LAPORAN PRAKTIKUM

STRUKTUR DATA

MODUL VII

DOUBLE LINKED LIST

Disusun Oleh :

Nama : Izzah Minkhotun Fannisa

NIM : 103112400198

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

C++ merupakan bahasa pemrograman yang pertama kali dikembangkan oleh Bjarne Stroustrop, seorang ilmuwan komputer asal Denmark. Bahasa pemrograman C++ disempurnakan dari bahasa C yang di temukan oleh Dennis Ritchie pada awal 1970-an. Lalu resmi dirilis pada tahun 1985. C++ cukup banyak digunakan karena kemampuannya dalam memberikan kecepatan eksekusi tinggi serta dukungan terhadap manajemen memori tingkat rendah.

Doubly Linked List, atau daftar berantai ganda, adalah jenis struktur data yang fleksibel. Ia terdiri dari serangkaian simpul (node), di mana setiap simpul punya tiga bagian utama: pointer ke simpul sebelumnya (prev), data yang disimpan, dan pointer ke simpul berikutnya *(next)*. Berbeda dari *Single Linked List* yang cuma bisa dilalui satu arah, *Doubly Linked List* bisa dilalui ke depan atau ke belakang berkat dua pointer itu. Ini bikin operasi seperti nambah atau hapus simpul di posisi mana saja jadi lebih gampang, tanpa perlu mulai dari awal list. Tapi, ia butuh lebih banyak memori karena ada pointer ekstra.

Di C++, *Doubly Linked List* biasanya dibuat pakai struct atau class yang punya pointer prev dan next. Ia juga punya operasi dasar seperti sisip *(insert)*, hapus (*delete),* telusur (*traverse*), dan cari (*search*). Struktur ini sering dipakai di aplikasi yang butuh navigasi dua arah, misalnya fitur *undo-redo*, riwayat browser, atau *playlis*t musik.

B. Guide

1. Guide 1

a. Source Code

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

struct Node {
    int data;
    Node *prev;
    Node *next;

    Node(int val, Node *p = nullptr, Node *n = nullptr)
        : data(val), prev(p), next(n) {}
};

Node *ptr_first = nullptr;
Node *ptr_last  = nullptr;

void delete_last();

void add_first(int Value) {
    Node *newNode = new Node(Value, nullptr, ptr_first);

    if (ptr_first == nullptr) {
        ptr_last = newNode;
```

```
    } else {
        ptr_first->prev = newNode;
    }
    ptr_first = newNode;
}

void add_last(int Value) {
    Node *newNode = new Node(Value, ptr_last, nullptr);

    if (ptr_last == nullptr) {
        ptr_first = newNode;
    } else {
        ptr_last->next = newNode;
    }
    ptr_last = newNode;
}

void add_target(int targetValue, int newValue) {
    Node *current = ptr_first;

    while (current != nullptr && current->data != targetValue)
{
        current = current->next;
    }

    if (current != nullptr) {
        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 == nullptr) {
        cout << "list kosong\n";
        return;
    }

    while (current != nullptr) {
        cout << current->data;
        if (current->next != nullptr) {
```

```
            cout << " <-> ";
        }
        current = current->next;
    }
    cout << endl;
}

void delete_first() {
    if (ptr_first == nullptr) return;

    Node *temp = ptr_first;

    if (ptr_first == ptr_last) {
        ptr_first = nullptr;
        ptr_last = nullptr;
    } else {
        ptr_first = ptr_first->next;
        ptr_first->prev = nullptr;
    }
    delete temp;
}

void delete_last() {
    if (ptr_last == nullptr) return;

    Node *temp = ptr_last;

    if (ptr_first == ptr_last) {
        ptr_first = nullptr;
        ptr_last = nullptr;
    } else {
        ptr_last = ptr_last->prev;
        ptr_last->next = nullptr;
    }
    delete temp;
}

void delete_target(int targetValue) {
    Node *current = ptr_first;

    while (current != nullptr && current->data != targetValue)
{
        current = current->next;
    }

    if (current != nullptr) {
        if (current == ptr_first) {
            delete_first();
```

```
        } else if (current == ptr_last) {
            delete_last();
        } else {
            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 != nullptr && current->data != targetValue)
{
        current = current->next;
    }

    if (current != nullptr) {
        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();
```

