

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL III
SINGLE AND DOUBLE LINKED LIST**



Disusun Oleh :

NAMA : ANISA YASAROH

NIM : 2311102063

Dosen :

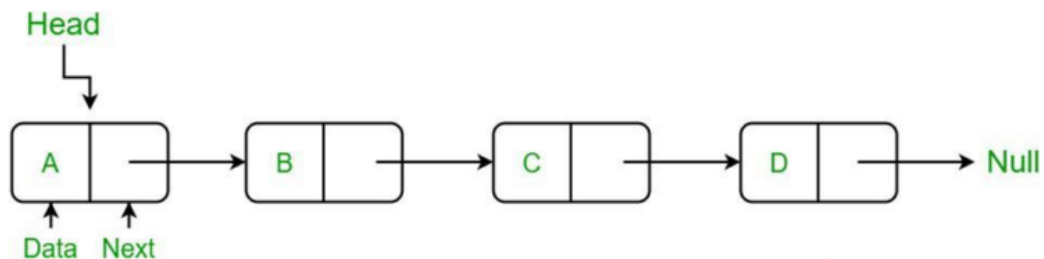
WAHYU ANDI SAPUTRA, S.Pd., M.Eng.

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

A. DASAR TEORI

1. Singke Linked List

Single linked list merupakan jenis linked list yang bersifat searah, yakni dapat dilintasi hanya dalam satu arah dari head hingga node terakhir. Setiap elemen dalam daftar tertaut disebut node. Sebuah node berisi data dan sebuah pointer ke node berikutnya, yang membantu menjaga struktur list. Node terakhir menunjuk ke nullptr, yang membantu kita menentukan kapan list berakhir. Dengan menggunakan struktur seperti ini, linked list dibentuk dengan cara menunjuk pointer next suatu elemen ke elemen yang mengikutinya. Pointer next pada elemen terakhir merupakan NULL, yang menunjukkan akhir dari suatu list. Elemen pada awal suatu list disebut head dan elemen terakhir dari suatu list disebut tail.

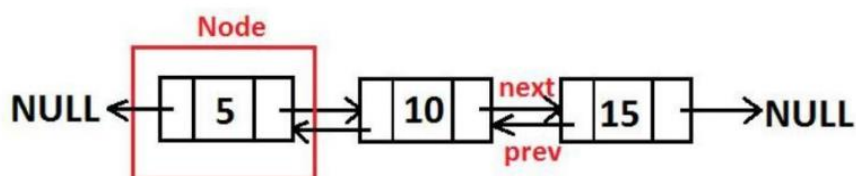


Single linked list efisien dalam penggunaan memori karena hanya memerlukan satu pointer untuk setiap simpul, memungkinkan operasi penambahan, penghapusan, pencarian, dan pengambilan nilai secara efektif.

2. Double Linked List

Double linked list mirip dengan single linked list, tetapi memiliki tambahan pointer prev pada setiap simpul yang menunjuk ke simpul sebelumnya. Dengan pointer prev ini, double linked list memungkinkan operasi penambahan dan penghapusan pada simpul mana saja dengan efisiensi. Tiap simpul memiliki elemen data, pointer next ke simpul berikutnya, dan pointer prev ke simpul sebelumnya. Kelebihan double linked list adalah kemampuannya dalam operasi penambahan dan penghapusan yang efisien dimana saja, serta kemudahan traversal dari depan dan belakang. Namun, kekurangannya adalah penggunaan memori lebih besar karena setiap simpul memerlukan satu pointer tambahan. Selain itu, operasi penambahan dan penghapusan dapat memakan waktu lebih lama dibandingkan dengan single linked list.

Representasi sebuah double linked list dapat dilihat pada gambar berikut ini :



Dalam linked list, terdapat dua pointer penting yaitu HEAD yang menunjuk pada node pertama, dan TAIL yang menunjuk pada node terakhir. Linked list kosong jika nilai pointer HEAD adalah NULL. Pointer prev dari HEAD selalu NULL karena itu merupakan node pertama, dan pointer next dari TAIL selalu NULL karena itu menandakan akhir dari data.

B. GUIDED

1. Guided 1 – Latihan Single Linked List

Source Code

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

// Deklarasi Struct Node
struct Node {
    int data;
    Node* next;
};

Node* head;
Node* tail;

// Inisialisasi Node
void init() {
    head = NULL;
    tail = NULL;
}

// Pengecekan apakah list kosong
bool isEmpty() {
    return head == NULL;
}

// Tambah Node di depan
void insertDepan(int nilai) {
    Node* baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty()) {
        head = tail = baru;
    } else {
        baru->next = head;
        head = baru;
    }
}

// Tambah Node di belakang
void insertBelakang(int nilai) {
    Node* baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty()) {
        head = tail = baru;
    } else {
```

```

        tail->next = baru;
        tail = baru;
    }
}

// Hitung jumlah Node di list
int hitungList() {
    Node* hitung = head;
    int jumlah = 0;
    while (hitung != NULL) {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

// Tambah Node di posisi tengah
void insertTengah(int data, int posisi) {
    if (posisi < 1 || posisi > hitungList()) {
        cout << "Posisi diluar jangkauan" << endl;
    } else if (posisi == 1) {
        cout << "Posisi bukan posisi tengah" << endl;
    } else {
        Node* baru = new Node();
        baru->data = data;
        Node* bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1) {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

