LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL IV
SINGLY LINKED LIST 1

Disusun Oleh :
NAMA : Zidane Aji Noegroho
NIM : 103112430006

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Algoritma merupakan fondasi dalam pembuatan program komputer. Secara sederhana, algoritma adalah serangkaian langkah logis dan sistematis yang disusun untuk menyelesaikan suatu masalah. C++, berfungsi sebagai alat untuk mengimplementasikan algoritma tersebut agar dapat dimengerti dan dieksekusi oleh komputer. C++ sering digunakan sebagai bahasa pengantar untuk mempelajari konsep pemrograman dasar karena strukturnya yang terorganisir dan kemampuannya untuk menangani operasi tingkat rendah.

Untuk membangun logika dalam program sesuai dengan algoritma yang dirancang, C++ menyediakan struktur kontrol, yang terbagi menjadi dua jenis utama:

Struktur Percabangan (Conditional): Digunakan untuk pengambilan keputusan, di mana program akan menjalankan blok kode tertentu jika suatu kondisi terpenuhi. Struktur ini mencakup if-else untuk mengevaluasi kondisi boolean dan switch-case untuk memilih blok kode berdasarkan nilai dari sebuah variabel.

Struktur Perulangan (Looping): Digunakan untuk mengeksekusi blok kode yang sama secara berulang kali selama kondisi tertentu masih terpenuhi. C++ menyediakan tiga jenis perulangan utama: for, while, dan do-while, yang masing-masing memiliki karakteristik penggunaan yang spesifik dalam implementasi algoritma.

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

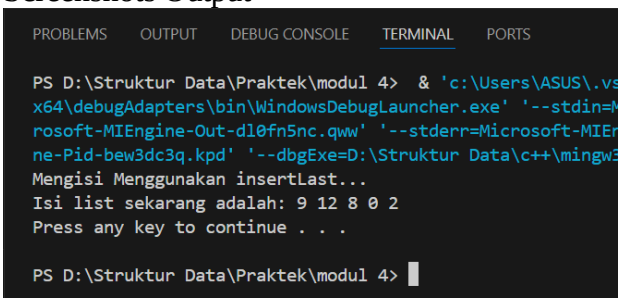
Guided 1

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 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:

Berisi program utama. Di sini sebuah list (linked list) dibuat, lalu diisi angka-angka menggunakan fungsi insertLast. Program mulai dengan membuat list kosong memakai CreateList(L). Setelah itu, beberapa node baru dibuat memakai alokasi(…), lalu dimasukkan ke bagian belakang list. Setiap kali membuat node baru, nilainya diisi (misalnya 9, 12, 8, dst). Setelah semua nilai dimasukkan, list ditampilkan dengan PrintInfo(L) sehingga semua angka yang sudah diinput muncul berurutan. Terakhir program berhenti setelah menunggu input (system("pause")).

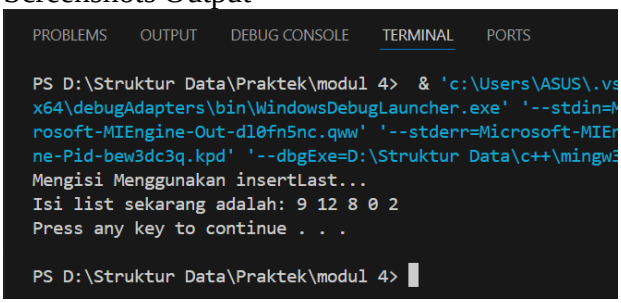
Singlylist.cpp

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

```
}
```

Screenshots Output



Deskripsi:

