LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VI
DOUBLY LINKED LIST
(BAGIAN PERTAMA)

Disusun Oleh :
NAMA : KHOIRUL ADDIFA
NIM : 103112400172

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Doubly Linked List adalah jenis linked list yang lebih fleksibel, di mana setiap elemen (node) tidak hanya menyimpan data (infotype) dan alamat ke elemen berikutnya (next atau successor), tetapi juga menyimpan alamat ke elemen sebelumnya (prev atau predecessor). Keberadaan dua pointer ini (next dan prev) memungkinkan navigasi (traversal) list dapat dilakukan secara dua arah (maju dan mundur), membuatnya sangat efisien untuk operasi penyisipan (insert) dan penghapusan (delete) di posisi mana pun dalam list karena kita dapat dengan mudah mengakses elemen sebelum elemen target.

Guided

Doublylist.h

```h
#ifndef DOUBLYLIST_H
#define DOUBLYLIST_H

#include <iostream>
#include <string>

using namespace std;

// Definisi struktur kendaraan
struct kendaraan {
    string nopol;
    string warna;
    int thnBuat;
};

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

// Typedef untuk pointer
typedef ElmList* address;

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

// --- PROTOTIPE FUNGSI ---
void CreateList(List& L);
address alokasi(kendaraan x);
void dealokasi(address& P);
```

```
void printInfo(List L);

void insertFirst(List& L, address P); // Digunakan untuk Soal 1
bool isNopolExist(List L, string nopol); // Digunakan untuk cek duplikasi

address findElm(List L, kendaraan x); // Soal No. 2
void deleteFirst(List& L);
void deleteLast(List& L);
void deleteAfter(List& L, address prec);
void deleteElmByNopol(List& L, string nopolHapus); // Soal No. 3

#endif
```

Doublylist.cpp

```
#include "Doublylist.h"

// 1. CreateList
void CreateList(List& L) {
    L.First = NULL;
    L.Last = NULL;
}

// 2. alokasi
address alokasi(kendaraan x) {
    address P = new ElmList;
    if (P == NULL) {
        // Penanganan jika alokasi memori gagal
        cout << "Gagal alokasi memori!" << endl;
    } else {
        P->info = x;
        P->next = NULL;
        P->prev = NULL;
    }
    return P;
}

// 5. insertFirst
void insertFirst(List& L, address P) {
    if (P == NULL) return; // Pastikan P bukan NULL

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

```
        L.First->prev = P;
        L.First = P;
    }
}

// 7. deleteFirst (Lebih aman saat list hanya berisi 1 elemen)
void deleteFirst(List& L) {
    if (L.First == NULL) return; // List Kosong

    address P = L.First;
    if (L.First == L.Last) {
        // Hanya 1 elemen
        L.First = NULL;
        L.Last = NULL;
    } else {
        // Lebih dari 1 elemen
        L.First = P->next;
        L.First->prev = NULL;
    }
    dealokasi(P);
}

// 9. deleteAfter (Perbaikan penanganan jika yang dihapus adalah Last)
void deleteAfter(List& L, address prec) {
    if (prec == NULL || prec->next == NULL) return; // Tidak ada elemen
yang dihapus

    address P = prec->next; // Elemen yang akan dihapus

    if (P == L.Last) {
        // Jika yang dihapus elemen terakhir
        L.Last = prec;
        L.Last->next = NULL;
    } else {
        // Jika elemen di tengah
        prec->next = P->next;
        P->next->prev = prec;
    }
    dealokasi(P);
}

// 10. deleteElmByNopol (Menggunakan logika pointer yang sudah
diperbaiki)
void deleteElmByNopol(List& L, string nopolHapus) {
    address P = L.First;
    // Cari elemen
    while (P != NULL && P->info.nopol != nopolHapus) {
        P = P->next;
```

```
    }

    if (P == NULL) {
        cout << "Nomor Polisi " << nopolHapus << " tidak ditemukan." <<
endl;
        return;
    }

    if (P == L.First) {
        deleteFirst(L);
    } else if (P == L.Last) {
        deleteLast(L);
    } else {
        // Hapus elemen di tengah. Lebih aman jika manipulasi prev dan
next dilakukan di sini
        P->prev->next = P->next;
        P->next->prev = P->prev;
        dealokasi(P);
    }
    cout << "Data dengan nomor polisi **" << nopolHapus << "** berhasil
dihapus." << endl;
}
// 3. dealokasi
void dealokasi(address& P) {
    if (P != NULL) {
        delete P;
        P = NULL;
    }
}

