

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

MODUL 5



Disusun Oleh :

NAMA : IBTIDA ZADA UTOMO

NIM : 103112430037

Dosen

FAHRUDIN MUKTI WIBOWO

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

A. Dasar Teori

Struktur data linked list merupakan salah satu bentuk penyimpanan data dinamis yang terdiri atas node-node saling terhubung melalui pointer. Setiap node umumnya berisi dua komponen, yaitu data dan pointer ke node berikutnya. Pada doubly linked list (double linked list), setiap node memiliki dua pointer — satu menunjuk ke node sebelumnya (prev) dan satu menunjuk ke node berikutnya (next). Hal ini membuat proses traversal dapat dilakukan dua arah, baik dari depan ke belakang maupun sebaliknya.

Kelebihan dari doubly linked list dibandingkan singly linked list adalah kemudahan dalam melakukan operasi seperti penyisipan dan penghapusan elemen di tengah list, karena node memiliki informasi arah ganda. Struktur ini sangat berguna dalam berbagai aplikasi seperti sistem navigasi data (misalnya pada *browser history*), manajemen memori, serta sistem antrian yang memerlukan fleksibilitas tinggi.

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;
}

```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\struktur data\MODUL 5\GUIDED> cd "d:\struktur data\MODUL 5\GUIDED\" ; if ($?) { g++ tempCode
nerFile }
Awal                                : 10 <-> 5 <-> 20
Setelah delete_first      : 5 <-> 20
Setelah delete_last       : 5
Setelah tambah            : 5 <-> 30 <-> 40
Setelah delete_target     : 5 <-> 40
PS D:\struktur data\MODUL 5\GUIDED>
```

Deskripsi: program diatas adalah program untuk implementasi double linked list menggunakan bahasa c++ dan memiliki 2 pointer yaitu prev yaitu untuk menunjuk ke node sebelumnya, dan ada next untuk menunjuk ke node sesudahnya

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

Unguided 1

doublilyst.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;
};

// Prosedur dan Fungsi
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);

#endif
```

doublilyst.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;
}

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 printInfo(List L) {
    address P = L.First;
    cout << "\nDATA LIST:\n";
    while (P != NULL) {
        cout << "No 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 kendaraan : ";
cin >> x.warna;
cout << "Masukkan tahun kendaraan : ";
cin >> x.thnbuat;

if (isExist(L, x.nopol)) {
    cout << "Nomor polisi sudah terdaftar!\n";
} else {
    address P = alokasi(x);
    insertLast(L, P);
}
cout << "\nTambah data lagi? (y/n): ";
cin >> pilih;
cout << endl;
}
printInfo(L);
return 0;
}
```

Screenshots Output

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\struktur data\MODUL 5\UNGUIDED> cd "d:\struktur data\MODUL 5\UNGUIDED"
Masukkan nomor polisi : D004
Masukkan warna kendaraan : kuning
Masukkan tahun kendaraan : 90

Tambah data lagi? (y/n): y

Masukkan nomor polisi : D003
Masukkan warna kendaraan : putih
Masukkan tahun kendaraan : 70

Tambah data lagi? (y/n): y

Masukkan nomor polisi : D001
Masukkan warna kendaraan : hitam
Masukkan tahun kendaraan : 90

Tambah data lagi? (y/n): n

DATA LIST:
No Polisi : D004
Warna      : kuning
Tahun      : 90
-----
No Polisi : D003
Warna      : putih
Tahun      : 70
-----
No Polisi : D001
Warna      : hitam
Tahun      : 90
-----
PS D:\struktur data\MODUL 5\UNGUIDED> 
```

Deskripsi: program diatas adalah program yang digunakan untuk menambahkan data pada double linked list jadi nanti disuruh memasukan data berupa nomor polisi lalu warna kendaraan dan yang terakhir itu tahun kendaraan nantinya program akan meyimpan semua data di double linked list

Nomer 2

list.cpp

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

struct infotype {
    string nopol;
    string warna;
    int tahun;
```

```

};

struct ElmList {
    infotype info;
    ElmList* next;
};

typedef ElmList* address;
typedef address List;

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

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

void insertLast(List &L, address P) {
    if (L == NULL) {
        L = P;
    } else {
        address Q = L;
        while (Q->next != NULL) {
            Q = Q->next;
        }
        Q->next = P;
    }
}

void printInfo(List L) {
    address P = L;
    while (P != NULL) {
        cout << "Nomor Polisi : " << P->info.nopol << endl;
        cout << "Warna          : " << P->info.warna << endl;
        cout << "Tahun            : " << P->info.tahun << endl;
        cout << "-----" << endl;
        P = P->next;
    }
}

