LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 4
SINGLY LINKED LIST (BAGIAN PERTAMA)

Disusun Oleh :

NAMA : Herdian Abdillah Purnomo

NIM : 103112430048

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Singly Linked List adalah struktur data dinamis yang terdiri dari kumpulan node, di mana setiap node berisi data dan pointer yang menunjuk ke node berikutnya. Struktur ini memungkinkan penambahan dan penghapusan data secara fleksibel tanpa perlu memindahkan elemen lain seperti pada array. Dalam praktikum ini, singly linked list digunakan untuk mengelola playlist lagu yang menyimpan data berupa judul, penyanyi, dan durasi. Dengan konsep pointer, setiap lagu dapat dihubungkan secara berurutan membentuk daftar lagu yang dapat ditambah atau dihapus secara dinamis.

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; // Cukup satu pointer untuk digunakan berulang kali

    createList(L);

    cout << "Mengisi list menggunaka 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 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 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) {
        // Jika list kosong, interLast sama dengan insertFirst
        insertFirst(L, P);
    } else {
        // Jika list tidak kosong, cari element terakhir
        address Last = L.first;
        while (Last->next != NIL) {
            Last = Last->next;
```

```
        }
        // Sambungkan elemen terakhir dengan elemen terbaru (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 SINGLYLISH_H_INLCLUDED
#define SINGLYLISH_H_INLCLUDED
#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
```

Screenshots Output

```
PS C:\Users\Lenovo\Documents\PRAKTIKUM STRUKDAT\FILE\MODUL 4\GUIDED> cd "c:\Users\Lenovo\Documents\PRAKTIKUM STRUKDAT\FILE\MODUL 4\GUI
DED\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Mengisi list menggunaka interLast...
Isi list sekarang adalah: 9 12 8 0 2
Press any key to continue . . .
```

Deskripsi:

Program Guided 1 menunjukkan cara dasar membuat singly linked list menggunakan operasi seperti createList, alokasi, insertLast, dan printInfo. Program menambahkan beberapa data ke dalam list lalu menampilkannya di layar. Melalui percobaan ini, mahasiswa belajar bagaimana node saling terhubung melalui pointer serta bagaimana traversal dilakukan untuk menampilkan isi list

.

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

Unguided 1

main.cpp

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

using namespace std;

int main() {
    Playlist playlist;
    int pilihan;
    string judul, penyanyi;
    float durasi;

    do {
        cout << "\n=== MENU PLAYLIST LAGU ===\n";
        cout << "1. Tambah lagu di awal\n";
        cout << "2. Tambah lagu di akhir\n";
        cout << "3. Tambah lagu setelah lagu ke-3\n";
```

```
        cout << "4. Hapus lagu berdasarkan judul\n";
        cout << "5. Tampilkan playlist\n";
        cout << "0. Keluar\n";
        cout << "Pilih menu: ";
        cin >> pilihan;
        cin.ignore();

        switch (pilihan) {
            case 1:
                cout << "Judul lagu   : ";
                getline(cin, judul);
                cout << "Penyanyi     : ";
                getline(cin, penyanyi);
                cout << "Durasi (menit): ";
                cin >> durasi;
                cin.ignore();
                playlist.tambahDepan(judul, penyanyi, durasi);
                break;

            case 2:
                cout << "Judul lagu   : ";
                getline(cin, judul);
                cout << "Penyanyi     : ";
                getline(cin, penyanyi);
                cout << "Durasi (menit): ";
                cin >> durasi;
                cin.ignore();
                playlist.tambahBelakang(judul, penyanyi, durasi);
                break;

            case 3:
                cout << "Judul lagu   : ";
                getline(cin, judul);
                cout << "Penyanyi     : ";
                getline(cin, penyanyi);
                cout << "Durasi (menit): ";
                cin >> durasi;
                cin.ignore();
                playlist.tambahSetelahKe3(judul, penyanyi,
durasi);
                break;

            case 4:
```

```
                cout << "Masukkan judul lagu yang akan dihapus: ";
                getline(cin, judul);
                playlist.hapusLagu(judul);
                break;

            case 5:
                playlist.tampilkan();
                break;

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

            default:
                cout << "Pilihan tidak valid.\n";
        }
    } while (pilihan != 0);

    return 0;
}
```

