LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL IV & V
PENGENALAN CODE BLOCKS

Disusun Oleh :

NAMA : Rikza Nur Zaki

NIM : 103112430030

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori Singly Linked List

Singly Linked List adalah salah satu struktur data dinamis linier di mana setiap elemen data atau node atau elemen list, terhubung ke elemen berikutnya melalui sebuah pointer, dan tidak memerlukan alokasi memori yang berdekatan di memori seperti array.

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

Guided 1

Singlylist.h

```h
#ifndef LIST_H_INCLUDED
#define LIST_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
```

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) {
        cout << "List kosong" << endl;
    } else {
        while (P != Nil) {
            cout << P->info << " ";
            P = P->next;
        }
        cout << endl;
    }
}
```

Main.cpp

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

int main() {
    list L;
    address P;

    createList(L);
    cout << "membuat 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

```
membuat list menggunakan insertLast...
isi list sekarang adalah: 9 12 8 0 2
Press any key to continue . . .
```

Deskripsi:

Program ini memakai fungsi ADT, dengan membuat createlist agar menjamin kondisi awal list terdefinisi dengan benar, address alokasi untuk menyediakan unit memori untuk data yang baru, dealokasi untuk mencegah kebocoran memori, insertfirst untuk menyisipkan elemen di posisi yang paling cepat diakses, insertlast untuk menyisipkan elemen di posisi yang paling sesuai dengan urutan alami, dan printinfo untuk mengakses acak fungsional. Kemudian di main.cpp tinggal memasukan setiap lokasinya maka akan muncul secara berurutan.

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

Unguided 1

playlist.h

```cpp
#ifndef PLAYLIST_H
#define PLAYLIST_H
#include <string>

struct infotype {
    std::string judul;
    std::string penyanyi;
    float durasi;
};

struct ElmLagu;
typedef ElmLagu* address;

struct ElmLagu {
    infotype info;
    address next;
};

struct List {
    address First;
};

void CreateList(List& L);
address alokasi(std::string judul, std::string penyanyi, float durasi);
void dealokasi(address& P);
void insertFirst(List& L, address P);
void insertLast(List& L, address P);
void insertAfterKe3(List& L, address P);
void delP(List& L, std::string judul_hapus);
address findElm(List L, std::string judul_cari);
void printInfo(List L);
int nbElmt(List L);
#endif
```

playlist.cpp

```cpp
#include "playlist.h"
#include <iostream>
#include <string>

using namespace std;


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

address alokasi(string judul, string penyanyi, float durasi) {
    address P = new ElmLagu;
    if (P != NULL) {
        P->info.judul = judul;
        P->info.penyanyi = penyanyi;
        P->info.durasi = durasi;
        P->next = NULL;
    }
    return P;
}

void dealokasi(address& P) {
    delete P;
    P = NULL;
}

int nbElmt(List L) {
    int count = 0;
    address P = L.First;
    while (P != NULL) {
        count++;
        P = P->next;
    }
    return count;
}

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

```
void printInfo(List L) {
    address P = L.First;
    cout << "\n=== DAFTAR PLAYLIST ===\n";
    if (P == NULL) {
        cout << "Playlist kosong." << endl;
        return;
    }

    int no = 1;
    cout << "No. | Judul Lagu | Penyanyi | Durasi (menit)" << endl;
    cout << "-------------------------------------------------" << endl;

    while (P != NULL) {
        cout << no++ << ". | "
             << P->info.judul << " | "
             << P->info.penyanyi << " | "
             << P->info.durasi << endl;
        P = P->next;
    }
    cout << "-------------------------------------------------" << endl;
    cout << "Total Lagu: " << nbElmt(L) << endl;
}

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

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

```
void insertAfterKe3(List& L, address P) {
    if (nbElmt(L) < 3) {
        cout << "ERROR: Jumlah lagu kurang dari 3. Tidak bisa insert after lagu ke-3." << endl;
        dealokasi(P);
        return;
    }

    address Prec = L.First;
    for (int i = 1; i < 3; i++) {
        Prec = Prec->next;
    }

    P->next = Prec->next;
    Prec->next = P;
    cout << "Lagu '" << P->info.judul << "' berhasil ditambahkan setelah playlist ke-3." << endl;
}

void delP(List& L, string judul_hapus) {
    address P_del = findElm(L, judul_hapus);

    if (P_del == NULL) {
        cout << "Lagu dengan judul '" << judul_hapus << "' tidak ditemukan." << endl;
        return;
    }

    if (P_del == L.First) {
        L.First = P_del->next;
    } else {
        address Prec = L.First;
        while (Prec->next != P_del) {
            Prec = Prec->next;
            if (Prec == NULL) break;
        }

        if (Prec != NULL) {
            Prec->next = P_del->next;
        }
    }

    cout << "Lagu '" << P_del->info.judul << "' berhasil dihapus." << endl;
    dealokasi(P_del);
}
```

main.cpp

