

**LAPORAN PRAKTIKUM STRUKTUR
DATA**

**MODUL
DOUBLY LINKED LIST (DLL)**



Disusun Oleh :

NAMA : Ayu Setyaning Tyas

NIM : 103112430119

Dosen

FAHRUDIN MUKTI WIBOWO

**PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

A. Dasar Teori

Daftar tertaut ganda (double linked list) merupakan struktur data yang lebih kompleks daripada daftar tertaut tunggal (single linked list), tetapi menawarkan beberapa keuntungan. Keuntungan utama dari daftar tertaut ganda adalah memungkinkan traversal daftar yang efisien di kedua arah. Hal ini karena setiap simpul dalam daftar berisi penunjuk ke simpul sebelumnya dan penunjuk ke simpul berikutnya. Hal ini memungkinkan penyisipan dan penghapusan simpul dari daftar dengan cepat dan mudah, serta traversal daftar yang efisien di kedua arah. Daftar Tertaut Ganda (DLL) merupakan jenis daftar tertaut yang setiap simpulnya berisi tiga bagian: data, penunjuk ke simpul berikutnya, dan penunjuk ke simpul sebelumnya. Struktur ini memungkinkan penelusuran daftar dalam arah maju dan mundur, tidak seperti daftar tertaut tunggal yang hanya dapat ditelusuri maju. Dalam doubly linked list tautan sebelumnya dan berikutnya dari simpul awal dan akhir, masing-masing, menunjuk ke suatu jenis terminator, biasanya simpul sentinel atau null, untuk memudahkan penelusuran daftar. Jika hanya ada satu simpul sentinel, maka daftar tersebut terhubung secara melingkar melalui simpul sentinel. Daftar ini dapat dikonseptualisasikan sebagai dua daftar tertaut tunggal yang dibentuk dari item data yang sama, tetapi dalam urutan berurutan yang berlawanan.

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

Guided 1

Code Program

```
#ifndef SINGLYLIST_H_INCLUDE
#define SINGLYLIST_H_INCLUDE

#include <iostream>

#define Nill NULL

typedef int infotype;
typedef struct Elmtlist *address;

struct Elmtlist {
    infotype info;
    address next;
};

struct List {
    address First;
};

//deklarasi prosedur dan fungsi primitif
void CreateList(List &L);
address Alokasi(infotype x);
void Dealokasi(address &P);
void InsertFirst(List &L, address P);
void InsertLast(List &L, address P);
void printInfo(List L);
```

```
#endif
```

Code *singlylist.cpp*

```
#include "singlylist.h"

void CreateList(List &L) {
    L.First = Nill;
}

address Alokasi(infotype x) {
    address P = new Elmtlist;
    P->info = x;
    P->next = Nill;
    return P;
}

void Dealokasi(address &P) {
    delete P;
}

void InsertFirst(List &L, address P) {
    P->next = L.First;
    L.First = P;
}

void InsertLast(List &L, address P) {
    if (L.First == Nill) {
        //jika list kosong, insertlast sama dengan insertfirst
        InsertFirst(L, P);
    } else {
        // jika list tidak kosong, cari elemen terakhir
        address Last = L.First;
        while (Last->next != Nill) {
            Last = Last->next;
        }
        // sambungkan elemen terakhir dengan elemen baru (p)
        Last->next = P;
    }
}

void printInfo(List L) {
    address P = L.First;
    if (P == Nill) {
        std::cout << "List kosong! " << std::endl;
    } else {
        while (P != Nill) {
            std::cout << P->info << " ";
            P = P->next;
        }
    }
}
```

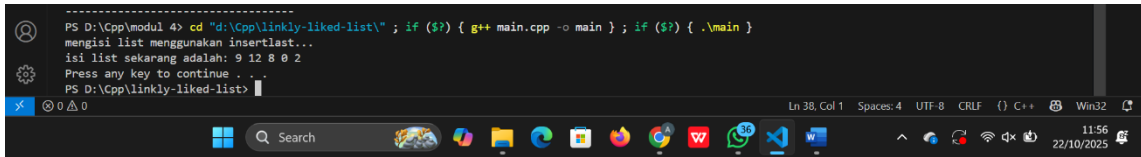
```
}  
std::cout << std::endl;  
}  
}
```

