LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 4&5
SINGLY LINKED LIST

Disusun Oleh :

NAMA : Loh Suwargi Nitis Hamengku Bintang

NIM : 103112400116

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Singly Linked List atau daftar tertaut tunggal merupakan salah satu struktur data dinamis yang banyak digunakan dalam pemrograman. Struktur ini tersusun atas sejumlah node, di mana setiap node memiliki dua komponen utama, yaitu data dan pointer yang menunjuk ke node berikutnya. Node pertama disebut *head*, sedangkan node terakhir menunjuk ke NULL, menandakan akhir dari daftar. Tidak seperti array yang menyimpan data secara bersebelahan di memori, Singly Linked List menyimpan elemen secara terpisah, namun tetap saling terhubung melalui pointer. Hal ini menjadikannya lebih fleksibel dalam pengelolaan memori karena ukuran list dapat berubah secara dinamis sesuai kebutuhan program (Almadhoun & Parham-Mocello, 2023).

Operasi dasar yang umum dilakukan pada Singly Linked List meliputi penambahan (*insertion*), penghapusan (*deletion*), dan penelusuran (*traversal*) data. Penambahan di awal list memiliki kompleksitas waktu O(1), sedangkan penyisipan di akhir atau pencarian berdasarkan posisi biasanya membutuhkan waktu O(n) karena harus menelusuri seluruh node dari *head* hingga posisi target. Struktur ini memiliki kelebihan seperti efisiensi dalam penyisipan atau penghapusan elemen di awal, serta fleksibilitas dalam alokasi memori. Namun, kelemahannya terletak pada tidak tersedianya akses acak terhadap data dan kebutuhan memori tambahan untuk menyimpan pointer di setiap node. Beberapa penelitian terbaru menunjukkan bahwa meskipun linked list menawarkan fleksibilitas tinggi, dalam situasi tertentu struktur berbasis array justru lebih cepat dan efisien (Sonntag & Colnet, 2023).

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

Guided 1

```
#ifndef SINGLYLIST_H_INCLUDED
#define SINGLYLIST_H_INCLUDED
#include <iostream>
#define Nil NULL


typedef int infotype;
typedef struct ElmList *address;


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

```
struct List{
    address First;
};

//Deklarasi Prosedur dan FFungsi Primitif
void CreateList(List &L);
address alokasi(infotype x);
void dealokasi( address &p);
void insertFirst(List &L, address P);
void insertLast(List &L, address P);
void printInfo(List L);

#endif
```

```
#include "Singlylist.h"

void CreateList(List &L) {
    L.First = Nil;
}

address alokasi(infotype x) {
    address P = new ElmList;
    P->info = x;
    P->next = Nil;
    return P;
}

void dealokasi(address &P) {
```

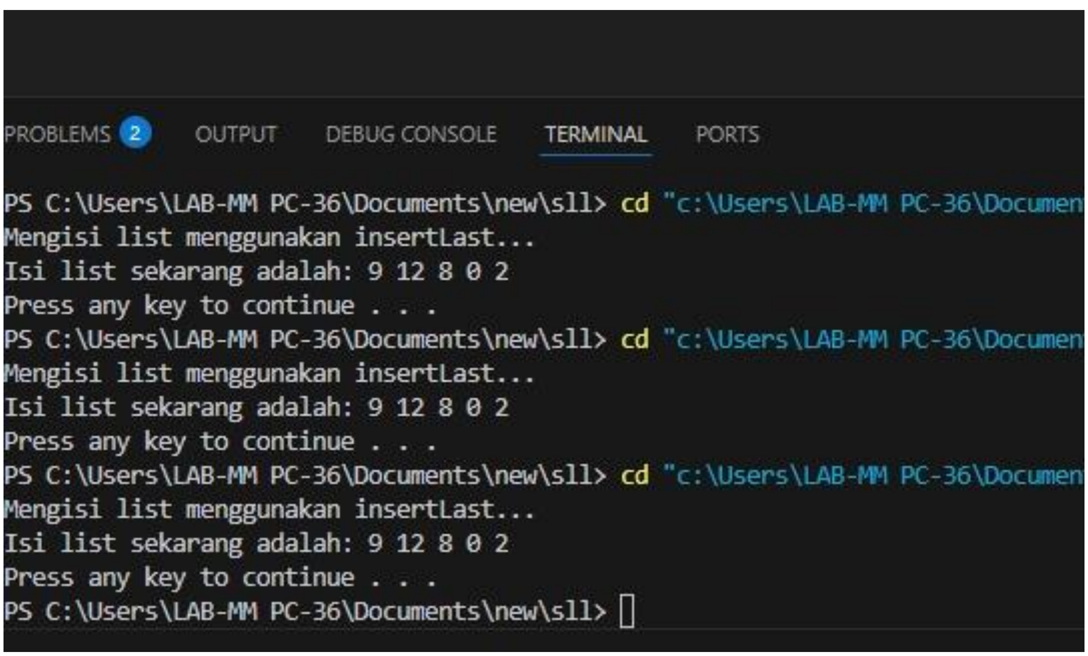
```
    delete P;
}

