LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL IV
SINGLY LINKED LIST

Disusun Oleh :

NAMA : AGUNG RAMADHAN

NIM : 103112430060

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Single Linked List merupakan struktur data linier yang dinamis dan fleksibel, tersusun dari serangkaian node yang tidak harus berdekatan di memori, di mana setiap node terdiri dari dua bagian: data (informasi yang disimpan) dan pointer (next) yang menyimpan alamat memori dari node berikutnya, sehingga membentuk sebuah 'rantai'. Seluruh daftar dikontrol oleh pointer Head yang menunjuk ke node pertama, dengan node terakhir memiliki pointer next yang bernilai NULL sebagai penanda akhir daftar. Keunggulan utamanya adalah kemampuannya untuk melakukan alokasi memori dinamis dan efisiensi waktu konstan ($O(1)$) untuk operasi penyisipan dan penghapusan di awal daftar. Namun, kelemahan mendasarnya adalah tidak adanya akses acak, yang berarti untuk mengakses elemen di tengah, seluruh daftar harus ditelusuri dari Head, menjadikan operasi pencarian dan akses elemen $O(n)$.

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

Guided 1

Kode SinglyList.h

```
// AGUNG RAMADHAN
// 103112430060

#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 fungsi dan prosedur
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
```

Kode SinglyList.cpp

```
// AGUNG RAMADHAN
// 103112430060

#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;
    }
}
```

Kode main.cpp

```
// AGUNG RAMADHAN
// 103112430060

#include <iostream>
#include <cstdlib>
#include "singlylist.h"
#include "singlylist.cpp"


using namespace std;
int main() {
    List L;
    address P; // Cukup satu pointer untuk digunakan berulang kali

    CreateList(L);

    cout << "Mengisis list menggunakan insertLast..." << endl;

    // Mengisi list sesuai dengan 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: ";
    printInfo(L);

    system("pause");
```

```
    return 0;
}
```

Screenshots Output

```
Mengisis list menggunakan insertLast...
Isi list sekarang: 9 12 8 0 2
Press any key to continue . . .
```

Deskripsi:

Kode ini merupakan implementasi struktur data singly linked list dalam bahasa C++ yang terdiri dari tiga file utama: singlylist.h, singlylist.cpp, dan main.cpp. File singlylist.h mendefinisikan struktur node (ElmtList) yang menyimpan data bertipe integer dan pointer ke elemen berikutnya, serta struktur List yang menyimpan pointer ke elemen pertama. File singlylist.cpp berisi implementasi fungsi-fungsi utama seperti CreateList() untuk menginisialisasi list kosong, alokasi() dan dealokasi() untuk manajemen memori dinamis, insertFirst() dan insertLast() untuk menambahkan elemen di awal atau akhir list, serta printInfo() untuk menampilkan isi list secara berurutan. Sedangkan main.cpp berfungsi sebagai program utama yang membuat list baru, menambahkan beberapa elemen menggunakan insertLast(), lalu mencetak hasilnya ke layar. Secara keseluruhan, program ini memperlihatkan bagaimana konsep linked list bekerja dalam mengelola data secara dinamis menggunakan pointer tanpa batasan ukuran seperti array, sekaligus menjadi dasar penting dalam memahami struktur data dan algoritma berbasis node.

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

Unguided 1

Kode Playlist.h

```
// AGUNG RAMADHAN
// 103112430060

#ifndef PLAYLIST_H
#define PLAYLIST_H

#include <iostream>
#include <string>
#include <algorithm> // untuk transform()
using namespace std;

struct Lagu {
    string namaLagu;
    string artis;
    float panjangWaktu;
    Lagu* next;
```

```
};

class Playlist {
private:
    Lagu* awal;
    string toLowerCase(const string& str); // helper function

public:
    Playlist();
    void insertFirst(const string& nama, const string& musisi, float
waktu);
    void insertLast(const string& nama, const string& musisi, float
waktu);
    void insertAfter3(const string& nama, const string& musisi, float
waktu);
    void deleteByTitle(const string& nama);
    void showPlaylist();
};

#endif
```