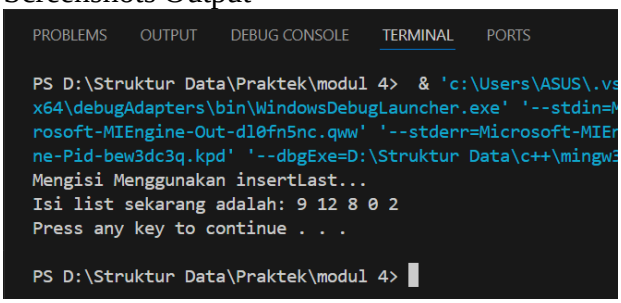
berisi implementasi fungsi-fungsi untuk mengelola linked list. CreateList membuat list kosong dengan mengatur pointer awal menjadi NULL. alokasi membuat node baru di memori, mengisi nilai info, dan next = NULL. dealokasi menghapus node dari memori. insertFirst menambahkan node di bagian depan list dengan mengarahkan next dari node baru ke head list, lalu memindahkan head ke node baru. insertLast menambahkan node di bagian belakang list: jika list kosong, cukup gunakan insertFirst; jika tidak, iterasi sampai node terakhir lalu sambungkan node baru di ujung. PrintInfo mencetak isi list dari depan sampai akhir. Jika list kosong, akan menulis “List Kosong!”.

Singlylist.h

```h
#ifndef SINGLYLIST_H_INCLUDED
#define SINGLYLIST_H_INCLUDED

#include <iostream>

#define Nil NULL

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
```

Screenshots Output



Deskripsi:

header yang mendefinisikan struktur data dan deklarasi fungsi. Di sini didefinisikan tipe data node (ElmList) yang menyimpan integer dan pointer ke node berikutnya. Struktur List menyimpan pointer head bernama First. Bagian bawah file ini berisi deklarasi fungsi seperti CreateList, alokasi, dealokasi, insertFirst, insertLast, dan printInfo, sehingga file lain bisa menggunakannya. File ini juga mendefinisikan Nil sebagai NULL untuk mempermudah penulisan.

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

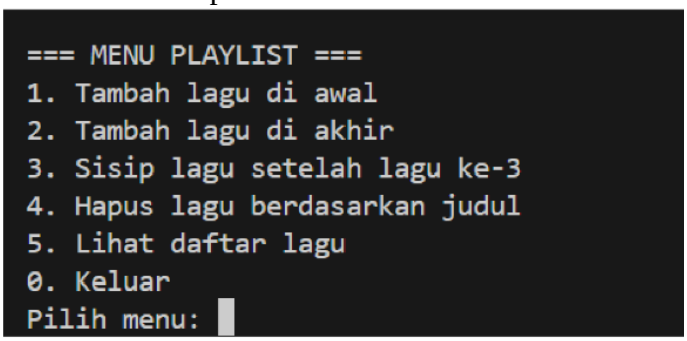
Unguided (Main.cpp)

```
#include "Playlist.h"
#include <iostream>
#include <limits>

using namespace std;

int main() {
    Playlist daftarLagu;
    int menu;
    string judul, artis;
    float waktu;

    do {
        cout << "\n=== MENU PLAYLIST ===\n";
        cout << "1. Tambah lagu di awal\n";
        cout << "2. Tambah lagu di akhir\n";
        cout << "3. Sisip lagu setelah lagu ke-3\n";
        cout << "4. Hapus lagu berdasarkan judul\n";
        cout << "5. Lihat daftar lagu\n";
        cout << "0. Keluar\n";
        cout << "Pilih menu: ";
        cin >> menu;
        cin.ignore(); // menghapus newline agar getline tidak terlewat
```

```
        switch (menu) {
            case 1:
                cout << "Judul lagu   : ";
                getline(cin, judul);
                cout << "Artis        : ";
                getline(cin, artis);
                cout << "Durasi (menit): ";
                cin >> waktu;
                daftarLagu.tambahDiAwal(judul, artis, waktu);
                break;

            case 2:
                cout << "Judul lagu   : ";
                getline(cin, judul);
                cout << "Artis        : ";
                getline(cin, artis);
                cout << "Durasi (menit): ";
                cin >> waktu;
                daftarLagu.tambahDiAkhir(judul, artis, waktu);
                break;

            case 3:
                cout << "Judul lagu   : ";
                getline(cin, judul);
                cout << "Artis        : ";
                getline(cin, artis);
                cout << "Durasi (menit): ";
                cin >> waktu;
                daftarLagu.sisipSetelahKe3(judul, artis, waktu);
                break;

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

            case 5:
                daftarLagu.tampilkanDaftar();
                break;

            case 0:
                cout << "Terima kasih! Program selesai.\n";
                break;

            default:
                cout << "Menu tidak tersedia, coba lagi.\n";
        }
```

```
        cin.ignore(numeric_limits<streamsize>::max(), '\n'); // mencegah
input nyangkut
    } while (menu != 0);

    return 0;
}
```

