LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VI
DOUBLY LINKED LIST

Disusun Oleh :
NAMA : HILMI HAKIM RAMADANI
NIM : 103112430016

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

C++ adalah bahasa pemrograman yang dikembangkan oleh Bjarne Stroustrup di Bell Labs pada awal 1980-an sebagai pengembangan dari bahasa C. Awalnya dirancang untuk sistem Unix, C++ kemudian menjadi bahasa pemrograman tujuan umum (general-purpose programming language).

Linked list adalah suatu bentuk struktur data yang berupa sekumpulan elemen data yang bertipe sama dimana tiap elemen saling berkait atau dihubungkan dengan elemen lain melalui suatu pointer. Pointeritu sendiri adalah alamat elemen data yang tersimpan di memori. Penggunaan pointer untuk mengacu elemen berakibat elemen-elemen bersebelahan secara logik walau tidak bersebelahan secara fisik di memori.

Linked list terdiri dari node-node (simpul-simpul) yang saling terhubung(linked).Simpul berupa struct, sedangkan link berupa komponen yang bertipe pointerke simpul. Ada dua jenis pointeryang digunakan, yaitu head (menunjukkan alamat pointer paling depan) dan tail (menunjukkan simpul terakhir). Operasi penambahan atau penghapusan sebuah simpul akan mengubah nilai pointer linknya.Sedangkan pointer link disimpul terkahir diberi nilai null.

Doubly Linked List merupakan linked list dengan menggunakan pointer, dimana setiap node memiliki tiga buah field, yaitu: field pointer yang menunjuk ke pointer berikutnya, field pointer yang menunjuk ke pointer sebelumnya dan field yang berisi data dari node tersebut. Semenatara pointer next dan prev-nya menunjuk ke null.

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

Guided

#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)
```

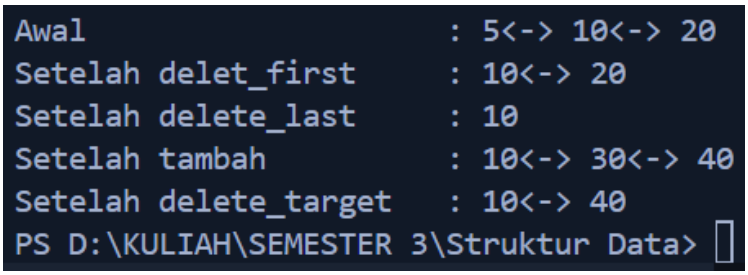
```
{
    Node *newNode = new Node{value, NULL, ptr_first};

    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;
    }
}

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_mode(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 delet_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



Deskripsi:

Program ini merupakan implementasi Doubly Linked List dalam bahasa C++ yang menggunakan dua pointer global, yaitu ptr_first dan ptr_last, untuk menunjuk node pertama dan terakhir. Setiap node memiliki tiga bagian, yaitu data, pointer ke node sebelumnya (prev), dan pointer ke node berikutnya (next). Program menyediakan operasi penambahan data di awal (add_first), di akhir (add_last), dan setelah node tertentu (add_target), serta operasi penghapusan di awal (delete_first), di akhir (delete_last), dan berdasarkan nilai tertentu (delete_target). Selain itu, terdapat fungsi edit_mode untuk mengubah data pada node tertentu dan view untuk menampilkan seluruh isi list. Pada fungsi main, seluruh operasi tersebut diuji untuk menunjukkan perubahan isi list setelah setiap proses penambahan dan penghapusan dilakukan.

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

Soal1

#main.cpp

```
#include <iostream>
#include "Doublylist.h"
#include "Doublylist.cpp"
using namespace std;

int main()
{
```

```
    List L;
    CreateList(L);

    kendaraan x;
    string cek;
    int n;

    for (int i = 0; i < 3; i++)
    {
        cout << "masukkan nomor polisi: ";
        cin >> x.nopol;

        // cek duplikat
        bool ada = false;
        address P = L.First;
        while (P != Nil)
        {
            if (P->info.nopol == x.nopol)
            {
                ada = true;
                break;
            }
            P = P->next;
        }

        if (ada)
        {
            cout << "nomor polisi sudah terdaftar\n\n";
            i--;
            continue;
        }

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

        insertLast(L, alokasi(x));
    }

    cout << "DATA LIST 1\n\n";
    printInfo(L);

    return 0;
}
```

#Doublylist.cpp

```
#include "Doublylist.h"
```

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

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

void dealokasi(address &P)
{
   delete P;
   P = Nil;
}

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

void printInfo(List L)
{
   address P = L.First;
   while (P != Nil)
   {
      cout << "no polisi : " << P->info.nopol << endl;
      cout << "warna     : " << P->info.warna << endl;
      cout << "tahun     : " << P->info.thnBuat << endl;
      cout << endl;
      P = P->next;
   }
}
```

#Doublylist.h

```
#ifndef DOUBLYLIST_H
#define DOUBLYLIST_H

#include <iostream>
#include <string>
using namespace std;

#define Nil NULL

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

typedef struct ElmList *address;

// elemen list
struct ElmList
{
   kendaraan info;
   address next;
   address prev;
};

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

// prototype ADT
void CreateList(List &L);
address alokasi(kendaraan x);
void dealokasi(address &P);
void insertLast(List &L, address P);
void printInfo(List L);

