LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 2
PENGENALAN BAHASA C++

Disusun Oleh :

NAMA : RAIHAN DZAKY MUFLIH

NIM : 103112430029

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Double Linked List adalah struktur data yang terdiri dari beberapa node yang saling terhubung dua arah yaitu ke node sebelumnya dan ke node berikutnya. Setiap node memiliki tiga bagian utama yaitu data, pointer ke node sebelumnya, dan pointer ke node berikutnya. Dengan dua arah ini proses pencarian, penambahan, dan penghapusan data bisa dilakukan lebih mudah, baik dari depan maupun dari belakang. Meskipun lebih fleksibel dibanding single linked list, double linked list membutuhkan lebih banyak memori karena setiap node harus menyimpan dua pointer.

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

Guided 1

```
#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_last(5);
    add_last(20);
    cout << "Awal\t\t\t\t\t : ";
    view();

    delete_first();
    cout << "Setelah delete_first\t : ";
    view();
    delete_last();
    cout << "Setelah delete_last\t\t : ";
    view();

    add_last(30);
    add_last(40);
    cout << "Setelah tambah\t\t\t : ";
    view();

    delete_target(30);
    cout << "Setelah delete_target\t : ";
    view();

    return 0;
}
```

Screenshots Output

```
[Running] cd "d:\C++\MODUL 5\" && g++ main.cpp -o main && "d:\C++\MODUL 5\"main
Awal                    : 10 <-> 5 <-> 20
Setelah delete_first    : 5 <-> 20
Setelah delete_last     : 5
Setelah tambah          : 5 <-> 30 <-> 40
Setelah delete_target   : 5 <-> 40
```

Deskripsi:
Program di atas adalah implementasi doubly linked list yang digunakan untuk menambah atau menghapus data di depan, di belakang, atau setelah data tertentu, serta menampilkan semua isi daftar. Setiap data disimpan dalam node yang saling terhubung ke depan dan ke belakang.

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

Unguided 1

#doublylist.h

```
#ifndef DOUBLYLIST_H
#define DOUBLYLIST_H

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

struct kendaraan {
    string nopol;
    string warna;
    int thnbuat;
};
typedef kendaraan infotype;

struct ElmList;
typedef ElmList* address;

struct ElmList {
    infotype info;
    address next;
    address prev;
};

struct List {
```

```
    address First;
    address Last;
};

void CreateList(List &L);
address alokasi(infotype x);
void dealokasi(address &P);
void insertLast(List &L, address P);
void printInfo(List L);
bool isExist(List L, string nopol);
address findElm(List L, string nopol);

void deleteFirst(List &L, address &P);
void deleteLast(List &L, address &P);
void deleteAfter(address Prec, address &P);

#endif
```

#doublylist.cpp

```
#include "doublylist.h"

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

address alokasi(infotype x) {
    address P = new ElmList;
    P->info = x;
    P->next = NULL;
    P->prev = NULL;
    return P;
}

void dealokasi(address &P) {
    delete P;
    P = NULL;
}
```

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

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

void insertLast(List &L, address P) {
    if (L.First == NULL) {
        L.First = P;
        L.Last = P;
    } else {
        L.Last->next = P;
        P->prev = L.Last;
        L.Last = P;
    }
}

void deleteFirst(List &L, address &P) {
    if (L.First == NULL) {
        P = NULL;
        return;
    }
    P = L.First;
    if (L.First == L.Last) {
        L.First = NULL;
        L.Last = NULL;
```

```
    } else {
        L.First = P->next;
        L.First->prev = NULL;
        P->next = NULL;
    }
}

void deleteLast(List &L, address &P) {
    if (L.Last == NULL) {
        P = NULL;
        return;
    }
    P = L.Last;
    if (L.First == L.Last) {
        L.First = NULL;
        L.Last = NULL;
    } else {
        L.Last = P->prev;
        L.Last->next = NULL;
        P->prev = NULL;
    }
}

void deleteAfter(address Prec, address &P) {
    if (Prec == NULL || Prec->next == NULL) {
        P = NULL;
        return;
    }
    P = Prec->next;
    Prec->next = P->next;
    if (P->next != NULL) {
        P->next->prev = Prec;
    }
    P->next = NULL;
    P->prev = NULL;
}

void printInfo(List L) {
    address P = L.First;
    cout << "\nDATA LIST:\n";
    while (P != NULL) {
        cout << "Nomor Polisi : " << P->info.nopol << endl;
        cout << "Warna        : " << P->info.warna << endl;
```

```
        cout << "Tahun         : " << P->info.thnbuat << endl;
        cout << "------------------------\n";
        P = P->next;
    }
}
```

#main.cpp

```
#include "doublylist.h"
#include "doublylist.cpp"

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

    infotype x;
    char pilih = 'y';


    while (pilih == 'y' || pilih == 'Y') {
        cout << "Masukkan Nomor Polisi : ";
        cin >> x.nopol;
        cout << "Masukkan Warna        : ";
        cin >> x.warna;
        cout << "Masukkan Tahun        : ";
        cin >> x.thnbuat;

        if (isExist(L, x.nopol)) {
            cout << "Nomor polisi sudah terdaftar!\n";
        } else {
            address P = alokasi(x);
            insertLast(L, P);
        }

        cout << "Tambah data lagi? (y/n): ";
        cin >> pilih;
        cout << endl;
    }
```