Code *main.cpp*

```
#include <iostream>  
#include <cstdlib>  
#include "singlylist.h"  
#include "singlylist.cpp"  
  
using namespace std;  
  
int main() {  
    List L;  
    address P; // cukup satu pointer untuk digunakan berulang kali  
  
    CreateList(L);  
  
    cout << "mengisi list menggunakan insertlast..." << endl;  
    // mengisi list sesuai urutan  
    P = Alokasi(9);  
    InsertLast(L, P);  
  
    P = Alokasi(12);  
    InsertLast(L, P);  
  
    P = Alokasi(8);  
    InsertLast(L, P);  
  
    P = Alokasi(0);  
    InsertLast(L, P);  
  
    P = Alokasi (2);  
    InsertLast(L, P);  
  
    cout << "isi list sekarang adalah: ";  
    printInfo(L);  
  
    system ("pause");  
    return 0;  
}
```

Screenshots Output

```
-----
PS D:\Cpp\modul 4> cd "d:\Cpp\linkly-liked-list" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
mengisi list menggunakan insertlast...
isi list sekarang adalah: 9 12 8 0 2
Press any key to continue . . .
PS D:\Cpp\linkly-liked-list>
```



Deskripsi:

Program ini dibuat menggunakan Single Linked List. Program ini dibuat bertujuan untuk mempelajari bagaimana cara menambahkan dan menghapus tanpa bantuan array.

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

Unguided 1

Code Program

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

struct kendaraan {
    string nomopolisi;
    string warna;
    int thnBuat;
};

struct Node {
    kendaraan info;
    Node* next;
    Node* prev;
};

struct List {
    Node* first;
    Node* last;
};

void createList(List &L) {
    L.first = NULL;
    L.last = NULL;
}

Node* alokasi(kendaraan x) {
    Node* P = new Node;
    P->info = x;
    P->next = NULL;
    P->prev = NULL;
    return P;
}

void insertFirst(List &L, Node* P) {
    if (L.first == NULL) {
        L.first = P;
        L.last = P;
    }
}
```

```

    } else {
        P->next = L.first;
        L.first->prev = P;
        L.first = P;
    }
}

Node* findElm(List L, string nomorpolisi) {
    Node* P = L.first;
    while (P != NULL) {
        if (P->info.nomorpolisi == nomorpolisi) {
            return P;
        }
        P = P->next;
    }
    return NULL;
}

void printInfo(List L) {
    Node* P = L.first;
    int i = 1;
    cout << endl << "DATA LIST " << i << endl;
    while (P != NULL) {
        cout << "no polisi : " << P->info.nomorpolisi << endl;
        cout << "warna    : " << P->info.warna << endl;
        cout << "tahun    : " << P->info.thnBuat << endl;
        P = P->next;
    }
}

int main() {
    List L;
    createList(L);
    kendaraan x;
    Node* P;

    // Input 1
    cout << "masukkan nomor polisi : ";
    cin >> x.nomorpolisi;
    cout << "masukkan warna kendaraan : ";
    cin >> x.warna;
    cout << "masukkan tahun kendaraan : ";
    cin >> x.thnBuat;
    P = alokasi(x);
    insertFirst(L, P);
    cout << endl;

    // Input 2

```

```

cout << "masukkan nomor polisi : ";
cin >> x.nomorpolisi;
cout << "masukkan warna kendaraan : ";
cin >> x.warna;
cout << "masukkan tahun kendaraan : ";
cin >> x.thnBuat;
if (findElm(L, x.nomorpolisi) != NULL)
    cout << "nomor polisi sudah terdaftar" << endl;
else {
    P = alokasi(x);
    insertFirst(L, P);
}
cout << endl;

