LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL IV
SINGLY LINKED LIST

Disusun Oleh :

NAMA : RISKY CAHAYU

NIM : 103112430121

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Single Linked List merupakan salah satu struktur data dinamis yang terdiri atas rangkaian elemen yang disebut node, di mana setiap node menyimpan dua bagian utama yaitu data dan pointer (next) yang menunjuk ke node berikutnya. Struktur ini memungkinkan penyimpanan dan pengelolaan data secara efisien karena ukurannya dapat berubah secara dinamis selama program berjalan, berbeda dengan array yang memiliki ukuran tetap. Operasi dasar pada Single Linked List meliputi penambahan data di awal, tengah, atau akhir daftar, penghapusan data, serta penelusuran seluruh elemen dari node pertama (head) hingga node terakhir (tail) yang menunjuk ke NULL. Struktur ini banyak digunakan dalam berbagai aplikasi seperti sistem playlist lagu, antrian, dan implementasi stack, karena memungkinkan penyisipan dan penghapusan data tanpa perlu menggeser elemen lain. Meskipun demikian, Single Linked List memiliki kelemahan yaitu tidak dapat mengakses elemen secara langsung dan membutuhkan memori tambahan untuk penyimpanan pointer. (Sumber: Wikipedia, “Linked List,” 2024; Knuth, D.E., The Art of Computer Programming, 1998).

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

Guided 1

Kode SinglyList.h

```
// Risky Cahayu
// 103112430121

#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 Fungsi 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
```

Kode SinglyList.cpp

```
// Risky Cahayu
// 103112430121

#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 ) {
    if (L.First == Nil) {
        // Jika list kosong, insertLast sama dengan insertFirst
        insertFirst(L,P);
    } else {
        // Jika list tidak kosong, cari elemen terakhir
        address Last = L.First;
        while (Last->next != Nil) {
            Last = Last->next;
        }
        // Sambungkan elemen terakhir ke elemen baru (P)
        Last->next = P;
    }
}
```

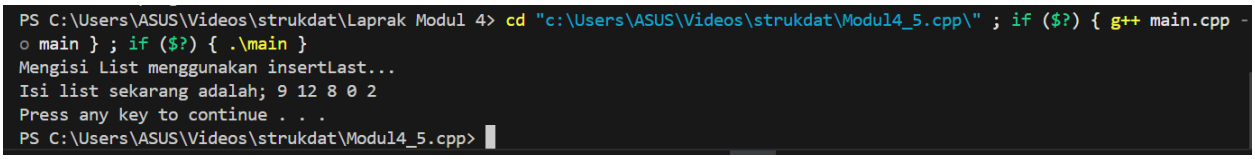
```
void printInfo(List L) {
    address P = L.First;
    if (P == Nil) {
        std::cout << "List Kosong!" << std::endl;
    } else {
        while (P != Nil) {
        std::cout << P ->info << " ";
        P = P ->next;
    }
    std::cout << std::endl;
    }
}
```

Kode main.cpp

```
// Risky Cahayu
// 103112430121

#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 di atas mengimplementasikan singly linked list (daftar berantai tunggal) dalam C++, ada file header yang mendefinisikan tipe data (infotype = int), struktur node (ElmList) dan struktur list (List) beserta operasi dasar, lalu file implementasi yang menyediakan fungsi untuk membuat list kosong (CreateList), membuat node baru di heap (alokasi), membebaskan memori node (dealokasi), menambah node di depan (insertFirst) atau di belakang (insertLast), dan menampilkan isi list (printInfo). Intinya, kode ini menunjukkan bagaimana menyimpan sekumpulan nilai secara dinamis menggunakan pointer — setiap nilai berada di node yang menunjuk ke node berikutnya — sehingga ukuran list bisa bertambah tanpa menentukan ukuran awal. Program main membuat list kosong, lalu menambahkan beberapa angka (9, 12, 8, 0, 2) menggunakan insertLast sehingga urutan elemen terjaga, lalu mencetak isi list ke layar; fungsi alokasi memastikan setiap node dibuat di memori heap dan dealokasi tersedia untuk membebaskannya ketika tidak dipakai. Kegunaan praktisnya: struktur ini berguna bila Anda butuh koleksi data yang sering diubah ukurannya (penambahan/penghapusan) dengan overhead memori yang fleksibel, misalnya antrian sederhana, riwayat undo, atau representasi graf/struktur dinamis lainnya.

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

Unguided 1

Kode Playlist.h