```cpp
#include "playlist.h"
#include "playlist.cpp"
#include <iostream>

using namespace std;

int main() {
    List L;
    CreateList(L);
    address P;

    cout << "===== PLAYLIST LAGU  =====" << endl;
    cout << "\n--- Tambah Lagu di Awal ---" << endl;
    P = alokasi("Laskar Pelangi", "Nidji", 4.31);
    insertFirst(L, P);
    P = alokasi("Bendera", "Cokelat", 3.40);
    insertFirst(L, P);

    printInfo(L);

    cout << "\n--- Tambah Lagu di Akhir ---" << endl;
    P = alokasi("Indonesia Raya", "W.R. Supratman", 1.47);
    insertLast(L, P);
    P = alokasi("Sempurna", "Andra and The Backbone", 4.35);
    insertLast(L, P);

    printInfo(L);

    cout << "\n--- Tambah Lagu Setelah Playlist ke-3 ---" << endl;
    P = alokasi("Gundul-Gundul Pacul", "Lagu Daerah", 2.10);
    insertAfterKe3(L, P);

    printInfo(L);

    cout << "\n--- Hapus Lagu Berdasarkan Judul ---" << endl;

    delP(L, "Indonesia Raya");
    delP(L, "Bendera");

    printInfo(L);

    return 0;
}
```

Screenshots Output

Tambah lagu di awal playlist

```
===== PLAYLIST LAGU  =====

--- Tambah Lagu di Awal ---

=== DAFTAR PLAYLIST ===
No. | Judul Lagu | Penyanyi | Durasi (menit)
-------------------------------------------------
1. | Bendera | Cokelat | 3.4
2. | Laskar Pelangi | Nidji | 4.31
-------------------------------------------------
Total Lagu: 2
```

Tambah lagu di akhir playlist

```
--- Tambah Lagu di Akhir ---

=== DAFTAR PLAYLIST ===
No. | Judul Lagu | Penyanyi | Durasi (menit)
-------------------------------------------------
1. | Bendera | Cokelat | 3.4
2. | Laskar Pelangi | Nidji | 4.31
3. | Indonesia Raya | W.R. Supratman | 1.47
4. | Sempurna | Andra and The Backbone | 4.35
-------------------------------------------------
Total Lagu: 4
```

Tambah lagu setelah playlist ke 3

```
--- Tambah Lagu Setelah Playlist ke-3 ---
Lagu 'Gundul-Gundul Pacul' berhasil ditambahkan setelah playlist ke-3.
```

```
=== DAFTAR PLAYLIST ===
No. | Judul Lagu | Penyanyi | Durasi (menit)
-------------------------------------------------
1. | Bendera | Cokelat | 3.4
2. | Laskar Pelangi | Nidji | 4.31
3. | Indonesia Raya | W.R. Supratman | 1.47
4. | Gundul-Gundul Pacul | Lagu Daerah | 2.1
5. | Sempurna | Andra and The Backbone | 4.35
-------------------------------------------------
Total Lagu: 5
```

Hapus lagu berdasarkan judul

```
--- Hapus Lagu Berdasarkan Judul ---
Lagu 'Indonesia Raya' berhasil dihapus.
Lagu 'Bendera' berhasil dihapus.
```

Tampilkan seluruh lagu dalam playlist

```
=== DAFTAR PLAYLIST ===
No. | Judul Lagu | Penyanyi | Durasi (menit)
-------------------------------------------------
1. | Laskar Pelangi | Nidji | 4.31
2. | Gundul-Gundul Pacul | Lagu Daerah | 2.1
3. | Sempurna | Andra and The Backbone | 4.35
-------------------------------------------------
Total Lagu: 3
```

Deskripsi:

Program ini sama seperti guided diatas yaitu dengan fungsi ADT, dengan memberikan struct data pada playlist.h, kemudian diimplementasikan ke playlist.cpp, menambah awal playlist dengan insertfirst, menambah playlist akhir dengan insertlast, menambah lagu setelah playlist ketiga dengan insertafter3, menghapus lagu dengan delP, kemudian pada main.cpp tinggal melakukan uji coba apakah fungsi ADT yang telah dibuat berhasil untuk dijalankan.

E. Kesimpulan

Pada codingan diatas membuktikan bahawa implementasi Singly Linked List memiliki keunggulan alokasi memori dinamis untuk pengelolaan data yang fleksibel, memungkinkan penyisipan dan penghapusan data secara efisien melalui manipulasi pointer.

F. Referensi

Moh. Sjukani. (2012). Struktur Data (Algoritma dan Struktur Data dengan C, C++). Jakarta: Mitra.

Horowitz, E. & Sahni, S. (1984). Fundamentals of Data Structures in Pascal. Pitman Publishing Limited.

Wirth, N. (1986). Algorithms & Data Structures. Prentice Hall.