LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL IV
SINGLY LINKED LIST

Disusun Oleh :
NAMA : ZAKI MAULA DHIA
NIM : 103112400127

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

C++ adalah pengembangan dari dari bahasa c yang dibuat oleh Bjarne Stroustrup sekitar tahun 1980-an. C++ disebut bahasa multi-paradigma, artinya bisa dipakai dengan gaya prosedural (pakai fungsi biasa), beriorientasi objek (pakai class dan object), atau bahkan gabungan keduanya. C++ punya dasar-dasar seperti variabel, operator percabangan (if, switch), perulangan (for, while), dan bisa memakai class untuk membuat objek.

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

Guided 1

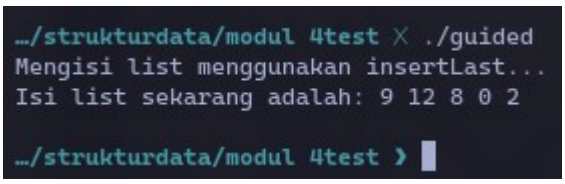
```
// file main1.cpp
#include <iostream>
#include <cstdlib>
#include "singly.h"
#include "singly.cpp"

using namespace std;

int main(){
    List L;
    address P;

    CreateList(L);

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

    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);
}
```

```cpp
// file singly.cpp
#include "singly.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)
    {
        insertFirst(L,P);
    } else {
        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;
```

```
    }
}
```

```
// file singly.h
#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 CreatList(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
```

Screenshots Output



Deskripsi:

Program Singly Linked List yang hampir mirip dengan array tetapi bisa jadi lebih cepat ketika dicoding dengan benar, program tersebut berguna untuk memasukan angka ke struck ‘ElmList’ dengan menggunakan pointer karena Singly Linked List lebih sering

menggunakan pointer untuk mencari data dan memasukan data, pertama angka akan masuk satu per satu kedalam struck dengan menggunakan Singly Linked List, lalu program akan menampilkan isi dari struck dengan menggunakan Singly Linked List juga dan menggunakan pointer untuk menampilkan datanya satu per satu secara urut sesuai input.

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

Unguided 1

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

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

    do {
        cout << "\n=== MENU PLAYLIST LAGU ===" << endl;
        cout << "1. Tambah lagu di awal playlist" << endl;
        cout << "2. Tambah lagu di akhir playlist" << endl;
        cout << "3. Tambah lagu setelah playlist ke-3" << endl;
        cout << "4. Hapus lagu berdasarkan judul" << endl;
        cout << "5. Tampilkan seluruh lagu" << endl;
        cout << "0. Keluar" << endl;
        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.tambahDepan(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.tambahBelakang(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.hapusLagu(judul);
            break;

        case 5:
            playlist.tampilkan();
            break;

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

        default:
            cout << "Pilihan tidak valid." << endl;
        }

    } while (pilihan != 0);

    return 0;
}
```

```
// Playlist.h
#ifndef PLAYLIST_H
#define PLAYLIST_H

#include <iostream>
```

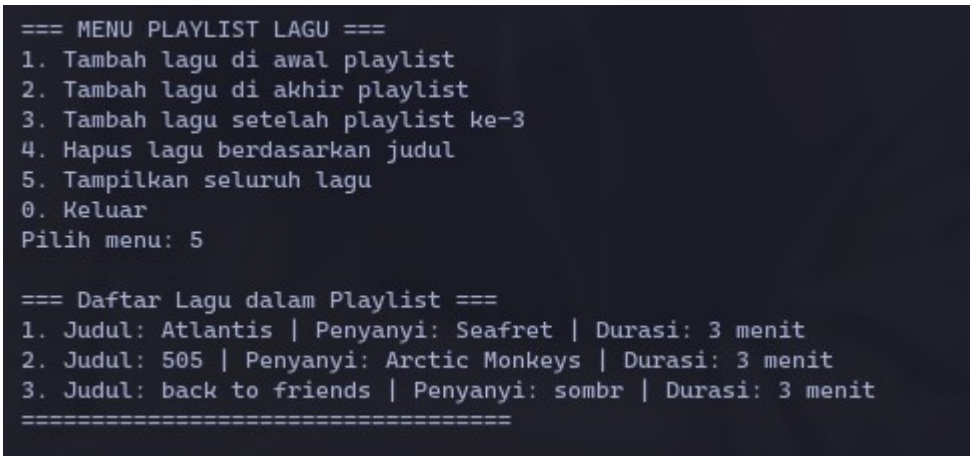
```
#include <string>
using namespace std;

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

class Playlist {
private:
    Lagu* head;

public:
    Playlist();
    ~Playlist();

    void 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
#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* laguBaru = new Lagu{judul, penyanyi, durasi, head};
    head = laguBaru;
}
```

```
void Playlist::tambahBelakang(string judul, string penyanyi, float durasi) {
    Lagu* laguBaru = new Lagu{judul, penyanyi, durasi, nullptr};
    if (head == nullptr) {
        head = laguBaru;
        return;
    }

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

void Playlist::tambahSetelahKe3(string judul, string penyanyi, float durasi) {
    Lagu* laguBaru = 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. Menambah di akhir." << endl;
        tambahBelakang(judul, penyanyi, durasi);
        delete laguBaru; // karena sudah dibuat baru di tambahBelakang
        return;
    }

    laguBaru->next = temp->next;
    temp->next = laguBaru;
}

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

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

```
    }

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

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

void Playlist::tampilkan() {
    if (head == nullptr) {
        cout << "Playlist kosong." << endl;
        return;
    }

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

Screenshots Output



Deskripsi:

Program untuk memiliki fungsi untuk menambah sebuah lagu kedalam struck ‘Lagu’ dengan fitur menambahkan lagu di awal playlist, menambahkan lagu di akhir playlist, menambah lagu setelah playlist ke tiga, hapus lagu berdasarkan judul, dan menampilkan semua lagu didalam playlist.