```
// Risky Cahayu
// 103112430121

#ifndef PLAYLIST_H
#define PLAYLIST_H

#include <iostream>
```

```
#include <string>
#include <algorithm> // untuk transform()
using namespace std;

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

class Playlist {
private:
    Lagu* head;
    string toLowerCase(const string& str); // helper function

public:
    Playlist();
    void tambahDepan(const string& judul, const string& penyanyi, float
durasi);
    void tambahBelakang(const string& judul, const string& penyanyi,
float durasi);
    void tambahSetelahKe3(const string& judul, const string& penyanyi,
float durasi);
    void hapusBerdasarkanJudul(const string& judul);
    void tampilkanPlaylist();
};

#endif
```

Kode Playlist.cpp

```
// Risky Cahayu
// 103112430121

#include "Playlist.h"

// Fungsi bantu untuk ubah ke huruf kecil semua
string Playlist::toLowerCase(const string& str) {
    string lower = str;
    transform(lower.begin(), lower.end(), lower.begin(), ::tolower);
    return lower;
}

// Konstruktor
Playlist::Playlist() {
    head = nullptr;
}
```

```
// Tambah lagu di depan
void Playlist::tambahDepan(const string& judul, const string& penyanyi,
float durasi) {
    Lagu* laguBaru = new Lagu{judul, penyanyi, durasi, head};
    head = laguBaru;
}

// Tambah lagu di belakang
void Playlist::tambahBelakang(const string& judul, const string&
penyanyi, float durasi) {
    Lagu* laguBaru = new Lagu{judul, penyanyi, durasi, nullptr};
    if (!head) {
        head = laguBaru;
        return;
    }
    Lagu* temp = head;
    while (temp->next) {
        temp = temp->next;
    }
    temp->next = laguBaru;
}

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

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

    if (!temp) {
        cout << "Playlist kurang dari 3 lagu, tidak bisa menambah setelah
lagu ke-3.\n";
        return;
    }

    Lagu* laguBaru = new Lagu{judul, penyanyi, durasi, temp->next};
    temp->next = laguBaru;
}

// Hapus lagu berdasarkan judul (tidak peka huruf besar-kecil)
void Playlist::hapusBerdasarkanJudul(const string& judul) {
    if (!head) {
        cout << "Playlist kosong.\n";
```

```
        return;
    }

    string target = toLowerCase(judul);
    Lagu* temp = head;
    Lagu* prev = nullptr;

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

    if (!temp) {
        cout << "Lagu dengan judul '" << judul << "' tidak ditemukan.\n";
        return;
    }

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

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

// Tampilkan seluruh lagu
void Playlist::tampilkanPlaylist() {
    if (!head) {
        cout << "Playlist kosong.\n";
        return;
    }