void insertFirst(List &L, address P) {
    P->next = L.First;
    L.First = P;
}

void insertLast(List &L, address P) {
    // Jika list kosong, insertLast sama dengan insertFisrt
    if (L.First == Nil) {
        insertFirst(L, P);
    } else {
        // Jika list
        address Last = L.First;
        while (Last->next != Nil) {
            Last = Last->next;
        }
        Last->next = P;
    }
}

void printInfo(List L) {
    address P = L.First;
    if (P == Nil) {
        std::cout << "List Kosong!" << std::endl;
    } else {
        while (P != Nil) {
            std::cout << P->info << " ";
            P = P->next;
```

```
        }
        std::cout << std::endl;
    }
}
```

```
#include <iostream>
#include <cstdlib>
#include "Singlylist.h"
#include "Singlylist.cpp"

using namespace std;

int main() {
    List L;
    address P; // Cukup satu pointer untuk digunakan berulang kali

    CreateList(L);

    cout << "Mengisi list menggunakan insertLast..." << endl;

    // Mengisi list sesuai urutan
    P = alokasi(9);
    insertLast(L, P);

    P = alokasi(12);
    insertLast(L, P);

    P = alokasi(8);
    insertLast(L, P);
```

```
        P = alokasi(0);
        insertLast(L, P);

        P = alokasi(2);
        insertLast(L, P);

        cout << "Isi list sekarang adalah: ";
        printInfo(L);

        system("pause");
        return 0;

    }
```

Screenshots Output



Deskripsi:Program terdiri dari tiga file utama, yaitu Singlylist.h, Singlylist.cpp, dan main.cpp. Pada file Singlylist.h didefinisikan struktur ElmList yang berisi nilai (info) dan pointer (next), serta fungsi-fungsi dasar seperti CreateList, insertFirst, insertLast, dan printInfo.

File Singlylist.cpp berisi implementasi fungsi-fungsi tersebut, di mana CreateList digunakan untuk membuat list kosong, insertFirst dan insertLast menambah elemen di awal atau akhir list, dan printInfo menampilkan seluruh data. Sementara itu, file main.cpp menjadi program utama

yang membuat list, menambahkan beberapa nilai (9, 12, 8, 0, 2) ke dalamnya, lalu menampilkan hasilnya.

D. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Unguided 1

```
#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();
    ~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 hapusBerdasarkanJudul(string judul);
    void tampilkan();
};

#endif
```

```
#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::tambahAwal(string judul, string penyanyi, float durasi) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, head};
    head = baru;
    cout << "Lagu \"" << judul << "\" berhasil ditambahkan di awal playlist.\n";
}

void Playlist::tambahAkhir(string judul, string penyanyi, float durasi) {
```

```
    Lagu* baru = new Lagu{judul, penyanyi, durasi, nullptr};

    if (head == nullptr) {
        head = baru;
    } else {
        Lagu* current = head;
        while (current->next != nullptr) {
            current = current->next;
        }
        current->next = baru;
    }
    cout << "Lagu \"" << judul << "\" berhasil ditambahkan di akhir playlist.\n";
}

void Playlist::tambahSetelahKe3(string judul, string penyanyi, float durasi) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, nullptr};
    Lagu* current = head;
    int posisi = 1;