// 4. printInfo
void printInfo(List L) {
    address P = L.First;
    int no = 1;

    if (P == NULL) {
        cout << "List kosong" << endl;
        return;
    }

    cout << "=============================================" << endl;
    cout << "          DATA KENDARAAN TEREGISTRASI" << endl;
    cout << "=============================================" << endl;

    while (P != NULL) {
        cout << "No " << no << " : " << endl;
        cout << "  Nomor Polisi : " << P->info.nopol << endl;
        cout << "  Warna        : " << P->info.warna << endl;
```

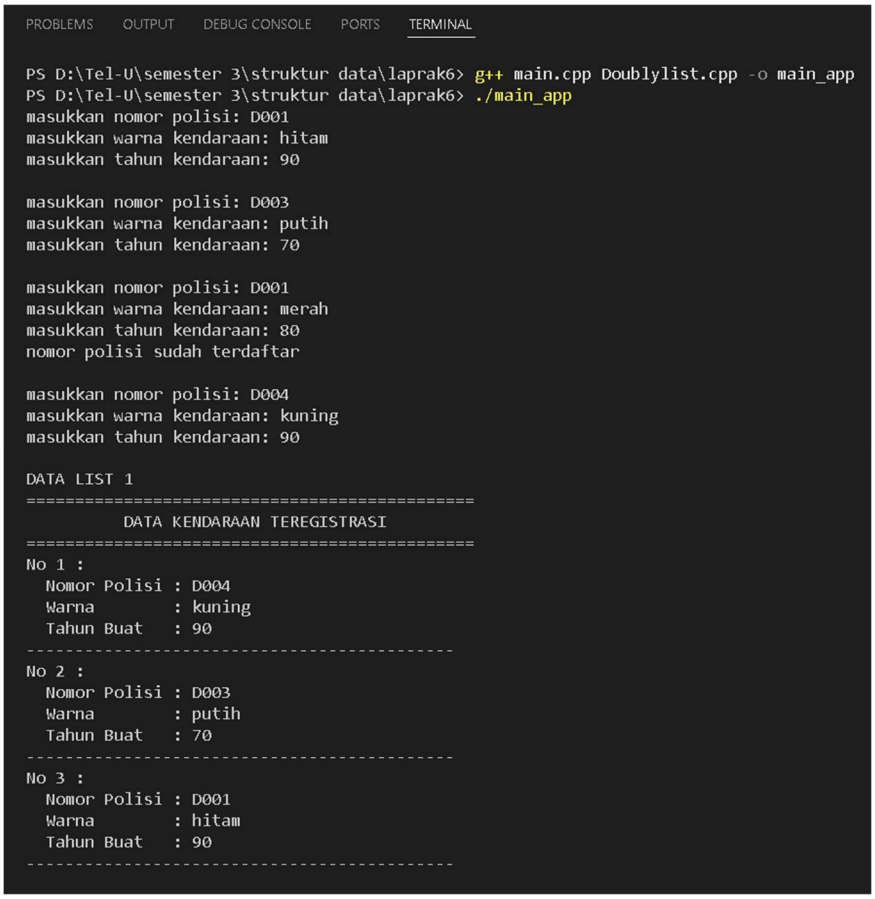
```
        cout << "  Tahun Buat    : " << P->info.thnBuat << endl;
        cout << "----------------------------------------" << endl;
        P = P->next;
        no++;
    }
}

// 6. isNopolExist
bool isNopolExist(List L, string nopol) {
    address P = L.First;
    while (P != NULL) {
        if (P->info.nopol == nopol) {
            return true;
        }
        P = P->next;
    }
    return false;
}

// 8. deleteLast
void deleteLast(List& L) {
    if (L.First == NULL) return; // List kosong

    address P = L.Last;
    if (L.First == L.Last) {
        // Hanya 1 elemen
        L.First = NULL;
        L.Last = NULL;
    } else {
        // Lebih dari 1 elemen
        L.Last = P->prev;
        L.Last->next = NULL;
    }
    dealokasi(P);
}