// Hapus Node di depan
void hapusDepan() {
    if (!isEmpty()) {
        Node* hapus = head;
        if (head->next != NULL) {
            head = head->next;
            delete hapus;
        } else {
            head = tail = NULL;
            delete hapus;
        }
    } else {

```

```

        cout << "List kosong!" << endl;
    }
}

// Hapus Node di belakang
void hapusBelakang() {
    if (!isEmpty()) {
        if (head != tail) {
            Node* hapus = tail;
            Node* bantu = head;
            while (bantu->next != tail) {
                bantu = bantu->next;
            }
            tail = bantu;
            tail->next = NULL;
            delete hapus;
        } else {
            head = tail = NULL;
        }
    } else {
        cout << "List kosong!" << endl;
    }
}

// Hapus Node di posisi tengah
void hapusTengah(int posisi) {
    if (posisi < 1 || posisi > hitungList()) {
        cout << "Posisi diluar jangkauan" << endl;
    } else if (posisi == 1) {
        cout << "Posisi bukan posisi tengah" << endl;
    } else {
        Node* hapus;
        Node* bantu = head;
        for (int nomor = 1; nomor < posisi - 1; nomor++) {
            bantu = bantu->next;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    }
}

// Ubah data Node di depan
void ubahDepan(int data) {
    if (!isEmpty()) {
        head->data = data;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

```

```

    }
}

// Ubah data Node di posisi tengah
void ubahTengah(int data, int posisi) {
    if (!isEmpty()) {
        if (posisi < 1 || posisi > hitungList()) {
            cout << "Posisi di luar jangkauan" << endl;
        } else if (posisi == 1) {
            cout << "Posisi bukan posisi tengah" << endl;
        } else {
            Node* bantu = head;
            for (int nomor = 1; nomor < posisi; nomor++) {
                bantu = bantu->next;
            }
            bantu->data = data;
        }
    } else {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah data Node di belakang
void ubahBelakang(int data) {
    if (!isEmpty()) {
        tail->data = data;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

// Hapus semua Node di list
void clearList() {
    Node* bantu = head;
    while (bantu != NULL) {
        Node* hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}

// Tampilkan semua data Node di list
void tampil() {
    if (!isEmpty()) {
        Node* bantu = head;
        while (bantu != NULL) {

```

```

        cout << bantu->data << " ";
        bantu = bantu->next;
    }
    cout << endl;
} else {
    cout << "List masih kosong!" << endl;
}
}

int main() {
    init();
    insertDepan(3); tampil();
    insertBelakang(5); tampil();
    insertDepan(2); tampil();
    insertDepan(1); tampil();
    hapusDepan(); tampil();
    hapusBelakang(); tampil();
    insertTengah(7, 2); tampil();
    hapusTengah(2); tampil();
    ubahDepan(1); tampil();
    ubahBelakang(8); tampil();
    ubahTengah(11, 2); tampil();
    return 0;
}

```

Screenshoot Output

The screenshot shows a terminal window on the left and a Notepad++ window on the right. The terminal window displays the output of the program, which is a linked list of numbers: 3, 3 5, 2 3 5, 1 2 3 5, 2 3 5, 2 3, 2 7 3, 2 3, 1 3, 1 8, 1 11. The Notepad++ window shows the input data: Nama : ANISA YASAROH, NIM : 2311102063.

```

3
3 5
2 3 5
1 2 3 5
2 3 5
2 3
2 7 3
2 3
1 3
1 8
1 11
PS D:\Struktur Data smt 2>

```

File Edit View
 Nama : ANISA YASAROH
 NIM : 2311102063
 Ln 2, Col 18 | 38 characters | 100% | Window UTF-8

Deskripsi Program

Program diatas yaitu implementasi sederhana dari operasi-operasi dasar pada linked list, seperti penambahan, penghapusan, dan pengubahan nilai pada node. Struktur data linked list diwakili oleh sebuah struct node yang memiliki dua variabel yaitu data (nilai integer) dan pointer next yang menunjuk ke node selanjutnya. Program juga memiliki fungsi-fungsi untuk menginisialisasi linked list, mengecek apakah linked list kosong, menghitung jumlah node, serta menampilkan dan membersihkan isi linked list. Fungsi-

fungsi tersebut kemudian digunakan dalam main function untuk menguji operasi-operasi dasar pada linked list.

2. Guided 2 – Latihan Double Linked List

Source Code

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

class Node {
public:
    int data;
    Node* prev;
    Node* next;
};

class DoublyLinkedList {
public:
    Node* head;
    Node* tail;

    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    void push(int data) {
        Node* newNode = new Node;
        newNode->data = data;
        newNode->prev = nullptr;
        newNode->next = head;

        if (head != nullptr) {
            head->prev = newNode;
        } else {
            tail = newNode;
        }

        head = newNode;
    }

    void pop() {
        if (head == nullptr) {
            return;
        }
        Node* temp = head;
        head = head->next;

        if (head != nullptr) {
```



```

        head->prev = nullptr;
    } else {
        tail = nullptr;
    }

    delete temp;
}

bool update(int oldData, int newData) {
    Node* current = head;

    while (current != nullptr) {
        if (current->data == oldData) {
            current->data = newData;
            return true;
        }
        current = current->next;
    }
    return false;
}

void deleteAll() {
    Node* current = head;
    while (current != nullptr) {
        Node* temp = current;
        current = current->next;
        delete temp;
    }
    head = nullptr;
    tail = nullptr;
}

void display() {
    Node* current = head;
    while (current != nullptr) {
        cout << current->data << " ";
        current = current->next;
    }
    cout << endl;
}

};

int main() {
    DoublyLinkedList list;
    while (true) {
        cout << "1. Add data" << endl;
        cout << "2. Delete data" << endl;
        cout << "3. Update data" << endl;
    }
}