Kode Playlist.cpp

```
// AGUNG RAMADHAN
// 103112430060

#include "Playlist.h"

string Playlist::toLowerCase(const string& str) {
    string lower = str;
    transform(lower.begin(), lower.end(), lower.begin(), ::tolower);
    return lower;
}

Playlist::Playlist() {
    awal = nullptr;
}

void Playlist::insertFirst(const string& nama, const string& musisi,
float waktu) {
    Lagu* nodeBaru = new Lagu{nama, musisi, waktu, awal};
    awal = nodeBaru;
}

void Playlist::insertLast(const string& nama, const string& musisi, float
waktu) {
    Lagu* nodeBaru = new Lagu{nama, musisi, waktu, nullptr};
    if (!awal) {
        awal = nodeBaru;
```

```
        return;
    }
    Lagu* ptr = awal;
    while (ptr->next) {
        ptr = ptr->next;
    }
    ptr->next = nodeBaru;
}

void Playlist::insertAfter3(const string& nama, const string& musisi,
float waktu) {
    Lagu* ptr = awal;
    int hitung = 1;

    while (ptr && hitung < 3) {
        ptr = ptr->next;
        hitung++;
    }

    if (!ptr) {
        cout << "Daftar lagu kurang dari 3 lagu, tidak bisa menambah
setelah lagu ke-3.\n";
        return;
    }

    Lagu* nodeBaru = new Lagu{nama, musisi, waktu, ptr->next};
    ptr->next = nodeBaru;
}

void Playlist::deleteByTitle(const string& nama) {
    if (!awal) {
        cout << "Daftar lagu kosong.\n";
        return;
    }

    string targetHurufKecil = toLowerCase(nama);
    Lagu* ptr = awal;
    Lagu* sebelum = nullptr;

    while (ptr && toLowerCase(ptr->namaLagu) != targetHurufKecil) {
        sebelum = ptr;
        ptr = ptr->next;
    }

    if (!ptr) {
        cout << "Lagu dengan judul '" << nama << "' tidak ditemukan.\n";
        return;
    }
```

```
    if (!sebelum) {
        awal = ptr->next;
    } else {
        sebelum->next = ptr->next;
    }

    delete ptr;
    cout << "Lagu '" << nama << "' berhasil dihapus dari daftar.\n";
}

void Playlist::showPlaylist() {
    if (!awal) {
        cout << "Daftar lagu kosong.\n";
        return;
    }

    Lagu* ptr = awal;
    int i = 1;
    cout << "\n=== Daftar Lagu Playlist ===\n";
    while (ptr) {
        cout << i++ << ". Nama Lagu : " << ptr->namaLagu << endl;
        cout << "   Musisi    : " << ptr->artis << endl;
        cout << "   Waktu     : " << ptr->panjangWaktu << " menit\n";
        cout << "-----------------------------\n";
        ptr = ptr->next;
    }
}
    if (!ptr) {
        cout << "Daftar lagu kurang dari 3 lagu, tidak bisa menambah
setelah lagu ke-3.\n";
        return;
    }

    Lagu* nodeBaru = new Lagu{nama, musisi, waktu, ptr->next};
    ptr->next = nodeBaru;
}

void Playlist::deleteByTitle(const string& nama) {
    if (!awal) {
        cout << "Daftar lagu kosong.\n";
        return;
    }

    string targetHurufKecil = toLowerCase(nama);
    Lagu* ptr = awal;
    Lagu* sebelum = nullptr;
```

```
    while (ptr && toLowerCase(ptr->namaLagu) != targetHurufKecil) {
        sebelum = ptr;
        ptr = ptr->next;
    }

    if (!ptr) {
        cout << "Lagu dengan judul '" << nama << "' tidak ditemukan.\n";
        return;
    }

    if (!sebelum) {
        awal = ptr->next;
    } else {
        sebelum->next = ptr->next;
    }

    delete ptr;
    cout << "Lagu '" << nama << "' berhasil dihapus dari daftar.\n";
}

void Playlist::showPlaylist() {
    if (!awal) {
        cout << "Daftar lagu kosong.\n";
        return;
    }

    Lagu* ptr = awal;
    int i = 1;
    cout << "\n=== Daftar Lagu Playlist ===\n";
    while (ptr) {
        cout << i++ << ". Nama Lagu : " << ptr->namaLagu << endl;
        cout << "   Musisi    : " << ptr->artis << endl;
        cout << "   Waktu     : " << ptr->panjangWaktu << " menit\n";
        cout << "-----------------------------\n";
        ptr = ptr->next;
    }
}
```

