LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL IV & V
SINGLY LINKED LIST

Disusun Oleh :

NAMA : HILMI HAKIM RAMADANI

NIM : 103112430016

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

C++ adalah bahasa pemrograman yang dikembangkan oleh Bjarne Stroustrup di Bell Labs pada awal 1980-an sebagai pengembangan dari bahasa C. Awalnya dirancang untuk sistem Unix, C++ kemudian menjadi bahasa pemrograman tujuan umum (general-purpose programming language).

Linked list adalah suatu bentuk struktur data yang berupa sekumpulan elemen data yang bertipe sama dimana tiap elemen saling berkait atau dihubungkan dengan elemen lain melalui suatu pointer. Pointeritu sendiri adalah alamat elemen data yang tersimpan di memori. Penggunaan pointer untuk mengacu elemen berakibat elemen-elemen bersebelahan secara logik walau tidak bersebelahan secara fisik di memori.

Linked list terdiri dari node-node (simpul-simpul) yang saling terhubung(linked).Simpul berupa struct, sedangkan link berupa komponen yang bertipe pointerke simpul. Ada dua jenis pointeryang digunakan, yaitu head (menunjukkan alamat pointer paling depan) dan tail (menunjukkan simpul terakhir). Operasi penambahan atau penghapusan sebuah simpul akan mengubah nilai pointer linknya.Sedangkan pointer link disimpul terkahir diberi nilai null.

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

Guided 1

#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;

    // 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 list sekarang adalah: ";
    printInfo(L);

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

#Singlylist.cpp

```
#include "Singlylist.h"

void CreateList(List &L)
{
    L.First = Nil;
}

address alokasi(infotype x)
{
    address P = new Elmtlist;
    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;
    }
}
```

#Singlylist.h

```
#ifndef SINGLYLIST_H_INCLUDED
#define SINGLYLIST_H_INCLUDED

#include <iostream>

#define Nil NULL

typedef int infotype;
typedef struct Elmtlist *address;

struct Elmtlist
{
    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
```

Screenshots Output

```
PS D:\KULIAH\SEMESTER 3\Struktur Data>
n32-x64\debugAdapters\bin\WindowsDebugL
ut=Microsoft-MIEngine-Out-0njkxlcm.wkv'
t-MIEngine-Pid-x1yjlx3c.khc' '--dbgExe=
Mengisi list menggunakan insertLast...
Isi list sekarang adalah: 9 12 8 0 2
Press any key to continue . . .
```

Deskripsi:

Ketiga file tersebut bekerja bersama untuk mengimplementasikan dan menggunakan Single Linked List. File Singlylist.h berisi definisi struktur data list, node, serta deklarasi fungsi-fungsi dasar agar dapat digunakan oleh file lain. File Singlylist.cpp berisi implementasi dari fungsi-fungsi tersebut, seperti pembuatan list, alokasi node, penyisipan data di awal dan akhir list, serta menampilkan isi list. Sementara itu, file Main.cpp berfungsi sebagai program utama yang membuat list, mengisi data menggunakan insertLast, dan menampilkan hasilnya untuk menguji apakah operasi Single Linked List berjalan dengan benar.

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

Unguided

#main.cpp

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

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

   tambahAkhir(P, buatLagu("Laskar Pelangi", "Nidji", 4.2));
```

```
    tambahAkhir(P, buatLagu("Separuh Aku", "Noah", 4.5));
    tambahAkhir(P, buatLagu("Ruang Rindu", "Letto", 4.1));

    cout << "Playlist awal:" << endl;
    tampilkan(P);

    tambahAwal(P, buatLagu("Akad", "Payung Teduh", 4.3));
    tambahSetelahKe3(P, buatLagu("Fix You", "Coldplay", 4.6));
    hapusJudul(P, "Separuh Aku");

    cout << "\nPlaylist akhir:" << endl;
    tampilkan(P);