```

```

cout << "4. Clear data" << endl;
cout << "5. Display data" << endl;
cout << "6. Exit" << endl;

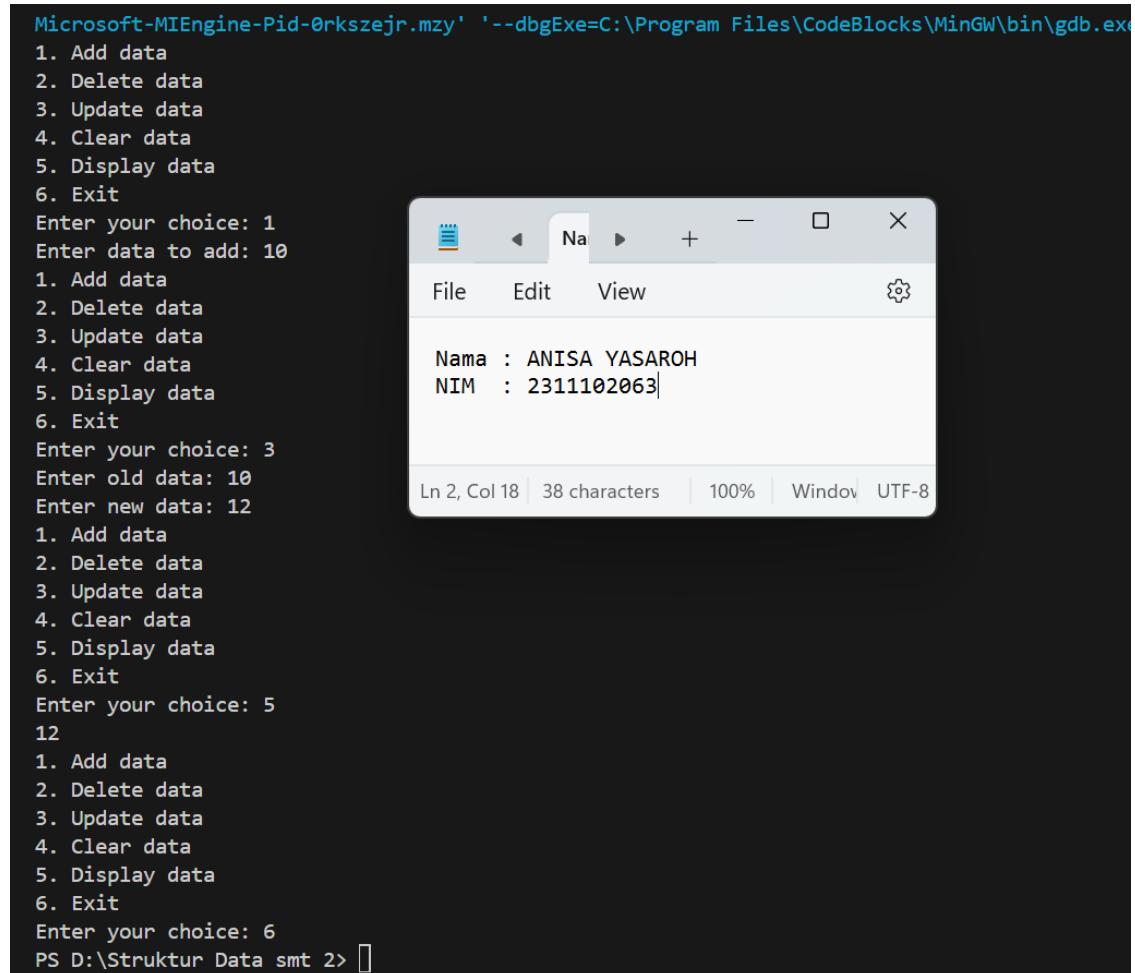
int choice;
cout << "Enter your choice: ";
cin >> choice;

switch (choice) {
    case 1: {
        int data;
        cout << "Enter data to add: ";
        cin >> data;
        list.push(data);
        break;
    }
    case 2: {
        list.pop();
        break;
    }
    case 3: {
        int oldData, newData;
        cout << "Enter old data: ";
        cin >> oldData;
        cout << "Enter new data: ";
        cin >> newData;
        bool updated = list.update(oldData, newData);
        if (!updated) {
            cout << "Data not found" << endl;
        }
        break;
    }
    case 4: {
        list.deleteAll();
        break;
    }
    case 5: {
        list.display();
        break;
    }
    case 6: {
        return 0;
    }
    default: {
        cout << "Invalid choice" << endl;
        break;
    }
}

```

```
}  
    return 0;  
}
```

Screenshoot Output



The screenshot shows a terminal window with the following text:

```
Microsoft-MIEngine-Pid-0rkszejr.mzy' '--dbgExe=C:\Program Files\CodeBlocks\MinGW\bin\gdb.exe  
1. Add data  
2. Delete data  
3. Update data  
4. Clear data  
5. Display data  
6. Exit  
Enter your choice: 1  
Enter data to add: 10  
1. Add data  
2. Delete data  
3. Update data  
4. Clear data  
5. Display data  
6. Exit  
Enter your choice: 3  
Enter old data: 10  
Enter new data: 12  
1. Add data  
2. Delete data  
3. Update data  
4. Clear data  
5. Display data  
6. Exit  
Enter your choice: 5  
12  
1. Add data  
2. Delete data  
3. Update data  
4. Clear data  
5. Display data  
6. Exit  
Enter your choice: 6  
PS D:\Struktur Data smt 2> |
```

Overlaid on the terminal is a small window titled 'Na' with a menu bar (File, Edit, View) and a settings icon. It displays the following text:

```
Nama : ANISA YASAROH  
NIM  : 2311102063|
```

The window also shows status information at the bottom: 'Ln 2, Col 18 | 38 characters | 100% | Window | UTF-8'.

