LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL IV & V
SINGLY LINKNED LIST

Disusun Oleh :
Nama: Dina Nadhyfa
NIM : 103112430052

Dosen
Fahrudin Mukti Wibowo

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Singly linked list adalah struktur data linear yang terdiri dari serangkaian elemen yang disebut node, di mana setiap node menyimpan data dan sebuah pointer ke node berikutnya dalam urutan. Berbeda dengan array yang memiliki alokasi memori statis, singly linked list memiliki alokasi memori yang dinamis sehingga ukuran dan distribusi memori dapat berubah-ubah selama eksekusi program. Node terakhir pada singly linked list menunjuk ke null sebagai tanda akhir daftar. Struktur ini memungkinkan operasi penyisipan dan penghapusan elemen menjadi lebih efisien karena tidak memerlukan penggeseran elemen elemen lain seperti pada array, tetapi tracersal hanya bisa dilakukan secara terus-menerus dari awal ke akhir (head ke tail) karena tidak memiliki pointer ke node sebelumnya.

Implementasi dasar single linked list melibatkan operasi-operasi seperti pembuatan linked list kosong, penambahan dan penghapusan node di awal, akhir, atau tengah daftar. Setiap node didefinisikan dengan dua bagian yaitu data yang disimpan dan pointer yang menunjuk ke node selanjutnya, biasanya direpresentasikan sebagai struktur atau kelas dalam bahasa pemrograman seperti C, C++, Java, atau Python. Karena sifatnya yang hanya memiliki satu arah, single linked list umumnya lebih sederhana dibandingkan double linked list yang mempunyai dua pointer per node. Penggunaan single linked list banyak ditemukan dalam aplikasi yang membutuhkan penambahan atau penghapusan data secara dinamis dan efisien, seperti antrian atau pengelolaan memori sementara.

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

Singlylist.h

```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 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_H_INCLUDED
```

Singlylist.cpp

```
#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 // SINGLYLIST_H_INCLUDED
```

Main.cpp

```
#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 // SINGLYLIST_H_INCLUDED
```

Screenshots Output

```
PS D:\COOLYEAH\SEMESTER 3\Strukdat\Modul 4> cd "d:\COOLYEAH\SEMESTER 3\Strukdat\Modul 4\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main
 }
Mengisi list menggunakan insertlast...
Isi list sekarang adalah: 9 12 8 0 2
Press any key to continue . . .
PS D:\COOLYEAH\SEMESTER 3\Strukdat\Modul 4>
```

Deskripsi:

Kode program ini mengimplementasikan struktur data singly linked list untuk menyimpan bilangan bulat (int) dan menguji operasi penambahan data di akhir list.

1. Singlylist.h mendefinisikan struktur node (Elmlist) yang menyimpa data (infotype = int) dan pointer ke node berikutnya, serta mendeklarasikan operasi-operasi dasar seperti membuat list, mengalokasikan node, dan meyisipkan elemen di awal atau di akhir.
2. Singlylist.cpp menyediakan implementasi dari fungsi-fungsi tersebut, termasuk untuk interLast yang menelususi list hingga node berakhir sebelum menyambungkan node baru.

3. main.cpp berfungsi untuk penguji list, di mana list kosong diinisialisasi, lima angka (9, 12, 8, 0, 2) kemudian ditambahkan secara berurutan ke akhir list menggunakan interLast, dan hasilnya ditampilkan ke terminal sebagai 9 12 8 0 2.

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

Playlist.h

```
//103112430052_Dina Nadhyfa

#ifndef PLAYLIST_H
#define PLAYLIST_H

#include <string>
using namespace std;

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

typedef struct elmlist *address;

struct elmlist {
    Lagu info;
    address next;
};

struct List {
    address first;
};

// Primitive functions
void createList(List &L);
address alokasi(string judul, string penyanyi, float durasi);
void dealokasi(address &P);
void insertFirst(List &L, address P);
void insertLast(List &L, address P);
void insertAfter(List &L, address P, address Prec);
void deleteLagu(List &L, string judul);
void printInfo(List L);
address findElm(List L, string judul);

#endif
```

Playlist.cpp

```cpp
//103112430052_Dina Nadhyfa

#include "Playlist.h"
#include <iostream>

void createList(List &L) {
    L.first = NULL;
}

address alokasi(string judul, string penyanyi, float durasi) {
    address P = new elmlist;
    P->info.judul = judul;
    P->info.penyanyi = penyanyi;
    P->info.durasi = durasi;
    P->next = NULL;
    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 == NULL) {
        L.first = P;
    } else {
        address last = L.first;
        while (last->next != NULL) {
            last = last->next;
        }
        last->next = P;
    }
}

void insertAfter(List &L, address P, address Prec) {
    if (Prec != NULL) {
        P->next = Prec->next;
        Prec->next = P;
    }
}

void deleteLagu(List &L, string judul) {
```

```
    address P = findElm(L, judul);
    if (P == NULL) {
        cout << "Lagu tidak ditemukan." << endl;
        return;
    }

    if (P == L.first) {
        L.first = P->next;
        dealokasi(P);
    } else {
        address prev = L.first;
        while (prev->next != P) {
            prev = prev->next;
        }
        prev->next = P->next;
        dealokasi(P);
    }
    cout << "Lagu \"" << judul << "\" berhasil dihapus." << endl;
}

void printInfo(List L) {
    address P = L.first;
    int i = 1;
    if (P == NULL) {
        cout << "Playlist kosong." << endl;
    } else {
        while (P != NULL) {
            cout << i << ". " << P->info.judul << " - " << P-
>info.penyanyi
                 << " (" << P->info.durasi << " menit)" << endl;
            P = P->next;
            i++;
        }
    }
}

address findElm(List L, string judul) {
    address P = L.first;
    while (P != NULL) {
        if (P->info.judul == judul) {
            return P;
        }
        P = P->next;
    }
    return NULL;
}
```

