LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 6
PENGENALAN CODE BLOCKS

Disusun Oleh :
NAMA :Muhamad Ilham Syahid
NIM :103112400155

Dosen
WAHYU ANDI SAPUTRA

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Doubly Linked List adalah struktur data linear yang setiap elemennya memiliki dua pointer, yaitu next (menunjuk ke elemen berikutnya) dan prev (menunjuk ke elemen sebelumnya). Struktur ini memungkinkan traversal data secara maju dan mundur. Doubly Linked List memiliki dua penunjuk utama, yaitu first sebagai elemen awal dan last sebagai elemen akhir. Operasi utama yang dapat dilakukan meliputi insert**,** delete, search, dan update data. Dibandingkan Singly Linked List, Doubly Linked List lebih fleksibel dalam pengolahan data, tetapi membutuhkan memori lebih besar karena penggunaan dua pointer pada setiap node.

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

Guided 1

Doublylist.h

```
#ifndef DOUBLYLIST_H
#define DOUBLYLIST_H


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


#define Nil NULL


// tipe data kendaraan
typedef struct {
string nopol;
string warna;
int thnBuat;
} infotype;


// pointer
typedef struct elmlist *address;


// node doubly linked list
struct elmlist {
infotype info;
address next;
address prev;
};
```

```
// list
struct List {
address First;
address Last;
};

void CreateList(List &L);
address alokasi(infotype x);
void dealokasi(address &P);
void insertLast(List &L, address P);
void printInfo(List L);
address findElm(List L, string nopol);
void deleteFirst(List &L, address &P);
void deleteLast(List &L, address &P);
void deleteAfter(address Prec, address &P);

#endif
```

Guided 2

Doublylist.cpp

```
#include "Doublylist.h"

void CreateList(List &L) {
    L.First = Nil;
    L.Last = Nil;
}

address alokasi(infotype x) {
    address P = new elmlist;
    P->info = x;
    P->next = Nil;
    P->prev = Nil;
    return P;
}

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

void insertLast(List &L, address P) {
```

```
    if (L.First == Nil) {
        L.First = P;
        L.Last = P;
    } else {
        P->prev = L.Last;
        L.Last->next = P;
        L.Last = P;
    }
}

void printInfo(List L) {
    address P = L.First;
    while (P != Nil) {
        cout << "No Polisi : " << P->info.nopol << endl;
        cout << "Warna     : " << P->info.warna << endl;
        cout << "Tahun     : " << P->info.thnBuat << endl;
        cout << "------------------" << endl;
        P = P->next;
    }
}

address findElm(List L, string nopol) {
    address P = L.First;
    while (P != Nil) {
        if (P->info.nopol == nopol)
            return P;
        P = P->next;
    }
    return Nil;
}

void deleteFirst(List &L, address &P) {
    P = L.First;
    if (P != Nil) {
        L.First = P->next;
        if (L.First != Nil)
            L.First->prev = Nil;
        else
            L.Last = Nil;
        P->next = Nil;
    }
}

void deleteLast(List &L, address &P) {
    P = L.Last;
    if (P != Nil) {
        L.Last = P->prev;
        if (L.Last != Nil)
```

```
            L.Last->next = Nil;
        else
            L.First = Nil;
        P->prev = Nil;
    }
}

void deleteAfter(address Prec, address &P) {
    if (Prec != Nil && Prec->next != Nil) {
        P = Prec->next;
        Prec->next = P->next;

        if (P->next != Nil)
            P->next->prev = Prec;
        else
            // jika yang dihapus adalah elemen terakhir
            Prec->next = Nil;

        P->next = Nil;
        P->prev = Nil;
    }
}
```

Guided 3

Main.cpp

```
#include "Doublylist.h"

int main() {
    List L;
    CreateList(L);

    infotype x;

    x = {"D001", "Merah", 2020};
    insertLast(L, alokasi(x));

    x = {"D002", "Hitam", 2019};
    insertLast(L, alokasi(x));

    x = {"D003", "Putih", 2021};
    insertLast(L, alokasi(x));

    cout << "DATA KENDARAAN" << endl;
    printInfo(L);
```

```
    cout << "\nCari D001" << endl;
    address P = findElm(L, "D001");
    if (P != Nil)
        cout << "Data ditemukan: " << P->info.nopol << endl;

    cout << "\nHapus D003" << endl;
    address Prec = findElm(L, "D002");
    deleteAfter(Prec, P);
    dealokasi(P);

    printInfo(L);

    return 0;
}
```

Screenshots Output

```
PS C:\Pratikum Struktur Data\modul 6 laprak> g++ main.cpp Doublylist.cpp -o main
PS C:\Pratikum Struktur Data\modul 6 laprak> ./main
PS C:\Pratikum Struktur Data\modul 6 laprak> ./main
DATA KENDARAAN
No Polisi : D001
DATA KENDARAAN
No Polisi : D001
No Polisi : D001
Warna     : Merah
Tahun     : 2020
------------------
No Polisi : D002
Tahun     : 2020
------------------
No Polisi : D002
Warna     : Hitam
Tahun     : 2019
------------------
No Polisi : D003
Warna     : Putih
Tahun     : 2021
------------------

Cari D001
Data ditemukan: D001

Hapus D003
No Polisi : D001
No Polisi : D003
Warna     : Putih
Tahun     : 2021
------------------

Cari D001
Data ditemukan: D001

Hapus D003
No Polisi : D001
Warna     : Merah
Tahun     : 2020
------------------
No Polisi : D002
Warna     : Hitam

Cari D001
Data ditemukan: D001
```

Deskripsi:

Program ini merupakan implementasi Doubly Linked List menggunakan bahasa C++ untuk menyimpan data kendaraan. Setiap node memiliki dua pointer, yaitu next dan prev**,** sehingga data dapat diakses dari dua arah. Program menyediakan operasi insertLast untuk menambah data, delete untuk menghapus data, findElm untuk mencari kendaraan berdasarkan nomor polisi, serta printInfo untuk menampilkan seluruh data. Struktur program dibagi menjadi tiga file agar lebih terorganisir dan mudah dipahami.

C. Kesimpulan

Program Doubly Linked List berhasil diimplementasikan untuk mengelola data kendaraan menggunakan bahasa C++. Struktur data ini memudahkan proses penambahan, penghapusan, dan pencarian data karena setiap node memiliki pointer ke elemen sebelumnya dan sesudahnya. Dengan pembagian program ke dalam beberapa file, kode menjadi lebih terstruktur, mudah dipelihara, dan aman dijalankan tanpa error.

Referensi

Modul Praktikum Struktur Data – Modul 6: Doubly Linked List.