Deskripsi Program

Program diatas yaitu implementasi dari double linked list dalam bentuk sebuah kelas 'DoublyLinkedList' yang memiliki fungsi-fungsi untuk melakukan operasi dasar pada double linked list, seperti penambahan (push), penghapusan (pop), pengubahan (update), penghapusan semua data (deleteAll), dan penampilan data (display). Setiap node dalam double linked list memiliki tiga pointer yaitu prev (menunjuk ke node sebelumnya), next (menunjuk ke node berikutnya), dan data (menyimpan nilai data). Program juga memiliki fungsi 'main' yang memberikan menu kepada pengguna untuk memilih operasi yang ingin dilakukan pada double linked list. Program berjalan secara terus-menerus hingga pengguna memilih untuk keluar (choice 6).

C. UNGUIDED

1. Unguided 1 – Membuat Program Menu Single Linked List

Buatlah program menu Single Linked List Non-Circular untuk menyimpan Nama dan usia mahasiswa, dengan menggunakan inputan dari user. Lakukan operasi berikut :

- a. Masukkan data sesuai urutan berikut. (Gunakan insert depan, belakang atau tengah). **Data pertama yang dimasukkan adalah nama dan usia anda.**

| [Nama_anda] | [Usia_anda] |
|-------------|-------------|
| John | 19 |
| Jane | 20 |
| Michael | 18 |
| Yusuke | 19 |
| Akechi | 20 |
| Hoshino | 18 |
| Karin | 18 |

- b. Hapus data Akechi
c. Tambahkan data berikut diantara John dan Jane : **Futaba 18**
d. Tambahkan data berikut diawal : **Igor 20**
e. Ubah data Michael menjadi : **Reyn 18**
f. Tampilkan seluruh data

Source Code

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

using namespace std;

struct node {
    string name;
    int age;
    node* next;
};

class linked_list{
private:
    node* head, * tail;

public:
    linked_list(){
        head = NULL;
```

```

        tail = NULL;
    }

    // tambah awal
    void add_first(string name, int age){
        node* temp = new node;
        temp->name = name;
        temp->age = age;
        temp->next = head;
        head = temp;
        if (tail == NULL)
        {
            tail = temp;
        }
    }

    // tambah akhir
    void add_last(string name, int age){
        node* temp = new node;
        temp->name = name;
        temp->age = age;
        temp->next = NULL;
        if (tail == NULL)
        {
            head = temp;
            tail = temp;
        } else{
            tail->next = temp;
            tail = temp;
        }
    }

    // tambah tengah
    void add_middle(int pos, string name, int age){
        node* prev = new node;
        node* cur = new node;
        node* temp = new node;
        cur = head;
        for (int a = 1; a < pos; a++)
        {
            prev = cur;
            cur = cur->next;
        }
        temp->name = name;
        temp->age = age;
        prev->next = temp;
        temp->next = cur;
    }

    // ubah tengah
    void edit_middle(int pos, string name, int age){

```

```

        node* temp = new node;
        temp = head;
        for (int a = 1; a < pos; a++)
        {
            temp = temp->next;
        }
        temp->name = name;
        temp->age = age;
    }
    // tampilkan jumlah data
    void count_data(){
        int count = 0;
        node* temp = new node;
        temp = head;
        while (temp != NULL)
        {
            count++;
            temp = temp->next;
        }
        cout << "Jumlah Data: " << count << endl;
    }

    // hapus data
    void delete_data(int pos)
    {
        node* prev = new node;
        node* cur = new node;
        cur = head;
        for (int a = 1; a < pos; a++)
        {
            prev = cur;
            cur = cur->next;
        }
        prev->next = cur->next;
    }

    // tampilkan data
    void display(){
        node* temp = new node;
        temp = head;
        while (temp != NULL)
        {
            cout << "Nama : " << temp->name << ", Usia : " <<
temp->age << endl;
            temp = temp->next;
        }
    }
};

```

```

int main(){
    linked_list l;

    // input data
    int choice;
    string name;
    int age;
    int pos;

    do {
        cout << endl;
        cout << "Menu Utama: " << endl;
        cout << "1. Tambahkan Awal" << endl;
        cout << "2. Tambahkan Akhir" << endl;
        cout << "3. Tambahkan Tengah" << endl;
        cout << "4. Ubah Tengah" << endl;
        cout << "5. Hapus Data" << endl;
        cout << "6. Tampilkan Jumlah Data" << endl;
        cout << "7. Tampilkan Seluruh Data" << endl;
        cout << "8. Keluar" << endl;
        cout << "Pilih Menu : ";
        cin >> choice;
    } while (choice != 8);

    switch (choice)
    {
        case 1:
            cout << "Masukkan Nama : ";
            cin >> name;
            cout << "Masukkan Usia : ";
            cin >> age;
            l.add_first(name, age);
            break;
        case 2:
            cout << "Masukkan Nama : ";
            cin >> name;
            cout << "Masukkan Usia : ";
            cin >> age;
            l.add_last(name, age);
            break;
        case 3:
            cout << "Masukkan Posisi : ";
            cin >> pos;
            cout << "Masukkan Nama : ";
            cin >> name;
            cout << "Masukkan Usia : ";
            cin >> age;
            l.add_middle(pos, name, age);
            break;
    }
}

```

