LAPORAN PRAKTIKUM
Modul 6
Double Linked List Bagian 1

Disusun Oleh :
Fauzan Rofif Ardiyanto
2211104036
S1SE06-02
Asisten Praktikum :
Aldi Putra
Andini Nur Hidayah

Dosen Pengampu :
Wahyu Andi Saputra

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2024

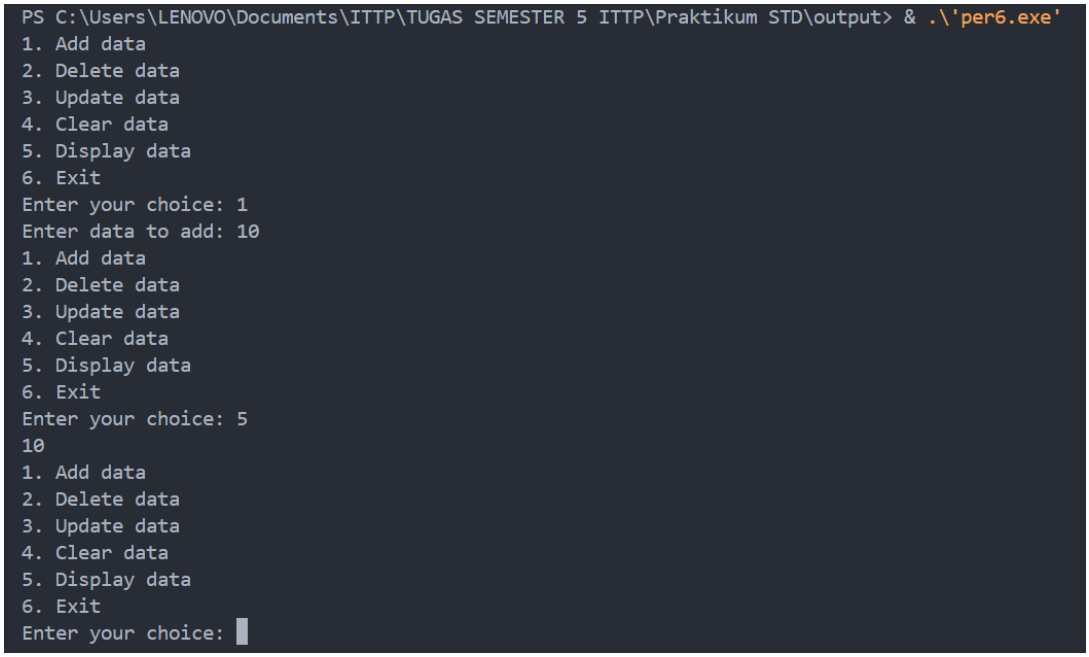
1. **Tujuan**
    a. Memahami konsep Double Linked List.
    b. Memahami operasi dasar dalam Double Linked List.
    c. Membuat program dengan menggunakan Double Linked List dengan prorotype yang ada.

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**

Guided



Source Code

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

//Struktur untuk node dalam linked list
struct Node {
    int data;
    Node* next;
};
// Fungsi untuk menambahkan elemen baru ke awal linked list

void insertFirst(Node*& head, Node*& tail, int new_data) {
    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = head;
    head = new_node;

    if(tail == nullptr){
        tail = new_node;
    }
}
// Fungsi untuk menambahkan elemen baru ke akhir linked list
void insertLast(Node*& head, Node*& tail, int new_data) {
    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = head;
    head = new_node;

    if(tail == nullptr){
        tail = new_node;
    }
}

// Fungsi untuk mencari elemen dalam linked list
int findElement(Node* head, int x) {
    Node* current = head;
    int index = 0;

    while (current != nullptr) {
        if (current->data == x) {
            return index;
        }
        current = current->next;
        index++;
```

```
    }
    return -1;
}

//Fungsi untuk menampilkan elemen dalam linked list
void display(Node* node) {
    while (node != nullptr){
        cout << node->data << " ";
        node = node->next;
    }
    cout << endl;
}

