LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VI
DOUBLY LINKED LIST

Disusun Oleh :
NAMA : ZAKI MAULA DHIA
NIM : 103112400127

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Dalam struktur data doubly linked list, setiap node tidak hanya memiliki pointer ke elemen berikutnya tetapi juga pointer ke elemen sebelumnya. Hal ini memungkinkan traversal dari depan ke belakang maupun dari belakang ke depan, suatu keunggulan yang tidak dimiliki oleh liked list satu arah biasa. Namun, keuntungan tersebut juga datang dengan sejumlah kompromi. Kerena tiap node menyimpan dua pointer(next dan prev), maka ruang memori yang diperlukan menjadi lebih besar dibandingan dengan singly linked list.

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){
```

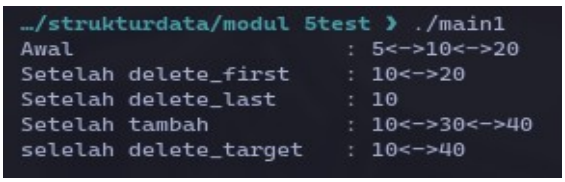
```
    Node *newNode = new Node{value, ptr_last, NULL};

    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;
        }
        if (current != NULL)
        {
            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_first(5);
```

```
    add_last(20);
    cout << "Awal\t\t\t: ";
    view();

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

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

    delete_Target(30);
    cout << "selelah delete_target\t: ";
    view();
}
```

Screenshots Output



Deskripsi:

Program ini membangun sebuah struktur data Doubly Linked List yang mampu melakukan operasi dasar seperti menambah, menampilkan, mengedit, dan menghapus data dari berbagai posisi dalam list. Pertama program akan menginputkan data ke dalam list, yang kemudian program akan menghapus data yang berada pada awal list dan yang berada pada akhir list, menyisakan node tengah, kemudian program akan menambahkan data baru kedalam list, lalu program akan melakukan penghapusan yang tertuju sesuai yang user codingkan, kemudian program akan menampilkan isi setelah data yang sama seperti yang user codingkan.

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

Unguided 1

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

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


    for (int i = 0; i < 3; i++) {
        kendaraan k;
        cout << "Masukkan nomor polisi: ";
        getline(cin, k.nopol);
        cout << "Masukkan warna kendaraan: ";
        getline(cin, k.warna);
        cout << "Masukkan tahun kendaraan: ";
        cin >> k.thnBuat;
        cin.ignore();

        cout << endl;

        insertLast(L, alokasi(k));
    }

    cout << "\n=== DAFTAR KENDARAAN ===" << endl;
    printInfo(L);

    return 0;
}
```

```
// Doublylist.h
#ifndef DOUBLYLIST_H
#define DOUBLYLIST_H
#include <iostream>
#include <string>
using namespace std;

struct kendaraan {
    string nopol;
```

```
    string warna;
    int thnBuat;
};

struct ElmList;
typedef ElmList* address;

struct ElmList {
    kendaraan info;
    address next;
    address prev;
};

struct List {
    address first;
    address last;
};

void createList(List &L);
address alokasi(kendaraan x);
void dealokasi(address &P);
void printInfo(List L);
void insertLast(List &L, address P);

#endif
```

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

void createList(List &L) {
    L.first = NULL;
    L.last = NULL;
}

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

void dealokasi(address &P) {
```

```
    delete P;
    P = NULL;
}

void insertLast(List &L, address P) {
    if (L.first == NULL) {
        L.first = P;
        L.last = P;
    } else {
        L.last->next = P;
        P->prev = L.last;
        L.last = P;
    }
}

void printInfo(List L) {
    address P = L.first;
    while (P != NULL) {
        cout << "Nopol   : " << P->info.nopol << endl;
        cout << "Warna   : " << P->info.warna << endl;
        cout << "Tahun   : " << P->info.thnBuat << endl;
        cout << "--------------------" << endl;
        P = P->next;
    }
}
```

Screenshots Output

```
…/strukturdata/modul 5test ❯ ./Doublylist
Masukkan nomor polisi: D001
Masukkan warna kendaraan: hitam
Masukkan tahun kendaraan: 90

Masukkan nomor polisi: D002
Masukkan warna kendaraan: putih
Masukkan tahun kendaraan: 70

Masukkan nomor polisi: D003
Masukkan warna kendaraan: merah
Masukkan tahun kendaraan: 80


