LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 4 & 5
SINGLY LINKED LIST

Disusun Oleh :
NAMA : Muhammad Luthfi Arrafi Ramadhani
NIM : 103112430043

Dosen Pengampu:
Fahrudin Mukti Wibowo
PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Dasar teori pada modul ini membahas konsep Singly Linked List, yaitu struktur data yang terdiri dari beberapa node yang saling terhubung menggunakan pointer. Setiap node menyimpan data dan penunjuk ke node berikutnya. Struktur ini bersifat dinamis, artinya jumlah datanya bisa bertambah atau berkurang sesuai kebutuhan. Operasi yang bisa dilakukan antara lain menambah, menghapus, menampilkan, dan mencari data dalam list. Singly Linked List sering digunakan untuk menyimpan data yang berubah-ubah.

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

Guided 1

(singlylist.h)

```
#ifndef LIST_H_INCLUDED
#define LIST_H_INCLUDED
#include <iostream>
#define Nil NULL
using namespace std;

/*deklarasi record dan struktur data list*/
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
```

Deskripsi:

File singlylist.h berisi berisi deklarasi struktur data dan fungsi yang digunakan dalam program. Di dalamnya terdapat struktur *elmlist* yang menyimpan data (*info*) dan penunjuk ke elemen berikutnya (*next*). Ada juga struktur *list* yang menyimpan alamat elemen pertama (*first*). Selain itu, file ini juga berisi deklarasi fungsi seperti createList, alokasi, dealokasi, insertFirst, insertLast, dan printInfo.

(singlylist.cpp)

```
#include "singlylist.h"

// Membuat sebuah list baru yang masih kosong
void createList(list &L){
   L.first = Nil;
}

// Mengalokasikan memori untuk sebuah elemen baru
// dan mengisi elemen tersebut dengan data (X)
address alokasi(infotype X){
   address P = new elmlist;
   P->info = X;
   P->next = Nil;
   return P;
}

// Menghapus atau membebaskan memori dari sebuah elemen
void dealokasi(address &P){
   delete P;
}

// Menambahkan sebuah elemen baru di awal list
void insertFirst(list &L, address P){
   P->next = L.first;
   L.first = P;
}

// Menambahkan sebuah elemen baru di akhir list
void insertLast(list &L, address P){
   if(L.first == Nil){
      // Jika list masih kosong, elemen baru menjadi elemen pertama
      insertFirst(L, P);
   } else {
      // Jika list sudah ada isinya, cari elemen terakhir
      address Last = L.first;
      while (Last->next != Nil){
         Last = Last->next;
      }
      // Sambungkan elemen terakhir dengan elemen baru
      Last->next = P;
   }
}

// Menampilkan semua isi data dari setiap elemen di list
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;
  }
}
```

Deskripsi:

File singlylist.cpp ini berisi implementasi dari fungsi-fungsi yang ada di file header. Fungsi *createList* digunakan untuk membuat list kosong, *alokasi* untuk membuat node baru, dan *dealokasi* untuk menghapus node. Fungsi *insertFirst* dan *insertLast* dipakai untuk menambah elemen di awal atau di akhir list, sedangkan *printInfo* digunakan untuk menampilkan semua data yang ada di list.

(main.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

Deskripsi:

File main.cpp ini berisi program utama untuk menjalankan dan menampilkan hasil dari fungsi-fungsi yang telah dibuat. Program diawali dengan membuat list kosong menggunakan *createList*, lalu menambah beberapa data menggunakan *insertLast*. Setelah itu, fungsi *printInfo* digunakan untuk menampilkan isi list ke layar.

