LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 4
PENGENALAN CODE BLOCKS

Disusun Oleh :
NAMA :Muhamad Ilham Syahid
NIM :103112400155

Dosen
WAHYU ANDI SAPUTRA

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Singly Linked List adalah struktur data dinamis yang terdiri dari kumpulan node yang saling terhubung menggunakan pointer. Setiap node menyimpan data dan satu pointer (next) yang menunjuk ke node berikutnya. Node terakhir menunjuk ke NULL, menandakan akhir dari list.

Linked list menggunakan pointer karena bersifat fleksibel, mudah dikembangkan, dan lebih efisien dalam proses penyisipan serta penghapusan data dibandingkan array yang bersifat statis.

Operasi dasar pada Singly Linked List meliputi pembuatan list (create), alokasi dan dealokasi memori, penyisipan data (insert first, insert last, insert after), penghapusan data (delete first, delete last, delete after), penelusuran/penampilan data (traversal), serta pembaruan data (update).

Dengan konsep Abstract Data Type (ADT), Singly Linked List diimplementasikan secara terpisah antara definisi struktur data dan fungsi operasinya sehingga program menjadi lebih terstruktur dan mudah dipelihara.

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;
};

typedef Lagu infotype;
typedef struct Node* address;

struct Node {
    infotype info;
    address next;
};
```

```
struct List {
    address first;
};

/* Prototype fungsi */
void createList(List &L);
address alokasi(infotype X);
void insertFirst(List &L, address P);
void insertLast(List &L, address P);
void insertAfterKe3(List &L, address P);
void deleteByJudul(List &L, string judul);
void printPlaylist(List L);

#endif
```

Guided 2

Playlist.cpp

```
#include "Playlist.h"

/* Membuat list kosong */
void createList(List &L) {
    L.first = NULL;
}

/* Alokasi node baru */
address alokasi(infotype X) {
    address P = new Node;
    P->info = X;
    P->next = NULL;
    return P;
}

/* Insert di awal */
void insertFirst(List &L, address P) {
    P->next = L.first;
    L.first = P;
}

/* Insert di akhir */
void insertLast(List &L, address P) {
    if (L.first == NULL) {
        L.first = P;
    } else {
```

```
        address Q = L.first;
        while (Q->next != NULL) {
            Q = Q->next;
        }
        Q->next = P;
    }
}

/* Insert setelah node ke-3 */
void insertAfterKe3(List &L, address P) {
    address Q = L.first;
    int i = 1;

    while (Q != NULL && i < 3) {
        Q = Q->next;
        i++;
    }

    if (Q != NULL) {
        P->next = Q->next;
        Q->next = P;
    }
}

/* Hapus lagu berdasarkan judul */
void deleteByJudul(List &L, string judul) {
    address P = L.first;
    address Prev = NULL;

    while (P != NULL && P->info.judul != judul) {
        Prev = P;
        P = P->next;
    }

    if (P != NULL) {
        if (Prev == NULL) {
            L.first = P->next;
        } else {
            Prev->next = P->next;
        }
        delete P;
    }
}

/* Tampilkan seluruh playlist */
void printPlaylist(List L) {
    address P = L.first;
    int no = 1;
```

```
    if (P == NULL) {
        cout << "Playlist kosong\n";
        return;
    }

    while (P != NULL) {
        cout << no++ << ". "
             << P->info.judul << " - "
             << P->info.penyanyi
             << " (" << P->info.durasi << " menit)" << endl;
        P = P->next;
    }
}
```

Screenshots Output

Deskripsi:

Guided 3

Main.cpp

```
  #include "Playlist.h"

int main() {
    List Playlist;
    createList(Playlist);

    infotype lagu;

    lagu = {"Laskar Pelangi", "Nidji", 4.2};
    insertFirst(Playlist, alokasi(lagu));

    lagu = {"Separuh Aku", "NOAH", 4.5};
    insertLast(Playlist, alokasi(lagu));

    lagu = {"Akad", "Payung Teduh", 4.1};
    insertLast(Playlist, alokasi(lagu));

    lagu = {"Hati-Hati di Jalan", "Tulus", 4.0};
    insertAfterKe3(Playlist, alokasi(lagu));
```

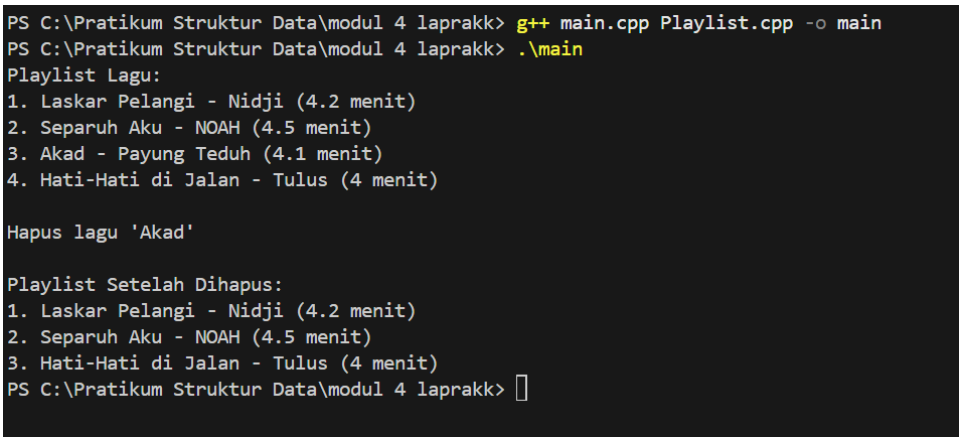
```
    cout << "Playlist Lagu:" << endl;
    printPlaylist(Playlist);

    cout << "\nHapus lagu 'Akad'\n";
    deleteByJudul(Playlist, "Akad");

    cout << "\nPlaylist Setelah Dihapus:" << endl;
    printPlaylist(Playlist);

    return 0;
}
```

Screenshots Output



Deskripsi:

Program ini mengelola playlist lagu menggunakan struktur data **Single Linked List** dalam bahasa C++. Setiap lagu disimpan dalam sebuah node yang berisi judul, penyanyi, dan durasi lagu serta pointer ke node berikutnya. Program menyediakan operasi untuk menambah lagu di awal, di akhir, dan setelah lagu ke-3, menghapus lagu berdasarkan judul, serta menampilkan seluruh isi playlist. Struktur ini bersifat dinamis sehingga memudahkan penambahan dan penghapusan data.

C. Kesimpulan

Program playlist lagu berbasis **Single Linked List** berhasil mengelola data lagu secara dinamis. Dengan struktur ini, penambahan dan penghapusan lagu dapat dilakukan dengan mudah tanpa bergantung pada ukuran tetap seperti array. Implementasi operasi insert, delete, dan traversal menunjukkan bahwa Single Linked List efektif digunakan untuk mengelola playlist lagu yang dapat berubah-ubah isinya.

D. Referensi

Modul Praktikum Struktur Data – Modul 4: Singly Linked List