=== DAFTAR KENDARAAN ===
Nopol   : D001
Warna   : hitam
Tahun   : 90
----------------------
Nopol   : D002
Warna   : putih
Tahun   : 70
----------------------
Nopol   : D003
Warna   : merah
Tahun   : 80
----------------------
```

Deskripsi:

Program memiliki fungsi untuk menginput data kendaraan dengan menggunakan metode DOUBLY LINKED LIST, program akan meminta input dari user dengan batas tiga buah data yang akan disimpan kedalam struct kendaraan, setelah user menerima input dari user program akan akan menampilkan data yang sudah di input menggunakan pointer yang cukup mirip dengan ADT.

Unguided 2

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

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


    for (int i = 0; i < 3; i++) {
        kendaraan k;
```

```
        cout << "Masukkan nomor polisi: ";
        getline(cin, k.nopol);
        cout << "Masukkan warna kendaraan: ";
        getline(cin, k.warna);
        cout << "Masukkan tahun kendaraan: ";
        cin >> k.thnBuat;
        cin.ignore();

        cout << endl;

        insertLast(L, alokasi(k));
    }

    cout << "\n=== DAFTAR KENDARAAN ===" << endl;
    printInfo(L);

    string cari;
    cout << "\nMasukkan Nomor Polisi yang dicari: ";
    getline(cin, cari);
    cout << endl;

    address hasil = findElm(L, cari);
    if (hasil != NULL) {
        cout << "Nomor Polisi : " << hasil->info.nopol << endl;
        cout << "Warna        : " << hasil->info.warna << endl;
        cout << "Tahun        : " << hasil->info.thnBuat << endl;
    } else {
        cout << "\nData dengan nomor polisi " << cari << " tidak ditemukan." << endl;
    }

    return 0;
}
```

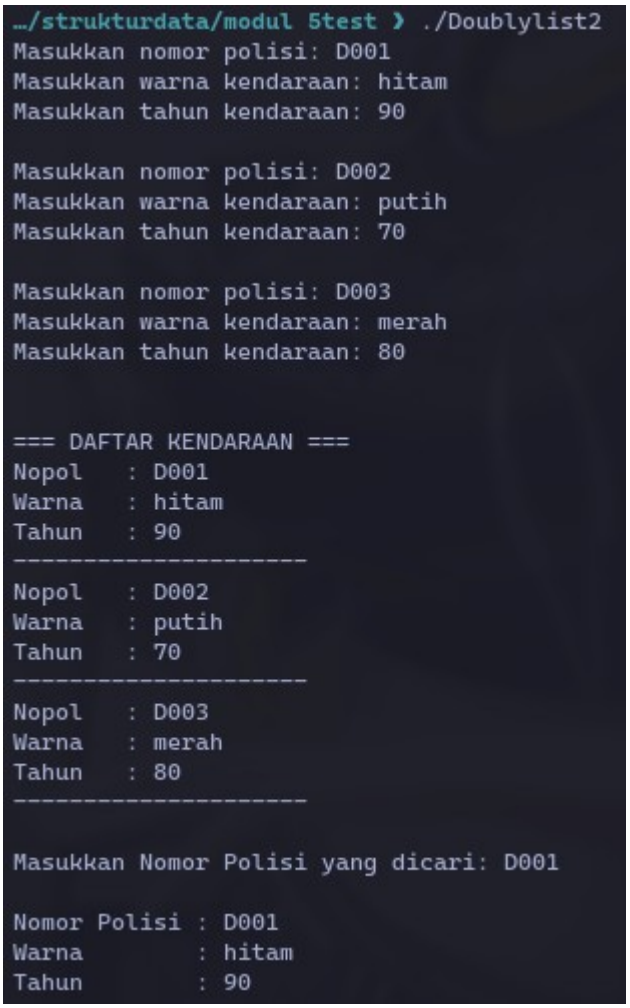
```
// Doublylist.h
#ifndef DOUBLYLIST_H
#define DOUBLYLIST_H
#include <iostream>
#include <string>
using namespace std;

struct kendaraan {
    string nopol;
    string warna;
```

```
    int thnBuat;
};

struct ElmList;
typedef ElmList* address;

struct ElmList {
    kendaraan info;
    address next;
    address prev;
};

struct List {
    address first;
    address last;
};

void createList(List &L);
address alokasi(kendaraan x);
void dealokasi(address &P);
void printInfo(List L);
address findElm(List L, string nopol);
void insertLast(List &L, address P);

#endif
```