```

        case 4:
            cout << "Masukkan Posisi : ";
            cin >> pos;
            cout << "Masukkan Nama : ";
            cin >> name;
            cout << "Masukkan Usia : ";
            cin >> age;
            l.edit_middle(pos, name, age);
            break;
        case 5:
            cout << "Masukkan Posisi : ";
            cin >> pos;
            l.delete_data(pos);
            break;
        case 6:
            l.count_data();
            break;
        case 7:
            l.display();
            break;
        case 8:
            cout << "Program Selesai" << endl;
            break;
        default:
            cout << "Pilihan tidak tersedia" << endl;
            break;
    }
} while (choice != 8);
return 0;
}

```

Screenshot Output

```

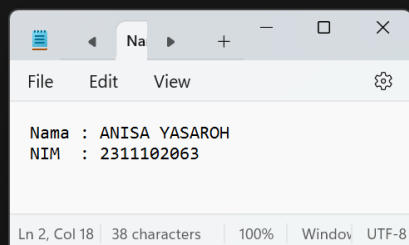
Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 2
Masukkan Nama : Anisa
Masukkan Usia : 19

```

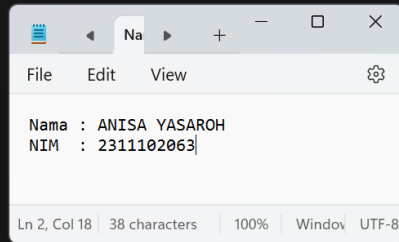
```

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 2
Masukkan Nama : John
Masukkan Usia : 19

```

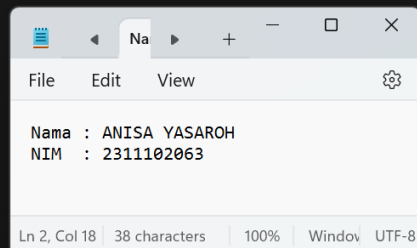


Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 2
Masukkan Nama : Jane
Masukkan Usia : 20



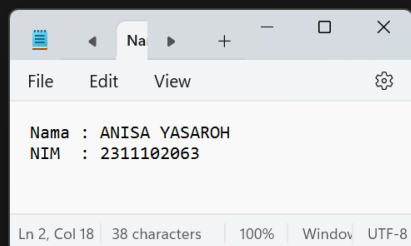
Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 2
Masukkan Nama : Michael
Masukkan Usia : 18

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 2
Masukkan Nama : Yusuke
Masukkan Usia : 19



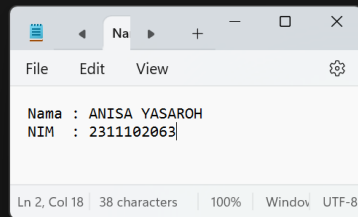
Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 2
Masukkan Nama : Akechi
Masukkan Usia : 20

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 2
Masukkan Nama : Hoshino
Masukkan Usia : 18



Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 2
Masukkan Nama : Karin
Masukkan Usia : 18

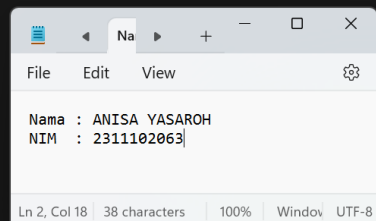
Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 7
Nama : Anisa, Usia : 19
Nama : John, Usia : 19
Nama : Jane, Usia : 20
Nama : Michael, Usia : 18
Nama : Yusuke, Usia : 19
Nama : Akechi, Usia : 20
Nama : Hoshino, Usia : 18
Nama : Karin, Usia : 18



Nama : ANISA YASAROH
NIM : 2311102063

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 5
Masukkan Posisi : 6

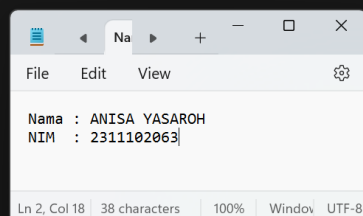
Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 7
Nama : Anisa, Usia : 19
Nama : John, Usia : 19
Nama : Jane, Usia : 20
Nama : Michael, Usia : 18
Nama : Yusuke, Usia : 19
Nama : Hoshino, Usia : 18
Nama : Karin, Usia : 18



Nama : ANISA YASAROH
NIM : 2311102063

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 3
Masukkan Posisi : 3
Masukkan Nama : Futuba
Masukkan Usia : 18

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 7
Nama : Anisa, Usia : 19
Nama : John, Usia : 19
Nama : Futuba, Usia : 18
Nama : Jane, Usia : 20
Nama : Michael, Usia : 18
Nama : Yusuke, Usia : 19
Nama : Hoshino, Usia : 18
Nama : Karin, Usia : 18



Nama : ANISA YASAROH
NIM : 2311102063

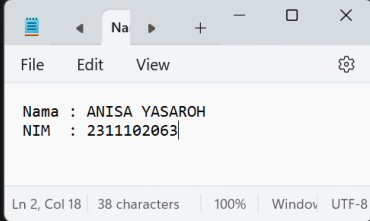
Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 1
Masukkan Nama : Igor
Masukkan Usia : 20