// 11. findElm
address findElm(List L, kendaraan x) {
    address P = L.First;
    while (P != NULL) {
        if (P->info.nopol == x.nopol) {
            return P;
        }
        P = P->next;
    }
    return NULL;
}
```

main.cpp

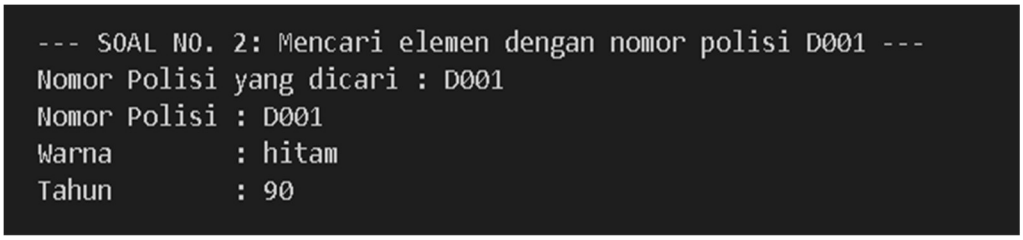
```cpp
#include "Doublylist.h"
#include <limits>

// ... (Fungsi inputKendaraan tetap sama)

// Prosedur khusus untuk menyisipkan dengan pengecekan duplikasi
void insertWithCheck(List& L, kendaraan x) {
    cout << "masukkan nomor polisi: " << x.nopol << endl;
    cout << "masukkan warna kendaraan: " << x.warna << endl;
    cout << "masukkan tahun kendaraan: " << x.thnBuat << endl;

    if (isNopolExist(L, x.nopol)) {
        cout << "nomor polisi sudah terdaftar" << endl;
    } else {
        address P = alokasi(x);
        if (P != NULL) {
            insertFirst(L, P);
        }
    }
    cout << endl;
}

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

    // ========================================================
    // BAGIAN 1: KASUS INSERT DENGAN DUPLIKASI (Sesuai Gambar)
    // ========================================================
    // 1. Masukkan D001 (Diterima)
    insertWithCheck(L, {"D001", "hitam", 90});

    // 2. Masukkan D003 (Diterima)
    insertWithCheck(L, {"D003", "putih", 70});

    // 3. Masukkan D001 lagi (Ditolak)
    insertWithCheck(L, {"D001", "merah", 80});

    // 4. Masukkan D004 (Diterima)
    insertWithCheck(L, {"D004", "kuning", 90});

    cout << "DATA LIST 1" << endl;
    printInfo(L);

    // ========================================================
    // BAGIAN 2: PENCARIAN ELEMEN (findElm 'D001')
```

```
    // ========================================================
    cout << "\n--- SOAL NO. 2: Mencari elemen dengan nomor polisi D001 ---" << endl;
    kendaraan cari = {"D001", "", 0};
    address hasilCari = findElm(L, cari);

    cout << "Nomor Polisi yang dicari : " << cari.nopol << endl;
    if (hasilCari != NULL) {
        cout << "Nomor Polisi : " << hasilCari->info.nopol << endl;
        cout << "Warna        : " << hasilCari->info.warna << endl;
        cout << "Tahun        : " << hasilCari->info.thnBuat << endl;
    } else {
        cout << "Nomor Polisi " << cari.nopol << " tidak ditemukan." << endl;
    }

    // ========================================================
    // BAGIAN 3: PENGHAPUSAN ELEMEN (delete 'D003')
    // ========================================================
    cout << "\n--- SOAL NO. 3: Hapus elemen dengan nomor polisi D003 ---" << endl;
    string nopolHapus = "D003";

    cout << "MASUKKAN Nomor Polisi yang akan dihapus : " << nopolHapus << endl;
    deleteElmByNopol(L, nopolHapus);

    cout << "\nDATA LIST 1 (Setelah Penghapusan):" << endl;
    printInfo(L);

