LAPORAN PRAKTIKUM STRUKTUR DATA

MODUL
SINGLE LINKD LIST

Disusun Oleh :
Nama : Jundi Amru Abbas Difaullah
NIM : 103112400143

Dosen:
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Single Linked List (daftar berantai tunggal) merupakan salah satu bentuk struktur data dinamis yang berbeda dari array karena elemen-elemennya tidak disimpan secara berurutan di memori. Struktur ini terdiri dari beberapa node yang saling terhubung. Setiap node biasanya memiliki dua bagian penting, yaitu bagian yang menyimpan data (seperti informasi sebuah item) dan bagian pointer yang berisi alamat node berikutnya. Node pertama disebut **Head**, sedangkan node terakhir akan menunjuk ke **NULL** sebagai tanda akhir rantai.

Kelebihan utama dari Single Linked List adalah kemampuannya untuk menambah dan menghapus data secara fleksibel tanpa perlu menggeser elemen lain seperti pada array. Hal ini dimungkinkan karena operasi cukup dilakukan dengan memodifikasi pointer antar node. Oleh karena itu, struktur ini sering digunakan untuk kasus yang membutuhkan perubahan data secara dinamis, misalnya untuk membuat daftar lagu dalam playlist.

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

Guided 1

Code SingLylist.h

```h
#ifndef SINGLYLIST_H_INCLUDED
#define SINGLYLIST_H_INCLUDED

#include <iostream>
using namespace std;

#define Nil NULL

typedef int infotype;
typedef struct ElmList *address;

struct ElmList {
    infotype info;
    address next;
};

struct List {
    address First;
};
// Deklarasi fungsi
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
```

Code SingLylist.cpp

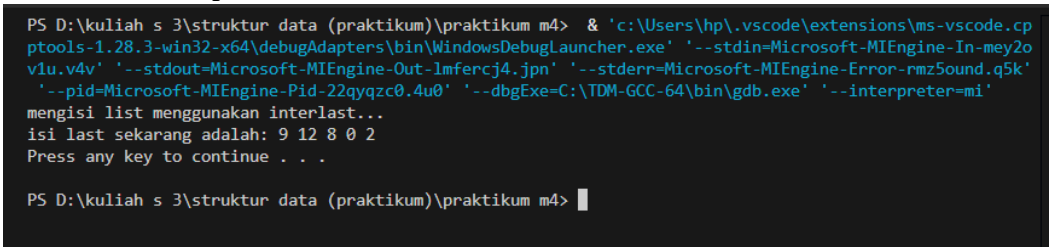
```cpp
#include "SingLylist.h"

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){
       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;
```

```
    }
}
```

Code main.cpp

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

using namespace std;
int main(){
    List L;
    address P;

    CreateList(L); //cukup satu pointer untuk di gunakan berulang kali

    cout << "mengisi list menggunakan interlast..." << 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 last sekarang adalah: ";
    printInfo(L);

    system("pause");
    return 0;
}
```

Screenshots Output :



Deskripsi :

Program ini nunjukin cara kerja dasar **Single Linked List** di C++. Di sini kita bikin list yang bisa nambah data ke belakang (pakai insertLast), lalu tampilkan semua isinya. Kodenya dibagi jadi tiga file: satu buat struktur dan deklarasi fungsi, satu buat isi fungsinya, dan satu lagi buat program utamanya. Intinya, program ini latihan dasar buat ngerti gimana node saling terhubung dalam linked list.

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

Guided 1

Code Playlist.h

```
#ifndef PLAYLIST_H_INCLUDED
#define PLAYLIST_H_INCLUDED

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

#define Nil NULL

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

typedef Lagu infotype;
typedef struct Node *address;

struct Node {
    infotype info;
    address next;
};

struct List {
    address head;
```

```
};

void buatList(List &L);
address buatNode(infotype data);
void hapusNode(address P);
void tambahDepan(List &L, address P);
void tambahBelakang(List &L, address P);
void tambahSetelahPosisi(List &L, address P, int posisi);
void hapusLagu(List &L, string judul);
void tampilPlaylist(List L);

