LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VI
DOUBLY LINKED LIST

Disusun Oleh :
NAMA : ABYAN RAHMAN AL FARIZ
NIM : 103112430021

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Doubly Linked List merupakan jenis linked list di mana setiap elemen memiliki dua penunjuk (pointer): satu yang mengarah ke elemen sebelumnya (prev) dan satu lagi yang mengarah ke elemen berikutnya (next).

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

Guided 1 (main.cpp)

```
 #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;
        }
        else if (current == ptr_last)
        {
            delete_last();
            return;
        }
        else
        {
            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 << "Setelah delete_target\t: ";
    view();

}
```

Screenshots Output:

```
PS D:\TelkomUniversity\Mata Kuliah\Semester 3\Struktur Data\Praktikum>  & 'c:\Users\ACER\.vscode\extensions\ms-vscod
e.cpptools-1.28.3-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-4q1w4obc.w40'
 '--stdout=Microsoft-MIEngine-Out-1faclimp.j00' '--stderr=Microsoft-MIEngine-Error-v3pfmorp.2jo' '--pid=Microsoft-MI
Engine-Pid-wcj3lnog.jvl' '--dbgExe=C:\Users\ACER\mingw32\bin\gdb.exe' '--interpreter=mi'
Awal                    : 5<->10<->20
Setelah delete_first    : 10<->20
Setelah delete_last     : 10
Setelah tambah          : 10<->30<->40
Setelah delete_target   : 10<->40
```

Deskripsi:

Program ini merupakan implementasi dari Doubly Linked List yang digunakan untuk menyimpan dan memanipulasi data bertipe integer secara dinamis. Setiap node memiliki tiga komponen utama, yaitu data sebagai nilai yang disimpan, serta dua pointer, prev untuk menunjuk ke elemen sebelumnya dan next untuk menunjuk ke elemen berikutnya. Program ini menyediakan beberapa operasi dasar pada list, seperti add_first() untuk menambah elemen di awal list, add_last() untuk menambah elemen di akhir, dan add_target() untuk menambah elemen setelah data tertentu.

Selain itu, terdapat prosedur penghapusan seperti delete_first() untuk menghapus elemen pertama, delete_last() untuk menghapus elemen terakhir, serta delete_target() untuk menghapus elemen tertentu berdasarkan nilainya. Program juga menyediakan edit_node() untuk mengubah nilai data pada node tertentu dan view() untuk menampilkan isi list secara berurutan menggunakan simbol <-> sebagai penghubung antar node.

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

Unguided 1 (Doublylist.cpp)

```cpp
#include "Doublylist.h"

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

address alokasi(infotype 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 printInfo(List L) {
    cout << "DATA LIST 1" << endl;
    address P = L.First;
    while (P != NULL) {
        cout << "Nomor Polisi : " << P->info.nopol << endl;
        cout << "Warna        : " << P->info.warna << endl;
        cout << "Tahun        : " << P->info.thnBuat << endl;
        cout << endl;
        P = P->next;
    }
}

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;
    }
}

address findElm(List L, infotype x) {
    address P = L.First;
    while (P != NULL) {
        if (P->info.nopol == x.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.Last == NULL) {
        P = NULL;
    } else if (L.First == L.Last) {
        P = L.Last;
        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;
    }
}
```

Unguided 1 (Doublylist.h)

```h
#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, infotype x);

void deleteFirst(List &L, address &P);
void deleteLast(List &L, address &P);
void deleteAfter(address Prec, address &P);

#endif
```