main.h

```cpp
//103112430052_Dina Nadhyfa

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

int main() {
    List L;
    createList(L);

    // Tambah lagu di awal
    address P1 = alokasi("Joyride", "CORTIS", 2.52);
    insertFirst(L, P1);

    address P2 = alokasi("Same Dream, Same Mind, Same Night",
"Seventeen", 4.08);
    insertFirst(L, P2);

    // Tambah lagu di akhir
    address P3 = alokasi("Fall In Love Again", "P1Harmony", 3.29);
    insertLast(L, P3);

    // Tambah lagu setelah lagu ke-3
    address P4 = alokasi("Blue Valentine", "NMIXX", 3.06);
    address Prec = L.first;
    for (int i = 1; i < 3 && Prec != NULL; i++) {
        Prec = Prec->next;
    }
    if (Prec != NULL) {
        insertAfter(L, P4, Prec);
    }

    // Tampilkan playlist
    cout << "Playlist Awal:" << endl;
    printInfo(L);
    cout << endl;

    // Hapus lagu berdasarkan judul
    deleteLagu(L, "Same Dream, Same Mind, Same Night");
    cout << endl;

    // Tampilkan playlist setelah penghapusan
    cout << "Playlist Setelah Penghapusan:" << endl;
    printInfo(L);

    return 0;
}
```

Screenshots Output

```
PS D:\COOLYEAH\SEMESTER 3\Strukdat\Modul 4 -5 unguided> g++ main.cpp Playlist.cpp -o Playlist_app
PS D:\COOLYEAH\SEMESTER 3\Strukdat\Modul 4 -5 unguided> ./playlist_app
Playlist Awal:
1. Same Dream, Same Mind, Same Night - Seventeen (4.08 menit)
2. Joyride - CORTIS (2.52 menit)
3. Fall In Love Again - P1Harmony (3.29 menit)
4. Blue Valentine - NMIXX (3.06 menit)

Lagu "Same Dream, Same Mind, Same Night" berhasil dihapus.

Playlist Setelah Penghapusan:
1. Joyride - CORTIS (2.52 menit)
2. Fall In Love Again - P1Harmony (3.29 menit)
3. Blue Valentine - NMIXX (3.06 menit)
PS D:\COOLYEAH\SEMESTER 3\Strukdat\Modul 4 -5 unguided>
```

Deskripsi:

Program ini mengimplementasikan singly linked list yang berfungsi sebagai aplikasi playlist musik.

1. Playlist.h mendefinisikan sturktur data untuk menyimpan informasi lagu sperti, judul, penyanyi, dan durasi di dalam node elmlist. File ini juga mendeklarasikan semua fungsi primitif untuk mengelola playlist (misalnya, insertFirst, insertlast, deleteLagu, printInfo).
2. Playlist.cpp menyediakan implementasi rinci untuk setiap fungsi.
3. main.cpp membuat list dan melakukan serangkaian operasi seperti, menambahkan dua lagu diawal, menambahkan satu lagu diakhir, menambahkan satu lagu setelah elemen ke-3, menampilkan playlist awal, menghapus lagu, dan menampilkan playlist setelah penghapusan.

D. Kesimpulan

Kesimpulan dari kode-kode program diatas adalah hasil pengimplementasian singly linked list. Memisahkannya secara modular menjadi header (.h) dan file implementasi (.cpp). Struktur ini memungkinkan untuk menyimpan data secara dinamis, baik itu bilangan bulat atau data kompleks seperti detail lagu (judul, penyanyi, durasi), menggunakan node yang saling terhubung. Secara fungsiona, list mampu melakukan operasi inti seperti penyisipan di awal, di akhir, dan di tengah, serta operasi penghapusan dan pencetakan isi list secara efisien.

E. Referensi

“Linked List: Pengertian dan Implementasi Dasar," Rumahcoding.co.id, 24 Mei 2024. [Online]. https://rumahcoding.co.id/linked-list-pengertian-dan-implementasi-dasar/

A. Morfotech, "Implementasi Linked List: Teori hingga Praktik," Morfotech.id, 2025. [Online]. https://morfotech.id/blog/morfogenesis-mengupas-tuntas-linked-list-struktur-data-rantai-yang-fleksibel

"Linked List Bahasa C Beserta Fungsi dan Penjelasannya," Penelitian.id, 3 Januari 2025. [Online]. https://www.penelitian.id/2023/04/linked-list-bahasa-c-.html

"Struktur Data: Pengertian, Tipe, dan Kegunaan," Codingstudio.id, 22 Juni 2023. [Online]. https://codingstudio.id/blog/struktur-data-pengertian-tipe-dan-kegunaan/

"Artikel Season 3 Single Linked List," Scribd.com, 28 September 2025. [Online]. https://id.scribd.com/document/441589995/artikel-season-3-single-linked-list-1