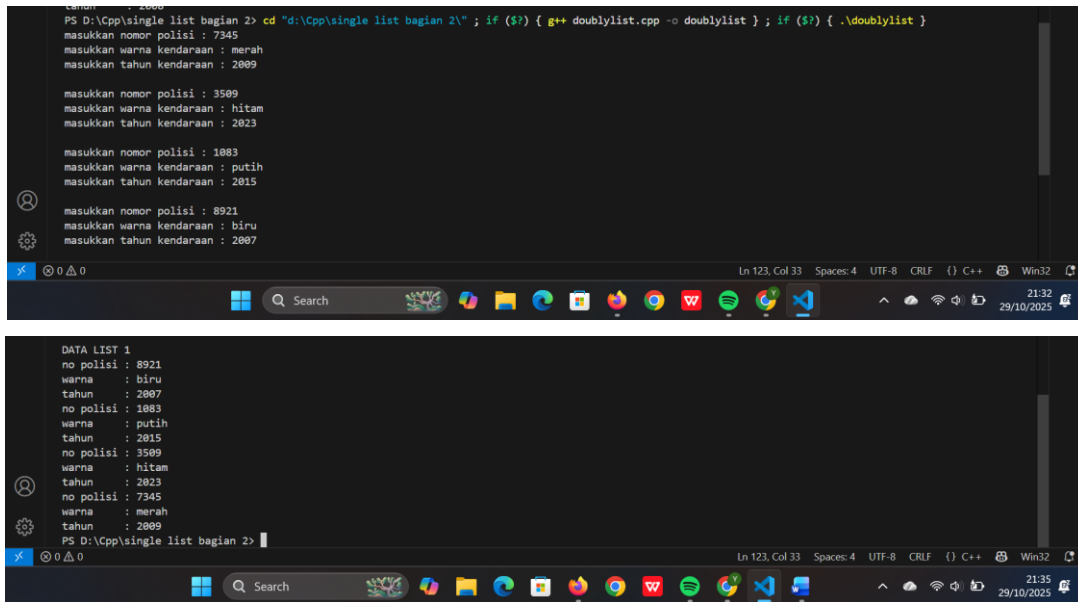
// Input 3
cout << "masukkan nomor polisi : ";
cin >> x.nomorpolisi;
cout << "masukkan warna kendaraan : ";
cin >> x.warna;
cout << "masukkan tahun kendaraan : ";
cin >> x.thnBuat;
if (findElm(L, x.nomorpolisi) != NULL)
    cout << "nomor polisi sudah terdaftar" << endl;
else {
    P = alokasi(x);
    insertFirst(L, P);
}
cout << endl;

// Input 4
cout << "masukkan nomor polisi : ";
cin >> x.nomorpolisi;
cout << "masukkan warna kendaraan : ";
cin >> x.warna;
cout << "masukkan tahun kendaraan : ";
cin >> x.thnBuat;
if (findElm(L, x.nomorpolisi) != NULL)
    cout << "nomor polisi sudah terdaftar" << endl;
else {
    P = alokasi(x);
    insertFirst(L, P);
}
printInfo(L);

return 0;
}

```


Screenshots Output



```
PS D:\Cpp\single list bagian 2> cd "d:\Cpp\single list bagian 2\" ; if ($?) { g++ doublylist.cpp -o doublylist } ; if ($?) { .\doublylist }  
masukkan nomor polisi : 7345  
masukkan warna kendaraan : merah  
masukkan tahun kendaraan : 2009  
  
masukkan nomor polisi : 3509  
masukkan warna kendaraan : hitam  
masukkan tahun kendaraan : 2023  
  
masukkan nomor polisi : 1083  
masukkan warna kendaraan : putih  
masukkan tahun kendaraan : 2015  
  
masukkan nomor polisi : 8921  
masukkan warna kendaraan : biru  
masukkan tahun kendaraan : 2007  
  
DATA LIST 1  
no polisi : 8921  
warna : biru  
tahun : 2007  
no polisi : 1083  
warna : putih  
tahun : 2015  
no polisi : 3509  
warna : hitam  
tahun : 2023  
no polisi : 7345  
warna : merah  
tahun : 2009  
PS D:\Cpp\single list bagian 2>
```

Deskripsi:

Program ini merupakan program yang dibuat dalam bentuk doublylist. Program ini dibuat untuk menyimpan, menambah, menghapus, dan menampilkan data secara dinamis tanpa batas ukuran tetap seperti array.

Unguided 2

Code Program

```
#include <iostream>  
  
#include <string>  
  
using namespace std;  
  
struct kendaraan {  
    string nomopolisi;  
    string warna;  
    int thnBuat;  
};  
  
struct Node {  
    kendaraan info;  
    Node* next;  
    Node* prev;  
};
```

```

struct List {
    Node* first;
    Node* last;
};

void createList(List &L) {
    L.first = NULL;
    L.last = NULL;
}

Node* alokasi(kendaraan x) {
    Node* P = new Node;
    P->info = x;
    P->next = NULL;
    P->prev = NULL;
    return P;
}

void insertFirst(List &L, Node* P) {
    if (L.first == NULL) {
        L.first = P;
        L.last = P;
    } else {
        P->next = L.first;
        L.first->prev = P;
        L.first = P;
    }
}

