LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VI
DOUBLY LINKED LIST (BAGIAN PERTAMA)

Disusun Oleh :
NAMA : Dealova Agta Syahlevi
NIM : 103112400124

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

C++ adalah bahasa pemrograman yang dikembangkan oleh Bjarne Stroustrup sebagai pengembangan dari bahasa C. Bahasa ini mendukung pemrograman prosedural maupun berorientasi ojek, jadi fleksibel digunakan dalam pembuatan aplikasi, game, sampai sistem operasi. Keunggulan C++ ada pada efisiennya yang tinggi serta kemampuannya menjrmbatani pemrograman tingkat rendah dan tinggi.

Dalam perkuliahan C++ sering digunakan karena strukturnya hampir sama dengan bahasa C namun memiliki fitur yang lebih modern. Jadi bisa lebih memudahkan kami mahasiswa memahami dasar-dasar pemrogrman, struktur data, dan algoritma, sekaligus mengenalkan konsep berorientasi objek. Selain itu, bahasa C++ juga membiasakan kami mahasiswa dengan bahasa yang banyak dipakai dalam industri perangkat lunak dan oengembangan sistem.

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

Guided 1

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

struct Node
{
    int data;
    Node *prev;
    Node *next;
};

Node *ptr_first = NULL;
Node *ptr_last = NULL;

void add_first(int value)
{
    Node *newNode = new Node{value, NULL, ptr_first};

    if (ptr_first == NULL)
```

```
        {
            ptr_last = newNode;

        }
        else
        {
            ptr_first->prev = newNode;
        }
        ptr_first = newNode;
    }

    void add_last (int value)
    {
        Node *newNode = new Node{value, ptr_last, NULL};

        if (ptr_last == NULL)
        {
            ptr_first = newNode;
        }
        else
        {
            ptr_last->next = newNode;
        }
        ptr_last = newNode;
    }

    void add_target(int targetValue, int newValue)
    {
        Node *current = ptr_first;
        while (current != NULL && current->data != targetValue)
        {
            current = current->next;
```

```
        }

        if (current != NULL)
        {
            if (current == ptr_last)
            {
                add_last(newValue);
            }
            else
            {
                 Node *newNode = new Node{newValue, current, current->next};
                current->next->prev = newNode;
                current->next = newNode;
            }
        }
    }

    void view()
    {
        Node *current = ptr_first;
        if (current == NULL)
        {
            cout << "List kosong\n";
            return;
        }
        while (current != NULL)
        {
            cout << current->data << (current->next != NULL ? " <-> " : "");
            current = current->next;
        }
        cout << endl;
```

```
}

void delete_first()
{
    if (ptr_first == NULL)
        return;

    Node *temp = ptr_first;

    if (ptr_first == ptr_last)
    {
        ptr_first = NULL;
        ptr_last = NULL;
    }
    else
    {
        ptr_first = ptr_first->next;
        ptr_first->prev = NULL;
    }
    delete temp;
}

void delete_last()
{
    if (ptr_last == NULL)
        return;

    Node *temp = ptr_last;

    if (ptr_first == ptr_last)
    {
        ptr_first = NULL;
```

```
            ptr_last = NULL;
        }
        else
        {
            ptr_last = ptr_last->prev;
            ptr_last->next = NULL;
        }
        delete temp;
    }

    void delete_target(int targetValue)
    {
        Node *current = ptr_first;
        while (current != NULL && current->data != targetValue)
        {
            current = current->next;
        }

        if (current != NULL)
        {
            if (current == ptr_first)
            {
                delete_first();
                return;
            }
            if (current == ptr_last)
            {
                delete_last();
                return;
            }

            current->prev->next = current->next;
```

```
            current->next->prev = current->prev;
            delete current;
        }
    }

    void edit_node(int targetValue, int newValue)
    {
        Node *current = ptr_first;
        while (current != NULL && current->data != targetValue)
        {
            current = current->next;
        }

        if (current != NULL)
        {
            current->data = newValue;
        }
    }

    int main()
    {
        add_first(10);
        add_last(5);
        add_last(20);
        cout << "Awal\t\t\t\t\t : ";
        view();

        delete_first();
        cout << "Setelah delete_first\t : ";
        view();
        delete_last();
        cout << "Setelah delete_last\t\t : ";
```

