LAPORAN PRAKTIKUM

STRUKTUR DATA

MODUL IV

SINGLY LINKED LIST

Disusun Oleh :

Nama : Izzah Minkhotun Fannisa

NIM : 103112400198

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

C++ merupakan bahasa pemrograman yang pertama kali dikembangkan oleh Bjarne Stroustrop, seorang ilmuwan komputer asal Denmark. Bahasa pemrograman C++ disempurnakan dari bahasa C yang di temukan oleh Dennis Ritchie pada awal 1970-an. Lalu resmi dirilis pada tahun 1985. C++ cukup banyak digunakan karena kemampuannya dalam memberikan kecepatan eksekusi tinggi serta dukungan terhadap manajemen memori tingkat rendah.

Single Linked List merupakan sebuah struktur data yang terdiri dari serangkaian node, di mana setiap node membawa data dan sebuah pointer yang mengarah ke node selanjutnya. Arah daftar bersifat tunggal (searah), yaitu dari node yang pertama (head) menuju node yang terakhir.

B. Guide

1. Guide 1

a. Source Code

singlylist.h

```
#ifndef SINGLYLIST_H_INCLUDED
#define SINGLYLIST_H_INCLUDED

#include <iostream>
#define Nil NULL

typedef int infotype;
typedef struct Elmist *address;

struct Elmist {
    infotype info;
    address next; //pointer
};

struct List {
    address first; //digunakan untuk menunjuk ke node pertama
};

//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
```

singlylist.cpp

```cpp
#include "singlylist.h" //jangan lupa include headerny

void CreateList(List &L) { //mendefinisikan atau membuat list pertama yang masih kosong
    L.first = Nil;
}

address alokasi(infotype x) { //digunakan untuk membuat alamat baru di sebuah memori
    address P = new Elmist; //digunakan untuk mengaokasikan memori untuk node baru
    P->info = x; //menyimpan data ke dalam x ke dalam file info nantinya
    P->next = Nil;
    return P; //mengembalikan alamat note baru
}

void dealokasi(address &P) {
    delete P; //menghapus atau mengembalikan ke note system
}

void insertFirst(List &L, address P) { //menyisipkan note baru di list yang sudah dibuat dibagian awal
    P->next = L.first; //note baru p dihubungkan ke note pertama, secara otomatis note pertama diganti ke P
    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 sistem 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;
    }
}
```

main.cpp

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


using namespace std;

int main(){
    List L; //berisi pointer utama
    address P; //Cukup satu pointer untuk digunakan berulang kali, menginisialisasi alamat memori pertama

    CreateList(L); //untuk memanggil prosedur list yang sudah didefinisikan di bagian body

    cout << "Mengisi List menggunakan insertLast..." << endl;

    //Mengisi list sesuai urutan
    P = alokasi(9); //untuk nilai yang pertama 9 menggunakan insert last
    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); //memanggil prosedur yang sudah didefinisikan
ke nilai nilai mengggunakan insert last

    system("pause");
    return 0;
}
```

b. Screenshot Output

```
PS C:\Users\ACER>  & 'c:\Users\ACER\.vscode\extens
rosoft-MIEngine-In-rqmtrdkq.ebw' '--stdout=Microso
IEngine-Pid-yrtukbto.qna' '--dbgExe=C:\Users\ACER\
Mengisi List menggunakan insertLast...
isi list sekarang adalah: 9 12 8 0 2
Press any key to continue . . .
```

c. Deskripsi Program

Program ini membuat singly linked list lalu mengisi daftar tersebut dengan angka (9, 12, 8, 0, 2) menggunakan fungsi insertLast, yaitu menambah data di bagian belakang list. Setelah semua data masuk, program menampilkan isi list dari awal sampai akhir.

C. Unguide

1. Unguide 1

a. Source Code

playlist.h

```
#ifndef PLAYLIST_H_INCLUDED
#define PLAYLIST_H_INCLUDED

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

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

typedef struct Node *address;

struct Node {
    Lagu info;
    address next;
};

struct Playlist {
```

```
    address first;
};

void createPlaylist(Playlist &P);
address alokasi(Lagu L);
void dealokasi(address &P);

void insertFirst(Playlist &P, address Q);
void insertLast(Playlist &P, address Q);
void insertAfter3(Playlist &P, address Q);

void deleteByTitle(Playlist &P, string judul);
void printPlaylist(Playlist P);

#endif
```

playlist.cpp

```
#include "playlist.h"

void createPlaylist(Playlist &P) {
    P.first = NULL;
}

address alokasi(Lagu L) {
    address P = new Node;
    P->info = L;
    P->next = NULL;
    return P;
}

void dealokasi(address &P) {
    delete P;
}

void insertFirst(Playlist &P, address Q) {
    Q->next = P.first;
    P.first = Q;
}

void insertLast(Playlist &P, address Q) {
    if (P.first == NULL) {
        insertFirst(P, Q);
    } else {
        address last = P.first;
        while (last->next != NULL) {
```

