LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 4 & 5
SINGLY LINKED LIST

Disusun Oleh :
NAMA : Muhammad Omar Nadiv
NIM : 103112430063

Dosen
Fahrudin Mukti Wibowo, S.Kom., M.Eng.

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Single linked list adalah jenis paling dasar dari linked list di mana setiap node terhubung ke node berikutnya dalam urutan linear. Setiap node memiliki dua bagian: data yang disimpan dan referensi ke node berikutnya dalam urutan. Single linked list tidak memiliki referensi kembali ke node sebelumnya, sehingga navigasi hanya bisa dilakukan dari depan ke belakang. Ini merupakan struktur data yang sederhana dan efisien untuk aplikasi yang memerlukan penyisipan atau penghapusan elemen di ujung linked list.

Implementasi dasar dari single linked list melibatkan beberapa operasi dasar seperti pembuatan linked list kosong, penambahan dan penghapusan elemen di awal dan akhir, serta penyisipan dan penghapusan elemen di tengah linked list.

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

Guided 1 (file main.cpp)

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

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 saat ini: ";
    printInfo(L);

    return 0;
}
```

Guided 1 (file 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;
    P = Nil;
}

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

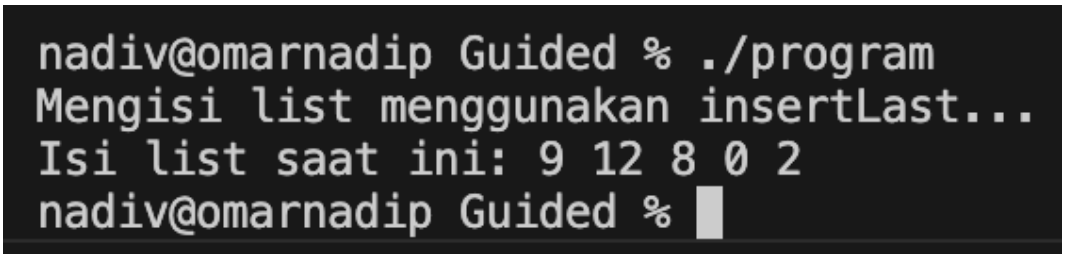
```
void insertLast(List &L, address P){
   if (L.First == Nil){
       insertFirst(L, P);
   } else {
       address Last = L.First;
       while (Last->next != Nil){
           Last = Last->next;
       }
       Last->next = P;
   }
}

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

       cout << endl;

   }
}
```

Guided 1 (file singlylist.h)

```
#ifndef SINGLYLIST_H_INCLUDED
#define SINGLYLIST_H_INCLUDED

#include<iostream>
#define Nil NULL
using namespace std;

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
```

Screenshots Output



Deskripsi:

Program ini merupakan implementasi struktur data Singly Linked List dalam bahasa C++. Program terdiri dari tiga file main.cpp, Singlylist.h, dan Singlylist.cpp.

File Singlylist.h berisi deklarasi struktur dan fungsi utama. Struktur Elmlist menyimpan data dan pointer ke elemen berikutnya, sedangkan List menyimpan alamat elemen pertama. Di dalamnya juga terdapat deklarasi fungsi seperti CreateList, alokasi, insertFirst, insertLast, dan printInfo.

File Singlylist.cpp berisi implementasi fungsi-fungsi tersebut. CreateList menginisialisasi list kosong, alokasi membuat node baru, insertFirst dan insertLast menambahkan elemen ke list, dan printInfo menampilkan seluruh isi list.

File main.cpp digunakan untuk menjalankan fungsi yang dibuat. Program membuat list kosong, menambahkan beberapa elemen menggunakan insertLast, lalu menampilkan hasilnya.

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

Unguided 1 (file 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;
};

struct Playlist {
   Lagu* head;
};

void createPlaylist(Playlist &P);
Lagu* createLagu(string judul, string penyanyi, float durasi);
void insertFirst(Playlist &P, Lagu* L);
void insertLast(Playlist &P, Lagu* L);
void insertAfter3(Playlist &P, Lagu* L);
void deleteByJudul(Playlist &P, string judul);
void tampilPlaylist(Playlist P);

#endif
```