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

Unguided 1

(Playlist.h)

```
// Muhammad Luthfi Arrafi Ramadhani
// 103112430043
// IF 12-06

#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
```

Deskripsi:

File Playlist.h ini berisi definisi struktur data dan deklarasi fungsi yang digunakan dalam program. Di dalamnya terdapat struktur Lagu yang menyimpan data berupa judul, penyanyi, dan durasi lagu, serta pointer *next* untuk menghubungkan ke lagu berikutnya. File ini juga mendeklarasikan fungsi-fungsi seperti *tambahDepan*, *tambahBelakang, tambahSetelahKe3, hapusLagu,* dan *tampilkan* yang akan diimplementasikan di file berikutnya.

(Playlist.cpp)

```
// Muhammad Luthfi Arrafi Ramadhani
// 103112430043
// IF 12-06

#include "Playlist.h"

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

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

void Playlist::tambahDepan(string judul, string penyanyi, float durasi) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, nullptr};
    baru->next = 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) {
    Lagu* baru = new Lagu{judul, penyanyi, durasi, nullptr};

    if (head == nullptr) {
        cout << "Playlist masih kosong.\n";
        delete baru;
        return;
    }

    Lagu* temp = head;
    int posisi = 1;
    while (temp != nullptr && posisi < 3) {
        temp = temp->next;
        posisi++;
    }

    if (temp == nullptr) {
        cout << "Jumlah lagu kurang dari 3, tidak bisa menambah setelah lagu ke-3.\n";
        delete baru;
        return;
    }

    baru->next = 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;
    }

    Lagu* temp = head;
    Lagu* prev = nullptr;

    while (temp != nullptr && temp->judul != judul) {
        prev = temp;
        temp = temp->next;
    }

    if (temp == nullptr) {
```

```
        cout << "Lagu dengan judul \"" << judul << "\" tidak ditemukan.\n";
        return;
    }

    if (prev == nullptr) {
        head = temp->next;
    } else {
        prev->next = temp->next;
    }

    delete temp;
    cout << "Lagu \"" << judul << "\" berhasil dihapus dari playlist.\n";
}

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

    Lagu* temp = head;
    int i = 1;
    cout << "\n=== Daftar Lagu di Playlist ===\n";
    while (temp != nullptr) {
        cout << i << ". Judul: " << temp->judul
             << " | Penyanyi: " << temp->penyanyi
             << " | Durasi: " << temp->durasi << " menit\n";
        temp = temp->next;
        i++;
    }
    cout << "==============================\n";
}
```

Deskripsi:

File Playlist.cpp ini berisi implementasi dari semua fungsi yang telah dideklarasikan di file header. Di dalamnya terdapat kode untuk menambah lagu di awal, di akhir, dan setelah lagu ke-3, menghapus lagu berdasarkan judul, serta menampilkan seluruh isi dari playlist.

(main.cpp)