Unguided 1 (main.cpp)

```cpp
#include "Doublylist.h"
#include "Doublylist.cpp"
#include <iostream>

using namespace std;

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

    infotype x;
    address P;

    for (int i = 0; i < 4; i++) {
        cout << "Masukkan Nomor Polisi : ";
        cin >> x.nopol;

        infotype temp;
        temp.nopol = x.nopol;

        if (findElm(L, temp) != NULL) {
            cout << "Nomor polisi sudah terdaftar\n\n";
            continue;
        }

        cout << "Masukkan Warna Kendaraan : ";
        cin >> x.warna;
        cout << "Masukkan Tahun Kendaraan : ";
        cin >> x.thnBuat;
        cout << endl;

        P = alokasi(x);
        insertLast(L, P);
    }

    cout << endl;
    printInfo(L);

    infotype cari;
    cout << "Masukkan Nomor Polisi yang dicari : ";
```

```
    cin >> cari.nopol;

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

    cout << endl;

    infotype hapus;
    cout << "Masukkan Nomor Polisi yang akan dihapus : ";
    cin >> hapus.nopol;

    address del = findElm(L, hapus);
    if (del != NULL) {
        if (del == L.First) {
            deleteFirst(L, del);
        } else if (del == L.Last) {
            deleteLast(L, del);
        } else {
            deleteAfter(del->prev, del);
        }
        cout << "Data dengan nomor polisi " << hapus.nopol << "
berhasil dihapus." << endl;
        dealokasi(del);
    } else {
        cout << "Data tidak ditemukan." << endl;
    }

    cout << endl;
    printInfo(L);

    return 0;
}
```

Screenshots Output:

```
PS D:\TelkomUniversity\Mata Kuliah\Semester 3\Struktur Data\Praktikum\Modul 6\Unguided> cd "d:\TelkomUniversity\Mata
 Kuliah\Semester 3\Struktur Data\Praktikum\Modul 6\Unguided\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main
}
Masukkan Nomor Polisi : D001
Masukkan Warna Kendaraan : hitam
Masukkan Tahun Kendaraan : 90

Masukkan Nomor Polisi : D003
Masukkan Warna Kendaraan : putih
Masukkan Tahun Kendaraan : 70

Masukkan Nomor Polisi : D001
Nomor polisi sudah terdaftar

Masukkan Nomor Polisi : D004
Masukkan Warna Kendaraan : kuning
Masukkan Tahun Kendaraan : 90


DATA LIST 1
Nomor Polisi : D001
Warna        : hitam
Tahun        : 90

Nomor Polisi : D003
Warna        : putih
Tahun        : 70

Nomor Polisi : D004
Warna        : kuning
Tahun        : 90
```

```
Masukkan Nomor Polisi yang dicari : D001

Nomor Polisi : D001
Warna        : hitam
Tahun        : 90
```

```
Masukkan Nomor Polisi yang akan dihapus : D003
Data dengan nomor polisi D003 berhasil dihapus.

DATA LIST 1
Nomor Polisi : D001
Warna        : hitam
Tahun        : 90

Nomor Polisi : D004
Warna        : kuning
Tahun        : 90
```

Deskripsi:

Program ini menerapkan konsep Doubly Linked List untuk mengelola data kendaraan secara dinamis. Setiap elemen pada list memiliki dua pointer, yaitu next yang menunjuk ke elemen berikutnya dan prev yang menunjuk ke elemen sebelumnya, sehingga memungkinkan penelusuran data dari dua arah. Tipe data kendaraan digunakan sebagai infotype, yang menyimpan atribut Nomor Polisi, Warna, dan Tahun Pembuatan. Program ini memanfaatkan beberapa fungsi utama, seperti CreateList() untuk inisialisasi list kosong, alokasi() untuk membuat elemen baru, insertLast() untuk menambahkan data kendaraan di akhir list, findElm() untuk mencari data berdasarkan nomor polisi, serta deleteFirst(), deleteLast(), dan deleteAfter() untuk menghapus elemen sesuai posisinya dalam list. Prosedur printInfo() digunakan untuk menampilkan seluruh data kendaraan yang tersimpan.

D. Kesimpulan

Double Linked List merupakan struktur data dinamis yang memungkinkan setiap elemen (node) terhubung dua arah melalui dua pointer, yaitu next (ke elemen berikutnya) dan prev (ke elemen sebelumnya). Dengan adanya dua arah ini, proses penelusuran, penambahan, dan penghapusan data menjadi lebih fleksibel dibandingkan dengan singly linked list, karena kita dapat bergerak maju maupun mundur di dalam list.

E. Referensi

https://www.w3schools.com/dsa/dsa_data_linkedlists_types.php

https://www.programiz.com/dsa/doubly-linked-list

https://en.wikipedia.org/wiki/Doubly_linked_list