LAPORAN PRAKTIKUM
Modul 6
Double Linked List Bagian 1

Disusun Oleh :
Satria Ariq Adelard Dompas/2211104033
SE 06 2

Asisten Praktikum :
Aldi Putra
Andini Nur Hidayah

Dosen Pengampu :
Wahyu Andi Saputra

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2024

1. **Tujuan**
    a. Mahasiswa mampu memahami konsep Double Linked List.
    b. Mahasiswa mampu memahami operasi dasar dalam Double Linked List.
    c. Mahasiswa mampu membuat program dengan menggunakan Double Linked List denganprorotype yang ada.

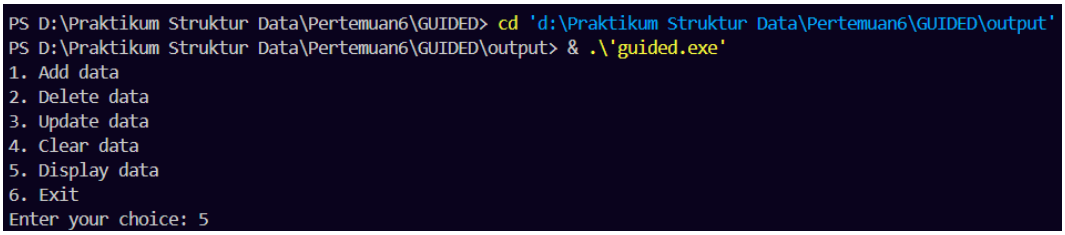
2. **Landasan Teori**
    a. Double Linked List

        Double linked list adalah struktur data yang terdiri dari elemen-elemen, di mana setiap elemen memiliki dua pointer: satu menunjuk ke elemen berikutnya (next) dan satu lagi ke elemen sebelumnya (prev). Keunggulannya adalah memungkinkan traversal dua arah, yang memudahkan operasi seperti penambahan, penghapusan, dan pencarian elemen. Meskipun membutuhkan lebih banyak memori untuk menyimpan pointer tambahan, double linked list menawarkan fleksibilitas dan efisiensi yang lebih baik dalam pengelolaan data.

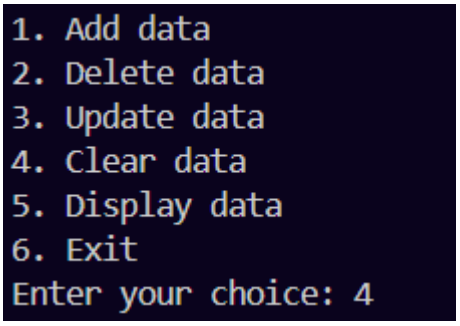
3. **Guided**
    a. Guided 1

    Output

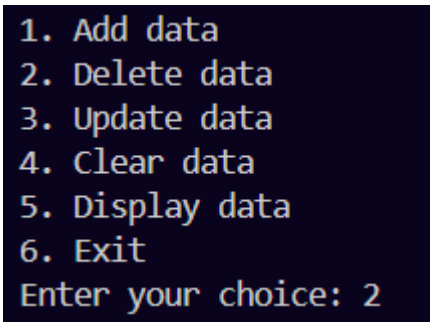
    - Add



    - Clear



    - Delete



- Display

```
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
15
```

- Update

```
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 1
Enter data to add: 20
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 3
Enter old data: 20
Enter new data: 15
```

- Exit

```
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 6
PS D:\Praktikum Struktur Data\Pertemuan6\GUIDED\output>
```

Source Code

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

class Node {
public:
    int data;
    Node* prev;
    Node* next;
};

class DoublyLinkedList {
public:
    Node* head;
    Node* tail;

    // Constructor untuk inisialisasi head dan tail
    DoublyLinkedList() {
        head = nullptr;
```

