LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL IV
PENGENALAN SINGLY LINKED LIST

Disusun Oleh :
NAMA : Sinta Sintiani
NIM : 103112430047

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Singly Linked List adalah struktur data yang digunakan untuk menyimpan kumpulan elemen secara berurutan dengan cara menghubungkan setiap elemen menggunakan pointer. Setiap elemen dalam daftar disebut node, yang terdiri dari dua bagian, yaitu data dan pointer yang menunjuk ke node berikutnya. Struktur ini bersifat dinamis karena dapat menambah atau menghapus elemen tanpa perlu mengatur ulang posisi data lain di memori. Keunggulan utama Singly Linked List terletak pada efisiensi dalam operasi penyisipan dan penghapusan data, terutama di bagian awal daftar. Namun, kelemahannya adalah proses pencarian data membutuhkan waktu lebih lama karena harus menelusuri setiap node secara berurutan dari node pertama hingga node yang diinginkan.

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

Guided 1

singlylist.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;
};

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
```

singlylist.cpp

```
#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) {
```

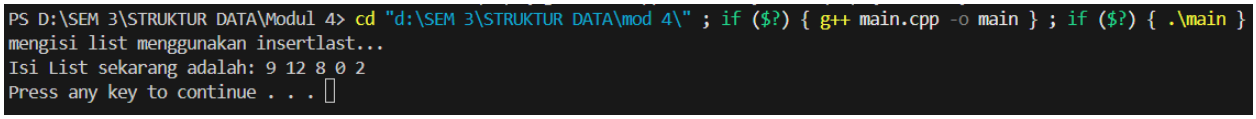
```
    address P = L.First;
    if (P == Nil) {
    std::cout << "List Kosong!" << std::endl;
} else {
    while (P != Nil) {
        std :: cout << P->info << " ";
        P = P->next;
    }
    std :: cout << std :: endl;
}
}
```

main.cpp

```
#include <iostream>
#include <cstdlib>
#include "singlylist.h"
#include "singlylist.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);

    system("pause");
    return 0;

}
```

Screenshots Output



Deskripsi:

Program ini mengimplementasikan Singly Linked List yaitu struktur data linear di mana setiap elemen (node) hanya memiliki satu penunjuk (pointer) ke elemen berikutnya. Program terdiri dari beberapa fungsi utama `CreateList` untuk menginisialisasi list kosong, `alokasi` untuk membuat node baru, `dealokasi` untuk menghapus node, `insertFirst` untuk menambahkan node di awal list, `insertLast` untuk menambahkan node di akhir list, dan `printInfo` untuk menampilkan seluruh elemen list. Pada `main()`, list dibuat dan diisi menggunakan `insertLast` dengan beberapa nilai integer, lalu ditampilkan ke layar. Konsep ini menunjukkan bagaimana linked list bersifat dinamis, memungkinkan penambahan elemen tanpa harus menentukan ukuran list di awal, berbeda dengan array yang memiliki ukuran tetap.

Unguided 1

playlist.h

```
#ifndef PLAYLIST_H
#define PLAYLIST_H

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

struct Lagu {
    string judul, penyanyi;
```

```
    float durasi;
    Lagu* next;
};

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"

Lagu* head = nullptr;

void tambahDepan(string judul, string penyanyi, float durasi) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, head};
    head = baru;
}

void tambahBelakang(string judul, string penyanyi, float durasi) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, nullptr};
    if (!head) head = baru;
    else {
        Lagu* temp = head;
        while (temp->next) temp = temp->next;
        temp->next = baru;
    }
}

void tambahSetelahKe3(string judul, string penyanyi, float durasi)
{
    Lagu* baru = new Lagu{judul, penyanyi, durasi, nullptr};
    Lagu* temp = head;
    int posisi = 1;
    while (temp && posisi < 3) {
        temp = temp->next;
        posisi++;
```

```
    }
    if (!temp) tambahBelakang(judul, penyanyi, durasi);
    else {
        baru->next = temp->next;
        temp->next = baru;
    }
}

void hapusLagu(string judul) {
    Lagu *temp = head, *prev = nullptr;
    while (temp && temp->judul != judul) {
        prev = temp;
        temp = temp->next;
    }
    if (!temp) {
        cout << "Lagu tidak ditemukan!\n";
        return;
    }
    if (!prev) head = head->next;
    else prev->next = temp->next;
    delete temp;
    cout << "Lagu \"" << judul << "\" dihapus.\n";
}

void tampilkan() {
    if (!head) {
        cout << "Playlist kosong.\n";
        return;
    }
    int i = 1;
    Lagu* temp = head;
    while (temp) {
        cout << i++ << ". " << temp->judul << " - " <<
temp->penyanyi
             << " (" << temp->durasi << " menit)\n";
        temp = temp->next;
    }
}
```

main.cpp