    return 0;
}
```

#Playlist.cpp

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

void createPlaylist(Playlist &P)
{
    P.first = NULL;
}

Lagu *buatLagu(string j, string p, float d)
{
    Lagu *L = new Lagu;
    L->judul = j;
    L->penyanyi = p;
    L->durasi = d;
    L->next = NULL;
    return L;
}

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

void tambahAkhir(Playlist &P, Lagu *L)
{
    if (P.first == NULL)
    {
        P.first = L;
    }
    else
```

```
    {
        Lagu *Q = P.first;
        while (Q->next != NULL)
            Q = Q->next;
        Q->next = L;
    }
}

void tambahSetelahKe3(Playlist &P, Lagu *L)
{
    Lagu *Q = P.first;
    int i = 1;
    while (Q != NULL && i < 3)
    {
        Q = Q->next;
        i++;
    }
    if (Q != NULL)
    {
        L->next = Q->next;
        Q->next = L;
    }
}

void hapusJudul(Playlist &P, string j)
{
    Lagu *Q = P.first;
    Lagu *prev = NULL;

    while (Q != NULL && Q->judul != j)
    {
        prev = Q;
        Q = Q->next;
    }

    if (Q != NULL)
    {
        if (prev == NULL)
            P.first = Q->next;
        else
            prev->next = Q->next;
        delete Q;
    }
}

void tampilkan(Playlist P)
{
    Lagu *Q = P.first;
    int no = 1;
    while (Q != NULL)
```

```
    {
        cout << no++ << ". " << Q->judul
             << " - " << Q->penyanyi
             << " (" << Q->durasi << " menit)" << endl;
        Q = Q->next;
    }
}
```

#Playlist.h

```
#ifndef PLAYLIST_H
#define PLAYLIST_H

#include <string>
using namespace std;

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

struct Playlist
{
    Lagu *first;
};

void createPlaylist(Playlist &P);
Lagu *buatLagu(string j, string p, float d);
void tambahAwal(Playlist &P, Lagu *L);
void tambahAkhir(Playlist &P, Lagu *L);
void tambahSetelahKe3(Playlist &P, Lagu *L);
void hapusJudul(Playlist &P, string j);
void tampilkan(Playlist P);

#endif
```

Screenshots Output

```
Playlist awal:
1. Laskar Pelangi - Nidji (4.2 menit)
2. Separuh Aku - Noah (4.5 menit)
3. Ruang Rindu - Letto (4.1 menit)

Playlist akhir:
1. Akad - Payung Teduh (4.3 menit)
2. Laskar Pelangi - Nidji (4.2 menit)
3. Fix You - Coldplay (4.6 menit)
4. Ruang Rindu - Letto (4.1 menit)
PS D:\KULIAH\SEMESTER 3\Struktur Data>
```

Deskripsi:

Program ini menggunakan konsep Single Linked List untuk mengelola playlist lagu yang dibagi ke dalam tiga file. File Playlist.h berisi struktur data lagu dan playlist serta deklarasi fungsi-fungsi dasar. File Playlist.cpp mengimplementasikan operasi linked list seperti menambah lagu di awal, akhir, setelah lagu ke-3, menghapus lagu berdasarkan judul, dan menampilkan isi playlist. Sementara itu, file main.cpp berfungsi sebagai program utama yang membuat playlist, menambahkan beberapa lagu, menghapus lagu tertentu, dan menampilkan hasilnya, sehingga program mudah dipahami dan sesuai untuk pembelajaran struktur data dasar.

D. Kesimpulan

Dari praktikum ini dapat disimpulkan bahwa struktur data ini memungkinkan pengelolaan data secara dinamis menggunakan pointer untuk menghubungkan setiap node. Melalui kegiatan guided dan unguided, mahasiswa mampu memahami cara membuat, mengalokasikan, menambahkan, menghapus, serta menampilkan data dalam Single Linked List menggunakan bahasa C++. Implementasi pada studi kasus playlist lagu menunjukkan bahwa Single Linked List efektif digunakan untuk menyimpan dan memanipulasi data yang ukurannya dapat berubah-ubah. Dengan pembagian program ke dalam file header, source, dan main, struktur kode menjadi lebih rapi, modular, dan mudah dipahami, sehingga mendukung pembelajaran konsep struktur data secara sistematis dan aplikatif.

E. Referensi

Dewi, L. J. E. (2010). Media Pembelajaran Bahasa Pemrograman C++. Jurnal Pendidikan Teknologi dan Kejuruan, 7(1).

Wikipedia. 2025, 29 September. C++. Diakses pada 1 Oktober. Dari https://en.wikipedia.org/wiki/C%2B%2B

Sihombing, J. (2019). Penerapan stack dan queue pada array dan linked list dalam java. INFOKOM (Informatika & Komputer), 7(2), 15-24.