```
        tail = nullptr;
    }

    // Fungsi untuk menambahkan elemen di depan list
    void insert(int data) {
        Node* newNode = new Node;
        newNode->data = data;
        newNode->prev = nullptr;
        newNode->next = head;

        if (head != nullptr) {
            head->prev = newNode;
        } else {
            tail = newNode; // Jika list kosong, tail juga mengarah ke node
baru
        }
        head = newNode;
    }

    // Fungsi untuk menghapus elemen dari depan list
    void deleteNode() {
        if (head == nullptr) {
            return; // Jika list kosong
        }
        Node* temp = head;
        head = head->next;
        if (head != nullptr) {
            head->prev = nullptr;
        } else {
            tail = nullptr; // Jika hanya satu elemen di list
        }
        delete temp; // Hapus elemen
    }

    // Fungsi untuk mengupdate data di list
    bool update(int oldData, int newData) {
        Node* current = head;
        while (current != nullptr) {
            if (current->data == oldData) {
                current->data = newData;
                return true; // Jika data ditemukan dan diupdate
            }
            current = current->next;
        }
        return false; // Jika data tidak ditemukan
    }

    // Fungsi untuk menghapus semua elemen di list
    void deleteAll() {
        Node* current = head;
        while (current != nullptr) {
            Node* temp = current;
            current = current->next;
```

```
            delete temp;
        }
        head = nullptr;
        tail = nullptr;
    }

    // Fungsi untuk menampilkan semua elemen di list
    void display() {
        Node* current = head;
        while (current != nullptr) {
            cout << current->data << " ";
            current = current->next;
        }
        cout << endl;
    }
};

int main() {
    DoublyLinkedList list;
    while (true) {
        cout << "1. Add data" << endl;
        cout << "2. Delete data" << endl;
        cout << "3. Update data" << endl;
        cout << "4. Clear data" << endl;
        cout << "5. Display data" << endl;
        cout << "6. Exit" << endl;

        int choice;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1: {
                int data;
                cout << "Enter data to add: ";
                cin >> data;
                list.insert(data);
                break;
            }
            case 2: {
                list.deleteNode();
                break;
            }
            case 3: {
                int oldData, newData;
                cout << "Enter old data: ";
                cin >> oldData;
                cout << "Enter new data: ";
                cin >> newData;
                bool updated = list.update(oldData, newData);
                if (!updated) {
                    cout << "Data not found" << endl;
                }
```

```
                break;
            }
            case 4: {
                list.deleteAll();
                break;
            }
            case 5: {
                list.display();
                break;
            }
            case 6: {
                return 0;
            }
            default: {
                cout << "Invalid choice" << endl;
                break;
            }
        }
    }
    return 0;
}
```

Deskripisi

Program di atas merupakan implementasi dari struktur data Double Linked List di C++, yang memungkinkan pengguna melakukan berbagai operasi dasar seperti menambah, menghapus, memperbarui, dan menampilkan data. Kelas Node mendefinisikan elemen dengan atribut untuk menyimpan data dan penunjuk ke node sebelumnya dan berikutnya. Kelas DoublyLinkedList mengelola daftar dengan metode untuk memasukkan data di awal, menghapus node pertama, memperbarui nilai, menghapus semua node, dan menampilkan semua elemen. Antarmuka berbasis teks menyediakan menu dimana pengguna dapat memilih tindakan yang diinginkan, dan perulangan berlanjut hingga pengguna memutuskan untuk keluar dari program. Program ini membantu Anda memahami dasar-dasar daftar tertaut ganda C++ dan manajemen memori.

4. **Unguided**
    a. Soal 1

    Source Code

    doublelist.h

```
#ifndef DOUBLELIST_H
#define DOUBLELIST_H

#include <string>

struct infotype {
    std::string nopol;  // Nomor Polisi
    std::string warna;  // Warna Kendaraan
    int thnBuat;        // Tahun Pembuatan
```

```
};

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

struct List {
    ElmList* First;
    ElmList* Last;
};