Playlist.cpp

```
#include "Playlist.h"

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

Playlist::~Playlist() {
    Lagu* temp;
    while (head != nullptr) {
        temp = head;
        head = head->next;
        delete temp;
    }
}

void Playlist::tambahDepan(string judul, string penyanyi, float
durasi) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, head};
    head = baru;
    cout << "Lagu \"" << judul << "\" berhasil ditambahkan di awal
```

```
playlist.\n";
}

void Playlist::tambahBelakang(string judul, string penyanyi, float
durasi) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, nullptr};
    if (head == nullptr) {
        head = baru;
    } else {
        Lagu* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = baru;
    }
    cout << "Lagu \"" << judul << "\" berhasil ditambahkan di
akhir playlist.\n";
}

void Playlist::tambahSetelahKe3(string judul, string penyanyi,
float durasi) {
    if (head == nullptr) {
        cout << "Playlist kosong. Tidak dapat menambahkan setelah
lagu ke-3.\n";
        return;
    }

    Lagu* temp = head;
    int count = 1;

    while (temp != nullptr && count < 3) {
        temp = temp->next;
        count++;
    }

    if (temp == nullptr) {
        cout << "Playlist kurang dari 3 lagu.\n";
    } else {
        Lagu* baru = new Lagu{judul, penyanyi, durasi,
temp->next};
        temp->next = baru;
        cout << "Lagu \"" << judul << "\" berhasil ditambahkan
setelah lagu ke-3.\n";
```

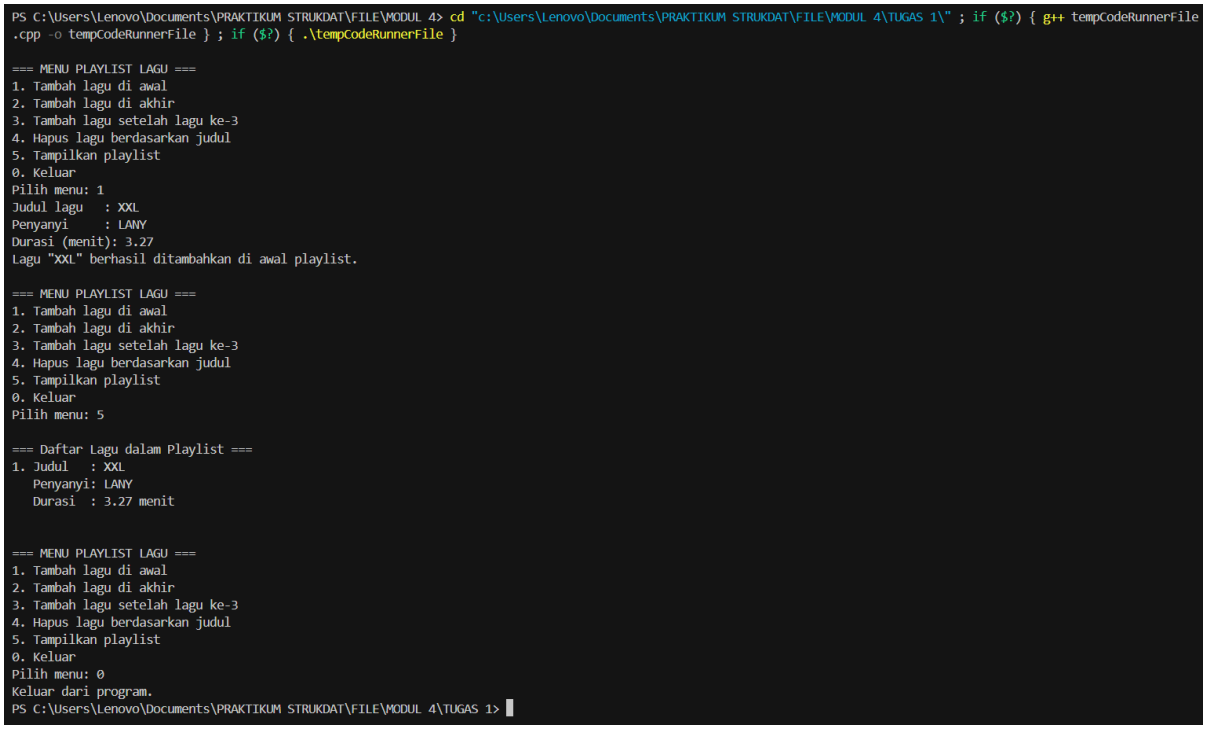
```
    }
}

void Playlist::hapusLagu(string judul) {
    if (head == nullptr) {
        cout << "Playlist kosong.\n";
        return;
    }

    if (head->judul == judul) {
        Lagu* hapus = head;
        head = head->next;
        delete hapus;
        cout << "Lagu \"" << judul << "\" berhasil dihapus.\n";
        return;
    }

    Lagu* temp = head;
    while (temp->next != nullptr && temp->next->judul != judul) {
        temp = temp->next;
    }

    if (temp->next == nullptr) {
        cout << "Lagu \"" << judul << "\" tidak ditemukan.\n";
    } else {
        Lagu* hapus = temp->next;
        temp->next = hapus->next;
        delete hapus;
        cout << "Lagu \"" << judul << "\" berhasil dihapus.\n";
    }
}

void Playlist::tampilkan() {
    if (head == nullptr) {
        cout << "Playlist kosong.\n";
        return;
    }

    Lagu* temp = head;
    int no = 1;
    cout << "\n=== Daftar Lagu dalam Playlist ===\n";
    while (temp != nullptr) {
        cout << no++ << ". Judul   : " << temp->judul << endl;
```

```
        cout << "   Penyanyi: " << temp->penyanyi << endl;
        cout << "   Durasi  : " << temp->durasi << " menit\n\n";
        temp = temp->next;
    }
}
```

