LAPORAN PRAKTIKUM

Modul 6

Double Linked List (Bagian Pertama)

Disusun Oleh:

Yogi Hafidh Maulana - 2211104061

SE06-02

Dosen :

Wahyu Andi

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

PURWOKERTO

2024

1. **Tujuan**
   - Memahami konsep Double Linked List.
   - Mengimplementasikan operasi dasar pada Double Linked List.
   - Mengelola memori dinamis dalam Double Linked List.
   - Mengembangkan program berbasis Double Linked List.
   - Menerapkan konsep Double Linked List dalam pemrograman praktis.

2. **Landasan Teori**

   - Pengertian Double Linked List
     Double linked list adalah salah satu jenis struktur data yang terdiri dari serangkaian elemen (node), di mana setiap node memiliki dua pointer yang mengarah ke elemen sebelumnya (prev) dan elemen berikutnya (next). Struktur ini memungkinkan traversal (penelusuran) baik ke depan maupun ke belakang, sehingga memberikan fleksibilitas yang lebih besar dibandingkan dengan single linked list.
   - Komponen Double Linked List
     Setiap node dalam double linked list terdiri dari tiga komponen utama:
     a) Data: Menyimpan informasi yang diinginkan (misalnya, nomor polisi, warna, tahun).
     b) Pointer Prev: Merujuk ke node sebelumnya dalam daftar.
     c) Pointer Next: Merujuk ke node berikutnya dalam daftar.

     Di samping itu, double linked list juga memiliki dua pointer tambahan:
     a) First: Menunjuk pada node pertama dalam daftar.
     b) Last: Menunjuk pada node terakhir dalam daftar.

   - Operasi Dasar pada Double Linked List Operasi dasar yang dapat dilakukan pada double linked list meliputi:

     a) Insert First: Menambahkan node baru di depan list.
     b) Insert Last: Menambahkan node baru di akhir list.
     c) Delete First: Menghapus node pertama dari list.
     d) Delete Last: Menghapus node terakhir dari list.
     e) Insert After: Menambahkan node setelah node tertentu.
     f) Insert Before: Menambahkan node sebelum node tertentu.
     g) Delete After: Menghapus node setelah node tertentu.
     h) Delete Before: Menghapus node sebelum node tertentu

3. Guided

a) Guided 1

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;
    }
};
```

```cpp
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;
}
```

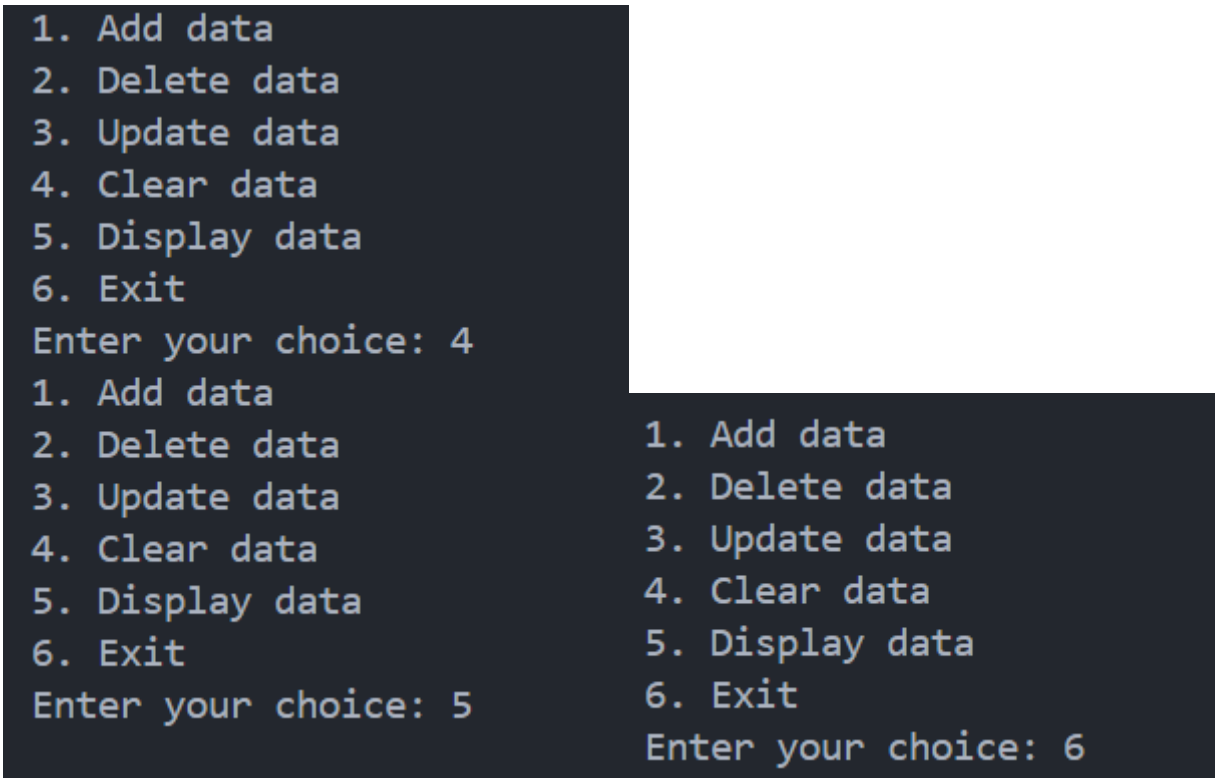
Output:

