LAPORAN PRAKTIKUM
Modul 6
Double Linked List Bagian 1

Disusun Oleh :
Ade Fatkhul Anam/2211104051
SE 06 02

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. Memahami konsep Double Linked List.
    b. Memahami operasi dasar dalam Double Linked List.
    c. Membuat program dengan menggunakan Double Linked List dengan prorotype yang ada.

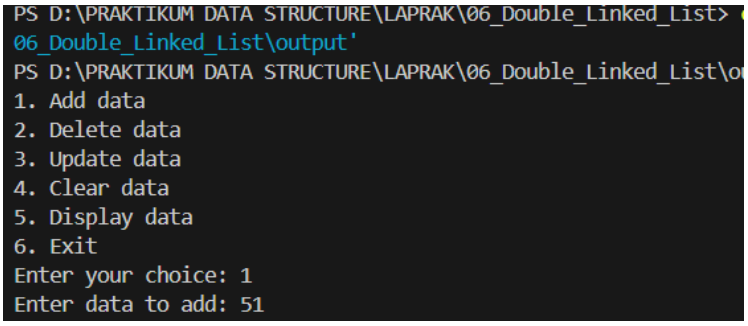
2. **Landasan Teori**
    a. Double Linked List

        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.

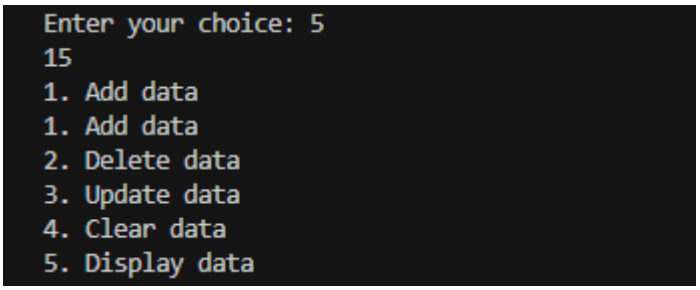
3. **Guided**
    a. Guided 1

    Output

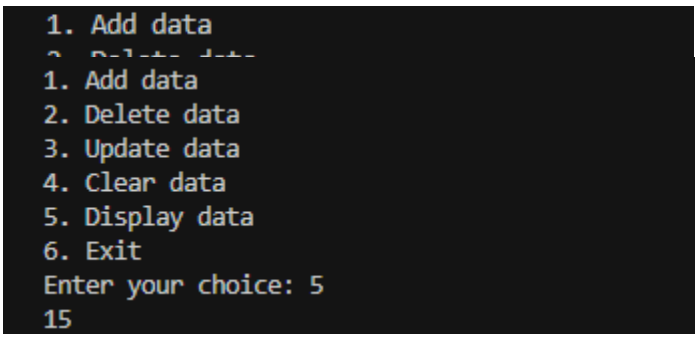
    - Add



    - Clear

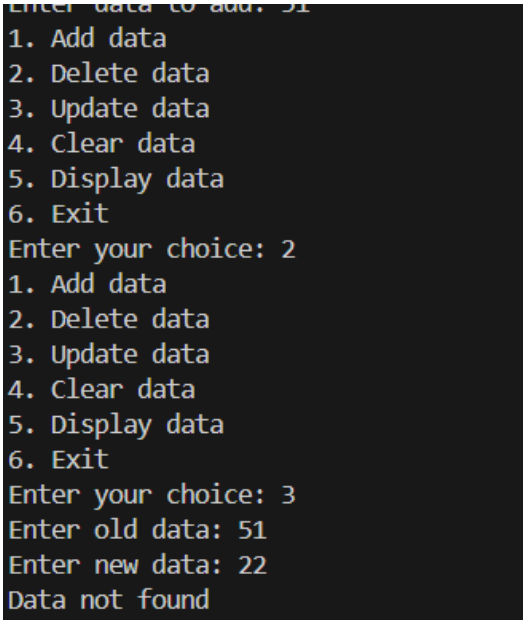


    - Delete

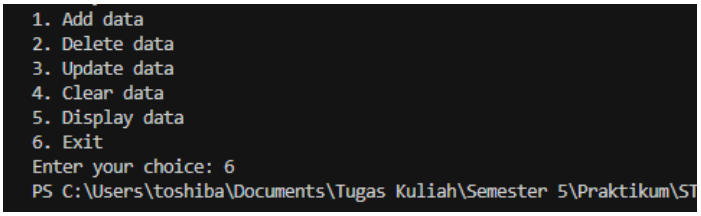


    - Display

- Update



- Exit



Source Code

```
#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 adalah implementasi dari struktur data Double Linked List dalam bahasa C++, yang memungkinkan pengguna untuk melakukan berbagai operasi dasar seperti menambahkan, menghapus, memperbarui, dan menampilkan data. Kelas `Node` mendefinisikan elemen dengan atribut untuk menyimpan data dan pointer ke node sebelumnya serta berikutnya. Kelas `DoublyLinkedList` mengelola daftar dengan metode untuk menyisipkan data di depan, menghapus node dari depan, memperbarui nilai, menghapus semua node, dan menampilkan semua elemen. Antarmuka pengguna berbasis teks menyediakan menu yang memungkinkan pengguna untuk memilih operasi yang diinginkan, dengan loop yang terus berjalan hingga pengguna memutuskan untuk keluar dari program. Program ini berguna untuk memahami dasar-dasar doubly linked list dan manajemen memori dalam C++.

