

**LAPORAN PRAKTIKUM STRUKTUR  
DATA DAN ALGORITMA**

**MODUL III  
SINGLE AND DOUBLE LINKED LIST**



**Disusun oleh :**

**MUHAMAD NASRULLOH**  
2311102044  
IF 11-B

**Dosen :**

**WAHYU ANDI SAPUTRA, S.PD., M.ENG**

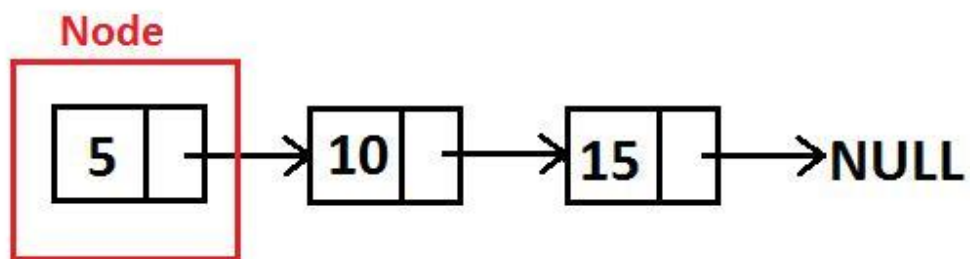
**PROGRAM STUDI S1 TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

## A. DASAR TEORI

### 1. Single Linked List

Linked List merupakan koleksi linear dari data yang disebut nodes, dimana setiap node akan menunjuk pada node lain melalui sebuah pointer, Linked list dapat didefinisikan juga sebagai kumpulan nodes yang mempresentasikan sebuah sequence.

Linked List yang hanya memiliki 1 penghubung ke node lain disebut sebagai single linked list.

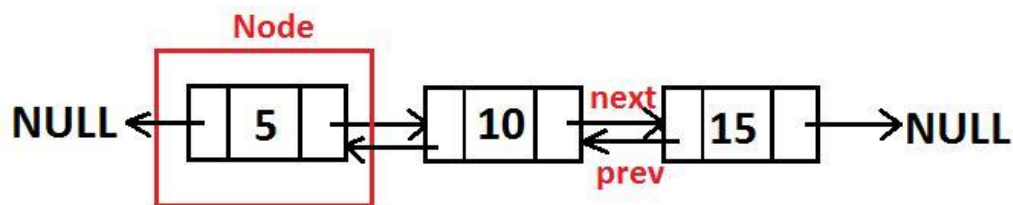


Jika Single Linked List didefinisikan sebagai berikut :

```
struct data{
    int x;
    struct data* next;
}*head=NULL, *curr, xx
```

### 2. DOUBLE LINKED LIST

Jika sebelumnya Single Linked List dengan 1 pointer penghubung, maka Double Linked List memiliki 2 pointer penghubung yaitu ke arah node sebelumnya (previous/prev) dan node sesudahnya (next).



Dengan adanya pointer prev memungkinkan untuk melakukan operasi penghapusan dan penambahan pada node mana saja secara efisien. Jika didefinisikan sebagai berikut :

```
struct data{
    int x;
    struct data* next;
    struct data* prev;
}*head=NULL, *curr, *tail;
```

Keuntungan memakai Double Linked List memungkinkan dapat melakukan operasi penghapusan dan penambahan simpul dimana saja secara efisien, sehingga sangat berguna dalam implementasi beberapa algoritma yang membutuhkan operasi tersebut. Selain itu juga memungkinkan melakukan transversal pada list baik dari depan (head) atau belakang (tail) dengan mudah. Hanya saja kekurangannya memori yang digunakan jauh lebih besar dibandingkan Single Linked List karena setiap node butuh 1 pointer tambahan dan juga memerlukan waktu eksekusi yang lama dalam operasi penambahan dan penghapusan.