#endif
```

Screenshots Output

```
masukkan nomor polisi: d001
masukkan warna kendaraan: htm
masukkan tahun kendaraan: 707

masukkan nomor polisi: d002
masukkan warna kendaraan: pth
masukkan tahun kendaraan: 80

masukkan nomor polisi: d002
nomor polisi sudah terdaftar

masukkan nomor polisi: d003
masukkan warna kendaraan: kng
masukkan tahun kendaraan: 90

DATA LIST 1
no polisi : d001
warna     : htm
tahun     : 707

no polisi : d002
warna     : pth
tahun     : 80

no polisi : d003
warna     : kng
tahun     : 90

PS D:\KULIAH\SEMESTER 3\Struktur Data>
```

Soal 2

- Menambah code di Doublylist.h

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

- Menambah code di Doublylist.cpp

```
address findElm(List L, string nopol) {
    address P = L.First;
    while (P != Nil) {
        if (P->info.nopol == nopol) {
            return P;
        }
        P = P->next;
    }
    return Nil;
}
```

- Menambah code di main.cpp

```
string cariNopol;
cout << "\nMasukkan nomor polisi yang ingin dicari: ";
cin >> cariNopol;

address hasil = findElm(L, cariNopol);

if (hasil != Nil) {
```

```
        cout << "\nData ditemukan\n";
        cout << "no polisi : " << hasil->info.nopol << endl;
        cout << "warna     : " << hasil->info.warna << endl;
        cout << "tahun     : " << hasil->info.thnBuat << endl;
    } else {
        cout << "\nData tidak ditemukan\n";
}
```

- Output

```
DATA LIST 1
no polisi : d001
warna     : kk
tahun     : 23

no polisi : d002
warna     : htm
tahun     : 9

no polisi : d003
warna     : htm
tahun     : 54


Masukkan nomor polisi yang dicari: d001

Data ditemukan
no polisi : d001
warna     : kk
tahun     : 23
PS D:\KULIAH\SEMESTER 3\Struktur Data>
```

Soal 3

- Tambahkan code di Doublylist.h

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

void deleteLast(List &L, address &P);

void deleteAfter(address Prec, address &P);
```

- Tambahkan code di Doublylist.cpp

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

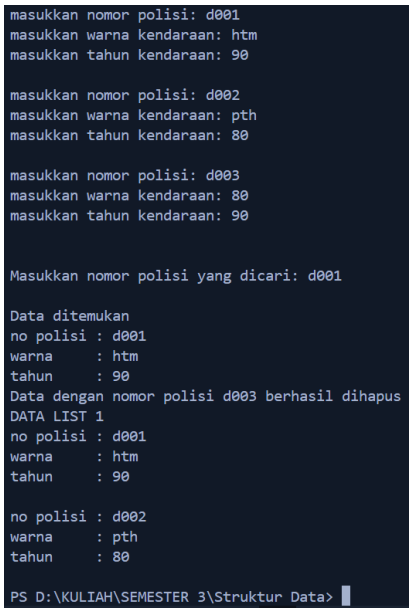
```
            }
            P->next = Nil;
            P->prev = Nil;
        }
    }

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

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

            if (P->next != Nil)
            {
                P->next->prev = Prec;
            }

            P->next = Nil;
            P->prev = Nil;
        }
}
```

- Tambahkan code di main.cpp

```
    address P = findElm(L, "d003");
```

```
    if (P != Nil)
    {
        address temp;

        if (P == L.First)
        {
            deleteFirst(L, temp);
        }
        else if (P == L.Last)
        {
            deleteLast(L, temp);
        }
        else
        {
            deleteAfter(P->prev, temp);
        }

        dealokasi(temp);
        cout << "Data dengan nomor polisi d003 berhasil dihapus\n";
    }
    else
    {
        cout << "Data d003 tidak ditemukan\n";
}
```

- Output



Deskripsi:

Program ini menerapkan ADT Doubly Linked List untuk menyimpan data kendaraan yang terdiri dari nomor polisi, warna, dan tahun pembuatan. Struktur list memiliki pointer First dan Last sehingga penambahan data di akhir dapat dilakukan secara efisien. Program utama menerima input data kendaraan, melakukan pengecekan agar nomor polisi tidak duplikat, kemudian menyimpan data ke dalam list dan menampilkannya sesuai format output yang diminta.

D. Kesimpulan

Dari praktikum ini dapat disimpulkan bahwa struktur data ini memungkinkan pengelolaan data secara lebih fleksibel karena setiap node memiliki pointer ke node sebelumnya dan berikutnya. Melalui kegiatan guided dan unguided, mahasiswa mampu mengimplementasikan ADT Doubly Linked List secara modular menggunakan bahasa C++, mulai dari pembuatan list, alokasi dan dealokasi node, penambahan data, pencarian data berdasarkan nomor polisi, hingga penghapusan node pada posisi awal, akhir, maupun tengah list. Penerapan studi kasus data kendaraan membantu memahami konsep kerja pointer prev dan next secara nyata, serta menunjukkan bahwa Doubly Linked List efektif digunakan untuk pengolahan data dinamis yang membutuhkan navigasi dua arah dan manipulasi elemen secara efisien.

E. Referensi

Dewi, L. J. E. (2010). Media Pembelajaran Bahasa Pemrograman C++. Jurnal Pendidikan Teknologi dan Kejuruan, 7(1).

Wikipedia. 2025, 29 September. C++. Diakses pada 1 Oktober. Dari https://en.wikipedia.org/wiki/C%2B%2B

Sihombing, J. (2019). Penerapan stack dan queue pada array dan linked list dalam java. INFOKOM (Informatika & Komputer), 7(2), 15-24.