```
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: 1
Enter data to add: 30
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
```

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

```
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: 30
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 3
Enter old data: 30
Enter new data: 40
```

```
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: 50
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
50 40 20
```



Deskripsi Program:

Kode di atas adalah implementasi dari struktur data **Double Linked List** dalam bahasa C++. Dalam struktur ini, setiap elemen (node) memiliki tiga atribut: data, prev (penunjuk ke node sebelumnya), dan next (penunjuk ke node berikutnya). Kelas DoublyLinkedList mengelola node-node ini dengan menyediakan fungsi-fungsi untuk menambahkan elemen (insert), menghapus elemen dari depan (deleteNode), memperbarui data (update), menghapus semua elemen (deleteAll), dan menampilkan seluruh elemen dalam list (display). Program ini berfungsi dengan antarmuka pengguna berbasis teks yang memungkinkan pengguna untuk memilih berbagai operasi pada list, seperti menambah, menghapus, memperbarui, dan menampilkan data, serta mengosongkan list. Dengan memanfaatkan double linked list, program ini dapat dengan mudah melakukan navigasi di antara elemen-elemen dengan efisien.

4. Unguided

a) Buatlah ADT Double Linked list sebagai berikut di dalam file “doublelist.h”:

```
Type infotype : kendaraan <
    nopol : string
    warna : string
    thnBuat : integer
>
Type address : pointer to ElmList
Type ElmList <
    info : infotype
    next :address
    prev : address
>

Type List <
    First : address
    Last : address
>
prosedur CreateList( in/out L : List )
fungsi alokasi( x : infotype ) : address
prosedur dealokasi( in/out P : address )
prosedur printInfo( in L : List )
prosedur insertLast( in/out L : List, in P : address )
```

Buatlah implementasi ADT Double Linked list pada file “doublelist.cpp” dan coba hasil implementasi ADT pada file “main.cpp”. Output seperti gambar dibawah:

```
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
masukkan warna kendaraan: merah
masukkan tahun kendaraan: 80
nomor polisi sudah terdaftar

masukkan nomor polisi: D004
masukkan warna kendaraan: kuning
masukkan tahun kendaraan: 90

DATA LIST 1

no polisi : D004
warna     : kuning
tahun     : 90
no polisi : D003
warna     : putih
tahun     : 70
no polisi : D001
warna     : hitam
tahun     : 90
```

Code:

Doublelist.h

```h
#ifndef DOUBLELIST_H
#define DOUBLELIST_H

#include <iostream>
#include <string>

using namespace std;

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

typedef struct ElmList *address;

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

struct List
{
    address First;
    address Last;
};

void CreateList(List &L);
address alokasi(Kendaraan x);
void dealokasi(address &P);
void printInfo(List L);
void insertLast(List &L, Kendaraan P);

#endif // DOUBLELIST_H
```

Doublelist.cpp

```cpp
#include "doublelist.h"

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

// Mengalokasikan memori untuk elemen baru
address alokasi(Kendaraan x)
{
    address P = new ElmList;
    P->info = x;
    P->next = nullptr;
    P->prev = nullptr;
    return P;
}

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

// Mencetak informasi dari list
void printInfo(List L)
{
    address current = L.First;
    while (current != nullptr)
    {
        cout << "no polisi : " << current->info.nopol << endl;
        cout << "warna : " << current->info.warna << endl;
        cout << "tahun : " << current->info.thnBuat << endl;
        current = current->next;
    }
}