    return 0;
}
```

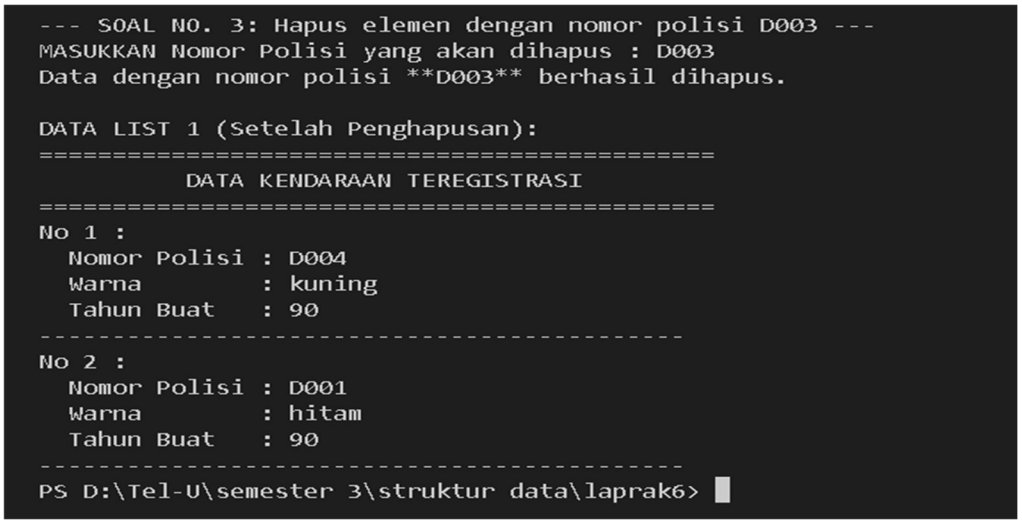
Screenshots Output



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL

PS D:\Tel-U\semester 3\struktur data\laprak6> g++ main.cpp Doublylist.cpp -o main_app
PS D:\Tel-U\semester 3\struktur data\laprak6> ./main_app
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
=========================================
         DATA KENDARAAN TEREGISTRASI
=========================================
No 1 :
  Nomor Polisi : D004
  Warna        : kuning
  Tahun Buat   : 90
-----------------------------------------
No 2 :
  Nomor Polisi : D003
  Warna        : putih
  Tahun Buat   : 70
-----------------------------------------
No 3 :
  Nomor Polisi : D001
  Warna        : hitam
  Tahun Buat   : 90
-----------------------------------------
```



```
--- SOAL NO. 2: Mencari elemen dengan nomor polisi D001 ---
Nomor Polisi yang dicari : D001
Nomor Polisi : D001
Warna        : hitam
Tahun        : 90
```



```
--- SOAL NO. 3: Hapus elemen dengan nomor polisi D003 ---
MASUKKAN Nomor Polisi yang akan dihapus : D003
Data dengan nomor polisi **D003** berhasil dihapus.

DATA LIST 1 (Setelah Penghapusan):
=============================================
          DATA KENDARAAN TEREGISTRASI
=============================================
No 1 :
  Nomor Polisi : D004
  Warna        : kuning
  Tahun Buat   : 90
---------------------------------------------
No 2 :
  Nomor Polisi : D001
  Warna        : hitam
  Tahun Buat   : 90
---------------------------------------------
PS D:\Tel-U\semester 3\struktur data\laprak6>
```

Deskripsi:

Tujuan program ini adalah untuk mengimplementasikan Daftar Tertaut Ganda (Doubly Linked List) , sebuah struktur data dinamis yang memungkinkan navigasi dua arah (maju dan mundur) . Fokusnya adalah menganalisis data kendaraan , serta menggunakan operasi dasar ADT seperti insertFirst (mencari data) , findElm (mencari data) , dan deleteElmByNopol (mencari data) dengan kasus Nomor Polisi ganda . Kesimpulan

Dalam praktikum Modul 5 ini, disimpulkan bahwa Doubly Linked List adalah struktur data fleksibel yang unggul karena setiap elemen memiliki dua pointer: next (ke elemen sesudah) dan prev (ke elemen sebelum). Kelebihan ini membuat operasi penghapusan dan pencarian menjadi lebih efisien dibandingkan Singly Linked List karena kita bisa langsung mengakses elemen sebelumnya. Praktikum ini berhasil membuktikan kemampuan Doubly Linked List dalam mengelola data kendaraan secara dinamis, memverifikasi field kunci (Nomor Polisi) sebelum penyisipan, dan menerapkan konsep ADT untuk kode yang terstruktur.

B. Referensi

- H. M. Deitel and P. J. Deitel, C++ How to Program. 10th ed. Boston, MA: Pearson Education, 2017.
- D. S. Malik, Data Structures Using C++. 2nd ed. Boston, MA: Cengage Learning, 2018.
- S. Sahni, Data Structures, Algorithms, and Applications in C++. Hyderabad, India: Universities Press, 2019.