LAPORAN PRAKTIKUM

STRUKTUR DATA

MODUL VI

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

Struktur data adalah metode untuk menyusun dan menyimpan informasi di dalam sistem komputer agar bisa diambil dan dikelola dengan efektif. Memilih struktur data yang sesuai sangat memengaruhi performa aplikasi, khususnya terkait konsumsi memori dan waktu pemrosesan informasi.

Salah satu jenis struktur data linear adalah *Linked List,* yang terdiri dari serangkaian node yang saling terhubung lewat pointer. *Linked List* dibedakan menjadi beberapa varian, termasuk *Doubly Linked List. Doubly Linked List* merupakan struktur di mana tiap node dilengkapi dengan dua pointer: next yang mengarah ke node selanjutnya dan prev yang mengarah ke node sebelumnya. Berkat kedua pointer ini, navigasi data bisa dilakukan ke dua arah, yaitu dari depan ke belakang atau sebaliknya.

Dibandingkan *Single Linked List, Doubly Linked List* unggul dalam operasi penghapusan dan pencarian data karena tidak perlu melacak node sebelumnya secara manual. Meski begitu, struktur ini memerlukan ruang memori ekstra akibat tambahan pointer prev di setiap node.

B. Guide

1. Guide 1

a. Source Code

dll.cpp

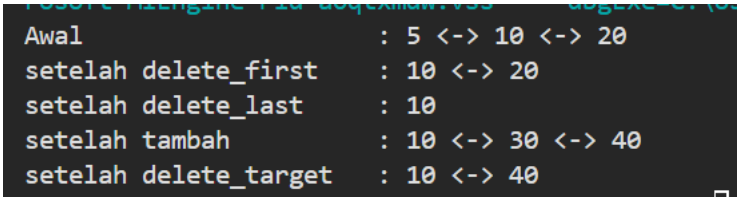
```
#include <iostream>
using namespace std;

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



c. Deskripsi Program

a) Program dll.cpp merupakan implementasi struktur data *Doubly Linked List* menggunakan bahasa pemrograman C++. Setiap node memiliki dua pointer, yaitu `prev` untuk menunjuk node sebelumnya dan `next` untuk menunjuk node berikutnya, sehingga data dapat ditelusuri secara dua arah.

b) Struktur `Node` digunakan untuk menyimpan data bertipe integer serta pointer `prev` dan `next`. Program juga menggunakan dua pointer global, yaitu `ptr_first` sebagai penunjuk node pertama dan `ptr_last` sebagai penunjuk node terakhir dalam linked list.

c) Fungsi `add_first()` dan `add_last()` berfungsi untuk menambahkan node baru masing-masing di awal dan di akhir list. Selain itu, fungsi `add_target()` digunakan untuk menambahkan node baru setelah node dengan nilai tertentu.

d) Fungsi `view()` digunakan untuk menampilkan seluruh isi *doubly linked list* dari node pertama hingga node terakhir dengan format penunjuk dua arah.

e) Operasi penghapusan data dilakukan melalui beberapa fungsi, yaitu `delete_first()` untuk menghapus node pertama, `delete_last()` untuk menghapus node terakhir, dan `delete_target()` untuk menghapus node berdasarkan nilai tertentu.

f) Program juga menyediakan fungsi `edit_node()` yang berfungsi untuk mengubah nilai data pada node tertentu tanpa mengubah struktur *linked list.*

g) Pada fungsi `main()`, dilakukan pengujian terhadap seluruh operasi *doubly linked list*, seperti penambahan data, penghapusan data, serta penampilan isi list untuk memastikan program berjalan dengan benar.

C. Unguide

1. Unguide 1

a. Source Code

doublylist.h

```
#ifndef DOUBLYLIST_H
#define DOUBLYLIST_H

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

```
typedef struct kendaraan {
    string nopol;
    string warna;
    int thnBuat;
} infotype;

typedef struct ElmList *address;

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

typedef struct {
    address First;
    address Last;
} List;

void CreateList(List &L);
address alokasi(infotype x);
void dealokasi(address &P);
void printInfo(List L);
void insertLast(List &L, 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;
}
```

