

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

**MODUL 6
DOUBLY LINKED LIST
(BAGIAN PERTAMA)**



Disusun Oleh :

NAMA : Muhammad Luthfi Arrafi

Ramadhani

NIM : 103112430043

Dosen Pengampu:

Fahrudin Mukti Wibowo

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Dasar teori pada modul ini membahas konsep Doubly Linked List, yaitu struktur data yang setiap elemennya memiliki dua pointer: *next* yang menunjuk ke elemen berikutnya dan *prev* yang menunjuk ke elemen sebelumnya. Struktur ini memungkinkan penelusuran data dilakukan dua arah, dari awal ke akhir maupun sebaliknya. Setiap node terdiri atas tiga bagian utama yaitu *info* sebagai data, serta *next* dan *prev* sebagai penghubung antar elemen. Operasi dasar yang umum dilakukan meliputi penambahan, penghapusan, pencarian, dan penelusuran data.

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

Guided 1

(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;
    }
    delete temp;
}

```

```

void delete_last()
{
    if (ptr_first == 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();
}

```

Screenshots Output

```

PS E:\BACKUP092024\VS CODE\3rd sm\Praktikum Muhammad Luthfi Arrafi Ramadhani-103112430043> cd "e:\BACKUP092024\VS CODE\3rd sm\Praktikum Muhammad Luthfi Arrafi Ramadhani-103112430043\Minggu 5\Guided\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
Awal          : 5 <-> 10 <-> 20
Setelah delete_first : 10 <-> 20
Setelah delete_last  : 10
Setelah tambah      : 10 <-> 30 <-> 40
Setelah delete_target : 10 <-> 40
PS E:\BACKUP092024\VS CODE\3rd sm\Praktikum Muhammad Luthfi Arrafi Ramadhani-103112430043\Minggu 5\Guided>

```

Deskripsi:

Program tersebut merupakan contoh penerapan Doubly Linked List di C++. Struktur Node berisi data dan dua penunjuk, yaitu *prev* untuk elemen sebelumnya dan *next* untuk elemen sesudahnya. Program ini punya beberapa fungsi seperti menambah data di awal (*add_first*), di akhir (*add_last*), atau setelah elemen tertentu (*add_target*). Ada juga fungsi untuk menghapus data (*delete_first*, *delete_last*, *delete_target*), mengedit nilai (*edit_node*), dan menampilkan isi list (*view*). Di fungsi *main*, semua operasi tersebut dijalankan untuk menunjukkan bagaimana data bisa ditambah, dihapus, dan ditampilkan.

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

Unguided 1

(Doublylist.h)

```

// Muhammad Luthfi Arrafi Ramadhani
// 103112430043
// IF 12-06

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

struct Kendaraan {
    string nopol;
    string warna;
    int thnBuat;
};

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

```

```
};

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

void createList(List &L);
Node* alokasi(Kendaraan x);
void dealokasi(Node* P);
void insertLast(List &L, Node* P);
void printInfo(List L);
bool isExist(List L, string nopol);

#endif
```

Deskripsi:

File Doublylist.h ini berisi deklarasi struktur data dan fungsi yang digunakan dalam program. Di dalamnya terdapat struktur *Kendaraan*, *Node*, dan *List*, serta deklarasi fungsi seperti *createList*, *alokasi*, *insertLast*, dan *printInfo* yang digunakan untuk mengelola data kendaraan.

(Doublylist.cpp)

```
// Muhammad Luthfi Arrafi Ramadhani
// 103112430043
// IF 12-06

#include "Doublylist.h"

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 dealokasi(Node* P) {
    delete P;
}

bool isExist(List L, string nopol) {
```

```

Node* temp = L.first;
while (temp != NULL) {
    if (temp->info.nopol == nopol)
        return true;
    temp = temp->next;
}
return false;
}

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

void printInfo(List L) {
    Node* temp = L.first;
    if (temp == NULL) {
        cout << "List kosong." << endl;
        return;
    }

    cout << "\nDATA LIST 1" << endl;
    while (temp != NULL) {
        cout << "no polisi : " << temp->info.nopol << endl;
        cout << "warna      : " << temp->info.warna << endl;
        cout << "tahun       : " << temp->info.thnBuat << endl;
        temp = temp->next;
    }
}

```

Deskripsi:

File Doublylist.cpp ini berisi implementasi dari semua fungsi yang dideklarasikan di Doublylist.h. Fungsinya mencakup pembuatan list baru, menambah data di akhir list, mengecek duplikasi nomor polisi, serta menampilkan seluruh isi dari list.

(main.cpp)

```
// Muhammad Luthfi Arrafi Ramadhani
// 103112430043
// IF 12-06

#include "Doublylist.h"