Unguided 1 (file playlist.cpp)

```
#include "playlist.h"

void createPlaylist(Playlist &P)
{
   P.head = nullptr;
}
```

```
Lagu *createLagu(string judul, string penyanyi, float durasi)
{
   Lagu *L = new Lagu;
   L->judul = judul;
   L->penyanyi = penyanyi;
   L->durasi = durasi;
   L->next = nullptr;
   return L;
}

void insertFirst(Playlist &P, Lagu *L)
{
   L->next = P.head;
   P.head = L;
}

void insertLast(Playlist &P, Lagu *L)
{
   if (P.head == nullptr)
   {
       P.head = L;
   }
   else
   {
       Lagu *temp = P.head;
       while (temp->next != nullptr)
       {
           temp = temp->next;
       }
       temp->next = L;
   }
}

void insertAfter3(Playlist &P, Lagu *L)
{
   Lagu *temp = P.head;
   int count = 1;
   while (temp != nullptr && count < 3)
   {
       temp = temp->next;
```

```
        count++;
    }
    if (temp != nullptr)
    {
        L->next = temp->next;
        temp->next = L;
    }
    else
    {
        cout << "Kurang dari 3 lagu dalam playlist." << endl;
    }
}

void deleteByJudul(Playlist &P, string judul)
{
    if (P.head == nullptr)
    {
        cout << "Playlist kosong!" << endl;
        return;
    }

    Lagu *temp = P.head;
    Lagu *prev = nullptr;

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

    if (temp == nullptr)
    {
        cout << "Lagu tidak ditemukan!" << endl;
        return;
    }

    if (prev == nullptr)
    {
        P.head = temp->next;
    }
    else
```

```
    {
        prev->next = temp->next;
    }

    delete temp;
    cout << "Lagu \"" << judul << "\" berhasil dihapus." << endl;
}

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

    Lagu *temp = P.head;
    int no = 1;
    while (temp != nullptr)
    {
        cout << no++ << ". " << temp->judul << " - " << temp->penyanyi
             << " (" << temp->durasi << " menit)" << endl;
        temp = temp->next;
    }
}
```

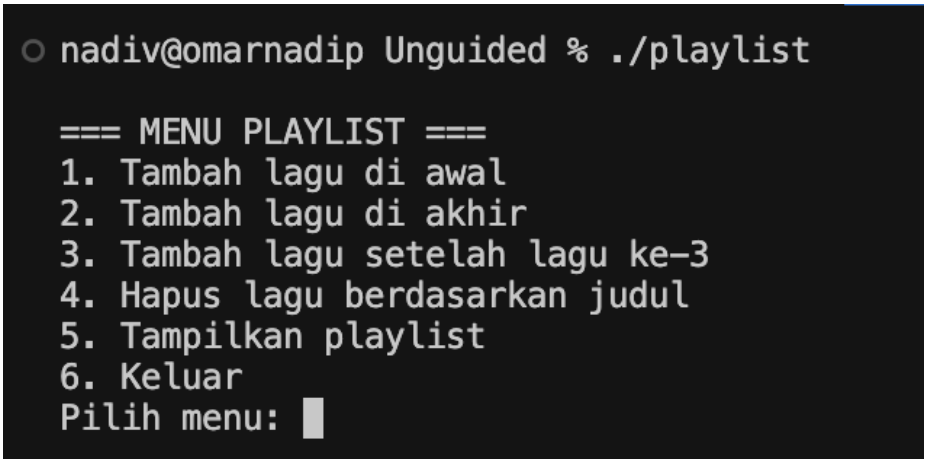
Unguided 1 (file main.cpp)

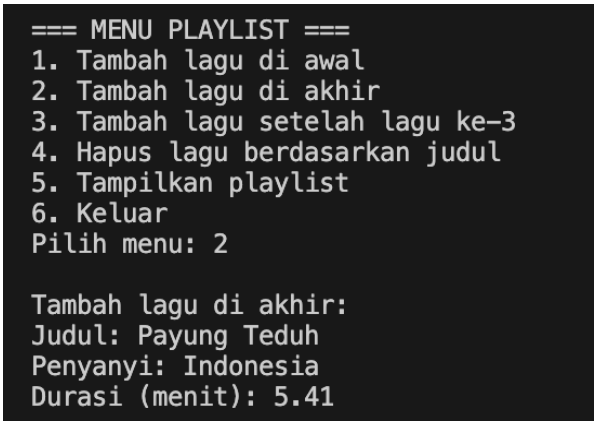
```
#include <iostream>
#include "playlist.h"
using namespace std;

void inputLagu(Playlist &P, int mode = 0)
{
    string judul, penyanyi;
    float durasi;

    cin.ignore();
    cout << "Judul: ";
    getline(cin, judul);
```

```
    cout << "Penyanyi: ";
    getline(cin, penyanyi);
    cout << "Durasi (menit): ";
    cin >> durasi;

    switch (mode)
    {
    case 1:
        insertFirst(P, createLagu(judul, penyanyi, durasi));
        break;
    case 2:
        insertLast(P, createLagu(judul, penyanyi, durasi));
        break;
    case 3:
        insertAfter3(P, createLagu(judul, penyanyi, durasi));
        break;
    }
}

int main()
{
    Playlist P;
    createPlaylist(P);

    int pilihan;
    do
    {
        cout << "\n=== MENU PLAYLIST ===\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 << "6. Keluar\n";
        cout << "Pilih menu: ";
        cin >> pilihan;

        switch (pilihan)
        {
        case 1:
            cout << "\nTambah lagu di awal:\n";
```

