LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL V
SINGLY LINKED LIST
(BAGIAN KEDUA)

Disusun Oleh :
NAMA : KHOIRUL ADDIFA
NIM : 103112400172

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Searching merupakan salah satu operasi dasar pada struktur data list yang berfungsi untuk mencari node tertentu berdasarkan nilai yang ingin ditemukan. Proses pencarian dilakukan dengan menelusuri setiap node secara berurutan mulai dari elemen pertama hingga node yang dicari ditemukan atau list berakhir. Karena list bersifat sekuensial, pencarian harus dilakukan dengan mengunjungi tiap node satu per satu. Melalui operasi searching, berbagai operasi lain seperti insert after, delete after, maupun update data menjadi lebih mudah dilakukan karena node yang menjadi acuan (posisi sebelum atau yang akan diubah) dapat ditemukan terlebih dahulu sehingga manipulasi data dalam list dapat dilakukan secara tepat dan efisien.

Guided

Singlylist.h

```h
#ifndef SINGLYLIST_H
#define SINGLYLIST_H

#include <iostream>
using namespace std;

typedef int infotype;

typedef struct ElmList *address;

struct ElmList {
    infotype info;
    address next;
};

struct List {
    address First;
};

// ADT PROCEDURES
void createList(List &L);
address alokasi(infotype x);
void dealokasi(address p);
void insertFirst(List &L, address p);
void printInfo(List L);

// Tambahan untuk soal nomor 3 dan 4
address findElm(List L, infotype x);
int totalInfo(List L);

#endif
```

Singlylist.cpp

```cpp
#include "Singlylist.h"

// Membuat list kosong
void createList(List &L) {
    L.First = NULL;
}

// Mengalokasi node baru
address alokasi(infotype x) {
    address p = new ElmList;
    if (p != NULL) {
        p->info = x;
        p->next = NULL;
    }
    return p;
}

// Dealokasi memory
void dealokasi(address p) {
    delete p;
}

// Insert di depan
void insertFirst(List &L, address p) {
    p->next = L.First;
    L.First = p;
}

// Cetak semua elemen
void printInfo(List L) {
    address p = L.First;
    while (p != NULL) {
        cout << p->info << " ";
        p = p->next;
    }
    cout << endl;
}

// Mencari elemen dengan info = x
address findElm(List L, infotype x) {
    address p = L.First;
    while (p != NULL) {
        if (p->info == x)
            return p;
        p = p->next;
    }
```

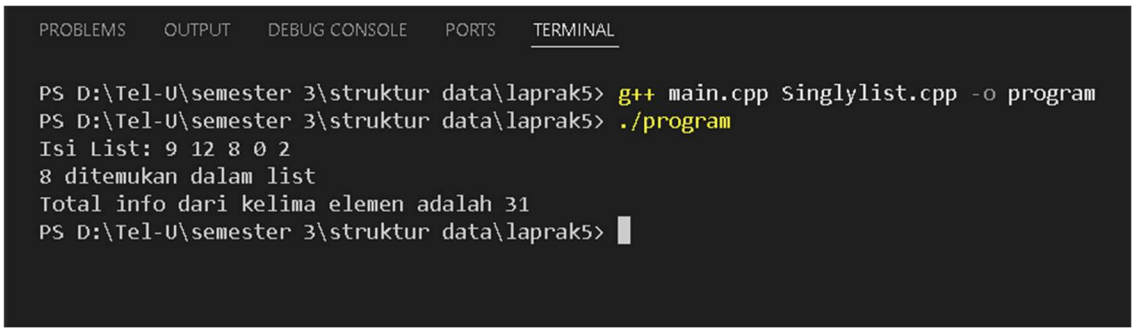
```
    return NULL;
}

// Menghitung total semua nilai info
int totalInfo(List L) {
    int sum = 0;
    address p = L.First;
    while (p != NULL) {
        sum += p->info;
        p = p->next;
    }
    return sum;
}
```

main.cpp

```
#include "Singlylist.h"

int main() {

    List L;
    address P1, P2, P3, P4, P5;

    createList(L);

    P1 = alokasi(2);
    insertFirst(L, P1);

    P2 = alokasi(0);
    insertFirst(L, P2);

    P3 = alokasi(8);
    insertFirst(L, P3);

    P4 = alokasi(12);
    insertFirst(L, P4);

    P5 = alokasi(9);
    insertFirst(L, P5);

    cout << "Isi List: ";
    printInfo(L);

    // Soal 3: cari info 8
    address cari = findElm(L, 8);
    if (cari != NULL)
        cout << cari->info << " ditemukan dalam list" << endl;
    else
```

```
        cout << "Elemen tidak ditemukan" << endl;

    // Soal 4: total info
    cout << "Total info dari kelima elemen adalah " << totalInfo(L) <<
endl;

    return 0;
}
```

Screenshots Output



Deskripsi:

Program ini bertujuan untuk mengimplementasikan konsep ADT Singly Linked List dalam pengelolaan data secara dinamis menggunakan pointer. Melalui program ini, mahasiswa dilatih untuk membuat list, menambahkan elemen ke bagian awal list, melakukan proses pencarian data (searching) pada node tertentu, serta menghitung total nilai dari seluruh elemen dalam list.

Kesimpulan

Dalam praktikum Modul 5 ini, disimpulkan bahwa Singly Linked List merupakan struktur data dinamis yang memungkinkan penambahan dan pengelolaan elemen secara fleksibel melalui pemanfaatan pointer. Mahasiswa mempelajari bagaimana node dibuat, disisipkan, ditelusuri, hingga dilakukan proses pencarian dan perhitungan terhadap data di dalamnya. Operasi searching menjadi bagian penting karena digunakan sebagai dasar untuk mempermudah operasi lain seperti insert after, delete after, maupun update elemen. Berbeda dengan array yang ukurannya tetap, linked list dapat bertambah sesuai kebutuhan sehingga lebih efisien untuk pengelolaan data yang tidak tetap jumlahnya. Secara keseluruhan, praktikum ini meningkatkan pemahaman mahasiswa terhadap struktur data dinamis, manajemen memori, serta penerapan ADT untuk membangun program yang terstruktur dan mudah dikembangkan.

B. Referensi

- Weiss, Mark Allen. Data Structures and Algorithm Analysis in C++. Pearson.
- Malik, D.S. Data Structures Using C++. Course Technology.
- GeeksforGeeks. “Singly Linked List in C++”.