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

    while (true) {
        Kendaraan k;
        cout << "\nmasukkan nomor polisi: ";
        cin >> k.nopol;

        if (k.nopol == "-") {
            break;
        }

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

        if (isExist(L, k.nopol)) {
            cout << "nomor polisi sudah terdaftar\n";
            continue;
        }

        insertLast(L, alokasi(k));
    }

    printInfo(L);
    return 0;
}
```

Screenshots Output

```
PS E:\BACKUP092024\VSCODE\3rd sm\Praktikum Muhammad Luthfi Arrafi Ramadhani-103112430043\Minggu 5\Unguided> g++ main.cpp Doublylist.cpp -o main
PS E:\BACKUP092024\VSCODE\3rd sm\Praktikum Muhammad Luthfi Arrafi Ramadhani-103112430043\Minggu 5\Unguided> ./main

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

masukkan nomor polisi: -

DATA LIST 1
no polisi : D001
warna : hitam
tahun : 90
no polisi : D003
warna : putih
tahun : 70
no polisi : D004
warna : kuning
tahun : 90
PS E:\BACKUP092024\VSCODE\3rd sm\Praktikum Muhammad Luthfi Arrafi Ramadhani-103112430043\Minggu 5\Unguided> |
```

Deskripsi:

File main.cpp ini merupakan program utama yang menjalankan semua fungsi sebelumnya. Program meminta input data kendaraan, memeriksa apakah nomor polisi sudah ada, menambahkan data baru ke list, dan menampilkan hasil akhir seluruh data kendaraan.

Unguided 2

(Doublylist.h)

```
// Muhammad Luthfi Arrafi Ramadhani
// 103112430043
// IF 12-06

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

struct Kendaraan {
    string nopol;
    string warna;
    int thnBuat;
};

struct Node {
    Kendaraan info;
```

```

    Node *next;
    Node *prev;
};

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

void createList(List &L);
Node* alokasi(Kendaraan x);
void dealokasi(Node* P);
void insertLast(List &L, Node* P);
void printInfo(List L);
bool isExist(List L, string nopol);
Node* findElm(List L, string nopol);

#endif

```

(Doublylist.cpp)

```

// Muhammad Luthfi Arrafi Ramadhani
// 103112430043
// IF 12-06

#include "Doublylist.h"

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 dealokasi(Node* P) {
    delete P;
}

bool isExist(List L, string nopol) {
    Node* temp = L.first;
    while (temp != NULL) {
        if (temp->info.nopol == nopol)
            return true;
    }
}

```

```

        temp = temp->next;
    }
    return false;
}

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

void printInfo(List L) {
    Node* temp = L.first;
    if (temp == NULL) {
        cout << "List kosong." << endl;
        return;
    }

    cout << "\nDATA LIST 1" << endl;
    while (temp != NULL) {
        cout << "no polisi : " << temp->info.nopol << endl;
        cout << "warna      : " << temp->info.warna << endl;
        cout << "tahun       : " << temp->info.thnBuat << endl;
        temp = temp->next;
    }
}

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

```

(main.cpp)

```
// Muhammad Luthfi Arrafi Ramadhani
// 103112430043
// IF 12-06

#include "Doublylist.h"

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

    while (true) {
        Kendaraan k;
        cout << "\nmasukkan nomor polisi: ";
        cin >> k.nopol;

        if (k.nopol == "-") {
            break;
        }

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

        if (isExist(L, k.nopol)) {
            cout << "nomor polisi sudah terdaftar\n";
            continue;
        }

        insertLast(L, alokasi(k));
    }

    printInfo(L);

    string cari;
    cout << "\nmasukkan nomor polisi yang ingin dicari: ";
    cin >> cari;

    Node* hasil = findElm(L, cari);
    if (hasil != NULL) {
        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";
    }

    return 0;
}
```

```
}
```

Screenshots Output

```
DATA LIST 1
no polisi : D001
warna     : hitam
tahun     : 90
no polisi : D003
warna     : putih
tahun     : 70
no polisi : D004
warna     : kuning
tahun     : 90

Masukkan nomor polisi yang ingin dicari: D001

Data ditemukan:
Nomor Polisi : D001
Warna        : hitam
Tahun        : 90
PS E:\BACKUP092024\VSCODE\3rd sm\Praktikum Muhammad Luthfi Arrafi Ramadhani-103112430043\Minggu 5\Unguided>
```

Deskripsi:

Program ini menambahkan fungsi *findElem()* untuk mencari data kendaraan nomor polisi. Fitur ini memungkinkan pengguna untuk mencari dan menampilkan detail kendaraan tertentu tanpa harus menelusuri seluruh list secara manual. Jika nomor polisi yang dimasukkan ditemukan, program akan menampilkan data lengkap kendaraan tersebut, sedangkan jika tidak ditemukan, akan muncul pesan bahwa data tidak tersedia.

Unguided 3

(Doublylist.h)

```
// Muhammad Luthfi Arrafi Ramadhani
// 103112430043
// IF 12-06

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

struct Kendaraan {
    string nopol;
    string warna;
    int thnBuat;
};

struct Node {
    Kendaraan info;
```

```

    Node *next;
    Node *prev;
};