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

Kode main.cpp

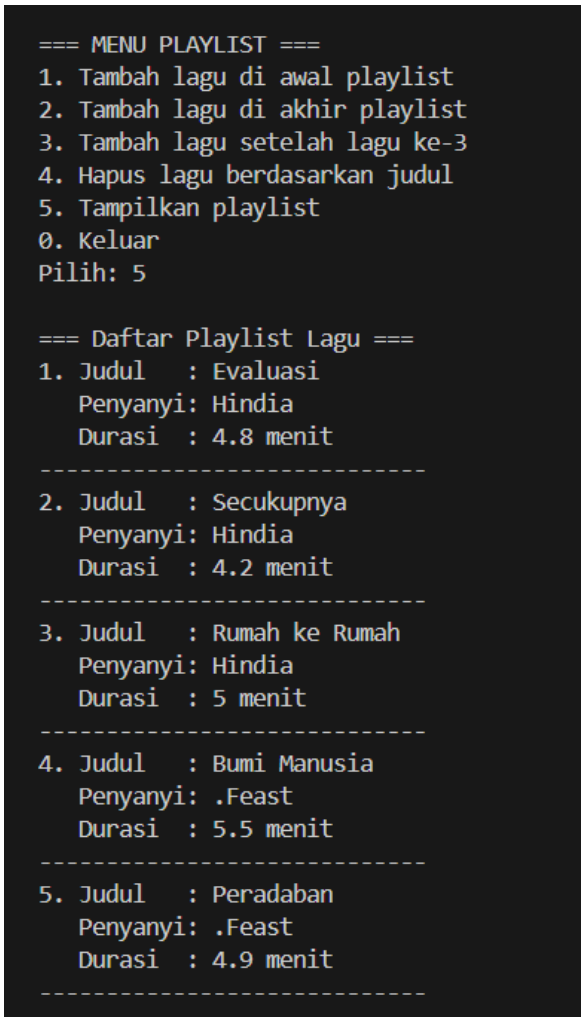
```cpp
// Risky Cahayu
// 103112430121

#include "Playlist.h"

int main() {
    Playlist myPlaylist;

    // Data dummy lagu
    myPlaylist.tambahBelakang("Evaluasi", "Hindia", 4.8);
    myPlaylist.tambahBelakang("Secukupnya", "Hindia", 4.2);
    myPlaylist.tambahBelakang("Rumah ke Rumah", "Hindia", 5.0);
    myPlaylist.tambahBelakang("Bumi Manusia", ".Feast", 5.5);
    myPlaylist.tambahBelakang("Peradaban", ".Feast", 4.9);

    int pilihan;
    string judul, penyanyi;
    float durasi;

    do {
        cout << "\n=== MENU PLAYLIST ===\n";
        cout << "1. Tambah lagu di awal playlist\n";
        cout << "2. Tambah lagu di akhir playlist\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: ";
        cin >> pilihan;
        cin.ignore();

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

        case 2:
            cout << "Masukkan judul lagu: "; getline(cin, judul);
            cout << "Masukkan penyanyi: "; getline(cin, penyanyi);
            cout << "Masukkan durasi (menit): "; cin >> durasi;
            myPlaylist.tambahBelakang(judul, penyanyi, durasi);
            break;

        case 3:
            cout << "Masukkan judul lagu: "; getline(cin, judul);
```

```
            cout << "Masukkan penyanyi: "; getline(cin, penyanyi);
            cout << "Masukkan durasi (menit): "; cin >> durasi;
            myPlaylist.tambahSetelahKe3(judul, penyanyi, durasi);
            break;

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

        case 5:
            myPlaylist.tampilkanPlaylist();
            break;

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

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

    } while (pilihan != 0);

    return 0;
}
```

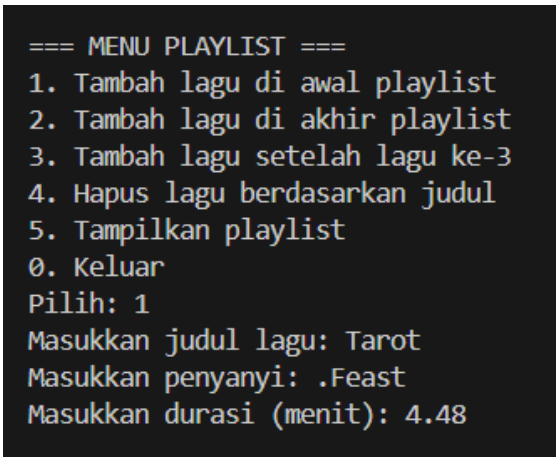
Screenshots Output

```
PS C:\Users\ASUS\Videos\strukdat\Laprak Modul 4> g++ main.cpp Playlist.cpp -o main
PS C:\Users\ASUS\Videos\strukdat\Laprak Modul 4> .\main

=== MENU PLAYLIST ===
1. Tambah lagu di awal playlist
2. Tambah lagu di akhir playlist
3. Tambah lagu setelah lagu ke-3
4. Hapus lagu berdasarkan judul
5. Tampilkan playlist
0. Keluar
```