Screenshots Output



Deskripsi:
program utama yang menjalankan menu interaktif untuk mengelola playlist lagu. Sebuah objek Playlist dibuat, kemudian pengguna memilih menu seperti menambah lagu di awal, menambah di akhir, menyisipkan setelah lagu ke-3, menghapus berdasarkan judul, atau melihat seluruh daftar lagu. Program membaca input judul, artis, dan durasi, lalu memanggil fungsi yang sesuai dari objek daftarLagu. Program terus menampilkan menu sampai pengguna memilih angka 0 untuk keluar.

Unguided 1 (playlist.cpp)

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

Playlist::Playlist() {
    awal = nullptr; // playlist kosong saat pertama kali dibuat
}

void Playlist::tambahDiAwal(string judul, string artis, float waktu) {
    // Membuat node baru dan menghubungkannya ke awal
    Lagu* baru = new Lagu{judul, artis, waktu, awal};
    awal = baru;
}

void Playlist::tambahDiAkhir(string judul, string artis, float waktu) {
    Lagu* baru = new Lagu{judul, artis, waktu, nullptr};

    if (awal == nullptr) {
```

```
        awal = baru;
        return;
    }

    Lagu* bantu = awal;
    while (bantu->berikut != nullptr)
        bantu = bantu->berikut;
    bantu->berikut = baru;
}

void Playlist::sisipSetelahKe3(string judul, string artis, float waktu) {
    Lagu* bantu = awal;
    int posisi = 1;

    while (bantu != nullptr && posisi < 3) {
        bantu = bantu->berikut;
        posisi++;
    }

    if (bantu != nullptr) {
        Lagu* baru = new Lagu{judul, artis, waktu, bantu->berikut};
        bantu->berikut = baru;
    } else {
        cout << "Jumlah lagu kurang dari tiga, penyisipan gagal.\n";
    }
}

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

    // Jika lagu yang dihapus berada di awal
    if (awal->judul == judul) {
        Lagu* hapus = awal;
        awal = awal->berikut;
        delete hapus;
        cout << "Lagu \"" << judul << "\" telah dihapus.\n";
        return;
    }

    // Mencari lagu di tengah/akhir
    Lagu* bantu = awal;
    while (bantu->berikut != nullptr && bantu->berikut->judul != judul)
        bantu = bantu->berikut;

    if (bantu->berikut == nullptr) {
        cout << "Lagu dengan judul \"" << judul << "\" tidak ditemukan.\n";
```

```
    } else {
        Lagu* hapus = bantu->berikut;
        bantu->berikut = hapus->berikut;
        delete hapus;
        cout << "Lagu \"" << judul << "\" berhasil dihapus.\n";
    }
}

void Playlist::tampilkanDaftar() {
    if (awal == nullptr) {
        cout << "Tidak ada lagu di playlist.\n";
        return;
    }