```
            last = last->next;
        }
        last->next = Q;
    }
}

void insertAfter3(Playlist &P, address Q) {
    if (P.first == NULL) {
        // jika kosong, langsung jadi first
        insertFirst(P, Q);
        return;
    }

    address temp = P.first;
    int count = 1;

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

    if (temp != NULL) {
        Q->next = temp->next;
        temp->next = Q;
    } else {
        insertLast(P, Q);
    }
}

void deleteByTitle(Playlist &P, string judul) {
    if (P.first == NULL) {
        cout << "Playlist kosong!\n";
        return;
    }

    address temp = P.first;
    address prev = NULL;
    if (temp->info.judul == judul) {
        P.first = temp->next;
        dealokasi(temp);
        cout << "Lagu \"" << judul << "\" berhasil dihapus!\n";
        return;
    }

    while (temp != NULL && temp->info.judul != judul) {
        prev = temp;
        temp = temp->next;
    }
```

```
    if (temp == NULL) {
        cout << "Lagu tidak ditemukan!\n";
        return;
    }

    prev->next = temp->next;
    dealokasi(temp);
    cout << "Lagu \"" << judul << "\" berhasil dihapus!\n";
}

void printPlaylist(Playlist P) {
    if (P.first == NULL) {
        cout << "Playlist kosong!\n";
        return;
    }

    address temp = P.first;
    int nomor = 1;

    while (temp != NULL) {
        cout << nomor++ << ". "
             << temp->info.judul << " - "
             << temp->info.penyanyi
             << " (" << temp->info.durasi << " menit)\n";
        temp = temp->next;
    }
}
```

main.cpp

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

int main() {
    Playlist P;
    createPlaylist(P);

    Lagu L;
    address Q;

    cout << "=== Tambah Lagu di Akhir ===\n";
    L = {"Fine", "Taeyeon", 3.30f};
    Q = alokasi(L);
```

```
    insertLast(P, Q);

    L = {"Like Water", "Wendy", 4.20f};
    Q = alokasi(L);
    insertLast(P, Q);

    L = {"Psycho", "Red Velvet", 3.30f};
    Q = alokasi(L);
    insertLast(P, Q);

    printPlaylist(P);
    cout << endl;

    cout << "=== Tambah Lagu di Awal ===\n";
    L = {"Russian Roulette", "Red Velvet", 3.10f};
    Q = alokasi(L);
    insertFirst(P, Q);

    printPlaylist(P);
    cout << endl;

    cout << "=== Tambah Lagu Setelah Lagu Ke-3 ===\n";
    L = {"Dandelions", "Ruth B", 3.5f};
    Q = alokasi(L);
    insertAfter3(P, Q);

    printPlaylist(P);
    cout << endl;

    cout << "=== Hapus Lagu Berdasarkan Judul ===\n";
    deleteByTitle(P, "Like Water");

    printPlaylist(P);

    return 0;
}
```

b. Screenshot Output

```
=== Tambah Lagu di Akhir ===
1. Fine - Taeyeon (3.3 menit)
2. Like Water - Wendy (4.2 menit)
3. Psycho - Red Velvet (3.3 menit)

=== Tambah Lagu di Awal ===
1. Russian Roulette - Red Velvet (3.1 menit)
2. Fine - Taeyeon (3.3 menit)
3. Like Water - Wendy (4.2 menit)
4. Psycho - Red Velvet (3.3 menit)

=== Tambah Lagu Setelah Lagu Ke-3 ===
1. Russian Roulette - Red Velvet (3.1 menit)
2. Fine - Taeyeon (3.3 menit)
3. Like Water - Wendy (4.2 menit)
4. Dandelions - Ruth B (3.5 menit)
5. Psycho - Red Velvet (3.3 menit)

=== Hapus Lagu Berdasarkan Judul ===
Lagu "Like Water" berhasil dihapus!
1. Russian Roulette - Red Velvet (3.1 menit)
2. Fine - Taeyeon (3.3 menit)
3. Dandelions - Ruth B (3.5 menit)
4. Psycho - Red Velvet (3.3 menit)
PS C:\Users\ACER\OneDrive\Desktop\Laprak 4>
```

c. Deskripsi Program

Program tersebut mengelola playlist lagu menggunakan struktur data *Single Linked List*. Setiap lagu disimpan sebagai node yang berisi judul, penyanyi, dan durasi. Program dapat menambah lagu di awal playlist, di akhir playlist, serta setelah lagu ke-3. Selain itu, program dapat menghapus lagu berdasarkan judul dan menampilkan seluruh daftar lagu yang ada. Program ini membantu pengguna memahami cara kerja *linked list* dalam pengelolaan data.

D. Referensi

GeeksforGeeks. (2024). *Linked list data structure*. https://www.geeksforgeeks.org/linked-list-data-structure/

Programiz. (2024). *Singly linked list*. https://www.programiz.com/dsa/linked-list

Tutorialspoint. (2024). *Linked list algorithms*. https://www.tutorialspoint.com/data_structures_algorithms/linked_list_algorithms.htm