// FUngsi untuk menghapus elemen dari linked list
void deleteElement(Node*& head, int x) {
    if (head == nullptr){
        cout << "Linked list kosong" << endl;
        return;
    }

    if (head->data == x){
        Node* temp = head;
        head = head->next;
        delete temp;
        return;
    }
    Node* current = head;
    while (current->next !=nullptr){
        if(current-> next-> data == x){
        Node* temp = current->next;
        current->next = current->next->next;
        delete temp;
        return;
        }
        current = current->next;
    }
}

int main() {
    Node* head = nullptr;
    Node* tail = nullptr;

    insertFirst(head, tail, 3);
    insertFirst(head, tail, 5);
```

```
    insertFirst(head, tail, 7);

    insertLast(head, tail, 11);
    insertLast(head, tail, 14);
    insertLast(head, tail, 15);

    cout << "Elemen dalam linked list: ";
    display(head);

    int x;
    cout << "Masukkan elemen yang ingin dicari ";
    cin >> x;

    int result = findElement(head, x);

    if(result == -1)
    cout << "Elemen tidak ditemukan dalam linked list" << endl;
    else
    cout << "Elemen ditemukan pada indeks" << result <<  endl;

    cout << "Masukkan elemen yang ingin dihapus: ";
    cin >> x;
    deleteElement(head, x);

    cout << "Elemen dalam linked list setelah penghapusan: ";
    display(head);
    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

Source Code

```
#ifndef DOUBLELIST_H
#define DOUBLELIST_H

#include <string>
using namespace std;

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

```h
// doublelist.h
#ifndef DOUBLELIST_H
#define DOUBLELIST_H

#include <iostream>
using namespace std;

// Type infotype
struct kendaraan
{
    string nopol;
    string warna;
    int thnBuat;
};

// Type address sebagai pointer to ElmList
typedef struct ElmList *address;

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

// Type List
struct List
{
    address First;
    address Last;
};

// Prosedur dan Fungsi ADT
void CreateList(List &L);
address alokasi(kendaraan x);
void dealokasi(address &P);
void printInfo(List L);
void insertLast(List &L, address P);
address findElm(List L, string nopol);
void deleteFirst(List &L, address &P);
void deleteLast(List &L, address &P);
void deleteAfter(address Prec, address &P);
```

```cpp
#endif

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

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

// Prosedur dealokasi
void dealokasi(address &P)
{
    delete P;
    P = nullptr;
}

// Prosedur untuk menampilkan semua elemen dalam list
void printInfo(List L)
{
    address P = L.First;
    while (P != nullptr)
    {
        cout << "no polisi : " << P->info.nopol << endl;
        cout << "warna     : " << P->info.warna << endl;
        cout << "tahun     : " << P->info.thnBuat << endl;
        cout << "--------------------------" << endl;
        P = P->next;
    }
}

// Prosedur untuk menambahkan elemen di akhir list
void insertLast(List &L, address P)
{
```

```
    if (L.First == nullptr)
    { // List kosong
        L.First = P;
        L.Last = P;
    }
    else
    {
        L.Last->next = P;
        P->prev = L.Last;
        L.Last = P;
    }
}

// Fungsi untuk mencari elemen berdasarkan nopol
address findElm(List L, string nopol)
{
    address P = L.First;
    while (P != nullptr)
    {
        if (P->info.nopol == nopol)
        {
            return P;
        }
        P = P->next;
    }
    return nullptr; // Tidak ditemukan
}

// Prosedur untuk menghapus elemen pertama
void deleteFirst(List &L, address &P)
{
    if (L.First != nullptr)
    {
        P = L.First;
        if (L.First == L.Last)
        { // Hanya satu elemen
            L.First = nullptr;
            L.Last = nullptr;
        }
        else
        {
            L.First = L.First->next;
            L.First->prev = nullptr;
```

```
    if (L.First != nullptr)
        else
            L.First->prev = nullptr;
        }
        P->next = nullptr;
        dealokasi(P);
    }
}