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

void printInfo(List L) {
    address P = L.Last;
    while (P != NULL) {
        cout << "no polisi : " << P->info.nopol << endl;
        cout << "warna     : " << P->info.warna << endl;
        cout << "tahun     : " << P->info.thnBuat << endl;
        P = P->prev;
    }
}
```

main.cpp

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

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

    infotype x;
    address P;
    int n = 4;

    for (int i = 1; i <= n; i++) {
        cout << "masukkan nomor polisi: ";
        cin >> x.nopol;
        cout << "masukkan warna kendaraan: ";
        cin >> x.warna;
        cout << "masukkan tahun kendaraan: ";
        cin >> x.thnBuat;

        bool ada = false;
        address Q = L.First;
        while (Q != NULL) {
            if (Q->info.nopol == x.nopol) {
                ada = true;
                break;
            }
```

```
            Q = Q->next;
        }

        if (ada) {
            cout << "nomor polisi sudah terdaftar" << endl <<
endl;
        } else {
            P = alokasi(x);
            insertLast(L, P);
            cout << endl;
        }
    }

    cout << "DATA LIST 1" << endl;
    printInfo(L);

    return 0;
}
```

b. Screenshot Output

```
masukkan nomor polisi: E003
masukkan warna kendaraan: ungu
masukkan tahun kendaraan: 1991

masukkan nomor polisi: E006
masukkan warna kendaraan: hitam
masukkan tahun kendaraan: 2005

masukkan nomor polisi: H889
masukkan warna kendaraan: hijau
masukkan tahun kendaraan: 2005

masukkan nomor polisi: E007
masukkan warna kendaraan: pink
masukkan tahun kendaraan: 1991

DATA LIST 1
no polisi : E007
warna     : pink
tahun     : 1991
no polisi : H889
warna     : hijau
tahun     : 2005
no polisi : E006
warna     : hitam
tahun     : 2005
no polisi : E003
warna     : ungu
tahun     : 1991
```

c. Deskripsi Program

Program ini merupakan implementasi *Abstract Data Type (ADT) Doubly Linked List* menggunakan bahasa pemrograman C++. ADT ini digunakan untuk menyimpan dan mengelola data kendaraan, di mana setiap data kendaraan memiliki atribut nomor polisi, warna, dan tahun pembuatan.

Struktur data *Doubly Linked List* memungkinkan setiap elemen (node) memiliki dua buah pointer, yaitu next yang menunjuk ke elemen berikutnya dan prev yang menunjuk ke elemen sebelumnya. Dengan struktur ini, proses penelusuran data dapat dilakukan dari depan maupun dari belakang.

Program dibagi ke dalam tiga file utama, yaitu:

a) doublylist.h

Berisi deklarasi struktur data (ADT) yang meliputi tipe data infotype, ElmList, dan List, serta deklarasi fungsi-fungsi dasar seperti pembuatan list, alokasi dan dealokasi memori, penambahan data di akhir list, dan pencetakan isi list.

b) doublylist.cpp

Berisi implementasi dari seluruh fungsi yang telah dideklarasikan pada file header, antara lain:

- CreateList untuk menginisialisasi list kosong
- alokasi untuk membuat node baru
- insertLast untuk menambahkan data kendaraan di bagian akhir list
- printInfo untuk menampilkan data kendaraan dari elemen terakhir ke elemen pertama
- dealokasi untuk membebaskan memori

c) main.cpp

Berfungsi sebagai program utama yang menangani proses input data kendaraan dari pengguna. Program akan menerima beberapa data kendaraan, melakukan pengecekan agar nomor polisi tidak duplikat, kemudian menyimpan data ke dalam *Doubly Linked List.* Setelah seluruh data dimasukkan, program akan menampilkan seluruh isi list.

2. Unguide 2

a. Source Code

doublylist.h

```
#ifndef DOUBLYLIST_H
#define DOUBLYLIST_H

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

```
typedef struct kendaraan {
    string nopol;
    string warna;
    int thnBuat;
} infotype;

typedef struct ElmList *address;

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

typedef struct {
    address First;
    address Last;
} List;

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

#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;
}

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

void printInfo(List L) {
    address P = L.Last;
    while (P != NULL) {
        cout << "no polisi : " << P->info.nopol << endl;
        cout << "warna     : " << P->info.warna << endl;
        cout << "tahun     : " << P->info.thnBuat << endl;
        P = P->prev;
    }
}