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

void createList(List &L);
Node* alokasi(Kendaraan x);
void dealokasi(Node* P);
void insertLast(List &L, Node* P);
void printInfo(List L);
bool isExist(List L, string nopol);
Node* findElm(List L, string nopol);
void deleteElm(List &L, string nopol);

#endif

```

(Doublylist.cpp)

```

// Muhammad Luthfi Arrafi Ramadhani
// 103112430043
// IF 12-06

#include "Doublylist.h"

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 dealokasi(Node* P) {
    delete P;
}

bool isExist(List L, string nopol) {
    Node* temp = L.first;
    while (temp != NULL) {
        if (temp->info.nopol == nopol)

```

```

        return true;
        temp = temp->next;
    }
    return false;
}

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

void printInfo(List L) {
    Node* temp = L.first;
    if (temp == NULL) {
        cout << "List kosong." << endl;
        return;
    }

    cout << "\nDATA LIST 1" << endl;
    while (temp != NULL) {
        cout << "no polisi : " << temp->info.nopol << endl;
        cout << "warna      : " << temp->info.warna << endl;
        cout << "tahun       : " << temp->info.thnBuat << endl;
        temp = temp->next;
    }
}

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

void deleteElm(List &L, string nopol) {
    Node* target = findElm(L, nopol);

    if (target == NULL) {
        cout << "data dengan nomor polisi " << nopol << " tidak ditemukan\n";
        return;
    }
}

```



```

    if (L.first == L.last) {
        L.first = L.last = NULL;
    }

    else if (target == L.first) {
        L.first = target->next;
        L.first->prev = NULL;
    }

    else if (target == L.last) {
        L.last = target->prev;
        L.last->next = NULL;
    }

    else {
        target->prev->next = target->next;
        target->next->prev = target->prev;
    }

    cout << "data dengan nomor polisi " << nopol << " berhasil dihapus\n";
    dealokasi(target);
}

```

(main.cpp)

```

// Muhammad Luthfi Arrafi Ramadhani
// 103112430043
// IF 12-06

#include "Doublylist.h"

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

    while (true) {
        Kendaraan k;
        cout << "\nmasukkan nomor polisi: ";
        cin >> k.nopol;

        if (k.nopol == "-") {
            break;
        }

        cout << "masukkan warna kendaraan: ";
        cin >> k.warna;
        cout << "masukkan tahun kendaraan: ";
        cin >> k.thnBuat;
    }
}

```

```

    if (isExist(L, k.nopol)) {
        cout << "nomor polisi sudah terdaftar\n";
        continue;
    }

    insertLast(L, alokasi(k));
}

printInfo(L);

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

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

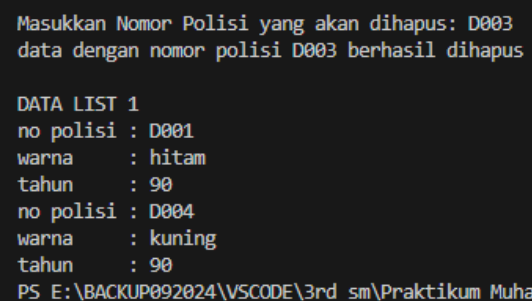
string hapus;
cout << "\nMasukkan Nomor Polisi yang akan dihapus: ";
cin >> hapus;
deleteElm(L, hapus);

printInfo(L);

return 0;
}

```

Screenshots Output



```

Masukkan Nomor Polisi yang akan dihapus: D003
data dengan nomor polisi D003 berhasil dihapus

DATA LIST 1
no polisi : D001
warna     : hitam
tahun     : 90
no polisi : D004
warna     : kuning
tahun     : 90
PS E:\BACKUP092024\VSCODE\3rd sm\Praktikum Muhammad Luthfi Arrafi Ramadhani-103112430043\Minggu 5\Unguided>

```

Deskripsi:

Program ini menambahkan fungsi *deleteElm()* untuk menghapus data kendaraan nomor polisi. Fitur ini bekerja dengan mencari elemen yang sesuai, lalu menghapus node tersebut dari posisi mana pun baik di awal, tengah, atau akhir list. Jika data berhasil

dihapus, program menampilkan pesan konfirmasi, dan list hasil penghapusan akan ditampilkan kembali di layar.

D. Kesimpulan

Dari latihan membuat program Doubly Linked List untuk data kendaraan di atas, dapat disimpulkan bahwa struktur data ini mempermudah proses penyimpanan, pencarian, dan penghapusan data secara efisien karena tiap elemen saling terhubung dua arah melalui pointer *next* dan *prev*. Dengan menerapkan operasi seperti *insert*, *find*, dan *delete*, kita bisa lebih memahami cara kerja hubungan antar node dan manfaat penggunaan pointer dalam pengelolaan data dinamis di C++.

E. Referensi

Gautam, A. (2024). *Exploring time complexity of doubly linked list operations*. Programiz Blog. <https://programiz.pro/resources/dsa-doubly-linked-list-complexity/>

PVS-Studio Team. (2023). *Implementation of a doubly linked list in C++*. PVS-Studio Blog. <https://pvs-studio.com/en/blog/terms/6683/>