Kode main.cpp

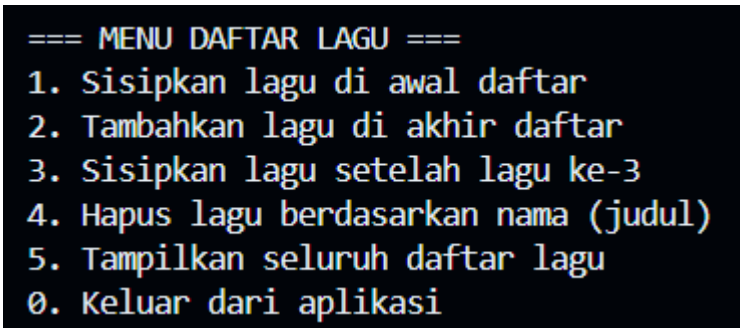
```
// Agung Ramadhan
// 103112430060

#include "Playlist.h"

int main() {
    Playlist songList;
```

```
    songList.insertLast("Alexandra", "Hindia", 4.1);
    songList.insertLast("Amin Paling Serius", "Sal Priadi", 4.5);
    songList.insertLast("Alamak", "Rizky Febian", 4.0);
    songList.insertLast("Everything You Are", "Hindia", 4.7);
    songList.insertLast("Kita Usahakan Rumah Itu", "Sal Priadi", 5.2);

    int aksi;
    string judulLagu, penyanyiLagu;
    float waktu;

    do {
        cout << "\n=== MENU DAFTAR LAGU ===\n";
        cout << "1. Sisipkan lagu di awal daftar\n";
        cout << "2. Tambahkan lagu di akhir daftar\n";
        cout << "3. Sisipkan lagu setelah lagu ke-3\n";
        cout << "4. Hapus lagu berdasarkan nama (judul)\n";
        cout << "5. Tampilkan seluruh daftar lagu\n";
        cout << "0. Keluar dari aplikasi\n";
        cout << "Pilih: ";
        cin >> aksi;
        cin.ignore();

        switch (aksi) {
        case 1:
            cout << "Masukkan nama lagu: "; getline(cin, judulLagu);
            cout << "Masukkan nama penyanyi: "; getline(cin,
penyanyiLagu);
            cout << "Masukkan waktu (menit): "; cin >> waktu;
            songList.insertFirst(judulLagu, penyanyiLagu, waktu);
            break;

        case 2:
            cout << "Masukkan nama lagu: "; getline(cin, judulLagu);
            cout << "Masukkan nama penyanyi: "; getline(cin,
penyanyiLagu);
            cout << "Masukkan waktu (menit): "; cin >> waktu;
            songList.insertLast(judulLagu, penyanyiLagu, waktu);
            break;

        case 3:
            cout << "Masukkan nama lagu: "; getline(cin, judulLagu);
            cout << "Masukkan nama penyanyi: "; getline(cin,
penyanyiLagu);
            cout << "Masukkan waktu (menit): "; cin >> waktu;
            songList.insertAfter3(judulLagu, penyanyiLagu, waktu);
            break;

        case 4:
```

```
            cout << "Masukkan nama lagu yang ingin dihapus: ";
            getline(cin, judulLagu);
            songList.deleteByTitle(judulLagu);
            break;

        case 5:
            songList.showPlaylist();
            break;

        case 0:
            cout << "Keluar dari program.\n";
            break;

        default:
            cout << "Pilihan tidak valid.\n";
        }

    } while (aksi != 0);

    return 0;
}
```