```

```

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

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

    infotype a = {"D001", "Hitam", 90};
    infotype b = {"D002", "Merah", 92};
    infotype c = {"D003", "Biru", 93};

    insertLast(L, alokasi(a));
    insertLast(L, alokasi(b));
    insertLast(L, alokasi(c));

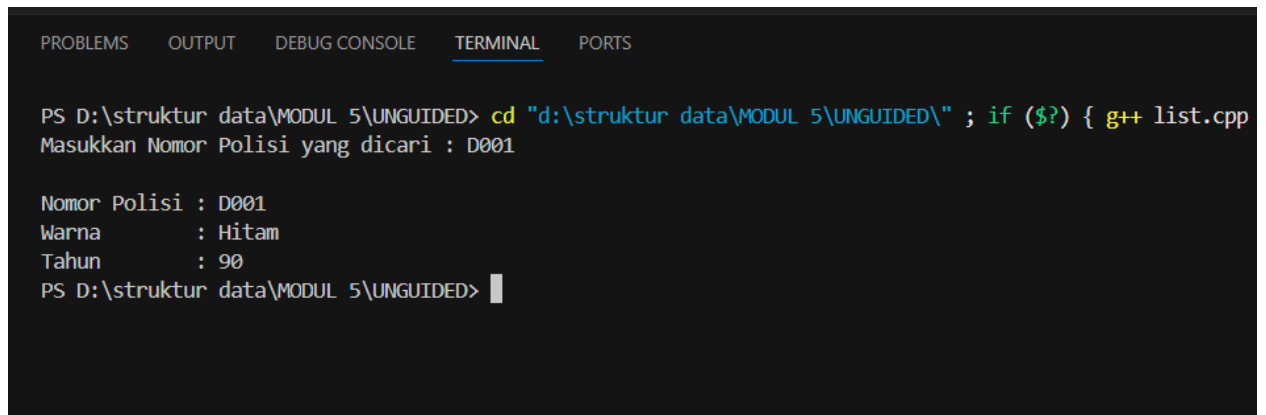
    string cari;
    cout << "Masukkan Nomor Polisi yang dicari : ";
    cin >> cari;

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

    return 0;
}

```

Screenshots Output



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\struktur data\MODUL 5\UNGUIDED> cd "d:\struktur data\MODUL 5\UNGUIDED\" ; if ($?) { g++ list.cpp
Masukkan Nomor Polisi yang dicari : D001

Nomor Polisi : D001
Warna       : Hitam
Tahun      : 90
PS D:\struktur data\MODUL 5\UNGUIDED> 
```

Deskripsi:

program diatas adalah program untuk mencari data di double linked list nanti program akan meminta kita memasukan nomor polisi yang tadi di program pertama, dan program akan mencari di double linked list dan memunculkan pada output

unguided 3

delete.cpp

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

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

struct ElmList {
    infotype info;
    ElmList* next;
};
```

```

typedef ElmList* address;
typedef address List;

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

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

void insertLast(List &L, address P) {
    if (L == NULL) {
        L = P;
    } else {
        address Q = L;
        while (Q->next != NULL) {
            Q = Q->next;
        }
        Q->next = P;
    }
}

void printInfo(List L) {
    address P = L;
    while (P != NULL) {
        cout << "Nomor Polisi : " << P->info.nopol << endl;
        cout << "Warna          : " << P->info.warna << endl;
        cout << "Tahun            : " << P->info.tahun << endl;
        cout << "-----" << endl;
        P = P->next;
    }
}

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