// Deklarasi fungsi
void CreateList(List &L);
ElmList* alokasi(infotype x);
void dealokasi(ElmList* &P);
void printInfo(const List &L);
void insertLast(List &L, ElmList* P);
ElmList* findElm(const List &L, infotype x);
void deleteFirst(List &L, ElmList* &P);
void deleteLast(List &L, ElmList* &P);
void deleteAfter(ElmList* Prec, ElmList* &P);

#endi
```

singlelist.cpp

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

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

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

void dealokasi(ElmList* &P) {
    delete P;
    P = nullptr;
}

void insertLast(List &L, ElmList* P) {
    if (L.First == nullptr) {
        L.First = P;
        L.Last = P;
    } else {
        L.Last->next = P;
        P->prev = L.Last;
        L.Last = P;
    }
}

void printInfo(const List &L) {
    ElmList* P = L.First;
    while (P != nullptr) {
        std::cout << "no polisi : " << P->info.nopol << std::endl;
        std::cout << "warna     : " << P->info.warna << std::endl;
        std::cout << "tahun     : " << P->info.thnBuat << std::endl;
        P = P->next;
    }
}

ElmList* findElm(const List &L, infotype x) {
    ElmList* P = L.First;
    while (P != nullptr) {
        if (P->info.nopol == x.nopol) {
            return P;
        }
        P = P->next;
    }
    return nullptr;
```

```
}

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

void deleteLast(List &L, ElmList* &P) {
    if (L.Last != nullptr) {
        P = L.Last;
        if (L.First == L.Last) {
            L.First = nullptr;
            L.Last = nullptr;
        } else {
            L.Last = L.Last->prev;
            L.Last->next = nullptr;
        }
        P->prev = nullptr;
    }
}

void deleteAfter(ElmList* Prec, ElmList* &P) {
    if (Prec != nullptr && Prec->next != nullptr) {
        P = Prec->next;
        Prec->next = P->next;
        if (P->next != nullptr) {
            P->next->prev = Prec;
        }
        P->next = nullptr;
        P->prev = nullptr;
    }
}
```

main.cpp

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

void inputKendaraan(infotype &kendaraan) {
    std::cout << "masukkan nomor polisi: ";
    std::cin >> kendaraan.nopol;
    std::cout << "masukkan warna kendaraan: ";
    std::cin >> kendaraan.warna;
    std::cout << "masukkan tahun kendaraan: ";
    std::cin >> kendaraan.thnBuat;
}

void tampilkanMenu() {
    std::cout << "\nMENU\n";
    std::cout << "1. Tambah Data Kendaraan\n";
    std::cout << "2. Cari Data Kendaraan\n";
    std::cout << "3. Hapus Data Kendaraan Pertama\n";
    std::cout << "4. Hapus Data Kendaraan Terakhir\n";
    std::cout << "5. Hapus Data Setelah Kendaraan Tertentu\n";
    std::cout << "6. Tampilkan Semua Data Kendaraan\n";
    std::cout << "0. Keluar\n";
    std::cout << "Pilih menu: ";
}

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

    int pilihan;
    do {
        tampilkanMenu();
        std::cin >> pilihan;
        std::cin.ignore();  // Mengabaikan newline setelah pilihan

        switch (pilihan) {
            case 1: {
                // Tambah data kendaraan
                infotype kendaraan;
                inputKendaraan(kendaraan);

                // Cek apakah nomor polisi sudah ada
                if (findElm(L, kendaraan) != nullptr) {
                    std::cout << "nomor polisi sudah terdaftar\n";
                } else {
                    insertLast(L, alokasi(kendaraan));
                    std::cout << "Data kendaraan berhasil ditambahkan.\n";
                }
                break;
            }
```

```
            case 2: {
                // Cari data kendaraan
                infotype cari;
                std::cout << "Masukkan Nomor Polisi yang dicari : ";
                std::cin >> cari.nopol;