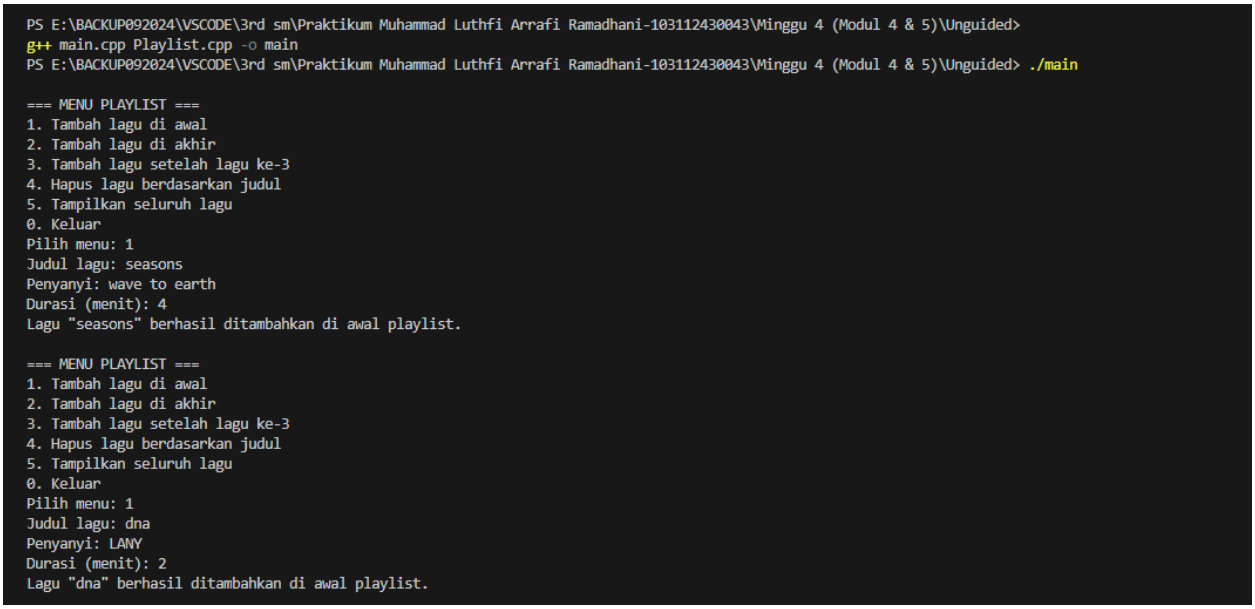
```
// Muhammad Luthfi Arrafi Ramadhani
// 103112430043
// IF 12-06

#include "Playlist.h"

int main() {
```

```
Playlist playlist;
int pilihan;
string judul, penyanyi;
float durasi;

do {
    cout << "\n=== MENU PLAYLIST ===\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 seluruh lagu\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;
            playlist.tambahDepan(judul, penyanyi, durasi);
            break;

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

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

        case 4:
            cout << "Masukkan judul lagu yang ingin dihapus: ";
            getline(cin, judul);
            playlist.hapusLagu(judul);
            break;

        case 5:
            playlist.tampilkan();
            break;

        case 0:
```

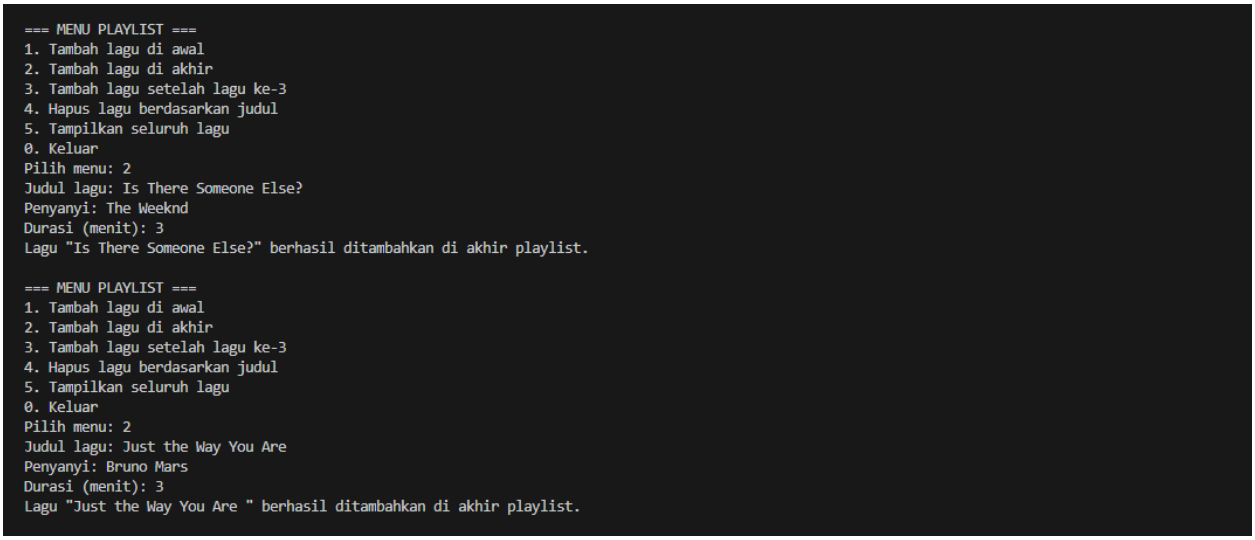
```
            cout << "Terima kasih! Program selesai.\n";
            break;

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

return 0;
}
```

Screenshots Output

Tambah lagu di awal playlist.