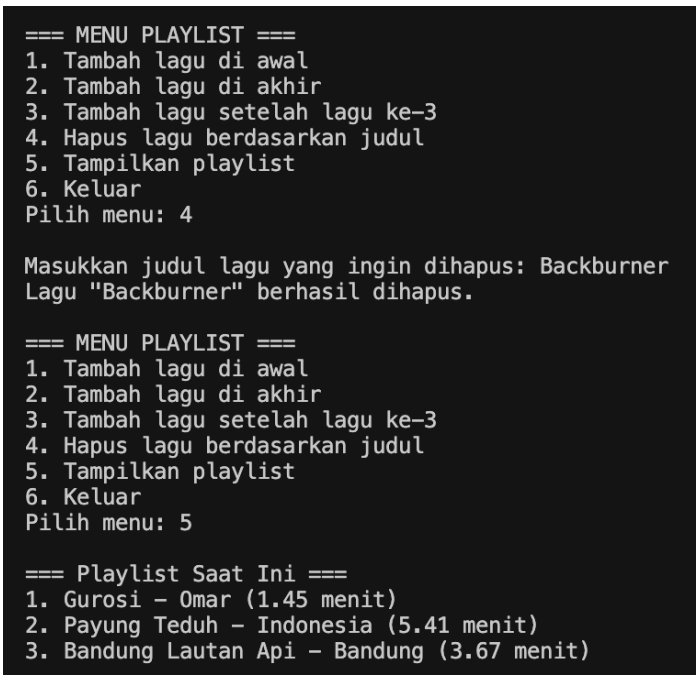
```
            inputLagu(P, 1);
            break;
        case 2:
            cout << "\nTambah lagu di akhir:\n";
            inputLagu(P, 2);
            break;
        case 3:
            cout << "\nTambah lagu setelah lagu ke-3:\n";
            inputLagu(P, 3);
            break;
        case 4:
        {
            cin.ignore();
            string judulHapus;
            cout << "\nMasukkan judul lagu yang ingin dihapus: ";
            getline(cin, judulHapus);
            deleteByJudul(P, judulHapus);
            break;
        }
        case 5:
            cout << "\n=== Playlist Saat Ini ===\n";
            tampilPlaylist(P);
            break;
        case 6:
            cout << "Keluar dari program.\n";
            break;
        default:
            cout << "Pilihan tidak valid, coba lagi.\n";
        }

    } while (pilihan != 6);

    return 0;
}
```

Screenshots Output





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

Tambah lagu di awal:
Judul: Gurosi
Penyanyi: Omar
Durasi (menit): 1.45

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

Tambah lagu setelah lagu ke-3:
Judul: Bandung Lautan Api
Penyanyi: Bandung
Durasi (menit): 3.67

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

=== Playlist Saat Ini ===
1. Gurosi - Omar (1.45 menit)
2. Backburner - Micky (3.27 menit)
3. Payung Teduh - Indonesia (5.41 menit)
4. Bandung Lautan Api - Bandung (3.67 menit)
```



Deskripsi:

Program ini digunakan untuk mengelola playlist lagu menggunakan single linked list di C++. Setiap lagu disimpan dalam struct Lagu yang berisi judul, penyanyi, durasi, dan pointer ke lagu berikutnya. Playlist diatur melalui struct Playlist dengan pointer head sebagai awal list.

Pengguna dapat menambahkan lagu di awal, akhir, atau setelah lagu ke-3, menghapus lagu berdasarkan judul, dan menampilkan seluruh playlist melalui menu interaktif. Fungsi-fungsi seperti createLagu, insertFirst, insertLast, insertAfter3, deleteByJudul, dan tampilPlaylist digunakan untuk memanipulasi dan menampilkan data lagu secara dinamis.

D. Kesimpulan

Pada modul 4 dan 5 ini, saya belajar bagaimana menggunakan struktur data dinamis berupa single linked list dalam pemrograman C++. Saya memahami cara membuat struct untuk menyimpan data kompleks, menggunakan pointer untuk menghubungkan node, serta membuat fungsi untuk menambah, menghapus, dan menampilkan node dalam linked list. Selain itu, saya juga belajar bagaimana alokasi memori dinamis bekerja dan bagaimana pointer dapat digunakan untuk memanipulasi data secara langsung. Pembelajaran ini membantu saya memahami konsep struktur data yang lebih fleksibel dibanding array statis dan membuat logika program menjadi lebih efisien dan terorganisir.

E. Referensi

https://daismabali.com/artikel_detail/54/1/Mengenal-Single-Linked-List-dalam-Struktur-Data.html

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