```
    printInfo(L);

    string cari;
    cout << "\nMasukkan Nomor Polisi yang dicari : ";
    cin >> cari;
    address found = findElm(L, cari);
    if (found != NULL) {
        cout << "\nNomor Polisi : " << found->info.nopol << endl;
        cout << "Warna        : " << found->info.warna << endl;
        cout << "Tahun        : " << found->info.thnbuat << endl;
    } else {
        cout << "Data tidak ditemukan!\n";
    }

    string hapus;
    cout << "\nMasukkan Nomor Polisi yang akan dihapus : ";
    cin >> hapus;

    address target = findElm(L, hapus);
    if (target == NULL) {
        cout << "Data tidak ditemukan.\n";
    } else if (target == L.First) {
        address P;
        deleteFirst(L, P);
        dealokasi(P);
        cout << "Data pertama berhasil dihapus.\n";
    } else if (target == L.Last) {
        address P;
        deleteLast(L, P);
        dealokasi(P);
        cout << "Data terakhir berhasil dihapus.\n";
    } else {
        address Prec = target->prev;
        address P;
        deleteAfter(Prec, P);
        dealokasi(P);
        cout << "Data dengan nomor polisi " << hapus << " berhasil
dihapus.\n";
    }

    cout << "\nDATA SETELAH PENGHAPUSAN:\n";
    printInfo(L);
```

```
    return 0;
}
```

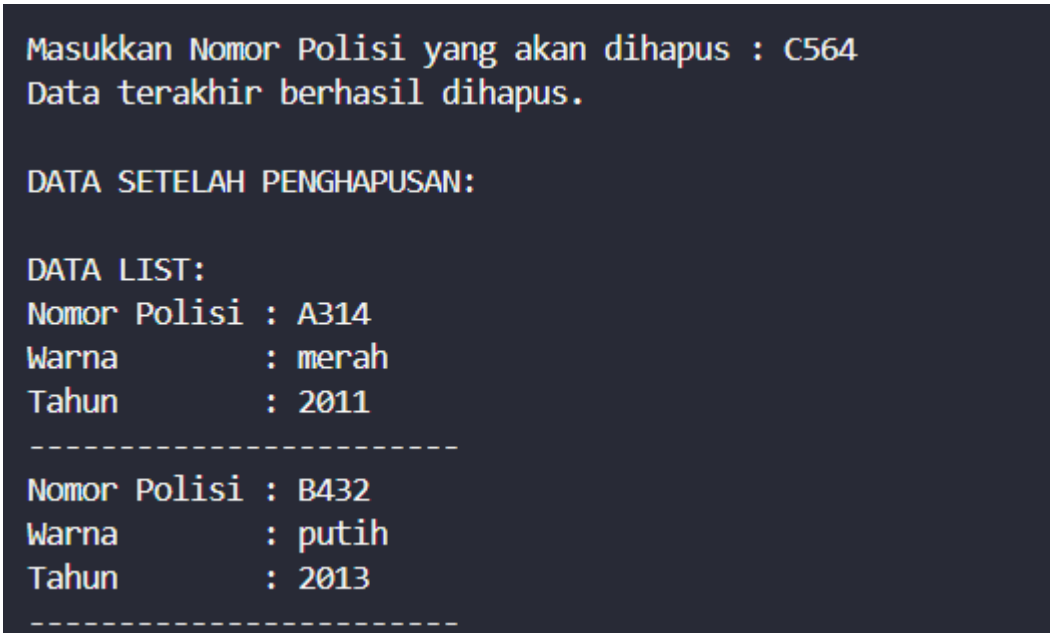
Screenshots Output 1

```
Masukkan nomor polisi : A314
Masukkan warna kendaraan : merah
Masukkan tahun kendaraan : 2011

Tambah data lagi? (y/n): y

Masukkan nomor polisi : B432
Masukkan warna kendaraan : putih
Masukkan tahun kendaraan : 2013

Tambah data lagi? (y/n): y

Masukkan nomor polisi : C564
Masukkan warna kendaraan : hitam
Masukkan tahun kendaraan : 2022

Tambah data lagi? (y/n): n


DATA LIST:
No Polisi : A314
Warna     : merah
Tahun     : 2011
-----------------------
No Polisi : B432
Warna     : putih
Tahun     : 2013
-----------------------
No Polisi : C564
Warna     : hitam
Tahun     : 2022
-----------------------
PS D:\C++\MODUL 5\UNGUIDED>
```

Screenshots Output 2

```
Masukkan Nomor Polisi yang dicari : A314

Nomor Polisi : A314
Warna        : merah
Tahun        : 2011
```

Screenshots Output 3



Deskripsi:
Program di atas adalah implementasi double linked list untuk menyimpan data kendaraan yang terdiri dari nomor polisi, warna, dan tahun pembuatan. Pengguna dapat menambahkan data kendaraan secara berulang, lalu program akan menampilkan seluruh data yang tersimpan. Selain itu terdapat fitur untuk mencari data berdasarkan nomor polisi, serta menghapus data tertentu dengan prosedur tergantung posisi data yang ingin dihapus. Setelah penghapusan program akan menampilkan kembali daftar kendaraan yang tersisa sebagai output.

D. Kesimpulan

Program ini menunjukkan bahwa double linked list dapat digunakan untuk mengelola data secara dinamis, di mana setiap elemen memiliki hubungan dua arah maju dan mundur. Dengan struktur ini proses penambahan, pencarian, dan penghapusan data menjadi lebih fleksibel dan efisien karena dapat dilakukan dari kedua arah. Selain itu program membantu memahami konsep dasar manipulasi pointer dan hubungan antar node dalam struktur data linked list secara praktis dan mudah.

E. Referensi

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

Kumar, S., & Singhal, M. (2004). A comparative performance analysis of single linked listand double linked list. International Journal of Computer Science and NetworkSecurity, 4(10), 248-254.

Lamhot Sitorus & David J.M. Sembiring, Konsep dan Implementasi Struktur Data dengan C++, Andi Offset, Yogyakarta