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

void createList(List &L) {
    L.first = NULL;
    L.last = NULL;
}

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

void dealokasi(address &P) {
```

```
    delete P;
    P = NULL;
}

void insertLast(List &L, address P) {
    if (L.first == NULL) {
        L.first = P;
        L.last = P;
    } else {
        L.last->next = P;
        P->prev = L.last;
        L.last = P;
    }
}

void printInfo(List L) {
    address P = L.first;
    while (P != NULL) {
        cout << "Nopol   : " << 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 != NULL) {
        if (P->info.nopol == nopol) {
            return P;
        }
        P = P->next;
    }
    return NULL;
}
```

Screenshots Output



Deskripsi:

Program memiliki fungsi untuk menginput data kendaraan dengan menggunakan metode DOUBLY LINKED LIST, program akan meminta input dari user dengan batas tiga buah data yang akan disimpan kedalam struct kendaraan, setelah user menerima input dari user program akan akan menampilkan data yang sudah di input menggunakan pointer yang cukup mirip dengan ADT. Yang kemudian user diminta untuk menginputkan nomor polisi untuk dicari didalam list, dengan menggunakan pointer dan menyamakan dengan inputan dari user maka didapatkan datanya kemudian ditampilkan dalam main.cpp, jika data tidak ada maka main.cpp akan menampilkan ‘Data dengan nomor polisi .... tidak tidemukan’.

Unguided 3

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

```
int main() {
    List L;
    createList(L);


    for (int i = 0; i < 3; i++) {
        kendaraan k;
        cout << "Masukkan nomor polisi: ";
        getline(cin, k.nopol);
        cout << "Masukkan warna kendaraan: ";
        getline(cin, k.warna);
        cout << "Masukkan tahun kendaraan: ";
        cin >> k.thnBuat;
        cin.ignore();

        cout << endl;

        insertLast(L, alokasi(k));
    }

    cout << "\n=== DAFTAR KENDARAAN ===" << endl;
    printInfo(L);

    string cari;
    cout << "\nMasukkan Nomor Polisi yang dicari: ";
    getline(cin, cari);
    cout << endl;

    address hasil = findElm(L, cari);
    if (hasil != NULL) {
        cout << "Nomor Polisi : " << hasil->info.nopol << endl;
        cout << "Warna        : " << hasil->info.warna << endl;
        cout << "Tahun        : " << hasil->info.thnBuat << endl;
    } else {
        cout << "\nData dengan nomor polisi " << cari << " tidak ditemukan." << endl;
    }

    string hapus;
    cout << "\nMasukkan Nomor Polisi yang akan dihapus: ";
    getline(cin, hapus);

    address target = findElm(L, hapus);
    address P;
```

```
    if (target != NULL) {
        if (target == L.first) {
            deleteFirst(L, P);
        } else if (target == L.last) {
            deleteLast(L, P);
        } else {
            deleteAfter(target->prev, P);
        }

        dealokasi(P);
        cout << "Data dengan nomor polisi " << hapus << " berhasil dihapus." << endl;
    } else {
        cout << "Data tidak ditemukan." << endl;
    }

    cout << "\n=== DAFTAR KENDARAAN ===" << endl;
    printInfo(L);
    return 0;
}
```

```
// Doublylist.h
#ifndef DOUBLYLIST_H
#define DOUBLYLIST_H
#include <iostream>
#include <string>
using namespace std;

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

struct ElmList;
typedef ElmList* address;

