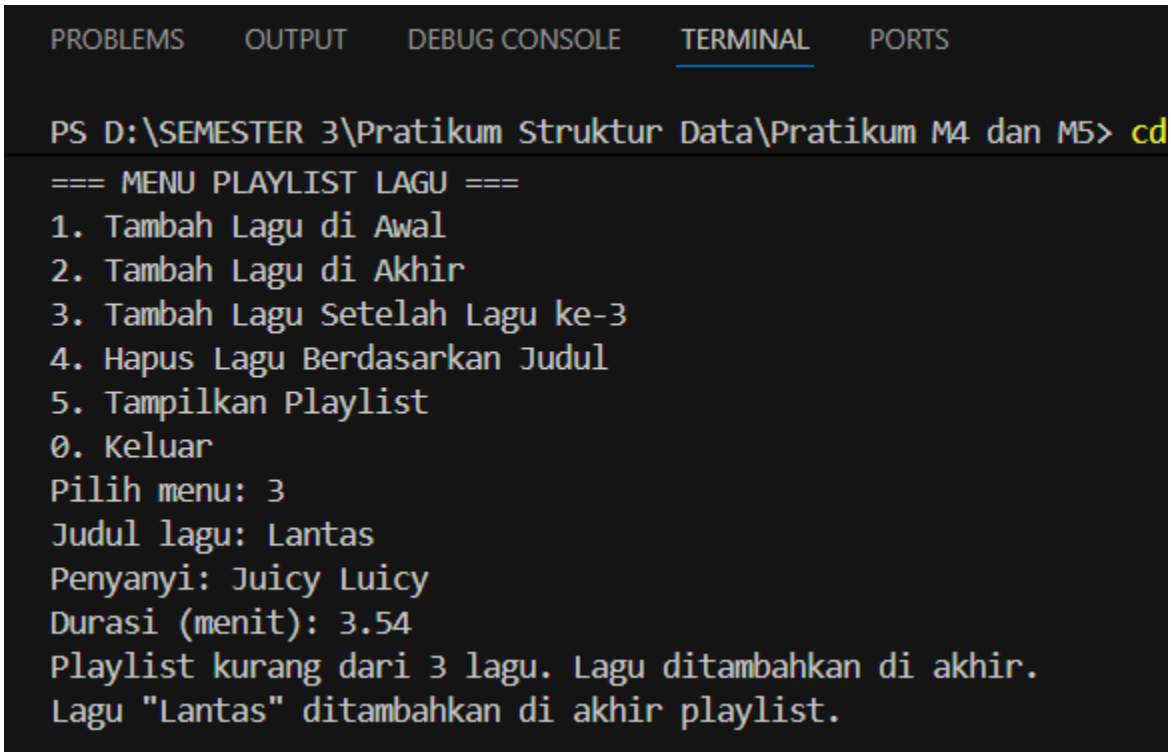
2. Menu Tambah Lagu di Akhir

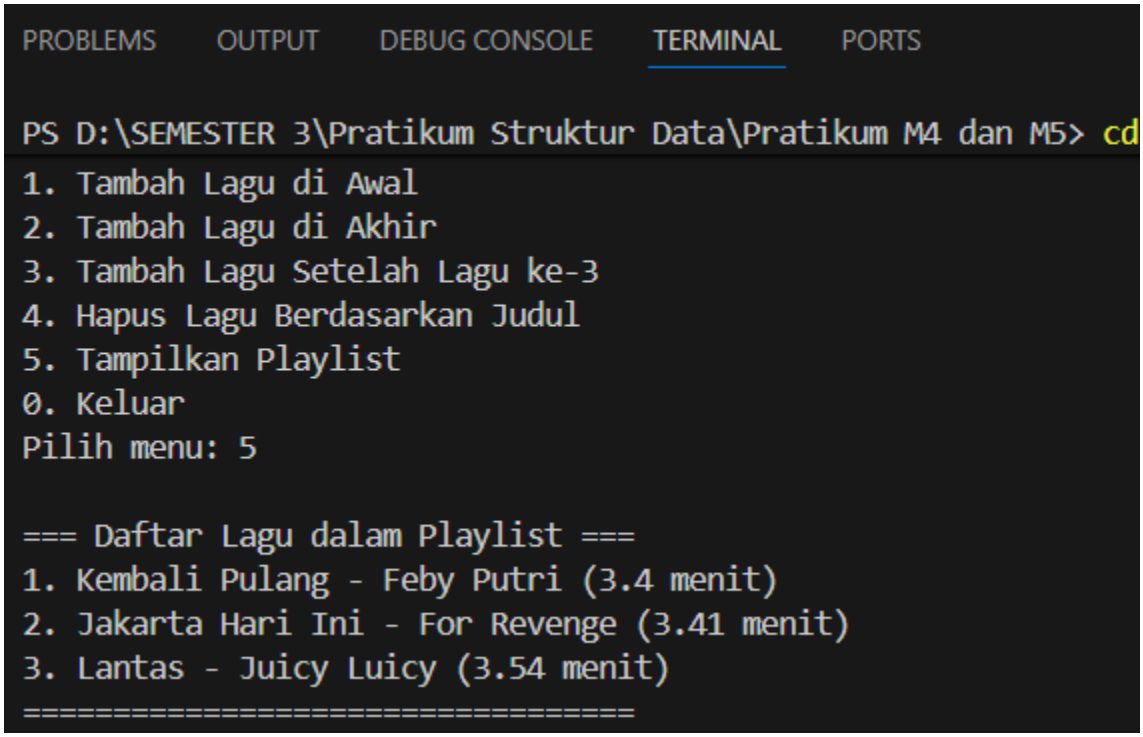
```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS D:\SEMESTER 3\Pratikum Struktur Data\Pratikum M4 dan M5> c

=== 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 Playlist
0. Keluar
Pilih menu: 2
Judul lagu: Jakarta Hari Ini
Penyanyi: For Revenge
Durasi (menit): 3.41
Lagu "Jakarta Hari Ini" ditambahkan di akhir playlist.
```

