LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 6
DOUBLY LINKED LIST

Disusun Oleh :
NAMA : Herdian Abdillah Purnomo
NIM : 103112430048

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Doubly Linked List adalah struktur data yang setiap elemennya terhubung dua arah, ke depan dan ke belakang.Setiap node memiliki tiga bagian: data, pointer ke node sebelumnya, dan pointer ke node berikutnya. Struktur ini memudahkan penambahan, penghapusan, dan pencarian data dari dua arah. Kelebihannya mudah diakses dua arah, sedangkan kekurangannya memakai memori lebih banyak karena dua pointer.

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

    return 0;
}
```

Screenshots Output

```
PS C:\Users\Lenovo\Documents\PRAKTIKUM STRUKDAT\FILE\MODUL 5> cd "c:\Users\Lenovo\Documents\PRAKTIKUM STRUKDAT\FILE\MODUL 5\GUIDED
\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
Awal                    : 10 <-> 5 <-> 20
Setelah delete_first    : 5 <-> 20
Setelah delete last     : 5
Setelah tambah          : 5 <-> 30 <-> 40
Setelah delete_target   : 5 <-> 40
PS C:\Users\Lenovo\Documents\PRAKTIKUM STRUKDAT\FILE\MODUL 5\GUIDED>
```

Deskripsi: Program berhasil menampilkan dan memanipulasi data dengan struktur Doubly Linked List. Semua operasi penambahan, penghapusan, dan tampilan data bekerja sesuai algoritma.

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

Unguided 1

main.cpp

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


int main() {
    List L;
    createList(L);
    int pilihan;
    do {
        cout << "\n===== MENU DOUBLY LINKED LIST =====\n";
        cout << "1. Tambah Data Kendaraan\n";
        cout << "2. Lihat Semua Data\n";
        cout << "3. Cari Data Kendaraan\n";
        cout << "4. Hapus Data Kendaraan\n";
        cout << "0. Keluar\n";
        cout << "Pilih menu: ";
        cin >> pilihan;

        if (pilihan == 1) {
            infotype x;
            cout << "\nMasukkan Nomor Polisi : ";
            cin >> x.nopol;
            cout << "Masukkan Warna        : ";
            cin >> x.warna;
            cout << "Masukkan Tahun Buat   : ";
            cin >> x.thnBuat;
            insertLast(L, alokasi(x));
            cout << "Data berhasil ditambahkan!\n";
        }

        else if (pilihan == 2) {
            printInfo(L);
        }

        else if (pilihan == 3) {
```

```
            string nopol;
            cout << "\nMasukkan nomor polisi yang dicari: ";
            cin >> nopol;
            address P = findElm(L, nopol);
            if (P != Nil) {
                cout << "\nData ditemukan:\n";
                cout << "Nopol  : " << P->info.nopol << endl;
                cout << "Warna  : " << P->info.warna << endl;
                cout << "Tahun  : " << P->info.thnBuat << endl;
            } else {
                 cout << "\nData dengan nopol " << nopol << " tidak
ditemukan.\n";
            }
        }

        else if (pilihan == 4) {
            string nopol;
            cout << "\nMasukkan nomor polisi yang akan dihapus: ";
            cin >> nopol;
            address P = findElm(L, nopol);
            if (P != Nil) {
                if (P == L.first)
                    deleteFirst(L, P);
                else if (P == L.last)
                    deleteLast(L, P);
                else
                    deleteAfter(P->prev, P);
                dealokasi(P);
                cout << "Data berhasil dihapus.\n";
            } else {
                cout << "Data tidak ditemukan.\n";
            }
        }

    } while (pilihan != 0);

    cout << "\nProgram selesai. Terima kasih!\n";
    return 0;
}
```

Doublylist.cpp

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

void printInfo(List L) {
    if (L.first == Nil) {
        cout << "\nList kosong.\n";
        return;
    }
    address P = L.first;
    cout << "\n=== DATA KENDARAAN ===\n";
    while (P != Nil) {
```

```
        cout << "Nopol  : " << P->info.nopol << endl;
        cout << "Warna  : " << P->info.warna << endl;
        cout << "Tahun  : " << P->info.thnBuat << endl;
        cout << "-----------------------\n";
        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) {
    if (L.first != Nil) {
        P = L.first;
        if (L.first == L.last) {
            L.first = Nil;
            L.last = Nil;
        } else {
            L.first = P->next;
            L.first->prev = Nil;
        }
        P->next = Nil;
    }
}

void deleteLast(List &L, address &P) {
    if (L.last != Nil) {
        P = L.last;
        if (L.first == L.last) {
            L.first = Nil;
            L.last = Nil;
        } else {
            L.last = P->prev;
            L.last->next = 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;
        P->next = Nil;
        P->prev = Nil;
    }
}
```

Doublylist.h

```
#ifndef DOUBLYLIST_H
#define DOUBLYLIST_H

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

#define Nil NULL

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

typedef kendaraan infotype;
typedef struct ElmList *address;

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

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
```

Screenshots Output

```
===== MENU DOUBLY LINKED LIST =====
1. Tambah Data Kendaraan
2. Lihat Semua Data
3. Cari Data Kendaraan
4. Hapus Data Kendaraan
0. Keluar
Pilih menu: 1

Masukkan Nomor Polisi : D001
Masukkan Warna        : HITAM
Masukkan Tahun Buat   : 90
Data berhasil ditambahkan!