// Menyisipkan elemen baru di akhir list
void insertLast(List &L, Kendaraan P)
{
    address newNode = alokasi(P);
    if (L.First == nullptr)
    {
        L.First = newNode;
        L.Last = newNode;
    }
    else
    {
        L.Last->next = newNode;
        newNode->prev = L.Last;
        L.Last = newNode;
    }
}
```

Main.cpp

```cpp
#include "doublelist.h"

int main()
{
    List L;
    CreateList(L);
    Kendaraan kendaraan;
    string nopol;

    while (true)
    {
        cout << "masukkan nomor polisi: ";
        cin >> kendaraan.nopol;

        cout << "masukkan warna kendaraan: ";
        cin >> kendaraan.warna;

        cout << "masukkan tahun kendaraan: ";
        cin >> kendaraan.thnBuat;

        // Cek jika nomor polisi sudah ada
        address current = L.First;
        bool exists = false;
        while (current != nullptr)
        {
            if (current->info.nopol == kendaraan.nopol)
            {
                exists = true;
                break;
            }
            current = current->next;
        }

        if (exists)
        {
            cout << "nomor polisi sudah terdaftar" << endl;
        }
        else
        {
            insertLast(L, kendaraan);
        }

        // Menampilkan informasi dari list
        cout << "DATA LIST 1" << endl;
        printInfo(L);
        cout << endl;
    }

    return 0;
}
```

Output:

```
D:\PROJECT\C++ Project\Pertemuan7>kendaraan
masukkan nomor polisi: D001
masukkan warna kendaraan: putih
masukkan tahun kendaraan: 20
DATA LIST 1
no polisi : D001
warna : putih
tahun : 20

masukkan nomor polisi: R001
masukkan warna kendaraan: pink
masukkan tahun kendaraan: 50
DATA LIST 1
no polisi : D001
warna : putih
tahun : 20
no polisi : R001
warna : pink
tahun : 50

masukkan nomor polisi: h4004
masukkan warna kendaraan: hitam
masukkan tahun kendaraan: 80
DATA LIST 1
no polisi : D001
warna : putih
tahun : 20
no polisi : R001
warna : pink
tahun : 50
no polisi : h4004
warna : hitam
tahun : 80
```

Deskripsi code

- doublelist.h : Mendefinisikan struktur data untuk Kendaraan, elemen list (ElmList), dan list itu sendiri (List). Juga mendeklarasikan fungsi-fungsi yang akan digunakan.
- doublelist.cpp : Mengimplementasikan fungsi-fungsi untuk mengelola list, termasuk membuat list, mengalokasikan memori untuk elemen baru, menghapus elemen, mencetak informasi, dan menyisipkan elemen di akhir list.
- main.cpp : Menggunakan fungsi yang didefinisikan untuk menerima input dari pengguna mengenai kendaraan dan memasukkan data ke dalam list, sambil memeriksa apakah nomor polisi sudah terdaftar sebelumnya.

b) Carilah elemen dengan nomor polisi D001 dengan membuat fungsi baru. fungsi findElm(L : List, x : infotype) : address

```
Masukkan Nomor Polisi yang dicari   : D001

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

Code:

Penambahan code pada doublelist.h

```
address findElm(List L, string x);
```

Fungsi find elemen

```
address findElm(List L, string x) {
    address current = L.First;
    while (current != nullptr) {
        if (current->info.nopol == x) {
            return current; // Mengembalikan alamat node yang ditemukan
        }
        current = current->next;
    }
    return nullptr; // Mengembalikan nullptr jika tidak ditemukan
}
```

Output:

```
Masukkan nomor polisi: H5000
Masukkan warna kendaraan: black
Masukkan tahun kendaraan: 80
DATA LIST 1
no polisi : H5000
warna : black
tahun : 80

Masukkan Nomor Polisi yang dicari : H5000
Nomor Polisi : H5000
Warna : black
Tahun : 80
Masukkan nomor polisi:
```

Deskripsi Code

- **findElm**. Fungsi ini akan mengiterasi seluruh list dan mencari node dengan nomor polisi yang sesuai. Jika ditemukan, fungsi akan mengembalikan alamat node tersebut; jika tidak, akan mengembalikan nullptr.
- **Modifikasi di main.cpp**: Di dalam main, program akan meminta nomor polisi yang dicari, dan jika ditemukan, informasi kendaraan akan ditampilkan sesuai format yang Anda inginkan. Jika tidak ditemukan, program akan memberi tahu bahwa nomor polisi tidak ada.