#endif
```

Code Playlist.cpp

```
#include "Playlist.h"

void buatList(List &L) {
    L.head = Nil;
}

address buatNode(infotype data) {
    address P = new Node;
    P->info = data;
    P->next = Nil;
    return P;
}

void hapusNode(address P) {
    delete P;
}

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

void tambahBelakang(List &L, address P) {
    if (L.head == Nil) {
        L.head = P;
    } else {
        address temp = L.head;
        while (temp->next != Nil) {
            temp = temp->next;
        }
```

```
        temp->next = P;
    }
}

void tambahSetelahPosisi(List &L, address P, int posisi) {
    address temp = L.head;
    int i = 1;

    while (temp != Nil && i < posisi) {
        temp = temp->next;
        i++;
    }

    if (temp != Nil) {
        P->next = temp->next;
        temp->next = P;
    } else {
        cout << "Posisi gak valid, lagu ditambah di akhir." << endl;
        tambahBelakang(L, P);
    }
}

void hapusLagu(List &L, string judul) {
    if (L.head == Nil) {
        cout << "Playlist masih kosong." << endl;
        return;
    }

    address P = L.head;
    address prev = Nil;

    while (P != Nil && P->info.judul != judul) {
        prev = P;
        P = P->next;
    }

    if (P == Nil) {
        cout << "Lagu \"" << judul << "\" gak ketemu." << endl;
    } else {
        if (prev == Nil) {
            L.head = P->next;
        } else {
            prev->next = P->next;
        }
        hapusNode(P);
```

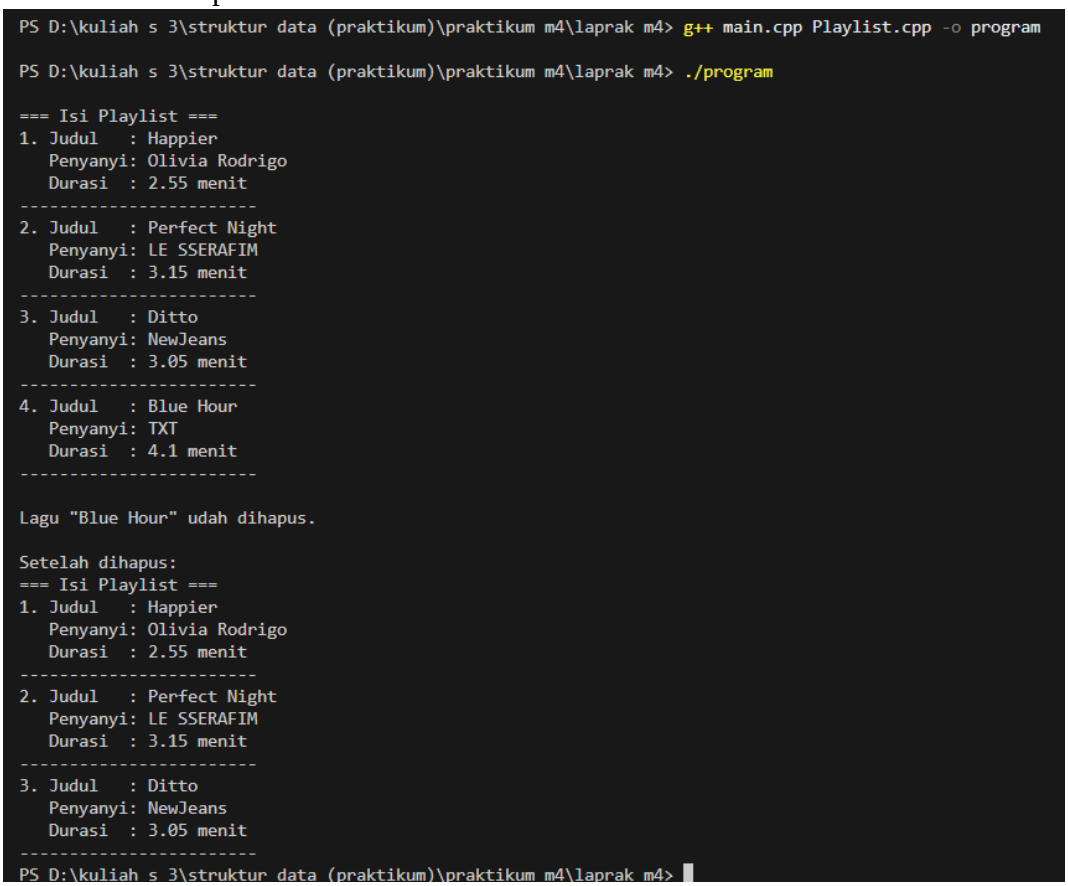
```
        cout << "Lagu \"" << judul << "\" udah dihapus." << endl;
    }
}