Playlist.h

```
#ifndef PLAYLIST_H
#define PLAYLIST_H

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

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

class Playlist {
private:
    Lagu* head;

public:
    Playlist();
    ~Playlist();

    void tambahDepan(string judul, string penyanyi, float durasi);
    void tambahBelakang(string judul, string penyanyi, float
durasi);
    void tambahSetelahKe3(string judul, string penyanyi, float
durasi);
    void hapusLagu(string judul);
    void tampilkan();
};

#endif
```

Screenshots Output



Deskripsi:

Program Unguided 1 menerapkan konsep singly linked list untuk mengelola playlist lagu. Program dibagi menjadi tiga file, yaitu Playlist.h, Playlist.cpp, dan main.cpp. Setiap node menyimpan judul, penyanyi, dan durasi lagu. Program dapat menambah lagu di awal, akhir, atau setelah lagu ke-3, menghapus lagu berdasarkan judul, serta menampilkan seluruh lagu dalam playlist. Percobaan ini menunjukkan penerapan linked list dalam kehidupan nyata untuk manajemen data dinamis seperti daftar lagu.

D. Kesimpulan

1. Singly Linked List memungkinkan pengelolaan data yang fleksibel karena elemen dapat ditambah atau dihapus tanpa perlu memindahkan data lain.
2. Struktur Linked List sangat berguna untuk aplikasi dinamis seperti playlist lagu, di mana jumlah data bisa berubah-ubah.
3. Pemahaman konsep pointer dan relasi antar-node sangat penting agar operasi seperti insert dan delete dapat dilakukan dengan benar.
4. Pemisahan kode menjadi file header (.H), implementasi (.cpp), dan main.cpp membuat program lebih terstruktur dan mudah dikelola.

E. Referensi

Nugroho, Adi. *Algoritma dan Struktur Data dalam Pemrograman C++*. Andi Offset, 2020.

Wibowo, Fahrudin Mukti. *Modul Praktikum Struktur Data - Telkom University Purwokerto*, 2025.

GeeksforGeeks. "Singly Linked List in C++." (https://www.geeksforgeeks.org/data-structures/linked-list/singly-linked-list/)

Tutorialspoint. "C++ Linked List." (https://www.tutorialspoint.com/cplusplus-program-to-implement-singly-linked-list)