Node* findElm(List L, string nomopolisi) {
    Node* P = L.first;

```

```

while (P != NULL) {
    if (P->info.nomorpolisi == nomorpolisi) {
        return P;
    }
    P = P->next;
}
return NULL;
}

void printInfo(List L) {
    Node* P = L.first;
    int i = 1;
    cout << endl << "DATA LIST " << i << endl;
    while (P != NULL) {
        cout << "no polisi : " << P->info.nomorpolisi << endl;
        cout << "warna    : " << P->info.warna << endl;
        cout << "tahun    : " << P->info.thnBuat << endl;
        P = P->next;
    }
}

int main() {
    List L;
    createList(L);
    kendaraan x;
    Node* P;

    // Input 1
    cout << "masukkan nomor polisi : ";
    cin >> x.nomorpolisi;
    cout << "masukkan warna kendaraan : ";
    cin >> x.warna;

```

```

cout << "masukkan tahun kendaraan : ";
cin >> x.thnBuat;
P = alokasi(x);
insertFirst(L, P);
cout << endl;

// Input 2
cout << "masukkan nomor polisi : ";
cin >> x.nomorpolisi;
cout << "masukkan warna kendaraan : ";
cin >> x.warna;
cout << "masukkan tahun kendaraan : ";
cin >> x.thnBuat;
if (findElm(L, x.nomorpolisi) != NULL)
    cout << "nomor polisi sudah terdaftar" << endl;
else {
    P = alokasi(x);
    insertFirst(L, P);
}
cout << endl;

// Input 3
cout << "masukkan nomor polisi : ";
cin >> x.nomorpolisi;
cout << "masukkan warna kendaraan : ";
cin >> x.warna;
cout << "masukkan tahun kendaraan : ";
cin >> x.thnBuat;
if (findElm(L, x.nomorpolisi) != NULL)
    cout << "nomor polisi sudah terdaftar" << endl;
else {
    P = alokasi(x);

```

```

    insertFirst(L, P);
}
cout << endl;

// Input 4
cout << "masukkan nomor polisi : ";
cin >> x.nomorpolisi;
cout << "masukkan warna kendaraan : ";
cin >> x.warna;
cout << "masukkan tahun kendaraan : ";
cin >> x.thnBuat;
if (findElm(L, x.nomorpolisi) != NULL)
    cout << "nomor polisi sudah terdaftar" << endl;
else {
    P = alokasi(x);
    insertFirst(L, P);
}

// menampilkan hasil
printInfo(L);

string cari;
cout << endl << "Masukkan Nomor Polisi yang dicari : ";
cin >> cari;

Node* hasil = findElm(L, cari);
if (hasil != NULL) {
    cout << endl;
    cout << "Nomor Polisi : " << hasil->info.nomorpolisi << endl;
    cout << "Warna      : " << hasil->info.warna << endl;
    cout << "Tahun      : " << hasil->info.thnBuat << endl;
} else {

```

```

        cout << "Data dengan nomor polisi " << cari << " tidak ditemukan." << endl;
    }

    return 0;
}

```

Screenshots Output

The first screenshot shows the program's execution in a Windows command prompt. It prompts the user to enter vehicle details: license plate number, color, and year. The user enters three vehicles: (7345, merah, 2009), (3509, hitam, 2023), and (1883, putih, 2015). The program then displays the data list.

The second screenshot shows the program displaying the data list and then searching for a specific license plate. The user enters 1883, and the program displays the details of the vehicle with that license plate: (1883, putih, 2015).

Deskripsi:

Program ini merupakan program yang dibuat dalam bentuk doublylist. Program ini dibuat untuk menyimpan, menambah, menghapus, dan menampilkan data secara dinamis tanpa batas ukuran tetap seperti array. Program ini juga memiliki sebuah fungsi untuk mencari data yang sudah di input.

Unguided 3

Code Program

```

#include <iostream>

#include <string>

using namespace std;

struct kendaraan {

    string nomopolisi;

    string warna;

    int thnBuat;

```

```
};

struct Node {
    kendaraan info;
    Node* next;
    Node* prev;
};

struct List {
    Node* first;
    Node* last;
};

void createList(List &L) {
    L.first = NULL;
    L.last = NULL;
}

Node* alokasi(kendaraan x) {
    Node* P = new Node;
    P->info = x;
    P->next = NULL;
    P->prev = NULL;
    return P;
}

void insertFirst(List &L, Node* P) {
    if (L.first == NULL) {
        L.first = P;
        L.last = P;
    } else {
```