```
        view();

        add_last(30);
        add_last(40);
        cout << "Setelah tambah\t\t\t : ";
        view();

        delete_target(30);
        cout << "Setelah delete_target\t : ";
        view();

        return 0;
    }
```

Screenshots Output

Deskripsi:

Program ini adalah implementasi Doubly Linked List (linked list ganda) dalam bahasa C++ yang menyimpan data bertipe integer dan menyediakan berbagai operasi dasar untuk memanipulasi data di dalam list.

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

Unguided 1

```
#ifndef DOUBLYLIST_H
#define DOUBLYLIST_H
#include <iostream>
#include <string>
using namespace std;
```

```
struct kendaraan {
    string nopol;
    string warna;
    int thnBuat;
};

typedef kendaraan infotype;

struct ElmList {
    infotype info;
    ElmList* next;
    ElmList* prev;
};

typedef ElmList* address;

struct List {
    address First;
    address Last;
};

void CreateList(List &L);
address alokasi(infotype x);
void dealokasi(address &P);
void printInfo(List L);
void insertLast(List &L, address P);
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
```

Deskripsi:

File Doublylist.h merupakan *header file* yang berisi deklarasi struktur data dan fungsi-fungsi dasar untuk mengelola Doubly Linked List dengan kasus data kendaraan.

Unguided 2

```
#include "Doublylist.h"

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

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

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

void printInfo(List L) {
    if (!L.First) { cout << "List kosong\n"; return; }
    address P = L.First; int i = 1;
    cout << "\nDATA LIST:\n";
    while (P) {
        cout << "Data ke-" << i++ << endl;
        cout << "Nomor Polisi : " << P->info.nopol << endl;
        cout << "Warna        : " << P->info.warna << endl;
        cout << "Tahun        : " << P->info.thnBuat << "\n\n";
        P = P->next;
    }
}

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

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

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

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

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

void deleteAfter(address Prec, address &P) {
```

```
    if (!Prec || !Prec->next) return;
    P = Prec->next;
    Prec->next = P->next;
    if (P->next) P->next->prev = Prec;
    P->next = P->prev = NULL;
}
```

Deskripsi:

File Doublylist.cpp merupakan implementasi dari fungsi-fungsi yang dideklarasikan dalam Doublylist.h, yang berfungsi untuk mengelola struktur data Doubly Linked List (daftar berantai ganda) dengan data bertipe kendaraan.

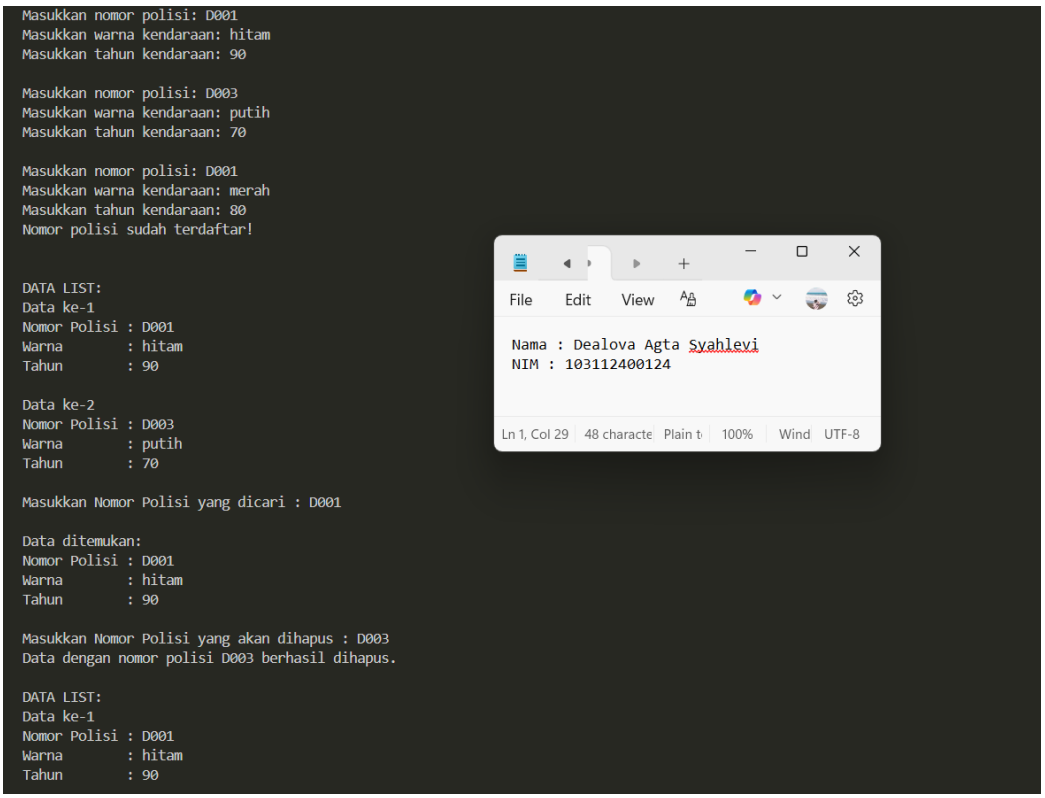
Unguided 3