Screenshots Output

Menampilkan Output berupa fitur playlist



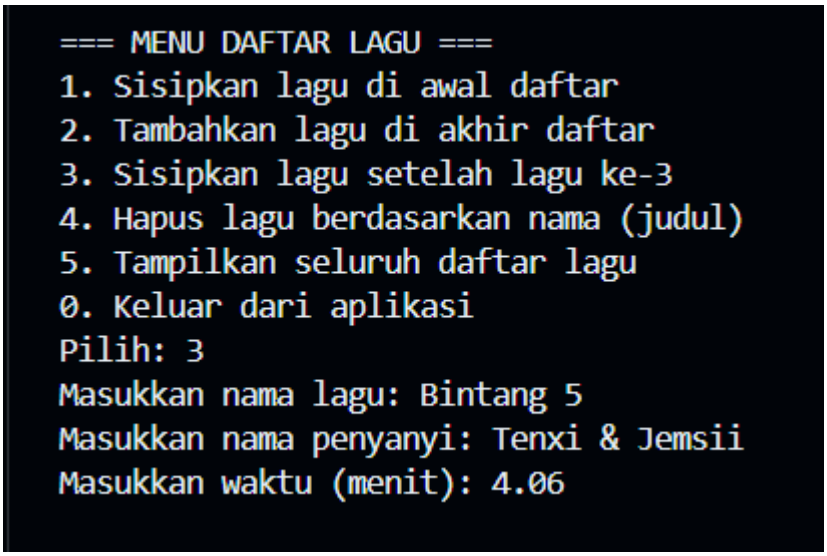
Menampilkan output pada menu 1 yaitu menambahkan lagu di awal playlist

```
=== MENU DAFTAR LAGU ===
1. Sisipkan lagu di awal daftar
2. Tambahkan lagu di akhir daftar
3. Sisipkan lagu setelah lagu ke-3
4. Hapus lagu berdasarkan nama (judul)
5. Tampilkan seluruh daftar lagu
0. Keluar dari aplikasi
Pilih: 1
Masukkan nama lagu: Cincin
Masukkan nama penyanyi: Hindia
Masukkan waktu (menit): 4.26
```

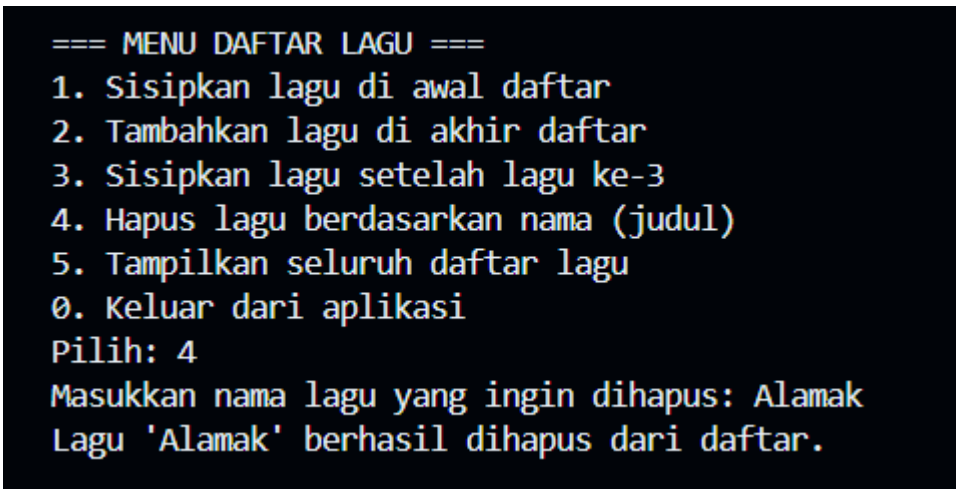
Menampilkan output pada menu 2 yaitu menambahkan lagu di akhir playlist

```
=== MENU DAFTAR LAGU ===
1. Sisipkan lagu di awal daftar
2. Tambahkan lagu di akhir daftar
3. Sisipkan lagu setelah lagu ke-3
4. Hapus lagu berdasarkan nama (judul)
5. Tampilkan seluruh daftar lagu
0. Keluar dari aplikasi
Pilih: 2
Masukkan nama lagu: o,Tuan
Masukkan nama penyanyi: .Feast
Masukkan waktu (menit): 5.06
```

Menampilkan Menu 3 yaitu menyisipkan lagu setelelah lagu ke-3



Menampilkan output pada menu 4 yaitu menghapus lagu yang ada diplaylis berdasarkan judul lagu



