LAPORAN PRAKTIKUM STRUKTUR DATA

MODUL 4
SINGLY LINKED LIST 1

Disusun Oleh :
NAMA : Gien Darrel Adli
NIM : 10312430008

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 (Singlylist.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
```

Deskripsi:

Kode ini mendefinisikan struktur dan fungsi untuk membuat serta mengelola singly linked list di C++. Struktur ElmList menyimpan data dan penunjuk ke elemen berikutnya, sedangkan List menyimpan alamat elemen pertama. Tersedia fungsi untuk membuat list, menambah elemen di awal atau akhir, menghapus memori, dan menampilkan isi list.

Guided 2 (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;
   }


}
```

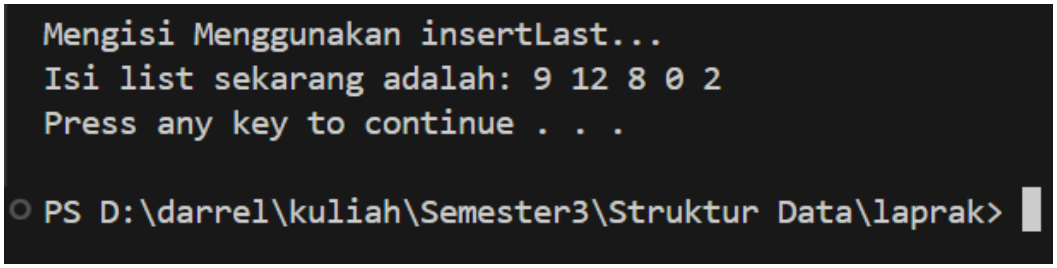
Deskripsi:

Kode di atas berisi implementasi fungsi-fungsi dari singly linked list. Fungsi CreateList mengosongkan list dengan membuat First = Nil. Fungsi alokasi membuat node baru berisi data x, sedangkan dealokasi menghapus node dari memori. Fungsi insertFirst menambahkan node di awal list, dan insertLast menambahkannya di akhir dengan mencari elemen terakhir terlebih dahulu. Terakhir, PrintInfo menampilkan semua data dalam list atau menuliskan “List Kosong!” jika tidak ada elemen.

Guided 3 (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:

Kode main.cpp ini adalah program utama untuk menjalankan dan menguji operasi pada singly linked list yang dibuat di file Singlylist.h dan Singlylist.cpp. Program dimulai dengan membuat list kosong menggunakan CreateList(L), lalu beberapa elemen baru (9, 12, 8, 0, 2) dimasukkan ke dalam list dengan fungsi insertLast, sehingga setiap elemen baru ditambahkan di bagian akhir. Setelah semua data dimasukkan, fungsi PrintInfo(L) menampilkan seluruh isi list di layar. Terakhir, system("pause") digunakan agar program tidak langsung tertutup setelah dijalankan, dan return 0 menandakan program berakhir dengan sukses.

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

Playlist.h

```
#ifndef PLAYLIST_H
#define PLAYLIST_H

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

Deskripsi:

File Playlist.h berfungsi sebagai rancangan struktur data dan fungsi untuk mengelola playlist lagu. Di dalamnya terdapat struktur Lagu yang menyimpan data setiap lagu berupa judul, artis, durasi, serta pointer berikut untuk menghubungkan lagu satu dengan lainnya dalam bentuk linked list. Kelas Playlist memiliki variabel privat awal sebagai penunjuk lagu pertama dan menyediakan beberapa fungsi publik seperti tambahDiAwal(), tambahDiAkhir(), sisipSetelahKe3(), hapusLagu(), dan tampilkanDaftar() yang masing-masing digunakan untuk menambah, menghapus, dan menampilkan lagu dalam playlist. Secara keseluruhan, file ini menjadi dasar pengelolaan playlist menggunakan konsep linked list, sehingga data lagu dapat dikelola secara dinamis dan terstruktur.

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;
    }
}
```

Deskripsi:

Playlist.cpp berisi implementasi fungsi-fungsi dari kelas Playlist yang mengatur pengelolaan daftar lagu menggunakan konsep linked list. Konstruktor Playlist() menginisialisasi playlist agar kosong. Fungsi tambahDiAwal() dan tambahDiAkhir() digunakan untuk menambah lagu di posisi awal dan akhir daftar, sedangkan sisipSetelahKe3() menambah lagu setelah posisi ketiga. Fungsi hapusLagu() berfungsi menghapus lagu berdasarkan judulnya, dan tampilkanDaftar() menampilkan seluruh lagu dalam playlist. File ini memastikan playlist dapat dikelola secara dinamis, mudah ditambah, dihapus, dan ditampilkan.

main.cpp

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

using namespace std;

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

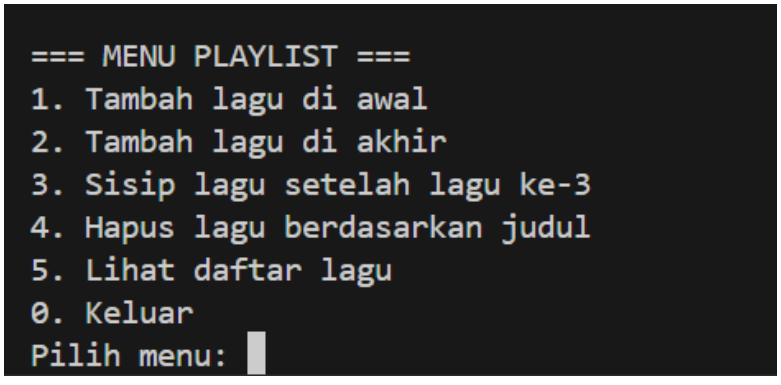
```
    float waktu;

    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;
}
```

Deskripsi:

main.cpp berfungsi sebagai program utama untuk menjalankan pengelolaan playlist lagu. Program ini menggunakan objek daftarLagu dari kelas **Playlist** dan menampilkan menu interaktif untuk menambah, menghapus, menyisipkan, serta menampilkan lagu. Data lagu seperti judul, artis, dan durasi dimasukkan oleh pengguna, lalu program akan memprosesnya menggunakan fungsi-fungsi yang ada di kelas **Playlist**. Program berjalan berulang hingga pengguna memilih keluar.

Screenshots Output Unguided:



D. Kesimpulan

Kesimpulan dari **Modul 4 Singly Linked List** adalah bahwa melalui praktikum ini mahasiswa memahami cara kerja dan penerapan struktur data **linked list** dalam pemrograman C++. Linked list memungkinkan pengelolaan data secara dinamis karena elemen-elemen (node) saling terhubung menggunakan pointer, sehingga penambahan, penghapusan, dan penyisipan data dapat dilakukan dengan efisien tanpa perlu menggeser elemen lain seperti pada array. Implementasi konsep ini melalui pembuatan program playlist lagu juga memperkuat pemahaman mahasiswa terhadap penggunaan pointer, struktur, dan fungsi dalam konteks pemrograman berorientasi objek. Dengan demikian, mahasiswa mampu memahami bagaimana linked list digunakan sebagai dasar dari berbagai struktur data yang lebih kompleks di bidang informatika.

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.