```
Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 7
Nama : Igor, Usia : 20
Nama : Anisa, Usia : 19
Nama : John, Usia : 19
Nama : Futuba, Usia : 18
Nama : Jane, Usia : 20
Nama : Michael, Usia : 18
Nama : Yusuke, Usia : 19
Nama : Hoshino, Usia : 18
Nama : Karin, Usia : 18

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 4
Masukkan Posisi : 6
Masukkan Nama : Reyn
Masukkan Usia : 18

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 7
Nama : Igor, Usia : 20
Nama : Anisa, Usia : 19
Nama : John, Usia : 19
Nama : Futuba, Usia : 18
Nama : Jane, Usia : 20
Nama : Reyn, Usia : 18
Nama : Yusuke, Usia : 19
Nama : Hoshino, Usia : 18
Nama : Karin, Usia : 18

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Seluruh Data
8. Keluar
Pilih Menu : 8
Program Selesai
PS D:\Struktur Data smt 2>
```



Deskripsi Program

Program diatas yaitu mengimplementasi dari sebuah single linked list. Linked list digunakan untuk menyimpan data dalam urutan tertentu, dimana setiap elemen (node) terdiri dari data (nama dan usia) serta pointer yang menunjuk ke node berikutnya. Kelas 'linked_list' menyediakan fungsi-fungsi untuk memanipulasi linked list, seperti menambahkan noe diawal, akhir, atau diposisi tertentu, mengubah data node diposisi tertentu, menghapus node dari posisi tertentu, menghitung jumlah data, dan menampilkan seluruh data. Dalam fungsi 'main', pengguna diberikan pilihan menu untuk melakukan operasi-operasi tersebut. Program akan terus berjalan dan meminta input pengguna sampai pengguna memilih untuk keluar.

2. Unguided 2 – Modifikasi Guided Double Linked List

Modifikasi Guided Double Linked List dilakukan dengan penambahan operasi untuk menambah data, menghapus, dan update di tengah / di urutan tertentu yang diminta. Selain itu, buatlah agar tampilannya menampilkan Nama produk dan harga.

| Nama Produk | Harga |
|-------------|---------|
| Originote | 60.000 |
| Somethinc | 150.000 |
| Skintific | 100.000 |

| | |
|---------|--------|
| Wardah | 50.000 |
| Hanasui | 30.000 |

Case:

1. Tambahkan produk Azarine dengan harga 65000 diantara Somethinc dan Skintific
2. Hapus produk wardah
3. Update produk Hanasui menjadi Cleora dengan harga 55.000
4. Tampilkan menu seperti dibawah ini

Toko Skincare Purwokerto

- 1. Tambah Data***
- 2. Hapus Data***
- 3. Update Data***
- 4. Tambah Data Urutan Tertentu***
- 5. Hapus Data Urutan Tertentu***
- 6. Hapus Seluruh Data***
- 7. Tampilkan Data***
- 8. Exit***

Pada menu 7, tampilan akhirnya akan menjadi seperti dibawah ini :

| Nama Produk | Harga |
|-------------|---------|
| Originote | 60.000 |
| Somethinc | 150.000 |
| Azarine | 65.000 |
| Skintific | 100.000 |
| Cleora | 55.000 |

Source Code

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

class Node { // Deklarasi Class Node untuk Double Linked List
public:
    string ProductName;
    int harga;
    Node* prev;
    Node* next;
};

class DoublyLinkedList { // Deklarasi Class DoublyLinkedList
    untuk Double Linked List
public:
    Node* head;
    Node* tail;
    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    void tambahproduk(string ProductName, int harga) { //
        Menambahkan produk ke dalam linked list di bagian atas
        Node* newNode = new Node;
        newNode->ProductName = ProductName;
        newNode->harga = harga;
        newNode->prev = nullptr;
        newNode->next = head;
```

```

        if (head != nullptr) {
            head->prev = newNode;
        }
        else {
            tail = newNode;
        }
        head = newNode;
    }

    void hapusproduk() { // Menghapus produk teratas dari
linked list
        if (head == nullptr) {
            return;
        }
        Node* temp = head;
        head = head->next;
        if (head != nullptr) {
            head->prev = nullptr;
        }
        else {
            tail = nullptr;
        }
        delete temp;
    }

    bool    ubahproduk(string    Oldproductname,    string
Newsproductname, int Hargaterbaru) { // Mengubah data produk
berdasarkan nama produk
        Node* current = head;
        while (current != nullptr) {
            if (current->ProductName == Oldproductname) {
                current->ProductName = Newsproductname;
                current->harga = Hargaterbaru;
                return true;
            }
            current = current->next;
        }
        return false; // Mengembalikan false jika data produk
tidak ditemukan
    }

    void    sisipposisi(string    ProductName,    int    harga,    int
posisi) { // Menambahkan data produk pada posisi tertentu
        if (posisi < 1) {
            cout << "Posisi tidak ada" << endl;
            return;
        }
        Node* newNode = new Node;

