**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**

1. 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](https://en.wikipedia.org/wiki/Sentinel_node" \o "Nodus sentinel) atau [null](https://en.wikipedia.org/wiki/Null_pointer) , 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](https://en.wikipedia.org/wiki/Linked_list) yang dibentuk dari item data yang sama, tetapi dalam urutan berurutan yang berlawanan.

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

Guided 1

Code Program

#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;

        }

        if (current == ptr\_last)

        {

            delete\_last();

            return;

        }

        current->prev->next = current->next;

        current->next->prev = current->prev;

        delete current;

    }

}

void edit\_node(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: ";

    view();

    delete\_target(30);

    cout << "Setelah delete\_target\t: ";

    view();

}

Screenshoot Output

A screenshot of a computer

AI-generated content may be incorrect.

Deskripsi:

Program ini dibuat menggunakan Single Linked List. Program ini dibuat untuk menamppilkan, menghapus data, menghapus target pointer dan menambahkan data di dalam node.

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

Unguided 1

Code Program

#include <iostream>

#include <string>

using namespace std;

struct kendaraan {

    string nomorpolisi;

    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

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

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 nomorpolisi;

    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);

    }

    // 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

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

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 nomorpolisi;

    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

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

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.

1. 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.

1. 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>