3. Menu Tambah Lagu setelah Lagu ke-3





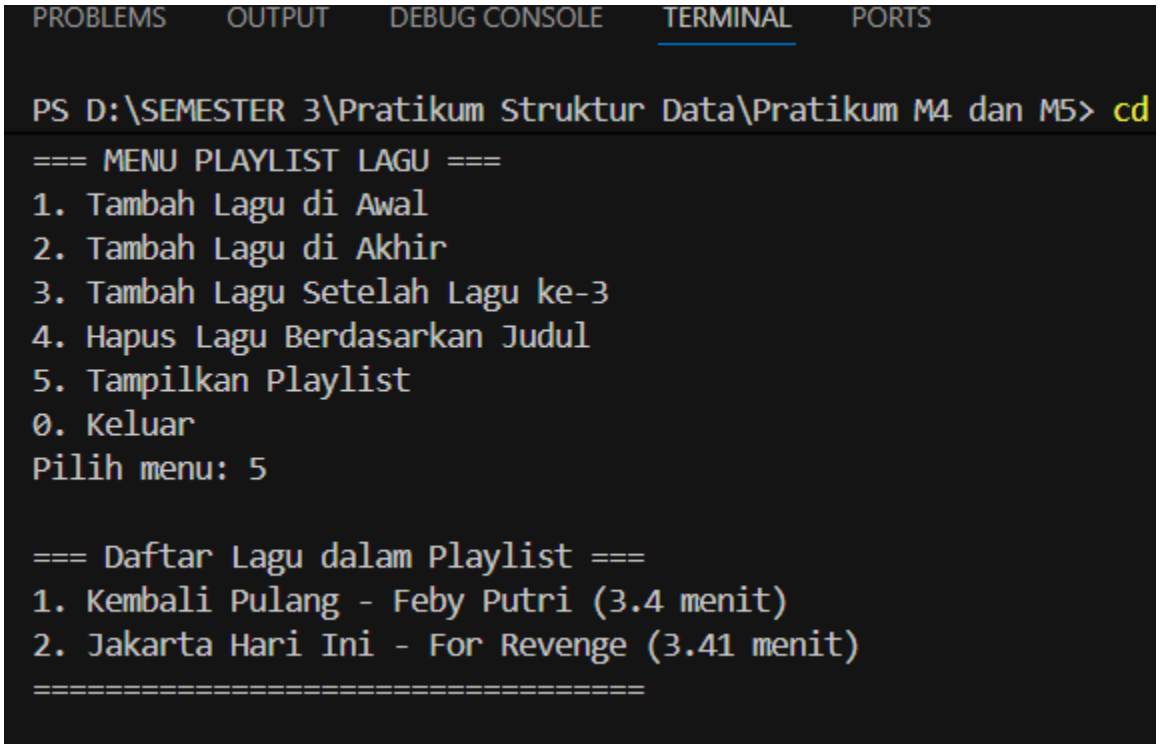
4. Hapus Lagu Berdasarkan Judul

```
=== 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 Playlist
0. Keluar
Pilih menu: 4
Masukkan judul lagu yang ingin dihapus: Lantas
Lagu "Lantas" berhasil dihapus.
```

```
PS D:\SEMESTER 3\Pratikum Struktur Data\Pratikum M4 dan M5>
=== 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 Playlist
0. Keluar
Pilih menu: 5

=== Daftar Lagu dalam Playlist ===
1. Kembali Pulang - Feby Putri (3.4 menit)
2. Jakarta Hari Ini - For Revenge (3.41 menit)
==================================
```

5. Tampilkan Playlist


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS D:\SEMESTER 3\Pratikum Struktur Data\Pratikum M4 dan M5> cd
=== 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 Playlist
0. Keluar
Pilih menu: 5
=== Daftar Lagu dalam Playlist ===
1. Kembali Pulang - Feby Putri (3.4 menit)
2. Jakarta Hari Ini - For Revenge (3.41 menit)
==========

Deskripsi:

Deskripsi: program diatas adalah program playlist lagu, jadi nanti kita bisa menambahkan lagu, dan tidak hanya menambahkan saja ada beberapa menu nya seperti tambah lagu di awal, tambah lagu di akhir, tambah lagu setelah lagu ketiga, hapus lagu bedasarkan judul, tampilkan seluruh lagi, dan keluar, jadi misal kita ingin menambahklan lagu makam pilih menu pertama nanti kita disuruh memasukan judul, lalu penyanyi nya , dan berapa menit durasi nya, dan jika memilih menu nomer 5 maka program akan menampilkan seluruh data lagu yang pernah dimasukan ke dalam data

C. Kesimpulan

Dari percobaan yang dilakukan, dapat disimpulkan bahwa *single linked list* merupakan struktur data yang efisien untuk menyimpan dan mengelola data secara dinamis. Dengan menggunakan *linked list*, proses penambahan maupun penghapusan data dapat dilakukan tanpa harus memindahkan elemen lain seperti pada array. Program playlist lagu yang dibuat menunjukkan penerapan nyata dari konsep *single linked list*, di mana pengguna dapat menambahkan lagu di awal, di akhir, atau setelah lagu ke-3, serta menghapus lagu berdasarkan judul dan menampilkan seluruh daftar lagu. Implementasi ini membantu memahami cara kerja pointer dan hubungan antar-node dalam membentuk suatu rangkaian data yang saling terhubung.

D. Referensi

"ISO/IEC TS 19841:2015". International Organization for Standardization. Archived from the original on 15 January 2019. Retrieved 15 February 2019.