Menampilkan Output berupa fitur playlist



Menampilkan Playlist dari output menu ke 5



Menampilkan output pada menu 1 yaitu menambahkan lagu di awal playlist

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

=== Daftar Playlist Lagu ===
1. Judul   : Tarot
   Penyanyi: .Feast
   Durasi  : 4.48 menit
----------------------------
2. Judul   : Evaluasi
   Penyanyi: Hindia
   Durasi  : 4.8 menit
----------------------------
3. Judul   : Secukupnya
   Penyanyi: Hindia
   Durasi  : 4.2 menit
----------------------------
4. Judul   : Rumah ke Rumah
   Penyanyi: Hindia
   Durasi  : 5 menit
----------------------------
5. Judul   : Bumi Manusia
   Penyanyi: .Feast
   Durasi  : 5.5 menit
----------------------------
6. Judul   : Peradaban
   Penyanyi: .Feast
   Durasi  : 4.9 menit
----------------------------
```

Mengecek Kembali output pada menu 5 untuk menampilkan playlist yang ditambah diawal lagu

```
=== MENU PLAYLIST ===
1. Tambah lagu di awal playlist
2. Tambah lagu di akhir playlist
3. Tambah lagu setelah lagu ke-3
4. Hapus lagu berdasarkan judul
5. Tampilkan playlist
0. Keluar
Pilih: 2
Masukkan judul lagu: Arteri
Masukkan penyanyi: .Feast
Masukkan durasi (menit): 4.30
```

Menampilkan output pada menu 2 yaitu menambahkan lagu di akhir playlist

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

=== Daftar Playlist Lagu ===
1. Judul   : Tarot
   Penyanyi: .Feast
   Durasi  : 4.48 menit
----------------------------
2. Judul   : Evaluasi
   Penyanyi: Hindia
   Durasi  : 4.8 menit
----------------------------
3. Judul   : Secukupnya
   Penyanyi: Hindia
   Durasi  : 4.2 menit
----------------------------
4. Judul   : Rumah ke Rumah
   Penyanyi: Hindia
   Durasi  : 5 menit
----------------------------
5. Judul   : Bumi Manusia
   Penyanyi: .Feast
   Durasi  : 5.5 menit
----------------------------
6. Judul   : Peradaban
   Penyanyi: .Feast
   Durasi  : 4.9 menit
----------------------------
7. Judul   : Arteri
   Penyanyi: .Feast
   Durasi  : 4.3 menit
----------------------------
```

Mengecek Kembali output pada menu 5 untuk menampilkan playlist yang ditambah diakhir lagu

```
=== MENU PLAYLIST ===
1. Tambah lagu di awal playlist
2. Tambah lagu di akhir playlist
3. Tambah lagu setelah lagu ke-3
4. Hapus lagu berdasarkan judul
5. Tampilkan playlist
0. Keluar
Pilih: 3
Masukkan judul lagu: kids
Masukkan penyanyi: Hindia
Masukkan durasi (menit): 3.02
```

Menampilkan output pada menu 2 yaitu menambahkan lagu di akhir playlist

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

=== Daftar Playlist Lagu ===
1. Judul   : Tarot
   Penyanyi: .Feast
   Durasi  : 4.48 menit
----------------------------
2. Judul   : Evaluasi
   Penyanyi: Hindia
   Durasi  : 4.8 menit
----------------------------
3. Judul   : Secukupnya
   Penyanyi: Hindia
   Durasi  : 4.2 menit
----------------------------
4. Judul   : kids
   Penyanyi: Hindia
   Durasi  : 3.02 menit
----------------------------
5. Judul   : Rumah ke Rumah
   Penyanyi: Hindia
   Durasi  : 5 menit
----------------------------
6. Judul   : Bumi Manusia
   Penyanyi: .Feast
   Durasi  : 5.5 menit
----------------------------
7. Judul   : Peradaban
   Penyanyi: .Feast
   Durasi  : 4.9 menit
----------------------------
8. Judul   : Arteri
   Penyanyi: .Feast
   Durasi  : 4.3 menit
----------------------------
```

Mengecek Kembali output pada menu 5 untuk menampilkan playlist yang ditambah setelah lagu ke-3