void tampilPlaylist(List L) {
    if (L.head == Nil) {
        cout << "Playlist kosong." << endl;
    } else {
        address P = L.head;
        int i = 1;
        cout << "=== Isi Playlist ===" << endl;
        while (P != Nil) {
            cout << i++ << ". Judul   : " << P->info.judul << endl;
            cout << "   Penyanyi: " << P->info.penyanyi << endl;
            cout << "   Durasi  : " << P->info.durasi << " menit" << endl;
            cout << "------------------------" << endl;
            P = P->next;
        }
    }
}
```

Code main.cpp

```
#include "Playlist.h"

int main() {
    List playlist;
    buatList(playlist);

    infotype lagu;
    address P;

    lagu = {"Perfect Night", "LE SSERAFIM", 3.15};
    P = buatNode(lagu);
    tambahDepan(playlist, P);

    lagu = {"Happier", "Olivia Rodrigo", 2.55};
    P = buatNode(lagu);
    tambahDepan(playlist, P);

    lagu = {"Blue Hour", "TXT", 4.10};
    P = buatNode(lagu);
    tambahBelakang(playlist, P);
```

```
    lagu = {"Ditto", "NewJeans", 3.05};
    P = buatNode(lagu);
    tambahSetelahPosisi(playlist, P, 2);

    tampilPlaylist(playlist);

    cout << endl;
    hapusLagu(playlist, "Blue Hour");

    cout << endl << "Setelah dihapus:" << endl;
    tampilPlaylist(playlist);

    return 0;
}
```

Screenshots Output :



Deskripsi :

Program ini dibuat untuk ngelola daftar lagu pakai **Single Linked List**. Jadi, setiap lagu disimpan sebagai node yang nyimpen judul, penyanyi, dan durasi. Dari program ini, kita bisa nambah lagu di awal, di akhir, atau di posisi tertentu, terus juga bisa hapus lagu berdasarkan judulnya. Hasil

akhirnya, program bakal nampilin semua lagu yang ada di playlist. Intinya, ini contoh sederhana gimana linked list bisa dipakai buat nyimpen data secara dinamis.

D. Kesimpulan

Single Linked List ini merupakan tipe struktur data dinamis yang sangat efektif untuk menagani data yang sering berubah tanpa perlu menggeser elemennya seperti array. Dengan menggunakan pointer yang berfungsi sebagai penghubung di antara node, struktur ini ini memungkinkan untuk melakukan penambahan , pengahapusan, dan pencarian data secara efisiensi, contoh penerapannya dalam mengelola palaylist lagu menunjukkan bagimana konsep linkd list bisa diaplikasikan dalam kehidupan nyata untuk menimpan serta memanipulasi seperti nama lagu, penyanyi, dan durasi lagunya, yang semuanya terhubung membentuk rantai data yang mudah dikelola, dan ini juga melatih cara berpikir yang sistematis melalui pendekatan pmrograman modular, di mana setiap fungsi dan file memiliki tugas spesifik untuk menjaga agar program tetap terorganisir dan mudah untuk berkembang

E. Referensi

- Wikipedia. (n.d.). *Linked list*. Diakses 29 Oktober 2025, https://en.wikipedia.org/wiki/Linked_list
- Programiz. (n.d.). *C++ Linked List (Single Linked List)*. Diakses 29 Oktober 2025, https://www.programiz.com/dsa/linked-list