c) Hapus elemen dengan nomor polisi D003 dengan prosedur delete.
- prosedur deleteFirst(in/out L : List, in/out P : address)
- prosedur deleteLast(in/out L : List, in/out P : address)
- prosedur deleteAfter(in Prec : address, in/out: P : address)

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

DATA LIST 1

Nomor Polisi    : D004
Warna           : kuning
Tahun           : 90
Nomor Polisi    : D001
Warna           : hitam
Tahun           : 90
```

Code
Doublelist.h

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

Doublelist.cpp

```cpp
void deleteFirst(List &L, address &P)
{
    if (L.First == nullptr)
    {
        cout << "List kosong, tidak ada yang dapat dihapus." << endl;
        return;
    }
    P = L.First;
    if (L.First == L.Last)
    { // Jika hanya ada satu elemen
        L.First = nullptr;
        L.Last = nullptr;
    }
    else
    {
        L.First = L.First->next;
        L.First->prev = nullptr;
    }
    delete P;
}

void deleteLast(List &L, address &P)
{
    if (L.First == nullptr)
    {
        cout << "List kosong, tidak ada yang dapat dihapus." << endl;
        return;
    }
    P = L.Last;
    if (L.First == L.Last)
    { // Jika hanya ada satu elemen
        L.First = nullptr;
        L.Last = nullptr;
    }
    else
    {
        L.Last = L.Last->prev;
        L.Last->next = nullptr;
    }
    delete P;
}

void deleteAfter(List &L, address Prec, address &P)
{
    if (Prec == nullptr || Prec->next == nullptr)
    {
        cout << "Tidak ada elemen setelah elemen yang diberikan." << endl;
        return;
    }
    P = Prec->next;
    Prec->next = P->next;
    if (P->next != nullptr)
    {
        P->next->prev = Prec;
    }
    else
    {
        // Jika P adalah elemen terakhir
        L.Last = Prec;
    }
    delete P;
}
```

Output

```
===== Menu =====
1. Tambah Kendaraan
2. Hapus Kendaraan
3. Cari Kendaraan
4. Tampilkan Semua Kendaraan
5. Keluar
Masukkan pilihan Anda: 2
Masukkan Nomor Polisi yang akan dihapus: R3000
Data dengan nomor polisi R3000 berhasil dihapus.

===== Menu =====
1. Tambah Kendaraan
2. Hapus Kendaraan
3. Cari Kendaraan
4. Tampilkan Semua Kendaraan
5. Keluar
Masukkan pilihan Anda: 4
DATA LIST 1
```

Deskripsi Program

- **Prosedur deleteFirst**: Menghapus elemen pertama dari list dan memperbarui pointer First dan Last jika diperlukan.
- **Prosedur deleteLast**: Menghapus elemen terakhir dari list dan memperbarui pointer Last.
- **Prosedur deleteAfter**: Menghapus elemen setelah node yang diberikan (Prec) dan memperbarui pointer yang relevan.
- **Modifikasi di main.cpp**: Dalam loop utama, program akan meminta nomor polisi yang akan dihapus. Jika ditemukan, elemen tersebut akan dihapus menggunakan prosedur yang sesuai, dan pesan akan ditampilkan. Setelah itu, daftar akan ditampilkan.

Setelah melakukan semua modifikasi tersebut, ikuti langkah-langkah kompilasi dan eksekusi program seperti sebelumnya. Program sekarang akan memiliki kemampuan untuk mencari dan menghapus kendaraan berdasarkan nomor polisi yang dimasukkan, serta menampilkan informasi daftar setelah setiap penghapusan.

5. Kesimpulan

Double linked list adalah struktur data yang memungkinkan penyimpanan dan pengelolaan elemen dengan lebih efisien, berkat kemampuannya untuk menavigasi baik ke depan maupun ke belakang melalui penggunaan dua pointer, yaitu prev dan next. Struktur ini mendukung berbagai operasi dasar seperti penambahan, penghapusan, dan pencarian elemen dengan fleksibilitas yang lebih tinggi dibandingkan dengan single linked list, meskipun memerlukan lebih banyak memori. Dengan pemahaman dan penerapan konsep double linked list, pengembang dapat menciptakan aplikasi yang lebih kompleks dan responsif, meningkatkan efisiensi dalam pengelolaan data dalam berbagai konteks pemrograman. Kesimpulannya, pemahaman yang mendalam tentang double linked list memberikan dasar yang kuat bagi siswa untuk mengembangkan keterampilan dalam struktur data dan algoritma yang lebih lanjut.