                ElmList* found = findElm(L, cari);
                if (found != nullptr) {
                    std::cout << "Nomor Polisi: " << found->info.nopol <<
std::endl;
                    std::cout << "Warna       : " << found->info.warna <<
std::endl;
                    std::cout << "Tahun       : " << found->info.thnBuat <<
std::endl;
                } else {
                    std::cout << "nomor polisi tidak ditemukan.\n";
                }
                break;
            }
            case 3: {
                // Hapus data kendaraan pertama
                ElmList* deleted;
                deleteFirst(L, deleted);
                if (deleted != nullptr) {
                    std::cout << "Data kendaraan pertama berhasil
dihapus.\n";
                    dealokasi(deleted);
                } else {
                    std::cout << "List kosong, tidak ada data untuk
dihapus.\n";
                }
                break;
            }
            case 4: {
                // Hapus data kendaraan terakhir
                ElmList* deleted;
                deleteLast(L, deleted);
                if (deleted != nullptr) {
                    std::cout << "Data kendaraan terakhir berhasil
dihapus.\n";
                    dealokasi(deleted);
                } else {
                    std::cout << "List kosong, tidak ada data untuk
dihapus.\n";
                }
                break;
            }
            case 5: {
                // Hapus data setelah kendaraan tertentu
                infotype cari;
                std::cout << "Masukkan nomor polisi kendaraan sebelum data
yang ingin dihapus: ";
                std::cin >> cari.nopol;
```

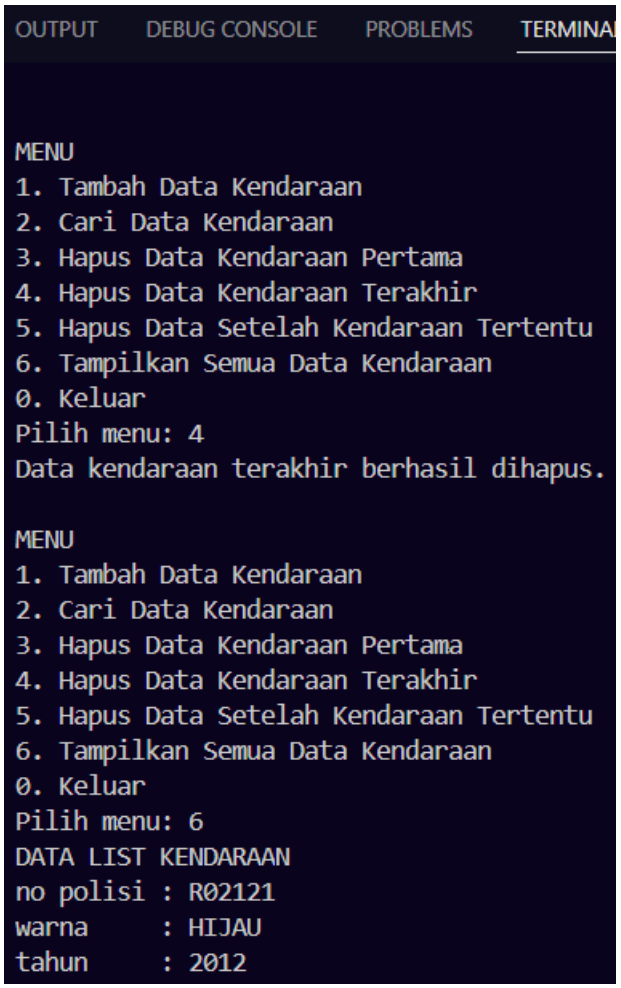
```
                ElmList* prec = findElm(L, cari);
                if (prec != nullptr && prec->next != nullptr) {
                    ElmList* deleted;
                    deleteAfter(prec, deleted);
                    if (deleted != nullptr) {
                        std::cout << "Data dengan nomor polisi " << deleted-
>info.nopol << " berhasil dihapus.\n";
                        dealokasi(deleted);
                    }
                } else {
                    std::cout << "Tidak ada data untuk dihapus setelah nomor
polisi tersebut.\n";
                }
                break;
            }
            case 6: {
                // Tampilkan semua data kendaraan
                std::cout << "DATA LIST KENDARAAN\n";
                printInfo(L);
                break;
            }
            case 0: {
                std::cout << "Keluar dari program.\n";
                break;
            }
            default:
                std::cout << "Pilihan tidak valid. Silakan coba lagi.\n";
                break;
        }
    } while (pilihan != 0);