```
#include "playlist.h"
```

```cpp
#include "playlist.cpp"

int main() {
    int pilih;
    string judul, penyanyi;
    float durasi;

    do {
        cout << "\n=== MENU PLAYLIST ===\n";
        cout << "1. Tambah di awal\n";
        cout << "2. Tambah di akhir\n";
        cout << "3. Tambah setelah ke-3\n";
        cout << "4. Hapus lagu\n";
        cout << "5. Tampilkan\n";
        cout << "0. Keluar\n";
        cout << "Pilih: ";
        cin >> pilih;
        cin.ignore();

        switch (pilih) {
            case 1:
                cout << "Judul: "; getline(cin, judul);
                cout << "Penyanyi: "; getline(cin, penyanyi);
                cout << "Durasi: "; cin >> durasi;
                tambahDepan(judul, penyanyi, durasi);
                break;
            case 2:
                cout << "Judul: "; getline(cin, judul);
                cout << "Penyanyi: "; getline(cin, penyanyi);
                cout << "Durasi: "; cin >> durasi;
                tambahBelakang(judul, penyanyi, durasi);
                break;
            case 3:
                cout << "Judul: "; getline(cin, judul);
                cout << "Penyanyi: "; getline(cin, penyanyi);
                cout << "Durasi: "; cin >> durasi;
                tambahSetelahKe3(judul, penyanyi, durasi);
                break;
            case 4:
                cout << "Judul lagu yang dihapus: ";
                getline(cin, judul);
                hapusLagu(judul);
                break;
```

```
            case 5:
                tampilkan();
                break;
        }
    } while (pilih != 0);
}
```

Screenshots Output

```
PS D:\SEM 3\STRUKTUR DATA\mod 4> cd "d:\SEM 3\STRUKTUR DATA\Modul 4\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCode

=== MENU PLAYLIST ===
1. Tambah di awal
2. Tambah di akhir
3. Tambah setelah ke-3
4. Hapus lagu
5. Tampilkan
0. Keluar
Pilih: 1
Judul: Alamak
Penyanyi: Rizky Febian
Durasi: 3.00
```

```
=== MENU PLAYLIST ===
1. Tambah di awal
2. Tambah di akhir
3. Tambah setelah ke-3
4. Hapus lagu
5. Tampilkan
0. Keluar
Pilih: 2
Judul: Bintang 5
Penyanyi: Tenxi
Durasi: 3.10
```

```
=== MENU PLAYLIST ===
1. Tambah di awal
2. Tambah di akhir
3. Tambah setelah ke-3
4. Hapus lagu
5. Tampilkan
0. Keluar
Pilih: 3
Judul: Pelangi
Penyanyi: Hivi
Durasi: 3.37
```

```
=== MENU PLAYLIST ===
1. Tambah di awal
2. Tambah di akhir
3. Tambah setelah ke-3
4. Hapus lagu
5. Tampilkan
0. Keluar
Pilih: 5
1. Alamak - Rizky Febian (3 menit)
2. Bintang 5 - Tenxi (3.1 menit)
3. Pelangi - Hivi (3.37 menit)

=== MENU PLAYLIST ===
1. Tambah di awal
2. Tambah di akhir
3. Tambah setelah ke-3
4. Hapus lagu
5. Tampilkan
0. Keluar
Pilih: 4
Judul lagu yang dihapus: Bintang 5
Lagu "Bintang 5" dihapus.
```

```
=== MENU PLAYLIST ===
1. Tambah di awal
2. Tambah di akhir
3. Tambah setelah ke-3
4. Hapus lagu
5. Tampilkan
0. Keluar
Pilih: 5
1. Alamak - Rizky Febian (3 menit)
2. Pelangi - Hivi (3.37 menit)

=== MENU PLAYLIST ===
1. Tambah di awal
2. Tambah di akhir
3. Tambah setelah ke-3
4. Hapus lagu
5. Tampilkan
0. Keluar
Pilih: 0
PS D:\SEM 3\STRUKTUR DATA\Modul 4>
```

Deskripsi:

Program ini menggunakan Singly Linked List untuk mengelola playlist lagu dalam bahasa C++. Setiap lagu disimpan sebagai sebuah node yang memuat informasi judul, penyanyi, durasi, serta pointer next yang menunjuk ke node berikutnya. Program mendukung beberapa operasi, seperti menambahkan lagu di awal (tambahDepan), di akhir (tambahBelakang), atau setelah lagu ke-3 (tambahSetelahKe3), menghapus lagu berdasarkan judul (hapusLagu), dan menampilkan seluruh lagu dalam playlist (tampilkan). Dengan struktur linked list, penambahan dan penghapusan lagu dapat dilakukan secara dinamis tanpa perlu memindahkan elemen lain, sehingga lebih efisien dibandingkan penggunaan array. Program ini juga menyediakan menu interaktif yang mempermudah pengguna dalam mengelola playlist secara langsung.

C. Kesimpulan

Dari praktikum ini dapat disimpulkan bahwa *Singly Linked List* merupakan struktur data dinamis yang efisien untuk operasi penambahan dan penghapusan elemen, terutama karena tidak memerlukan penggeseran data seperti pada array. Melalui implementasi dalam program, konsep pointer dan node dapat dipahami dengan lebih baik, di mana setiap node menyimpan data serta penunjuk ke node berikutnya. Selain itu, penerapan *linked list* pada studi kasus pengelolaan playlist menunjukkan fleksibilitas struktur ini dalam menyimpan dan memanipulasi data secara dinamis sesuai kebutuhan pengguna.

D. Referensi

https://www.geeksforgeeks.org/dsa/singly-linked-list-tutorial/

https://daismabali.medium.com/mengenal-single-linked-list-dalam-struktur-data-3b64efa6f10c

https://rumahcoding.co.id/linked-list-pengertian-dan-implementasi-dasar/