```
    return 0;
}
```

b. Screenshot Output

```
Awal                    : 5 <-> 10 <-> 20
setelah delete_first    : 10 <-> 20
setelah delete_last     : 10
setelah tambah          : 10 <-> 30 <-> 40
setelah delete_target   : 10 <-> 40
PS C:\Users\ACER\OneDrive\Desktop\Modul 5 & 6>
```

c. Deskripsi Program

Program ini dibuat buat nunjukin gimana *Doubly Linked List* bekerja di C++. Jadi, setiap data disimpan dalam node yang saling terhubung dua arah, bisa ke depan atau ke belakang. Programnya bisa nambah data di awal, di akhir, atau di tempat tertentu, bisa hapus atau ubah data berdasarkan nilai yang dicari. Setiap ada perubahan, isi list-nya langsung ditampilkan supaya hasilnya kelihatan jelas. Program ini mau kasih gambaran bagaimana struktur data *Doubly Linked List* bisa atur data dengan fleksibel, tapi tetap teratur dan gampang diubah.

C. Unguide

1. Unguide 1

a. Source Code

kendaraan.cpp

```
#include <iostream>
#include <string>

using namespace std;

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

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

struct List {
    Node* head;
    Node* tail;
```

```
};

void createList(List &L) {
    L.head = nullptr;
    L.tail = nullptr;
}

Node* buatNode(Kendaraan x) {
    Node* baru = new Node;
    baru->data = x;
    baru->next = nullptr;
    baru->prev = nullptr;
    return baru;
}

void tambahDepan(List &L, Node* P) {
    if (L.head == nullptr) {
        L.head = P;
        L.tail = P;
    } else {
        P->next = L.head;
        L.head->prev = P;
        L.head = P;
    }
}

Node* cariKendaraan(List L, string nopol) {
    Node* bantu = L.head;
    while (bantu != nullptr) {
        if (bantu->data.nopol == nopol) {
            return bantu;
        }
        bantu = bantu->next;
    }
    return nullptr;
}

void tampilData(List L) {
    Node* bantu = L.head;
    int i = 1;

    cout << "\n=== DATA KENDARAAN ===\n";
    while (bantu != nullptr) {
        cout << i++ << ". Nomor Polisi : " << bantu->data.nopol
<< endl;
        cout << "   Warna        : " << bantu->data.warna <<
endl;
```

```
        cout << "   Tahun        : " << bantu->data.tahun <<
endl << endl;
        bantu = bantu->next;
    }
}

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

    Kendaraan x;
    Node* P;

    for (int i = 1; i <= 4; i++) {
        cout << "Masukkan data kendaraan ke-" << i << endl;
        cout << "Nomor Polisi : ";
        cin >> x.nopol;
        cout << "Warna        : ";
        cin >> x.warna;
        cout << "Tahun        : ";
        cin >> x.tahun;

        if (cariKendaraan(L, x.nopol) != nullptr) {
            cout << "Nomor polisi sudah terdaftar!\n\n";
        } else {
            P = buatNode(x);
            tambahDepan(L, P);
            cout << "Data berhasil ditambahkan!\n\n";
        }
    }

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

b. Screenshot Output

```
   Tahun        : 2006
   Data berhasil ditambahkan!

   Masukkan data kendaraan ke-3
   Nomor Polisi : 9999
   Warna        : kuning
   Tahun        : 1998
   Data berhasil ditambahkan!

   Masukkan data kendaraan ke-4
   Nomor Polisi : 9997
   Warna        : abu-abu
   Tahun        : 1991
   Data berhasil ditambahkan!


   === DATA KENDARAAN ===
   1. Nomor Polisi : 9997
      Warna        : abu-abu
      Tahun        : 1991

   2. Nomor Polisi : 9999
      Warna        : kuning
      Tahun        : 1998

   3. Nomor Polisi : 9991
      Warna        : biru
      Tahun        : 2006

   4. Nomor Polisi : 1991
      Warna        : ungu
      Tahun        : 2007
```

c. Deskripsi Program

Program ini digunakan untuk mengelola data kendaraan menggunakan *Double Linked List*. Data yang disimpan meliputi nomor polisi, warna, dan tahun kendaraan. Program memungkinkan pengguna menambahkan data kendaraan dengan pengecekan nomor polisi agar tidak terjadi data ganda. Data kendaraan disimpan di bagian depan list dan seluruh data yang tersimpan dapat ditampilkan kembali.