    Lagu* bantu = awal;
    int nomor = 1;
    while (bantu != nullptr) {
        cout << nomor++ << ". Judul: " << bantu->judul
             << " | Artis: " << bantu->artis
             << " | Durasi: " << bantu->waktu << " menit\n";
        bantu = bantu->berikut;
    }
}
```

Screenshots Output

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

Deskripsi:

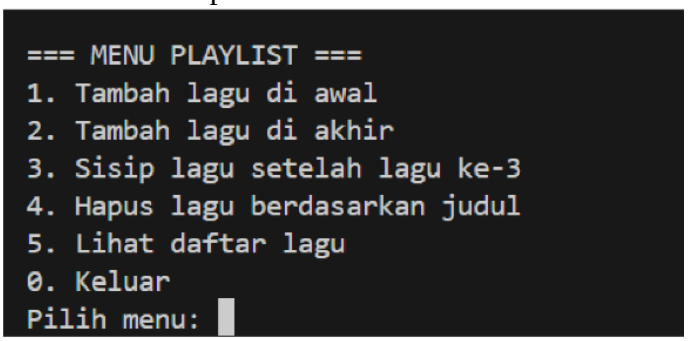
bagaimana sebuah playlist bekerja menggunakan struktur linked list. Konstruktor membuat playlist kosong dengan pointer awal = nullptr. Fungsi tambahDiAwal membuat node lagu baru dan menaruhnya di depan list. tambahDiAkhir menambah node di ujung list; jika playlist masih kosong, lagu langsung menjadi elemen pertama. sisipSetelahKe3 mencari lagu urutan ketiga, lalu menambahkan lagu baru setelahnya jika jumlah lagu cukup. hapusLagu mencari lagu berdasarkan judul dan menghapusnya, baik itu di awal, tengah, atau akhir. tampilkanDaftar mencetak semua lagu lengkap dengan nomor urutnya.

Unguided 1 (playlist.h)

```
#ifndef PLAYLIST_H
#define PLAYLIST_H

#include <iostream>
#include <string>
```

```cpp
using namespace std;

struct Lagu {
    string judul;
    string artis;
    float waktu;
    Lagu* berikut;
};

class Playlist {
private:
    Lagu* awal;

public:
    Playlist();

    void tambahDiAwal(string judul, string artis, float waktu);

    void tambahDiAkhir(string judul, string artis, float waktu);

    void sisipSetelahKe3(string judul, string artis, float waktu);

    void hapusLagu(string judul);

    void tampilkanDaftar();
};

#endif
```

Screenshots Output



Deskripsi:

mendefinisikan struktur data dan kelas yang digunakan. Struktur Lagu berisi judul, artis, durasi, dan pointer ke lagu berikutnya. Kelas Playlist menyimpan pointer ke lagu pertama dan mendeklarasikan fungsi-fungsi seperti menambah lagu, menyisipkan, menghapus, dan menampilkan isi playlist. Semua fungsi ini kemudian diimplementasikan di file cpp.

D. Kesimpulan

melalui praktikum Modul III ini saya dapat memahami konsep Abstract Data Type (ADT) serta penerapannya dalam pemrograman C++ melalui pemisahan antara deklarasi dan implementasi menggunakan file header dan file sumber. Dari kegiatan praktikum yang saya lakukan, saya belajar bagaimana membangun program yang terstruktur dengan memanfaatkan fungsi, struktur, array, dan pointer untuk mengelola serta memanipulasi data secara efisien. Praktikum ini memberikan saya pemahaman tentang pentingnya ADT dalam membuat program yang modular, mudah dikelola, dan mendukung penerapan prinsip pemrograman yang bersih serta terorganisir.

E. Referensi

Kaswar, A. B., & Zain, S. G. (2021). Mudah Belajar Pemrograman Dasar C++. Syiah Kuala University Press.

Hanief, S., Jepriana, I. W., & Kom, S. (2020). Konsep Algoritme dan Aplikasinya dalam Bahasa Pemrograman C++. Penerbit Andi.

Imamuddin, A., & Sobarnas, M. A. (2021). PEMBELAJARAN JARAK JAUH PEMROGRAMAN DASAR MENGGUNAKAN BAHASA C++ UNTUK UMUM: SEBUAH PROGRAM PENGABDIAN KEPADA MASYARAKAT. BEMAS: Jurnal Bermasyarakat, 1(2), 59-67