Menapilkan seluruh daftar lagu setelah menambahkan dan menghapus lagu yang kita pilih pada fitur ke-5

```
=== MENU DAFTAR LAGU ===
1. Sisipkan lagu di awal daftar
2. Tambahkan lagu di akhir daftar
3. Sisipkan lagu setelah lagu ke-3
4. Hapus lagu berdasarkan nama (judul)
5. Tampilkan seluruh daftar lagu
0. Keluar dari aplikasi
Pilih: 5

=== Daftar Lagu Playlist ===
1. Nama Lagu : Cincin
   Musisi    : Hindia
   Waktu     : 4.26 menit
----------------------------
2. Nama Lagu : Alexandra
   Musisi    : Hindia
   Waktu     : 4.1 menit
----------------------------
3. Nama Lagu : Amin Paling Serius
   Musisi    : Sal Priadi
   Waktu     : 4.5 menit
----------------------------
4. Nama Lagu : Bintang 5
   Musisi    : Tenxi & Jemsii
   Waktu     : 4.06 menit
----------------------------
5. Nama Lagu : Everything You Are
   Musisi    : Hindia
   Waktu     : 4.7 menit
----------------------------
6. Nama Lagu : Kita Usahakan Rumah Itu
   Musisi    : Sal Priadi
   Waktu     : 5.2 menit
----------------------------
7. Nama Lagu : o,Tuan
   Musisi    : .Feast
   Waktu     : 5.06 menit
----------------------------

=== MENU DAFTAR LAGU ===
1. Sisipkan lagu di awal daftar
2. Tambahkan lagu di akhir daftar
3. Sisipkan lagu setelah lagu ke-3
4. Hapus lagu berdasarkan nama (judul)
5. Tampilkan seluruh daftar lagu
0. Keluar dari aplikasi
```

Deskripsi:

Program ini merupakan aplikasi pengelola playlist lagu sederhana menggunakan konsep struktur data linked list dalam C++. File Playlist.h berfungsi sebagai deklarasi struktur dan kelas, di mana struct Lagu menyimpan informasi setiap lagu seperti nama lagu, artis, durasi, serta pointer next untuk menghubungkan ke lagu berikutnya. Kelas Playlist berisi pointer awal yang menandai elemen pertama dalam daftar serta beberapa fungsi operasi. Di Playlist.cpp, fungsi-fungsi tersebut diimplementasikan: insertFirst() menambahkan lagu di awal, insertLast() menambah di akhir, insertAfter3() menyisipkan setelah lagu ketiga, deleteByTitle() menghapus lagu tertentu berdasarkan judul dengan pencocokan tidak sensitif huruf besar kecil (melalui fungsi bantu toLowerCase()), dan showPlaylist() menampilkan semua lagu secara berurutan. Sementara itu, main.cpp mengatur antarmuka berbasis teks (menu) yang memungkinkan pengguna menambah, menghapus, menampilkan, atau keluar dari program. Seluruh sistem ini menggambarkan cara kerja dinamis dari *linked list*, di mana penambahan dan penghapusan elemen tidak memerlukan pergeseran data seperti pada array, menjadikannya efisien untuk manipulasi daftar lagu yang sering berubah.

D. Kesimpulan

Secara keseluruhan, kode ini menunjukkan penerapan **struktur data linked list** dalam bentuk nyata melalui simulasi **pengelolaan playlist lagu**. Program dirancang modular dengan pemisahan antara deklarasi, implementasi, dan fungsi utama, sehingga mudah dibaca dan dikembangkan. Melalui berbagai operasi seperti menambah, menyisipkan, menghapus, dan menampilkan lagu, program ini memperlihatkan bagaimana pointer bekerja dalam menghubungkan node-node data secara dinamis tanpa batasan ukuran tertentu seperti pada array. Kesimpulannya, kode ini tidak hanya berfungsi sebagai alat sederhana untuk mengelola daftar lagu, tetapi juga menjadi contoh yang efektif untuk memahami konsep dasar *linked list*, manipulasi memori dinamis, serta penerapan prinsip OOP (Object-Oriented Programming) dalam bahasa C++.

E. Referensi

Linked list. Diakses dari: https://en.wikipedia.org/wiki/Linked_list

Introduction to Linked List. Diakses dari: https://www.geeksforgeeks.org/introduction-to-linked-list-data-structure-and-algorithm-tutorials/

Speeding Up Search in Singly Linked List — Sarvesh Rakesh Bhatnagar. *International Journal of Computer Applications*, Vol. 182, No. 18, 2018, pp. 19-24.

Fundamental of Singly Linked List Creation and Insertion and Analysis — Sanjay S. Reddy, Saddanand Rao, Peruwam Thomas. International Journal of Humanities and Information Technology, Vol. 1 No. 02, 2016.

The Doubly Linked Tree of Singly Linked Rings: Providing Hard Real-Time Database Operations on an FPGA — Simon Lohmann & Dietmar Tutsch. Computers, 2024, 13(1):8.