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

main.cpp

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

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

    infotype x;
    address P;
    int n = 4;

    for (int i = 1; i <= n; i++) {
```

```
        cout << "masukkan nomor polisi: ";
        cin >> x.nopol;
        cout << "masukkan warna kendaraan: ";
        cin >> x.warna;
        cout << "masukkan tahun kendaraan: ";
        cin >> x.thnBuat;

        bool ada = false;
        address Q = L.First;
        while (Q != NULL) {
            if (Q->info.nopol == x.nopol) {
                ada = true;
                break;
            }
            Q = Q->next;
        }

        if (ada) {
            cout << "nomor polisi sudah terdaftar\n\n";
        } else {
            P = alokasi(x);
            insertLast(L, P);
            cout << endl;
        }
    }

    cout << "DATA LIST 1\n";
    printInfo(L);

    infotype cari;
    address hasil;

    cout << "\nMasukkan Nomor Polisi yang dicari : ";
    cin >> cari.nopol;

    hasil = findElm(L, cari);

    if (hasil != NULL) {
        cout << "\nNomor Polisi : " << hasil->info.nopol <<
endl;
        cout << "Warna        : " << hasil->info.warna << endl;
        cout << "Tahun        : " << hasil->info.thnBuat <<
endl;
    } else {
        cout << "\nNomor polisi tidak ditemukan" << endl;
    }

    return 0;
```

```
}
```

b. Screenshot Output

```
DATA LIST 1
no polisi : D007
warna     : biru
tahun     : 97
no polisi : D009
warna     : ungu
tahun     : 99
no polisi : D003
warna     : pink
tahun     : 98
no polisi : D001
warna     : hitam
tahun     : 90

Masukkan Nomor Polisi yang dicari : D001

Nomor Polisi : D001
Warna        : hitam
Tahun        : 90
```

c. Deskripsi Program

Program ini merupakan implementasi *ADT Doubly Linked List* menggunakan bahasa pemrograman C++ untuk menyimpan data kendaraan yang terdiri dari nomor polisi, warna, dan tahun pembuatan. Setiap node memiliki dua pointer, yaitu next dan prev, sehingga data dapat ditelusuri dari depan maupun dari belakang.

Program menyediakan fitur penambahan data kendaraan di akhir list, penampilan seluruh data, serta pencarian data kendaraan berdasarkan nomor polisi. Sebelum data dimasukkan, program melakukan pengecekan agar nomor polisi tidak terduplikasi. Implementasi ini menunjukkan penggunaan struktur data dinamis dan manajemen memori pada bahasa C++.

Unguide 3

a. Source Code

doublylist.h

```
#ifndef DOUBLYLIST_H
#define DOUBLYLIST_H

#include <iostream>
using namespace std;

typedef struct kendaraan {
    string nopol;
```

```
    string warna;
    int thnBuat;
} infotype;

typedef struct ElmList *address;

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

typedef struct {
    address First;
    address Last;
} List;

void CreateList(List &L);
address alokasi(infotype x);
void dealokasi(address &P);

void insertLast(List &L, address P);
void printInfo(List L);

address findElm(List L, infotype x);

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;
}

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

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

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

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

```
        dealokasi(P);
    }
}

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

void deleteAfter(address Prec, address &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;
        dealokasi(P);
    }
}
```