```

```

        }
        P = P->next;
    }
    return NULL;
}

void deleteFirst(List &L, address &P) {
    if (L != NULL) {
        P = L;
        L = L->next;
        P->next = NULL;
    } else {
        P = NULL;
    }
}

void deleteLast(List &L, address &P) {
    if (L != NULL) {
        if (L->next == NULL) {
            P = L;
            L = NULL;
        } else {
            address Q = L;
            while (Q->next->next != NULL) {
                Q = Q->next;
            }
            P = Q->next;
            Q->next = NULL;
        }
    } else {
        P = NULL;
    }
}

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

```



```

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

    infotype a = {"D004", "Kuning", 99};
    infotype b = {"D003", "Hijau", 91};
    infotype c = {"D002", "Hitam", 90};
    infotype d = {"D001", "Putih", 90};

    insertLast(L, alokasi(a));
    insertLast(L, alokasi(b));
    insertLast(L, alokasi(c));
    insertLast(L, alokasi(d));

    string cari;
    cout << "Masukkan Nomor Polisi yang akan dihapus : ";
    cin >> cari;

    address P = findElm(L, cari);
    if (P == NULL) {
        cout << "Data dengan nomor polisi " << cari << " tidak
ditemukan." << endl;
    } else {
        if (P == L) {
            deleteFirst(L, P);
        } else if (P->next == NULL) {
            deleteLast(L, P);
        } else {
            address Prec = L;
            while (Prec->next != P) {
                Prec = Prec->next;
            }
            deleteAfter(Prec, P);
        }
        cout << "Data dengan nomor polisi " << cari << " berhasil
dihapus." << endl;
    }

    cout << endl << "DATA LIST :" << endl;
    printInfo(L);

    return 0;
}

```



Screenshots Output

```
PS D:\struktur data\MODUL 5\UNGUIDED> cd "d:\struktur data\MODUL 5\UNGUIDED"
Masukkan Nomor Polisi yang akan dihapus : D004
Data dengan nomor polisi D004 berhasil dihapus.

DATA LIST :
Nomor Polisi : D003
Warna       : Hijau
Tahun       : 91
-----
Nomor Polisi : D002
Warna       : Hitam
Tahun       : 90
-----
Nomor Polisi : D001
Warna       : Putih
Tahun       : 90
-----
PS D:\struktur data\MODUL 5\UNGUIDED> cd "d:\struktur data\MODUL 5\UNGUIDED"
Masukkan Nomor Polisi yang akan dihapus : D001
Data dengan nomor polisi D001 berhasil dihapus.

DATA LIST :
Nomor Polisi : D004
Warna       : Kuning
Tahun       : 99
-----
Nomor Polisi : D003
Warna       : Hijau
Tahun       : 91
-----
Nomor Polisi : D002
Warna       : Hitam
Tahun       : 90
-----
PS D:\struktur data\MODUL 5\UNGUIDED> |
```

Deskripsi: program diatas adalah program untuk menghapus data pada double linked list nanti nya program akan meminta data yang akan dihapus berupa nomor polisi terus program menghapus data yang ada di double linked list, dan program diatas juga menggunakan procedur delete yaitu ada delete first, procedure delete last, dan procedur delete after

E. Kesimpulan

Dari praktikum ini dapat disimpulkan bahwa doubly linked list merupakan pengembangan dari singly linked list yang memungkinkan pergerakan dua arah melalui pointer prev dan next. Implementasi struktur ini di C++ mempermudah operasi manipulasi data seperti penambahan, penghapusan, dan pencarian elemen tanpa harus melakukan pergeseran data secara manual sebagaimana pada array.

Selain itu, penggunaan prosedur seperti insertLast, deleteFirst, dan findElm menunjukkan bagaimana pengelolaan memori dan pointer berperan penting dalam menjaga keutuhan hubungan antar node. Dengan memahami konsep dasar dan implementasinya, mahasiswa dapat membangun sistem penyimpanan data yang lebih fleksibel, efisien, dan terstruktur.

F. Referensi

- <https://www.geeksforgeeks.org/cpp/doubly-linked-list-in-cpp/>
- <https://www.programiz.com/dsa/doubly-linked-list>
- <https://www.w3schools.com/cpp/default.asp>