// Prosedur untuk menghapus elemen terakhir
void deleteLast(List &L, address &P)
{
    if (L.Last != nullptr)
    {
        P = L.Last;
        if (L.First == L.Last)
        { // Hanya satu elemen
            L.First = nullptr;
            L.Last = nullptr;
        }
        else
        {
            L.Last = L.Last->prev;
            L.Last->next = nullptr;
        }
        P->prev = nullptr;
        dealokasi(P);
    }
}

// Prosedur untuk menghapus elemen setelah Prec
void deleteAfter(address Prec, address &P)
{
    if (Prec != nullptr && Prec->next != nullptr)
    {
        P = Prec->next;
        Prec->next = P->next;
        if (P->next != nullptr)
        {
            P->next->prev = Prec;
        }
        else
        {
```

```
    if (Prec != nullptr && Prec->next != nullptr)
        else
            Prec->prev = nullptr;
        }
        P->next = nullptr;
        P->prev = nullptr;
        dealokasi(P);
    }
}

bool isNopolExist(List L, string nopol)
{
    return findElm(L, nopol) != nullptr;
}

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

    int choice;
    do
    {
        cout << "\nMENU:\n";
        cout << "1. Tambah data kendaraan\n";
        cout << "2. Cari data kendaraan berdasarkan nomor polisi\n
        cout << "3. Tampilkan seluruh data kendaraan\n";
        cout << "4. Hapus data kendaraan\n";
        cout << "5. Keluar\n";
        cout << "Masukkan pilihan: ";
        cin >> choice;

        if (choice == 1)
        {
            kendaraan k;
            cout << "Masukkan nomor polisi: ";
            cin >> k.nopol;
            if (isNopolExist(L, k.nopol))
            {
                cout << "Nomor polisi sudah terdaftar\n";
                continue;
            }
            cout << "Masukkan warna kendaraan: ";
```

```
            cin >> k.warna;
            cout << "Masukkan tahun kendaraan: ";
            cin >> k.thnBuat;

            insertLast(L, alokasi(k));
        }
        else if (choice == 2)
        {
            string nopol;
            cout << "Masukkan nomor polisi yang dicari: ";
            cin >> nopol;
            address found = findElm(L, nopol);
            if (found)
            {
                cout << "\nData ditemukan:\n";
                cout << "No Polisi : " << found->info.nopol << en
                cout << "Warna     : " << found->info.warna << en
                cout << "Tahun     : " << found->info.thnBuat <<
            }
            else
            {
                cout << "Data tidak ditemukan\n";
            }
        }
        else if (choice == 3)
        {
            cout << "\nDATA LIST:\n";
            printInfo(L);
        }

    } while (choice != 5);

    return 0;
}
```

Output

```
PS C:\Users\LENOVO\Documents\ITTP\TUGAS SEMESTER 5 ITTP\P
ikum STD\Week6\output> & .\'unguided.exe'

MENU:
1. Tambah data kendaraan
2. Cari data kendaraan berdasarkan nomor polisi
3. Tampilkan seluruh data kendaraan
4. Hapus data kendaraan
5. Keluar
Masukkan pilihan: 1
Masukkan nomor polisi: O113
Masukkan warna kendaraan: Merah
Masukkan tahun kendaraan: 2001

MENU:
1. Tambah data kendaraan
2. Cari data kendaraan berdasarkan nomor polisi
3. Tampilkan seluruh data kendaraan
4. Hapus data kendaraan
5. Keluar
Masukkan pilihan: 3

DATA LIST:
no polisi : O113
warna     : Merah
tahun     : 2001
-------------------------

MENU:
1. Tambah data kendaraan
2. Cari data kendaraan berdasarkan nomor polisi
3. Tampilkan seluruh data kendaraan
4. Hapus data kendaraan
5. Keluar
Masukkan pilihan:
```

```
MENU:
1. Tambah data kendaraan
2. Cari data kendaraan berdasarkan nomor polisi
3. Tampilkan seluruh data kendaraan
4. Hapus data kendaraan
5. Keluar
Masukkan pilihan: 2
Masukkan nomor polisi yang dicari: O113

Data ditemukan:
No Polisi : O113
Warna     : Merah
Tahun     : 2001
```

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.