    return 0;
}
```

Output

```
MENU
1. Tambah Data Kendaraan
2. Cari Data Kendaraan
3. Hapus Data Kendaraan Pertama
4. Hapus Data Kendaraan Terakhir
5. Hapus Data Setelah Kendaraan Tertentu
6. Tampilkan Semua Data Kendaraan
0. Keluar
Pilih menu: 2
Masukkan Nomor Polisi yang dicari : R02121
Nomor Polisi: R02121
Warna       : HIJAU
Tahun       : 2012

MENU
1. Tambah Data Kendaraan
2. Cari Data Kendaraan
3. Hapus Data Kendaraan Pertama
4. Hapus Data Kendaraan Terakhir
5. Hapus Data Setelah Kendaraan Tertentu
6. Tampilkan Semua Data Kendaraan
0. Keluar
Pilih menu:
```

OUTPUT DEBUG CONSOLE PROBLEMS TERMINAL PORTS COMMENTS

```
masukkan tahun kendaraan: 2012
Data kendaraan berhasil ditambahkan.

MENU
1. Tambah Data Kendaraan
2. Cari Data Kendaraan
3. Hapus Data Kendaraan Pertama
4. Hapus Data Kendaraan Terakhir
5. Hapus Data Setelah Kendaraan Tertentu
6. Tampilkan Semua Data Kendaraan
0. Keluar
Pilih menu: 6
DATA LIST KENDARAAN
no polisi : R02121
warna     : HIJAU
tahun     : 2012

MENU
1. Tambah Data Kendaraan
2. Cari Data Kendaraan
3. Hapus Data Kendaraan Pertama
4. Hapus Data Kendaraan Terakhir
5. Hapus Data Setelah Kendaraan Tertentu
6. Tampilkan Semua Data Kendaraan
0. Keluar
Pilih menu:
```


OUTPUT
DEBUG CONSOLE
PROBLEMS
TERMINA
MENU
1. Tambah Data Kendaraan
2. Cari Data Kendaraan
3. Hapus Data Kendaraan Pertama
4. Hapus Data Kendaraan Terakhir
5. Hapus Data Setelah Kendaraan Tertentu
6. Tampilkan Semua Data Kendaraan
0. Keluar
Pilih menu: 4
Data kendaraan terakhir berhasil dihapus.
MENU
1. Tambah Data Kendaraan
2. Cari Data Kendaraan
3. Hapus Data Kendaraan Pertama
4. Hapus Data Kendaraan Terakhir
5. Hapus Data Setelah Kendaraan Tertentu
6. Tampilkan Semua Data Kendaraan
0. Keluar
Pilih menu: 6
DATA LIST KENDARAAN
no polisi : R02121
warna : HIJAU
tahun : 2012

Deskripsi

Program ini merupakan implementasi dari struktur data double linked list yang digunakan untuk menyimpan informasi kendaraan seperti plat nomor, warna, dan tahun pembuatan. Program ini memungkinkan pengguna untuk menambahkan detail kendaraan, mencari kendaraan berdasarkan nomor polisi, dan menghapus detail kendaraan dari daftar. Langkah-langkah utamanya meliputi membuat daftar, mengalokasikan memori untuk item baru, mencari dan menghapus item, dan menangani pelat nomor untuk menghindari duplikat. Informasi ditampilkan dalam urutan LIFO (Last In First Out), sehingga ketika Anda mencetak daftar, item yang paling baru ditambahkan akan muncul terlebih dahulu.

.