4. Unguided

a. Soal 1

Source Code

singlelist.h

```
#ifndef DOUBLELIST_H
#define DOUBLELIST_H

#include <string>
using namespace std;

struct infotype {
    string nopol;
    string warna;
    int thnBuat;
```

```
};

typedef struct ElmList *address;

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

struct List {
    address First;
    address Last;
};

// Prosedur dan fungsi dasar
void CreateList(List &L);
address alokasi(infotype x);
void dealokasi(address &P);
void printInfo(List L);
void insertLast(List &L, address P);

// Fungsi tambahan
address findElm(List L, infotype x);
void deleteFirst(List &L, address &P);
void deleteLast(List &L, address &P);
void deleteAfter(address Prec, address &P);
void deleteData(List &L, string nopol); // Prototipe fungsi deleteData

#endif
```

singlelist.cpp

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

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) {
    address P = L.Last;  // Start from the last element
    cout << "DATA LIST (LIFO)" << endl;
    while (P != NULL) {
        cout << "No polisi : " << P->info.nopol << endl;
        cout << "Warna     : " << P->info.warna << endl;
        cout << "Tahun     : " << P->info.thnBuat << endl;
        P = P->prev;  // Move to the previous element
    }
}

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

```
    }
}

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

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

void deleteData(List &L, string nopol) {
    infotype searchData;
    searchData.nopol = nopol;

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

```
        cout << "Data tidak ditemukan!" << endl;
    }
}
```