```
=== MENU PLAYLIST ===
1. Tambah lagu di awal playlist
2. Tambah lagu di akhir playlist
3. Tambah lagu setelah lagu ke-3
4. Hapus lagu berdasarkan judul
5. Tampilkan playlist
0. Keluar
Pilih: 4
Masukkan judul lagu yang ingin dihapus: tarot
Lagu 'tarot' berhasil dihapus dari playlist.
```

Menampilkan output pada menu 4 yaitu menghapus lagu yang ada diplaylis berdasarkan judul lagu

Mengecek Kembali output pada menu 5 untuk menampilkan playlist setelah lagu yang dihapus oleh user dari menu ke 4

Deskripsi:

Program ini membuat playlist lagu sederhana menggunakan singly linked list di C++: setiap lagu disimpan dalam node (Lagu) yang berisi judul, penyanyi, durasi, dan penunjuk ke lagu berikutnya; kelas Playlist menyediakan operasi untuk menambah lagu di depan (tambahDepan), di belakang (tambahBelakang), atau tepat setelah lagu ke-3 (tambahSetelahKe3), menghapus lagu berdasarkan judul tanpa peka huruf besar/kecil (hapusBerdasarkanJudul — memakai fungsi bantu toLowerCase), dan menampilkan seluruh daftar (tampilkanPlaylist); main() mengisi playlist contoh, lalu menampilkan menu interaktif supaya pengguna bisa memasukkan lagu baru, menghapus, atau melihat isi playlist sampai memilih keluar. Intinya, kode ini memperagakan cara menyimpan dan mengelola koleksi lagu secara dinamis (bisa bertambah atau dihapus kapan saja)

menggunakan pointer, dengan antarmuka teks sederhana sehingga cocok sebagai latihan struktur data atau basis untuk aplikasi pemutar/daftar lagu yang lebih lengkap.

D. Kesimpulan

Dari kedua program di atas, dapat disimpulkan bahwa penggunaan struktur data linked list memberikan fleksibilitas dalam pengelolaan data yang bersifat dinamis, di mana elemen dapat ditambahkan atau dihapus tanpa perlu menggeser elemen lain seperti pada array. Pada program pertama, yaitu program Playlist, konsep singly linked list diterapkan untuk menyimpan data lagu yang terdiri dari judul, penyanyi, dan durasi. Program ini menunjukkan cara membuat daftar lagu yang bisa dimodifikasi dengan menambah lagu di depan, di belakang, atau di posisi tertentu, serta menghapus lagu berdasarkan judul. Melalui implementasi ini, pengguna dapat memahami bagaimana pointer digunakan untuk menghubungkan antar node (lagu) dan mengelola data secara efisien di memori.

Sementara itu, pada program kedua yaitu program SinglyList, penerapan konsep linked list difokuskan pada operasi dasar seperti membuat list baru, menambahkan elemen di awal dan akhir, serta menampilkan isi list. Program ini menggambarkan prinsip kerja linked list secara lebih sederhana menggunakan tipe data integer sebagai contoh. Dengan mempraktikkan kedua program ini, mahasiswa dapat memahami perbedaan dan manfaat penggunaan linked list dibandingkan struktur data statis, serta bagaimana konsep pointer dan alokasi memori dinamis berperan penting dalam membangun struktur data yang efisien, terorganisir, dan mudah dikembangkan untuk berbagai kebutuhan pemrograman.

E. Referensi

Linked list. Diakses dari: https://en.wikipedia.org/wiki/Linked_list

Introduction to Linked List. Diakses dari: https://www.geeksforgeeks.org/introduction-to-linked-list-data-structure-and-algorithm-tutorials/

Knuth, D. E. (1998). *The Art of Computer Programming, Volume 1: Fundamental Algorithms.* Addison-Wesley.

Patel, P., & Dave, M. (2016). *A comparative study of linked list and array data structures. International Journal of Computer Applications*, 139(7), 1–5.

Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2014). *Data Structures and Algorithms in C++*. Wiley