2. Unguide 2

a. Source Code

nopolis.cpp

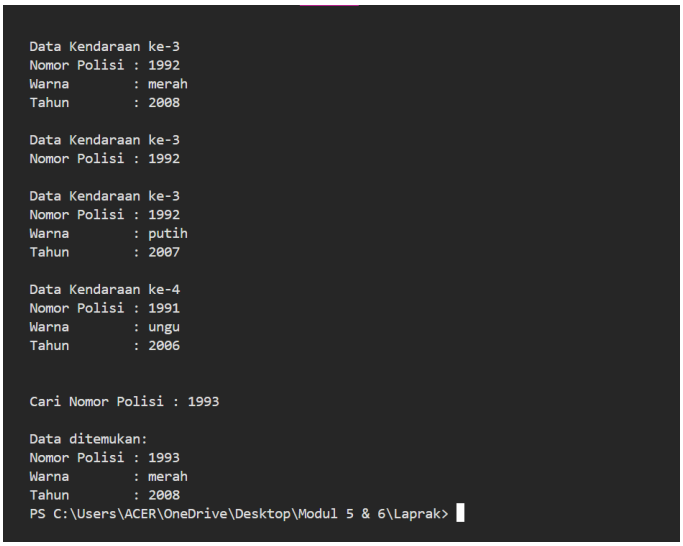
```cpp
#include <iostream>
#include <string>

using namespace std;

struct Kendaraan {
    string nopol;
    string warna;
    int tahun;
};
```

```
struct Node {
    Kendaraan data;
    Node* next;
    Node* prev;
};

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

void buatList(List &L) {
    L.first = nullptr;
    L.last  = nullptr;
}

Node* buatNode(Kendaraan x) {
    Node* P = new Node;
    P->data = x;
    P->next = nullptr;
    P->prev = nullptr;
    return P;
}

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

Node* cariData(List L, string nopol) {
    Node* P = L.first;
    while (P != nullptr) {
        if (P->data.nopol == nopol) {
            return P;
        }
        P = P->next;
    }
    return nullptr;
}

void tampilData(List L) {
    Node* P = L.first;
```

```
    int i = 1;

    cout << "\n=== DATA KENDARAAN ===\n";
    while (P != nullptr) {
        cout << "Data Kendaraan ke-" << i++ << endl;
        cout << "Nomor Polisi : " << P->data.nopol << endl;
        cout << "Warna        : " << P->data.warna << endl;
        cout << "Tahun        : " << P->data.tahun << endl <<
endl;
        P = P->next;
    }
}

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

    Kendaraan x;
    Node* P;

    for (int i = 1; i <= 4; i++) {
        cout << "Masukkan Nomor Polisi : ";
        cin >> x.nopol;
        cout << "Masukkan Warna        : ";
        cin >> x.warna;
        cout << "Masukkan Tahun        : ";
        cin >> x.tahun;

        if (cariData(L, x.nopol) != nullptr) {
            cout << "Nomor polisi sudah terdaftar!" << endl;
        } else {
            P = buatNode(x);
            tambahDepan(L, P);
        }
        cout << endl;
    }

    tampilData(L);

    string cari;
    cout << "\nCari Nomor Polisi : ";
    cin >> cari;

    Node* hasil = cariData(L, cari);
    if (hasil != nullptr) {
        cout << "\nData ditemukan:\n";
        cout << "Nomor Polisi : " << hasil->data.nopol << endl;
        cout << "Warna        : " << hasil->data.warna << endl;
```

```
        cout << "Tahun        : " << hasil->data.tahun << endl;
    } else {
        cout << "Data dengan nomor polisi " << cari << " tidak
ditemukan." << endl;
    }

    return 0;
}
```

b. Screenshot Output



c. Deskripsi Program

Program ini digunakan untuk mengelola data kendaraan menggunakan *Double Linked List*. Data kendaraan yang disimpan meliputi nomor polisi, warna, dan tahun kendaraan. Program memungkinkan pengguna untuk menambahkan data kendaraan, melakukan pencarian data berdasarkan nomor polisi, serta menampilkan seluruh data kendaraan yang tersimpan. Program juga melakukan pengecekan agar tidak terjadi data nomor polisi yang sama.