main.cpp

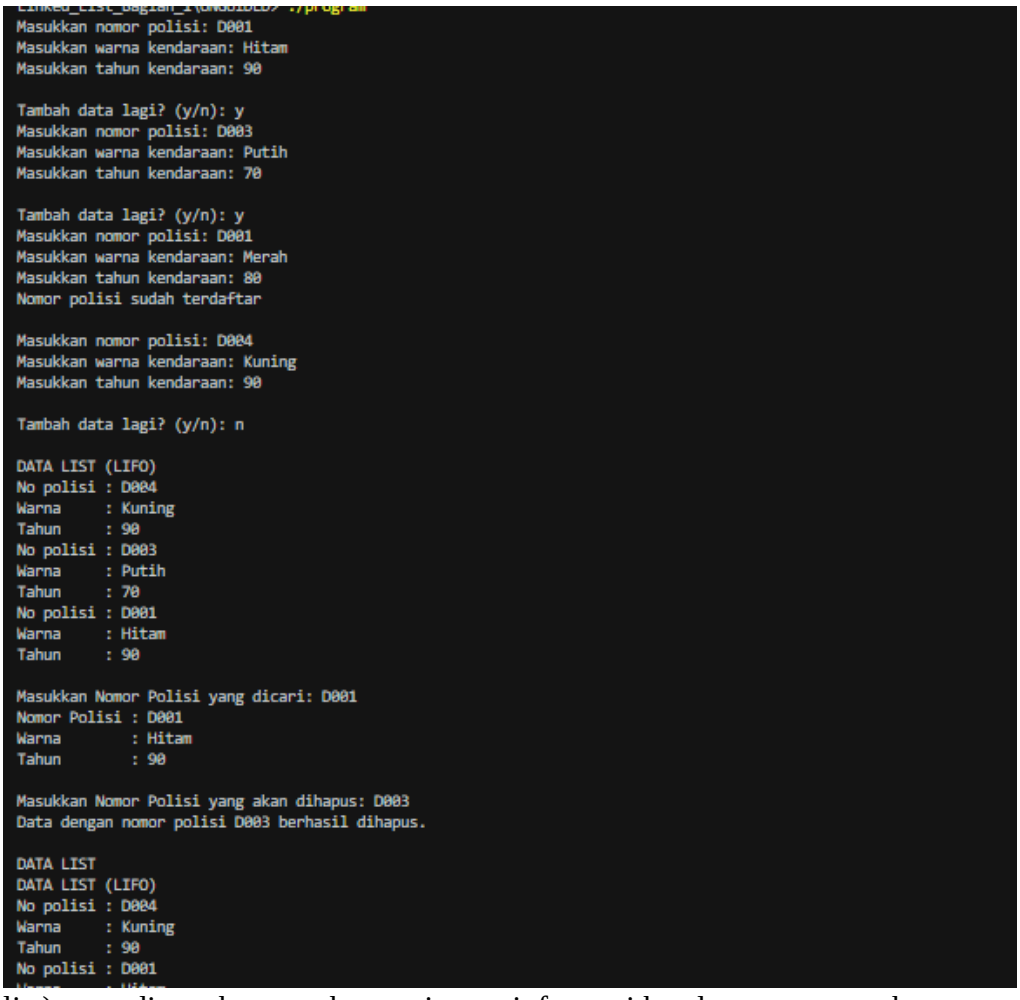
```
#include "doublelist.h"
#include <iostream>

using namespace std;

int main() {
    List L;
    CreateList(L);
    infotype kendaraan;
    char again;

    // Input data
    do {
        cout << "Masukkan nomor polisi: ";
        cin >> kendaraan.nopol;
        cout << "Masukkan warna kendaraan: ";
        cin >> kendaraan.warna;
        cout << "Masukkan tahun kendaraan: ";
        cin >> kendaraan.thnBuat;

        // Cek duplikasi nomor polisi
        if (findElm(L, kendaraan) != NULL) {
            cout << "Nomor polisi sudah terdaftar" << endl << endl;
            continue;
        }

        address P = alokasi(kendaraan);
        insertLast(L, P);

        cout << "\nTambah data lagi? (y/n): ";
        cin >> again;
    } while (again == 'y' || again == 'Y');

    cout << "\n";
    printInfo(L);

    // Fitur pencarian
    cout << "\nMasukkan Nomor Polisi yang dicari: ";
    cin >> kendaraan.nopol;

    address found = findElm(L, kendaraan);
    if (found != NULL) {
        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;
    }

    // Fitur penghapusan
    cout << "\nMasukkan Nomor Polisi yang akan dihapus: ";
    string nopolHapus;
    cin >> nopolHapus;

    deleteData(L, nopolHapus);

    cout << "\nDATA LIST" << endl;
    printInfo(L);

    return 0;
}
```

Output

Deskripsi

Program ini merupakan implementasi struktur data daftar ganda (double linked



list) yang digunakan untuk menyimpan informasi kendaraan, termasuk nomor polisi, warna, dan tahun pembuatan. Dalam program ini, pengguna dapat menambahkan data kendaraan, mencari kendaraan berdasarkan nomor polisi, serta menghapus data kendaraan dari daftar. Prosedur utama meliputi pembuatan daftar, alokasi memori untuk elemen baru, pencarian elemen, dan

penghapusan elemen, dengan penanganan untuk menghindari duplikasi nomor polisi. Informasi ditampilkan dalam urutan LIFO (Last In First Out), sehingga elemen terakhir yang ditambahkan akan ditampilkan terlebih dahulu saat mencetak daftar.