```

```

        newNode->ProductName = ProductName;
        newNode->harga = harga;
        if (posisi == 1) { // Jika posisi adalah 1 maka
tambahkan data produk di depan linked list
            newNode->next = head;
            newNode->prev = nullptr;
            if (head != nullptr) {
                head->prev = newNode;
            }
            else {
                tail = newNode;
            }
            head = newNode;
            return;
        }
        Node* current = head;
        for (int i = 1; i < posisi - 1 && current != nullptr;
++i) { // Looping sampai posisi sebelum posisi yang diinginkan
(Posisi - 1)
            current = current->next;
        }
        if (current == nullptr) {
            cout << "Posisi tidak ada" << endl;
            return;
        }
        newNode->next = current->next;
        newNode->prev = current;
        if (current->next != nullptr) {
            current->next->prev = newNode; // Pointer prev
node setelah current menunjuk ke newNode jika node setelah
current tidak nullptr
        }
        else {
            tail = newNode;
        }
        current->next = newNode;
    }

    void hapusposisi(int posisi) { // Menghapus data produk
pada posisi tertentu
        if (posisi < 1 || head == nullptr) {
            cout << "Posisi tidak ada atau list kosong" <<
endl;

            return;
        }
        Node* current = head;
        if (posisi == 1) {
            head = head->next;

```

```

        if (head != nullptr) {
            head->prev = nullptr;
        }
        else {
            tail = nullptr;
        }
        delete current;
        return;
    }
    for (int i = 1; current != nullptr && i < posisi; ++i)
    { // Looping sampai posisi yang diinginkan
        current = current->next;
    }
    if (current == nullptr) {
        cout << "Posisi tidak ada" << endl;
        return;
    }
    if (current->next != nullptr) {
        current->next->prev = current->prev;
    }
    else {
        tail = current->prev;
    }
    current->prev->next = current->next;
    delete current;
}

void hapussemua() { // Menghapus semua data produk
    Node* current = head;
    while (current != nullptr) {
        Node* temp = current;
        current = current->next;
        delete temp;
    }
    head = nullptr;
    tail = nullptr;
}

void tampilan() { // Menampilkan data produk
    Node* current = head;
    cout << "\nBerikut daftar Produk dan harga yang
tersedia saat ini:" << endl;
    cout << left << setw(20) << "Nama Produk" << "Harga"
<< endl;
    while (current != nullptr) {
        cout << left << setw(20) << current->ProductName
<< current->harga << endl;
        current = current->next;
    }
}

```



```

        }
        cout << endl;
    }
};

int main() {
    DoublyLinkedList list; // Deklarasi objek list dari class
    DoublyLinkedList

    list.tambahproduk("Hanasui", 30000);
    list.tambahproduk("Wardah", 50000);
    list.tambahproduk("Skintific", 100000);
    list.tambahproduk("Somethinc", 150000);
    list.tambahproduk("Originote", 60000);

    cout << "\n=====Selamat datang di Beautime
    Purwokerto===== " << endl;
    list.tampilan();

    while (true) { // Looping menu utama
        cout << "\nMenu Beautime Purwokerto" << endl;
        cout << "1. Tambah Data" << endl;
        cout << "2. Hapus Data" << endl;
        cout << "3. Update Data" << endl;
        cout << "4. Tambah Data Urutan Tertentu" << endl;
        cout << "5. Hapus Data Urutan Tertentu" << endl;
        cout << "6. Hapus Seluruh Data" << endl;
        cout << "7. Tampilkan Data" << endl;
        cout << "8. Exit" << endl;
        int pilihan;
        cout << "Pilih Menu: ";
        cin >> pilihan;
        switch (pilihan) { // Switch case untuk memilih menu
            case 1: {
                string ProductName;
                int harga;
                cout << "Masukkan nama produk: ";
                cin >> ProductName;
                cout << "Masukkan harga: ";
                cin >> harga;
                list.tambahproduk(ProductName, harga); //
                Memanggil fungsi tambah_produk
                cout << "Produk berhasil ditambahkan teratas" <<
                endl;
                break;
            }
            case 2: {

```

```

        list.hapusproduk(); // Memanggil fungsi
hapus_produk
        cout << "Produk teratas berhasil dihapus" << endl;
        break;
    }
    case 3: {
        string Oldproductname, Newsproductname;
        int Hargaterbaru;
        cout << "Input nama produk lama: ";
        cin >> Oldproductname;
        cout << "Input nama produk baru: ";
        cin >> Newsproductname;
        cout << "Input harga baru: ";
        cin >> Hargaterbaru;
        bool updated = list.ubahproduk(Oldproductname,
Newsproductname, Hargaterbaru); // Memanggil fungsi
ubah_produk
        if (!updated) {
            cout << "Data produk tidak ditemukan" << endl;
        }
        else {
            cout << "Data produk berhasil diupdate" <<
endl;
        }
        break;
    }
    case 4: {
        string ProductName;
        int harga, position;
        cout << "Input nama produk: ";
        cin >> ProductName;
        cout << "Input harga: ";
        cin >> harga;
        cout << "Input posisi: ";
        cin >> position;
        list.sisipposisi(ProductName, harga, position); //
Memanggil fungsi sisipkan_posisi_tertentu
        cout << "Produk berhasil ditambahkan pada posisi "
<< position << endl;
        break;
    }
    case 5: {
        int position;
        cout << "Input posisi yang ingin dihapus: ";
        cin >> position;
        list.hapusposisi(position); // Memanggil fungsi
hapus_posisi_tertentu

```

```

        break;
    }
    case 6: {
        list.hapussemua(); // Memanggil fungsi hapus_semua
        break;
    }
    case 7: {
        list.tampilan(); // Memanggil fungsi display
        break;
    }
    case 8: {
        return 0;
    }
    default: {
        cout << "Input Invalid" << endl;
        break;
    }
}
return 0;
}

```

Screenshoot Output

The screenshot shows a terminal window with the following output:

```

=====Selamat datang di Beautime Purwokerto=====

Berikut daftar Produk dan harga yang tersedia saat ini:
Nama Produk      Harga
Originote        60000
Somethinc         150000
Skintific         100000
Wardah            50000
Hanasui           30000

Menu Beautime Purwokerto
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan Data
8. Exit
Pilih Menu: 4
Input nama produk: Azarine
Input harga: 65000
Input posisi: 3
Produk berhasil ditambahkan pada posisi 3

```

Overlaid on the terminal is a Notepad window with the following text:

```

Nama : ANISA YASAROH
NIM  : 2311102063

```

The Notepad window status bar indicates: Ln 2, Col 18 | 38 characters | 100% | Window | UTF-8.