Tambah lagu di akhir playlist.



Tambah lagu setelah playlist ke-3.

```
=== MENU PLAYLIST ===
1. Tambah lagu di awal
2. Tambah lagu di akhir
3. Tambah lagu setelah lagu ke-3
4. Hapus lagu berdasarkan judul
5. Tampilkan seluruh lagu
0. Keluar
Pilih menu: 3
Judul lagu: Best Part
Penyanyi: Daniel Caesar
Durasi (menit): 3
Lagu "Best Part" berhasil ditambahkan setelah lagu ke-3.
```

Hapus lagu berdasarkan judul.

```
=== MENU PLAYLIST ===
1. Tambah lagu di awal
2. Tambah lagu di akhir
3. Tambah lagu setelah lagu ke-3
4. Hapus lagu berdasarkan judul
5. Tampilkan seluruh lagu
0. Keluar
Pilih menu: 4
Masukkan judul lagu yang ingin dihapus: Just the Way You Are
Lagu "Just the Way You Are " berhasil dihapus dari playlist.
```

Tampilkan seluruh lagu dalam playlist.

```
=== MENU PLAYLIST ===
1. Tambah lagu di awal
2. Tambah lagu di akhir
3. Tambah lagu setelah lagu ke-3
4. Hapus lagu berdasarkan judul
5. Tampilkan seluruh lagu
0. Keluar
Pilih menu: 5

=== Daftar Lagu di Playlist ===
1. Judul: dna | Penyanyi: LANY | Durasi: 2 menit
2. Judul: seasons | Penyanyi: wave to earth | Durasi: 4 menit
3. Judul: Is There Someone Else? | Penyanyi: The Weeknd | Durasi: 3 menit
4. Judul: Best Part | Penyanyi: Daniel Caesar | Durasi: 3 menit
============================

=== MENU PLAYLIST ===
1. Tambah lagu di awal
2. Tambah lagu di akhir
3. Tambah lagu setelah lagu ke-3
4. Hapus lagu berdasarkan judul
5. Tampilkan seluruh lagu
0. Keluar
Pilih menu:
```


Deskripsi:

File main.cpp ini merupakan program utama yang menjalankan semua fungsi dari playlist. Di sini terdapat menu yang memungkinkan pengguna menambah, menghapus, dan menampilkan lagu secara langsung. File ini berfungsi untuk menghubungkan seluruh bagian program sehingga pengguna bisa berinteraksi dengan playlist melalui pilihan menu yang disediakan.

D. Kesimpulan

Dari latihan membuat program Single Linked List untuk playlist lagu di atas, dapat disimpulkan bahwa struktur data tersebut memudahkan dalam menambah, menghapus, dan menampilkan data secara fleksibel. Dengan memanfaatkan pointer, setiap lagu dapat dihubungkan satu sama lain tanpa perlu menentukan jumlah data dari awal. Program ini juga membantu memahami cara kerja penyimpanan data yang dinamis serta konsep dasar operasi pada linked list di C++.

E. Referensi

Wijoyo, A., Prayudi, L. A., Fiqih, M., Santoso, R. D., Setiawan, R. T., & Farhan, A. (2024). *Penggunaan algoritma doubly linked list untuk insertion dan deletion*. *JRIIN: Jurnal Riset Informatika dan Inovasi, 1*(12), 1329–1331. https://jurnalmahasiswa.com/index.php/jriin/article/view/1282

Durán, E. (2021). *Understanding singly linked lists and some of their uses*. *Enmascript.com.* https://enmascript.com/articles/2021/04/12/understanding-singly-linked-lists-and-some-of-their-uses/