```

        P->next = L.first;
        L.first->prev = P;
        L.first = P;
    }
}

Node* findElm(List L, string nomorpolisi) {
    Node* P = L.first;
    while (P != NULL) {
        if (P->info.nomorpolisi == nomorpolisi) {
            return P;
        }
        P = P->next;
    }
    return NULL;
}

void printInfo(List L) {
    Node* P = L.first;
    int i = 1;
    cout << endl << "DATA LIST " << i << endl;
    while (P != NULL) {
        cout << "no polisi : " << P->info.nomorpolisi << endl;
        cout << "warna    : " << P->info.warna << endl;
        cout << "tahun    : " << P->info.thnBuat << endl;
        P = P->next;
    }
}

// penambahan fungsi untuk soal nomor
void deleteFirst(List &L, Node* &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, Node* &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(Node* Prec, Node* &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;
    }
}

int main() {
    List L;
    createList(L);
    kendaraan x;
    Node* P;

    // Input 1
    cout << "masukkan nomor polisi : ";
    cin >> x.nomorpolisi;
    cout << "masukkan warna kendaraan : ";
    cin >> x.warna;
    cout << "masukkan tahun kendaraan : ";
    cin >> x.thnBuat;
    P = alokasi(x);
    insertFirst(L, P);
    cout << endl;

    // Input 2
    cout << "masukkan nomor polisi : ";
    cin >> x.nomorpolisi;
    cout << "masukkan warna kendaraan : ";
    cin >> x.warna;
    cout << "masukkan tahun kendaraan : ";

```

```
cin >> x.thnBuat;

if (findElm(L, x.nomorpolisi) != NULL)

    cout << "nomor polisi sudah terdaftar" << endl;

else {

    P = alokasi(x);

    insertFirst(L, P);

}

cout << endl;
```

```
// Input 3
```

```
cout << "masukkan nomor polisi : ";

cin >> x.nomorpolisi;

cout << "masukkan warna kendaraan : ";

cin >> x.warna;

cout << "masukkan tahun kendaraan : ";

cin >> x.thnBuat;

if (findElm(L, x.nomorpolisi) != NULL)

    cout << "nomor polisi sudah terdaftar" << endl;

else {

    P = alokasi(x);

    insertFirst(L, P);

}

cout << endl;
```

```
// Input 4
```

```
cout << "masukkan nomor polisi : ";

cin >> x.nomorpolisi;

cout << "masukkan warna kendaraan : ";

cin >> x.warna;

cout << "masukkan tahun kendaraan : ";

cin >> x.thnBuat;
```

```

if (findElm(L, x.nomorpolisi) != NULL)

    cout << "nomor polisi sudah terdaftar" << endl;
else {
    P = alokasi(x);
    insertFirst(L, P);
}

// mencari data
string cari;
cout << endl << "Masukkan Nomor Polisi yang dicari : ";
cin >> cari;

Node* hasil = findElm(L, cari);
if (hasil != NULL) {
    cout << endl;
    cout << "Nomor Polisi : " << hasil->info.nomorpolisi << endl;
    cout << "Warna      : " << hasil->info.warna << endl;
    cout << "Tahun      : " << hasil->info.thnBuat << endl;
} else {
    cout << "Data dengan nomor polisi " << cari << " tidak ditemukan." << endl;
}

// Hapus elemen
string hapus;
cout << endl << "Masukkan Nomor Polisi yang akan dihapus : ";
cin >> hapus;

Node* target = findElm(L, hapus);
if (target != NULL) {
    if (target == L.first) {
        deleteFirst(L, P);
    }
}

```

```

    } else if (target == L.last) {
        deleteLast(L, P);
    } else {
        Node* Prec = target->prev;
        deleteAfter(Prec, P);
    }

    cout << "Data dengan nomor polisi " << hapus << " berhasil dihapus." << endl;
} else {
    cout << "Data tidak ditemukan." << endl;
}

// Tampilkan hasil akhir
printInfo(L);

return 0;
}

```

Screenshots Output

The first screenshot shows the program's execution in a terminal window. The user enters the following commands and responses:

```

PS D:\Cpp\single list bagian 2> cd "d:\Cpp\single list bagian 2\" ; if ($?) { g++ doublylist.cpp -o doublylist } ; if ($?) { .\doublylist }
masukkan nomor polisi : 7345
masukkan warna kendaraan : merah
masukkan tahun kendaraan : 2009

masukkan nomor polisi : 3509
masukkan warna kendaraan : hitam
masukkan tahun kendaraan : 2023

masukkan nomor polisi : 1083
masukkan warna kendaraan : putih
masukkan tahun kendaraan : 2015

masukkan nomor polisi : 8921
masukkan warna kendaraan : biru
masukkan tahun kendaraan : 2007

```

The second screenshot shows the program's output, displaying the list of vehicles and the result of deleting a vehicle with the license plate 1083:

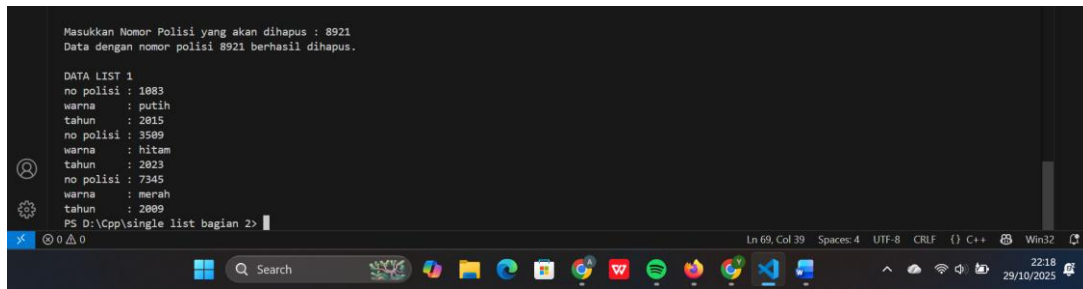
```

DATA LIST 1
no polisi : 8921
warna : biru
tahun : 2007
no polisi : 1083
warna : putih
tahun : 2015
no polisi : 3509
warna : hitam
tahun : 2023
no polisi : 7345
warna : merah
tahun : 2009

Masukkan Nomor Polisi yang dicari : 1083

Nomor Polisi : 1083
Warna : putih
Tahun : 2015

```



```
Masukkan Nomor Polisi yang akan dihapus : 8921
Data dengan nomor polisi 8921 berhasil dihapus.

DATA LIST 1
no polisi : 1083
warna : putih
tahun : 2015
no polisi : 3509
warna : hitam
tahun : 2023
no polisi : 7345
warna : merah
tahun : 2009
PS D:\Cpp\single list bagian 2>
```

Deskripsi:

Program ini merupakan program yang dibuat dalam bentuk doublylist. Program ini dibuat untuk menyimpan, menambah, menghapus, dan menampilkan data secara dinamis tanpa batas ukuran tetap seperti array. Program ini juga memiliki sebuah fungsi untuk mencari data yang sudah di input. Selain mencari program ini juga bisa menghapus data yang sudah di input sebelumnya.

D. Kesimpulan

Program ini mengimplementasikan konsep lengkap dari Doubly Linked List (DLL), di mana struktur data kendaraan (mencakup nomor polisi, warna, dan tahun pembuatan) dirancang dengan pointer prev dan next untuk memungkinkan akses dan traversal data dua arah. Fungsionalitas inti mencakup prosedur untuk {createList}, {alokasi} memori, dan penyisipan data di awal ({insertFirst}). Selain itu, program ini mengelola validasi data dengan kemampuan menolak input duplikat dan menyediakan fungsi {findElm} untuk mencari kendaraan berdasarkan nomor polisi, serta {printInfo} untuk menampilkan seluruh daftar. Program ini diperkuat dengan prosedur penghapusan data yang komprehensif, yaitu {deleteFirst}, {deleteLast}, dan {deleteAfter}. Dengan kemampuan untuk mencari dan menghapus data spesifik (misalnya {D003}) serta menghasilkan output yang sesuai dengan contoh modul, program ini secara keseluruhan berhasil menerapkan semua konsep dasar Doubly Linked List secara lengkap dan benar.

E. Referensi

<https://www.geeksforgeeks.org/dsa/doubly-linked-list/>
<https://www.geeksforgeeks.org/cpp/doubly-linked-list-in-cpp/>
https://en.wikipedia.org/wiki/Doubly_linked_list