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

    if (current == nullptr) {
        cout << "Playlist kurang dari 3 lagu. Lagu ditambahkan di akhir.\n";
        tambahAkhir(judul, penyanyi, durasi);
    } else {
        baru->next = current->next;
        current->next = baru;
```

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

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

    Lagu* current = head;
    Lagu* prev = nullptr;

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

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

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

    delete current;
```

```
    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* current = head;
    int i = 1;
    while (current != nullptr) {
        cout << i++ << ". Judul  : " << current->judul << endl;
        cout << "   Penyanyi : " << current->penyanyi << endl;
        cout << "   Durasi   : " << current->durasi << " menit\n";
        current = current->next;
    }
    cout << "==================================\n";
}
```

```
#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 playlist\n";
        cout << "0. Keluar\n";
        cout << "Pilih menu: ";
        cin >> pilihan;
        cin.ignore();

        switch (pilihan) {
            case 1:
                cout << "Masukkan judul lagu: ";
                getline(cin, judul);
                cout << "Masukkan nama penyanyi: ";
                getline(cin, penyanyi);
                cout << "Masukkan durasi (menit): ";
                cin >> durasi;
                playlist.tambahAwal(judul, penyanyi, durasi);
                break;

            case 2:
                cout << "Masukkan judul lagu: ";
                getline(cin, judul);
                cout << "Masukkan nama penyanyi: ";
                getline(cin, penyanyi);
                cout << "Masukkan durasi (menit): ";
                cin >> durasi;
```

```
            playlist.tambahAkhir(judul, penyanyi, durasi);
            break;

        case 3:
            cout << "Masukkan judul lagu: ";
            getline(cin, judul);
            cout << "Masukkan nama penyanyi: ";
            getline(cin, penyanyi);
            cout << "Masukkan durasi (menit): ";
            cin >> durasi;
            playlist.tambahSetelahKe3(judul, penyanyi, durasi);
            break;

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

        case 5:
            playlist.tampilkan();
            break;

        case 0:
            cout << "Terima kasih telah menggunakan program playlist.\n";
            break;

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

```
    return 0;
}
```

Screenshots Output

Deskripsi: Program ini dibuat untuk mengelola datar lagu. Setiap lagu disimpan node yang berisi judul, penyanyi, dan durasi lagu, lalu saling ke satu arah melalui pointer next. Program ini dibagi menjadi tiga file utama aitu Playlist.h, Playylist.cpp, main.cpp agar lebih tersutktur agar lebih terstruktur.

Fitur yang tersedia meliputi penambahan lagu di awal, di akhir, atau setelah lagu ke-3, penghapusan lagu berdasarkan judul, serta penampilan seluruh daftar lagu. Semua operasi dilakukan secara interaktif melalui menu di terminal. Secara keseluruhan, program ini menunjukkan penerapan konsep Single Linked List untuk mengelola data yang berubah-ubah secara efisien, sekaligus membantu memahami cara kerja pointer dan hubungan antar-node dalam struktur data dinamis.

E. Kesimpulan

Kedua program, yaitu Playlist Lagu dan Singly Linked List, sama-sama menerapkan konsep dasar struktur data Single Linked List yang memungkinkan pengelolaan data secara dinamis menggunakan pointer. Program Singly Linked List berfungsi sebagai contoh dasar yang menunjukkan cara kerja node, pointer, serta operasi seperti penambahan dan penampilan data. Sementara itu, program Playlist Lagu mengembangkan konsep tersebut ke dalam aplikasi yang lebih nyata, di mana setiap node merepresentasikan sebuah lagu dengan atribut judul, penyanyi, dan durasi. Pembagian program ke dalam beberapa file seperti header, implementasi, dan main juga menunjukkan penerapan prinsip modularitas agar kode lebih rapi dan mudah dikelola.

F. Referensi

Almadhoun, E., & Parham-Mocello, J. (2023). Students' difficulties with inserting and deleting nodes in a singly linked list in the C programming language. ***Journal of Computer Languages*****,** ***74*****, 101184.**

Sonntag, B., & Colnet, D. (2023). RIP Linked List. ***arXiv preprint arXiv:2306.06942*****.**