struct ElmList {
    kendaraan info;
    address next;
    address prev;
};
```

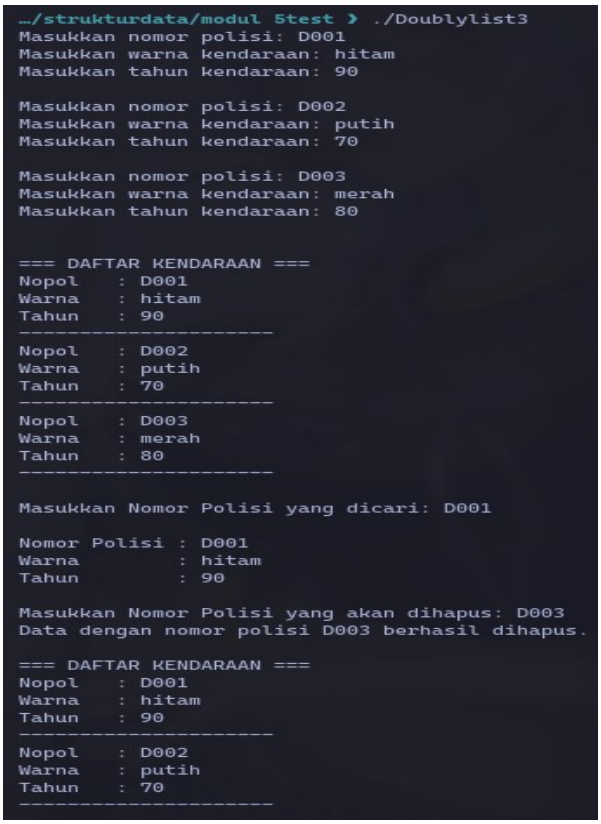
```
struct List {
    address first;
    address last;
};

void createList(List &L);
address alokasi(kendaraan x);
void dealokasi(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);
void insertLast(List &L, address P);

#endif
```

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

void createList(List &L) {
    L.first = NULL;
    L.last = NULL;
}

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

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

void insertLast(List &L, address P) {
    if (L.first == NULL) {
        L.first = P;
        L.last = P;
    } else {
```

```
        L.last->next = P;
        P->prev = L.last;
        L.last = P;
    }
}

void printInfo(List L) {
    address P = L.first;
    while (P != NULL) {
        cout << "Nopol   : " << 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 != NULL) {
        if (P->info.nopol == nopol) {
            return P;
        }
        P = P->next;
    }
    return NULL;
}

void deleteFirst(List &L, address &P) {
    if (L.first == NULL) {
        P = NULL;
    } else if (L.first == L.last) {
        P = L.first;
        L.first = NULL;
        L.last = NULL;
    } else {
        P = L.first;
        L.first = P->next;
        L.first->prev = NULL;
        P->next = NULL;
    }
}

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

```
    if (L.first == NULL) {
        P = NULL;
    } else if (L.first == L.last) {
        P = L.first;
        L.first = NULL;
        L.last = NULL;
    } else {
        P = L.last;
        L.last = P->prev;
        L.last->next = NULL;
        P->prev = NULL;
    }
}

void deleteAfter(address Prec, address &P) {
    if (Prec != NULL && Prec->next != NULL) {
        P = Prec->next;
        Prec->next = P->next;
        if (P->next != NULL) {
            P->next->prev = Prec;
        }
        P->next = NULL;
        P->prev = NULL;
    } else {
        P = NULL;
    }
}
```

Screenshots Output



Deskripsi:

Program yang sama seperti di nomor 1 dan 2 tetapi di soal 3 ditambah function untuk menghapus data D003 dengan menggunakan deleteFirst, deleteLast, deleteAfter. Ketiganya mempunyain fungsi yang sama yaitu menghapus data dalam list berbedaannya adalah deleteFirst menghapus data yang ada di awal list, deleteLast akan menghapus data yang berada di akhir list, dan deleteAfter biasanya digunakan untuk menghapus data di tengah list.

D. Kesimpulan

kesimpulan dari materi doubly linked list yang telah dipelajari pada minggu 6 bahwa program doubly list merupakan implementasi struktur data dinamis yang dapat menyimpan, menampilkan, mencari, dan menghapus data secara fleksibel. Seperti pada kasus list data kendaraan, program mampu menemukan elemen tertentu berdasarkan input dari user dab setelah data ditemukan user dapat menghapusnya dengan menggunakan prosedur penghapusan yang sesuai dengan struktur doubly linked list. Dengan menggunakan doubly linked list data kendaraan dapat di susun secara efisien dan terorganisir, sekaligus menunjukkan konsep dasar operasi pada struktur doubly linked list yang umum digunakan dalam berbagai aplikasi managemen data dinamis.

E. Referensi

- GeeksforGeeks. (n.d.). Applications, advantages and disadvantages of doubly linked list. Retrieved October 29, 2025, from https://www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-doubly-linked-list/
- Scaler. (n.d.). Doubly linked list. Retrieved October 29, 2025, from https://www.scaler.in/doubly-linked-list/