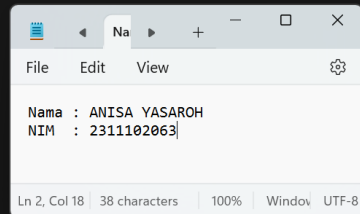
```
Menu Beautime Purwokerto
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan Data
8. Exit
Pilih Menu: 7
```

Berikut daftar Produk dan harga yang tersedia saat ini:

| Nama Produk | Harga |
|-------------|--------|
| Originote | 60000 |
| Somethinc | 150000 |
| Azarine | 65000 |
| Skintific | 100000 |
| Wardah | 50000 |
| Hanasui | 30000 |

```
Menu Beautime Purwokerto
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan Data
8. Exit
Pilih Menu: 5
```

Input posisi yang ingin dihapus: 5



```
Menu Beautime Purwokerto
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan Data
8. Exit
Pilih Menu: 7
```

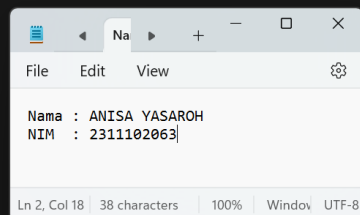
Berikut daftar Produk dan harga yang tersedia saat ini:

| Nama Produk | Harga |
|-------------|--------|
| Originote | 60000 |
| Somethinc | 150000 |
| Azarine | 65000 |
| Skintific | 100000 |
| Hanasui | 30000 |

```
Menu Beautime Purwokerto
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan Data
8. Exit
```

Pilih Menu: 3

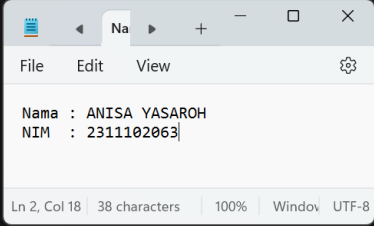
Input nama produk lama: Hanasui
Input nama produk baru: Cleora
Input harga baru: 55000
Data produk berhasil diupdate



```
Menu Beautime Purwokerto
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan Data
8. Exit
Pilih Menu: 7

Berikut daftar Produk dan harga yang tersedia saat ini:
Nama Produk      Harga
Originote        60000
Somethinc        150000
Azarine          65000
Skintific        100000
Cleora           55000

Menu Beautime Purwokerto
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan Data
8. Exit
Pilih Menu: 8
PS D:\Struktur Data smt 2> |
```



Deskripsi Program

Program diatas yaitu menggunakan konsep struktur data double linked list untuk menyimpan data produk skincare, seperti nama produk dan harganya. Pada saat pertama kali dijalankan, program akan menampilkan daftar produk skincare yang telah tersedia. Kemudian, program akan menampilkan pilihan kepada pengguna, yang mencakup fungsi-fungsi seperti menambah, menghapus, mengubah, dan menampilkan data produk. Setelah pengguna memilih operasi tertentu, program akan menjalankan fungsi yang sesuai dan melakukan operasi tersebut pada data produk yang tersimpan dalam linked list. Program akan terus berjalan dalam loop hingga pengguna memilih untuk keluar dari program.

D. KESIMPULAN

Berdasarkan praktikum yang telah dilakukan, dapat disimpulkan bahwa :

1. Single dan double linked list berguna dalam menangani masalah yang memerlukan manipulasi data secara dinamis. Dan digunakan untuk menerapkan struktur data seperti queue, strack, dan tree.
2. Single linked list mempelajari cara membuat dan menerapkan linked list yang menggunakan satu pointer untuk menunjuk ke node berikutnya.
3. Dalam double linked list mempelajari pembuatan dan implementasi linked list dengan dua pointer yaitu satu untuk menunjuk ke node sebelumnya dan satu lagi untuk menunjuk ke node berikutnya.
4. Double linked list memungkinkan operasi-operasi seperti penghapusan atau penambahan node ditengah-tengah linked list dilakukan dengan lebih efisien.

E. REFERENSI

Andrist, B., Sehr, V., & Garney, B. (2020). *C++ high Performance: master the art of optimizing the functioning of your C++ code*. Packt Publishing Ltd.

Krasnov, M. M. (2020). Functional programming library for C++. *Programming and Computer Software*, 46, 330-340.

Malik, D.S. (2023). *C++ Programming*. Cengage Learning, EMEA.

Mandala, S. K., & Gurrapu, N. (2021). *PROGRAMMING IN C++*. Horizon Books (A Division of Ignited Minds Edutech P Ltd).

Mohanty, S.N., & Tripathy, P.K. (2021). *Data structure and algorithms using C++: a practical implementation*. John Wiley & Sons.