LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VI
PENGENALAN CODE BLOCKS

Disusun Oleh :

NAMA : Rikza Nur Zaki

NIM : 103112430030

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Doubly Linked List adalah struktur data di mana setiap elemen memiliki dua penunjuk yaitu prev elemen dan next elemen, dikendalikan oleh penunjuk First dan Last, sehingga dapat mengakses elemen yang lebih mudah dengan iterasi maju dan mundur, dan list dianggap kosong jika penunjuk First bernilai Nil.

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

Guided 1

```
#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;
    }
    delete temp;
}

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 delete_first\t: ";
    view();
    delete_last();
    cout << "Setelah delete_last\t: ";
    view();

    add_last(30);
    add_last(40);
    cout << "Setelah tambah\t\t\t: ";
    view();

    delete_target(30);
    cout << "Setelah delete_target\t: ";
    view();

}
```

Screenshots Output

```
Awal                    : 5  <-> 10  <-> 20
Setelah delete_first    : 10  <-> 20
Setelah delete_last     : 10
Setelah tambah          : 10  <-> 30  <-> 40
Setelah delete_target   : 10  <-> 40
```

Deskripsi:

Kode diatas ini mengimplementasikan struktur data Doubly Linked List dengan node yang menyimpan nilai integer, kemudian mode penambahan (add_first, add_last, add_target), penghapusan (delete_first, delete_last, delete_target), pengeditan (edit_mode), dan penampilan (view) list, yang kemudian didemonstrasikan dalam fungsi main yang hasilnya terdapat di ss output.

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

Unguided 1

Doublylist.h

```h
#ifndef DOUBLYLIST_H
#define DOUBLYLIST_H
#include <iostream>
#include <string>
using namespace std;

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

struct ElmList;
typedef ElmList* address;

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

struct List {
    address first;
    address last;
};

void createList(List &L);
address alokasi(infotype x);
void dealokasi(address &P);
void insertLast(List &L, address P);
void printInfo(List L);
address findElm(List L, string nopol);
void deleteFirst(List &L, address &P);
void deleteLast(List &L, address &P);
void deleteAfter(address Prec, address &P);
#endif
```

Doubly.cpp

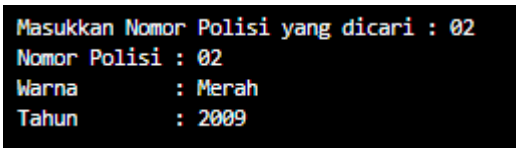
```cpp
#include "Doublylist.h"

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 insertLast(List &L, address P) {
    if (L.first == NULL) {
        L.first = P;
        L.last = P;
    } else {
        L.last->next = P;
        P->prev = L.last;
        L.last = P;
    }
}

void printInfo(List L) {
    address P = L.first;
    cout << "DATA LIST 1\n";
    while (P != NULL) {
        cout << "No Polisi : " << P->info.nopol << endl;
        cout << "Warna     : " << P->info.warna << endl;
        cout << "Tahun     : " << P->info.thnBuat << endl;
        cout << "------------------" << endl;
        P = P->next;
    }
}

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

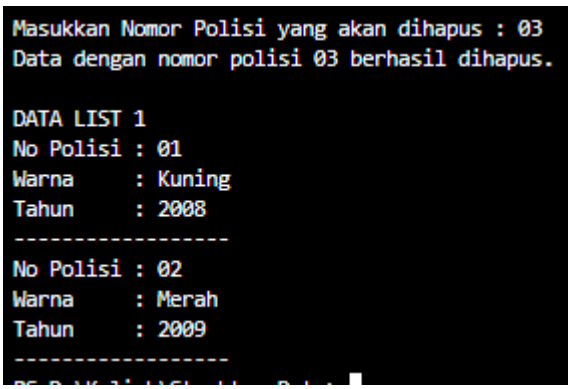
```cpp
void deleteFirst(List &L, address &P) {
    if (L.first != NULL) {
        P = L.first;
        if (L.first == L.last) {
            L.first = NULL;
            L.last = NULL;
        } else {
            L.first = L.first->next;
            L.first->prev = NULL;
        }
        P->next = NULL;
    }
}

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 = L.last->prev;
            L.last->next = NULL;
        }
        P->prev = NULL;
    }
}

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

main.cpp

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


int main() {
    List L;
    createList(L);
    infotype x;
    address P;

    for (int i = 0; i < 3; i++) {
        cout << "Masukkan nomor polisi: ";
        cin >> x.nopol;
        cout << "Masukkan warna kendaraan: ";
        cin >> x.warna;
        cout << "Masukkan tahun kendaraan: ";
        cin >> x.thnBuat;
        P = alokasi(x);
        insertLast(L, P);
        cout << endl;
    }

    cout << endl;
    printInfo(L);

    string cari;
    cout << "\nMasukkan Nomor Polisi yang dicari : ";
    cin >> cari;
    P = findElm(L, cari);
    if (P != NULL) {
        cout << "Nomor Polisi : " << P->info.nopol << endl;
        cout << "Warna        : " << P->info.warna << endl;
        cout << "Tahun        : " << P->info.thnBuat << endl;
    } else {
        cout << "Data tidak ditemukan!\n";
    }

    string hapus;
    cout << "\nMasukkan Nomor Polisi yang akan dihapus : ";
    cin >> hapus;
    P = findElm(L, hapus);
    if (P != NULL) {
        if (P == L.first) {
            deleteFirst(L, P);
        } else if (P == L.last) {
            deleteLast(L, P);
        } else {
            deleteAfter(P->prev, P);
        }
        dealokasi(P);
        cout << "Data dengan nomor polisi " << hapus << " berhasil dihapus.\n";
    } else {
        cout << "Data tidak ditemukan!\n";
    }

    cout << endl;
    printInfo(L);
    return 0;
}
```

Screenshots Output





Deskripsi:

Kode diatas menggunakan Double Linked List dengan memberikan informasi data kendaraan (nopol, warna, thnBuat), kemudian memberikan fungsi primitif seperti createList, insertLast, findElm, deleteFirst, deleteLast, deleteAfter. Lalu program dijalankan yang dimulai dari memasukan nomor polisi, warna kendaraan, tahun kendaraan, pada bagian findElm digunakan untuk mencari no polisi, dan pada bagian delete elm untuk menghapus no polisi.

E. Kesimpulan

Laprak ini melakukan uji coba terhadap Double Linked List, dengan mengimplementasikan konsep ADT dengan adanya alokasi, penyisipan, pencarian dan penghapusan.

F. Referensi

Wirth, N. (2013). *Algorithms and Data Structures*. Springer.

Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2014). *Data Structures and Algorithms in Java* (6th ed.). Wiley.