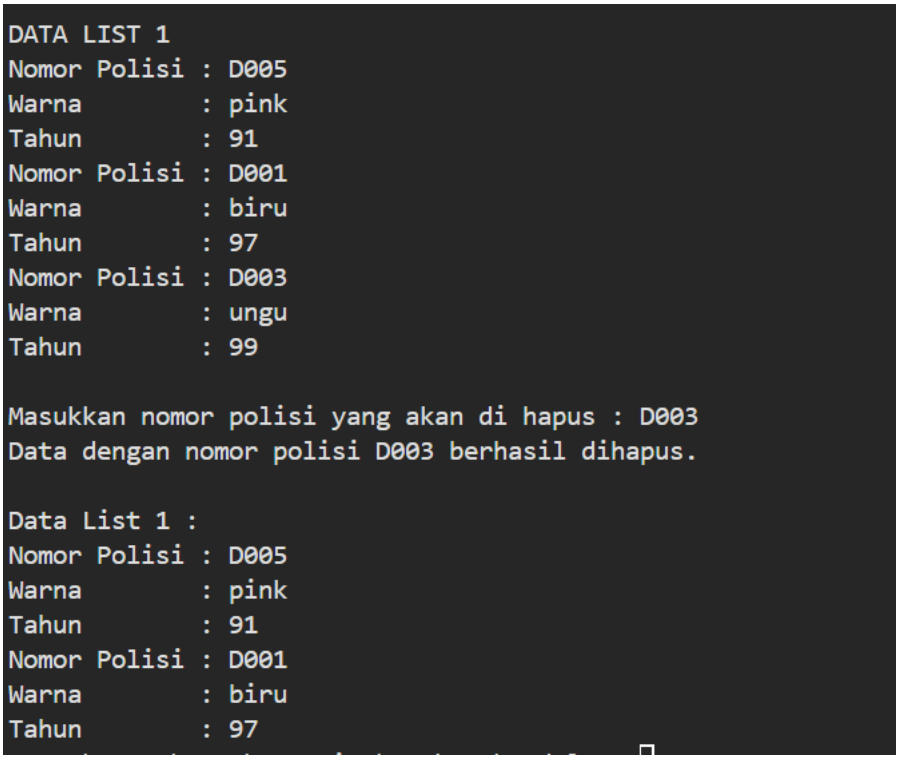
main.cpp

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

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

    infotype x;
    address P;
    int n = 4;

    for (int i = 1; i <= n; i++) {
        cout << "masukkan nomor polisi: ";
        cin >> x.nopol;
        cout << "masukkan warna kendaraan: ";
```

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

        bool ada = false;
        address Q = L.First;
        while (Q != NULL) {
            if (Q->info.nopol == x.nopol) {
                ada = true;
                break;
            }
            Q = Q->next;
        }

        if (ada) {
            cout << "nomor polisi sudah terdaftar\n\n";
        } else {
            insertLast(L, alokasi(x));
            cout << endl;
        }
    }

    cout << "DATA LIST 1\n";
    printInfo(L);

    infotype hapus;
    address target, prec;

    cout << "\nMasukkan nomor polisi yang akan di hapus : ";
    cin >> hapus.nopol;

    target = findElm(L, hapus);

    if (target != NULL) {
        if (target == L.First) {
            deleteFirst(L, P);
        } else if (target == L.Last) {
            deleteLast(L, P);
        } else {
            prec = target->prev;
            deleteAfter(prec, P);
        }

        cout << "Data dengan nomor polisi " << hapus.nopol
             << " berhasil dihapus.\n";
    } else {
        cout << "Data tidak ditemukan\n";
    }
```

```
    cout << "\nData List 1 :\n";
    printInfo(L);

    return 0;
}
```

b. Screenshot Output



c. Deskripsi Program

Program ini merupakan implementasi *ADT Doubly Linked List* menggunakan bahasa pemrograman C++ untuk mengelola data kendaraan yang terdiri dari nomor polisi, warna, dan tahun pembuatan. Setiap node memiliki dua pointer (next dan prev) sehingga memungkinkan penelusuran data dari depan maupun dari belakang.

Program menyediakan fitur penambahan data di akhir list, penampilan seluruh data, pencarian data berdasarkan nomor polisi, serta penghapusan data menggunakan prosedur deleteFirst, deleteLast, dan deleteAfter. Sebelum data dimasukkan, program memastikan nomor polisi tidak terduplikasi. Implementasi ini menunjukkan penggunaan struktur data dinamis, pointer, dan manajemen memori pada bahasa C++.

D. Kesimpulan

Berdasarkan hasil sesi praktikum yang telah dilaksanakan, dapat ditarik kesimpulan bahwa struktur informasi *Doubly Linked List* memungkinkan pengaturan data secara fleksibel dengan navigasi dua arah melalui pointer next dan prev. Struktur

ini mempermudah operasi penyisipan, pencarian, dan penghilangan data jika dibandingkan dengan jenis struktur linear lainnya.

Penerapan program Doubly Linked List dengan bahasa pemrograman C++ dalam praktikum ini berhasil menangani data kendaraan secara efektif, mulai dari penyisipan data di bagian akhir daftar, pencarian berdasarkan plat nomor, sampai penghilangan data melalui fungsi deleteFirst, deleteLast, dan deleteAfter. Lebih lanjut, verifikasi terhadap duplikasi plat nomor memastikan bahwa informasi yang tersimpan tetap akurat dan tidak berulang.

Dengan praktikum ini, mahasiswa dapat memahami aplikasi konsep struktur data yang dinamis, penggunaan pointer, serta pengelolaan memori, yang krusial dalam pembuatan aplikasi perangkat lunak dan pemrograman tingkat lanjut.

E. Referensi

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. Introduction to Algorithms. MIT Press.

GeeksforGeeks. Doubly Linked List Data Structure. https://www.geeksforgeeks.org/doubly-linked-list/

Stroustrup, B. *The C++ Programming Language*. Addison-Wesley.