3. Unguide 3

a. Source Code

keloladata.cpp

```
#include <iostream>
#include <string>

using namespace std;

struct Kendaraan {
    string nopol;
    string warna;
    int tahun;
};
```

```cpp
struct Node {
    Kendaraan data;
    Node* next;
    Node* prev;
};

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

void buatList(List &L) {
    L.first = nullptr;
    L.last = nullptr;
}

Node* buatNode(Kendaraan x) {
    Node* P = new Node;
    P->data = x;
    P->next = nullptr;
    P->prev = nullptr;
    return P;
}

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

Node* cariNode(List L, string nopol) {
    Node* P = L.first;
    while (P != nullptr) {
        if (P->data.nopol == nopol) {
            return P;
        }
        P = P->next;
    }
    return nullptr;
}

void hapusDepan(List &L, Node* &P) {
```

```
    if (L.first != nullptr) {
        P = L.first;
        if (L.first == L.last) {
            L.first = nullptr;
            L.last = nullptr;
        } else {
            L.first = L.first->next;
            L.first->prev = nullptr;
            P->next = nullptr;
        }
    }
}

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

void hapusSetelah(Node* Prec, Node* &P) {
    if (Prec != nullptr && Prec->next != nullptr) {
        P = Prec->next;
        Prec->next = P->next;
        if (P->next != nullptr) {
            P->next->prev = Prec;
        }
        P->next = nullptr;
        P->prev = nullptr;
    }
}

void tampilList(List L) {
    Node* P = L.first;
    int i = 1;

    cout << endl << "Data Kendaraan ke-" << i << endl;
    while (P != nullptr) {
        cout << "Nomor Polisi : " << P->data.nopol << endl;
        cout << "Warna        : " << P->data.warna << endl;
```

```
        cout << "Tahun        : " << P->data.tahun << endl <<
endl;
        P = P->next;
    }
}

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

    Kendaraan x;
    Node* P;

    for (int i = 1; i <= 4; i++) {
        cout << "Masukkan Nomor Polisi : ";
        cin >> x.nopol;
        cout << "Masukkan Warna        : ";
        cin >> x.warna;
        cout << "Masukkan Tahun        : ";
        cin >> x.tahun;

        if (cariNode(L, x.nopol) != nullptr) {
            cout << "Nomor polisi sudah terdaftar!" << endl;
        } else {
            P = buatNode(x);
            tambahDepan(L, P);
        }
        cout << endl;
    }

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

    Node* hasil = cariNode(L, cari);
    if (hasil != nullptr) {
        cout << endl;
        cout << "Nomor Polisi : " << hasil->data.nopol << endl;
        cout << "Warna        : " << hasil->data.warna << endl;
        cout << "Tahun        : " << hasil->data.tahun << endl;
    } else {
        cout << "Data dengan nomor polisi " << cari << " tidak
ditemukan." << endl;
    }

    string hapus;
    cout << endl << "Masukkan Nomor Polisi yang mau dihapus :
";
```

```
    cin >> hapus;

    Node* target = cariNode(L, hapus);
    if (target != nullptr) {
        if (target == L.first) {
            hapusDepan(L, P);
        } else if (target == L.last) {
            hapusBelakang(L, P);
        } else {
            Node* Prec = target->prev;
            hapusSetelah(Prec, P);
        }
        cout << "Data dengan nomor polisi " << hapus << "
berhasil dihapus." << endl;
    } else {
        cout << "Data tidak ditemukan." << endl;
    }

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

b. Screenshot Output

```
Masukkan Tahun        : 1992

Masukkan Nomor Polisi : 9995
Masukkan Warna        : biru
Masukkan Tahun        : 1993

Masukkan Nomor Polisi : 9999
Masukkan Warna        : hijau
Masukkan Tahun        : 1998
Nomor polisi sudah terdaftar!

Masukkan Nomor Polisi yang ingin dicari : 9999

Nomor Polisi : 9999
Warna        : ungu
Tahun        : 1991

Masukkan Nomor Polisi yang mau dihapus : 9995
Data dengan nomor polisi 9995 berhasil dihapus.

Data Kendaraan ke-1
Nomor Polisi : 9997
Warna        : ungu
Tahun        : 1992

Nomor Polisi : 9999
Warna        : ungu
Tahun        : 1991
```

c. Deskripsi Program

Program ini dibuat untuk mengelola data kendaraan menggunakan struktur data *Double Linked List*. Data kendaraan yang disimpan meliputi nomor polisi, warna, dan tahun kendaraan. Program memungkinkan pengguna untuk menambahkan data kendaraan, mencari data berdasarkan nomor polisi, serta menghapus data kendaraan baik di bagian depan, belakang, maupun di tengah list. Selain itu, program juga melakukan pengecekan nomor polisi agar tidak terjadi data ganda. Setelah proses penghapusan, seluruh data kendaraan yang tersisa akan ditampilkan kembali.

D. Kesimpulan

Dari hasil pembuatan program, dapat disimpulkan bahwa Double Linked List dapat digunakan untuk menyimpan dan mengelola data kendaraan dengan baik. Program ini mampu melakukan penambahan, pencarian, dan penghapusan data kendaraan berdasarkan nomor polisi. Penggunaan pointer next dan prev memudahkan pengelolaan data, serta pengecekan nomor polisi membantu mencegah terjadinya data yang sama.

E. Referensi

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms (3rd ed.). MIT Press.

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

Weiss, M. A. (2014). Data structures and algorithm analysis in C++ (4th ed.). Pearson Education.

Sedgewick, R., & Wayne, K. (2011). Algorithms (4th ed.). Addison-Wesley.