===== MENU DOUBLY LINKED LIST =====
1. Tambah Data Kendaraan
2. Lihat Semua Data
3. Cari Data Kendaraan
4. Hapus Data Kendaraan
0. Keluar
Pilih menu: 1

Masukkan Nomor Polisi : D003
Masukkan Warna        : PUTIH
Masukkan Tahun Buat   : 70
Data berhasil ditambahkan!
```

Deskripsi: Program berhasil menerapkan operasi dasar Doubly Linked List dengan tipe data kompleks (struct kendaraan). Fungsi-fungsi bekerja sesuai spesifikasi ADT, dan proses tambah, cari, hapus, serta tampilkan data berjalan dengan benar.

Unguided 2

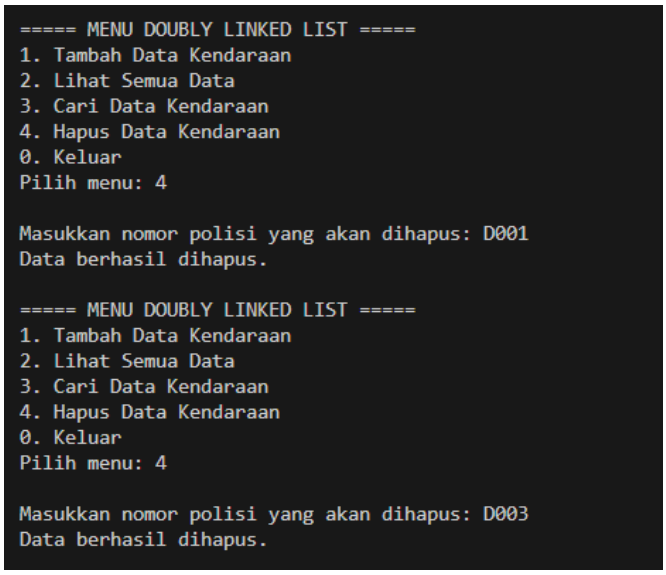
Screenshots Output

```
===== MENU DOUBLY LINKED LIST =====
1. Tambah Data Kendaraan
2. Lihat Semua Data
3. Cari Data Kendaraan
4. Hapus Data Kendaraan
0. Keluar
Pilih menu: 3

Masukkan nomor polisi yang dicari: D003

Data ditemukan:
Nopol  : D003
Warna  : PUTIH
Tahun  : 70

===== MENU DOUBLY LINKED LIST =====
1. Tambah Data Kendaraan
2. Lihat Semua Data
3. Cari Data Kendaraan
4. Hapus Data Kendaraan
0. Keluar
Pilih menu: 3

Masukkan nomor polisi yang dicari: D001

Data ditemukan:
Nopol  : D001
Warna  : HITAM
Tahun  : 90
```

Deskripsi: Percobaan tambahan untuk menguji kemampuan Doubly Linked List dengan melakukan operasi pencarian.

Unguided 3

Screenshots Output



Deskripsi: Percobaan tambahan untuk menguji kemampuan Doubly Linked List dengan melakukan operasi mengahpus data yang ditambahkan sebelumnya.

D. Kesimpulan

Dari percobaan yang dilakukan pada Modul 6:

- Doubly Linked List memiliki dua pointer di setiap node (next dan prev) yang memungkinkan traversal dua arah.
- Operasi dasar seperti tambah, hapus, dan cari data dapat dilakukan dengan efisien tanpa perlu menelusuri seluruh list dari awal.
- Implementasi ADT dengan tiga file (.h, .cpp, .main) membantu memisahkan logika, struktur, dan eksekusi program sehingga lebih modular dan mudah dikelola.
- Program berhasil menampilkan hasil sesuai dengan contoh output dan menunjukkan pemahaman konsep list berantai ganda.

E. Referensi

1. Modul Praktikum Struktur Data, **Modul 6 – Doubly Linked List**, Telkom University Purwokerto, 2025.
2. Nugroho, Adi. *Algoritma dan Struktur Data dengan C++*. Informatika Bandung, 2020.
3. Malik, D.S. *Data Structures Using C++*, Cengage Learning, 2018.
4. Materi kuliah dan diskusi praktikum Struktur Data, Dosen **Fahrudin Mukti Wibowo**, Telkom University Purwokerto.