```
#include "Doublylist.h"
#include "Doublylist.cpp"

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

    infotype x;
    string cari, hapus;
    address P, found;

    // Input data kendaraan
    for (int i = 0; i < 3; i++) {
        cout << "Masukkan nomor polisi: ";
        cin >> x.nopol;
        cout << "Masukkan warna kendaraan: ";
        cin >> x.warna;
        cout << "Masukkan tahun kendaraan: ";
        cin >> x.thnBuat;
```

```
    // Cek duplikasi nopol
    if (findElm(L, x.nopol) != NULL) {
        cout << "Nomor polisi sudah terdaftar!\n";
    } else {
        P = alokasi(x);
        insertLast(L, P);
    }
    cout << endl;
}

printInfo(L);

// Cari data
cout << "Masukkan Nomor Polisi yang dicari : ";
cin >> cari;
found = findElm(L, cari);
if (found != NULL) {
    cout << "\nData ditemukan:\n";
    cout << "Nomor Polisi : " << found->info.nopol << endl;
    cout << "Warna        : " << found->info.warna << endl;
    cout << "Tahun        : " << found->info.thnBuat << endl;
} else {
    cout << "Data tidak ditemukan.\n";
}

// Hapus data
cout << "\nMasukkan Nomor Polisi yang akan dihapus : ";
cin >> hapus;
found = findElm(L, hapus);
if (found != NULL) {
    if (found == L.First) {
```

```
            deleteFirst(L, P);
        } else if (found == L.Last) {
            deleteLast(L, P);
        } else {
            deleteAfter(found->prev, P);
        }
        dealokasi(P);
        cout << "Data dengan nomor polisi " << hapus << " berhasil dihapus.\n";
    } else {
        cout << "Data tidak ditemukan.\n";
    }

    printInfo(L);

    return 0;
}
```

Screenshots Output



Deskripsi:

Program ini merupakan implementasi lengkap dari program manajemen data kendaraan menggunakan struktur data Doubly Linked List.

D. Kesimpulan

Program ini mengimplementasikan struktur data Doubly Linked List untuk mengelola data kendaraan secara dinamis, dengan kemampuan menambah, menampilkan, mencari, dan menghapus data berdasarkan nomor polisi. Setiap node menyimpan informasi nomor polisi, warna, dan tahun pembuatan kendaraan. Program juga memastikan tidak ada duplikasi data serta mendemonstrasikan cara kerja linked list ganda yang efisien dan fleksibel dalam pengelolaan data.

E. Referensi

GeeksforGeeks. "C++ Programming Language – Introduction." https://www.geeksforgeeks.org/c-plus-plus/

Wikipedia. "C++." https://id.wikipedia.org/wiki/C%2B%2B