## B. GUIDED

### I – 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;
    }
}

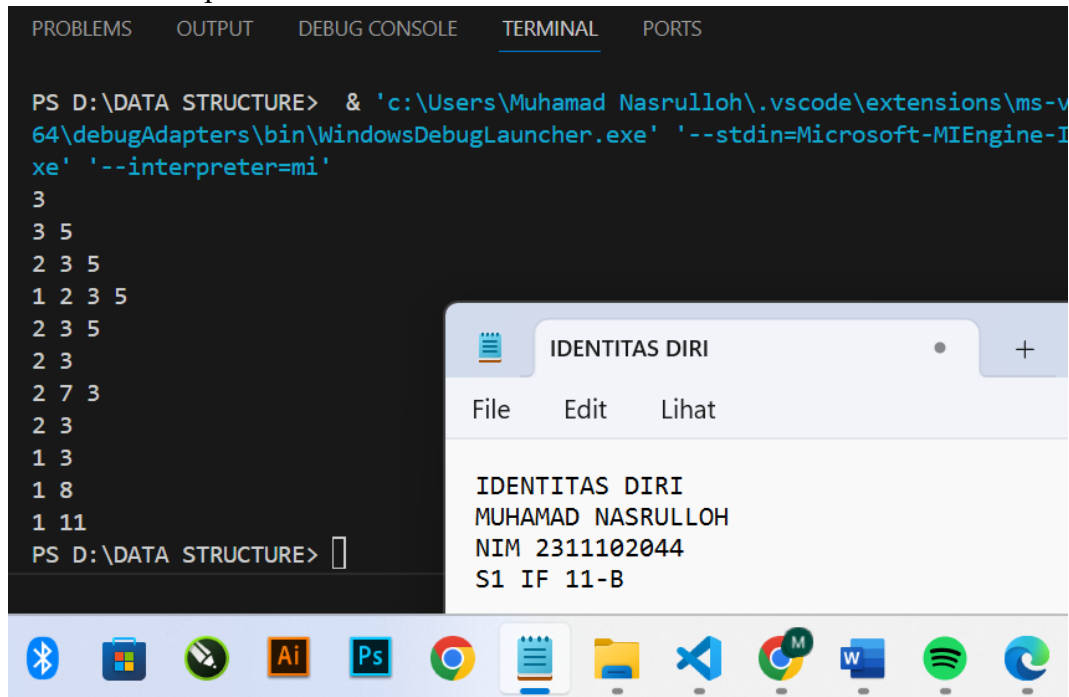
// Hapus semua Node di list
void clearList() {
    Node* bantu = head;
    while (bantu != NULL) {
        Node* hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}

// 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;
}

```

## Screenshot Output



The screenshot shows a VS Code interface with a terminal window and a Notepad window. The terminal window displays the output of a C++ program, which is a single linked list. The output is as follows:

```
PS D:\DATA STRUCTURE> & 'c:\Users\Muhamad Nasrulloh\.vscode\extensions\ms-vs-64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-Interpreter=mi'
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:\DATA STRUCTURE>
```

The Notepad window, titled "IDENTITAS DIRI", contains the following text:

```
IDENTITAS DIRI
MUHAMAD NASRULLOH
NIM 2311102044
S1 IF 11-B
```

## Deskripsi

Program diatas merupakan contoh implementasi Single Linked List dalam bahasa C++, yang memiliki fungsi untuk mengelola data dalam list, seperti :

- Menambah data

insertDepan(data) untuk menambahkan data di depan list, insertBelakang(data) untuk menambah data di belakang list, insertTengah(data, posisi) untuk menambah data di posisi tengah list

- Menghapus data

hapusDepan() untuk menghapus data di depan list, hapusBelakang() untuk menghapus data di belakang list, hapusTengah(posisi) untuk menghapus data di posisi tengah list

- Menampilkan data

tampil() untuk menampilkan semua data dalam list

- Fungsi lain

isEmpty() untuk mengecek apakah list kosong, hitungList() untuk menghitung jumlah data dalam list, clearList() untuk menghapus semua data dalam list

## II – 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;
}

```

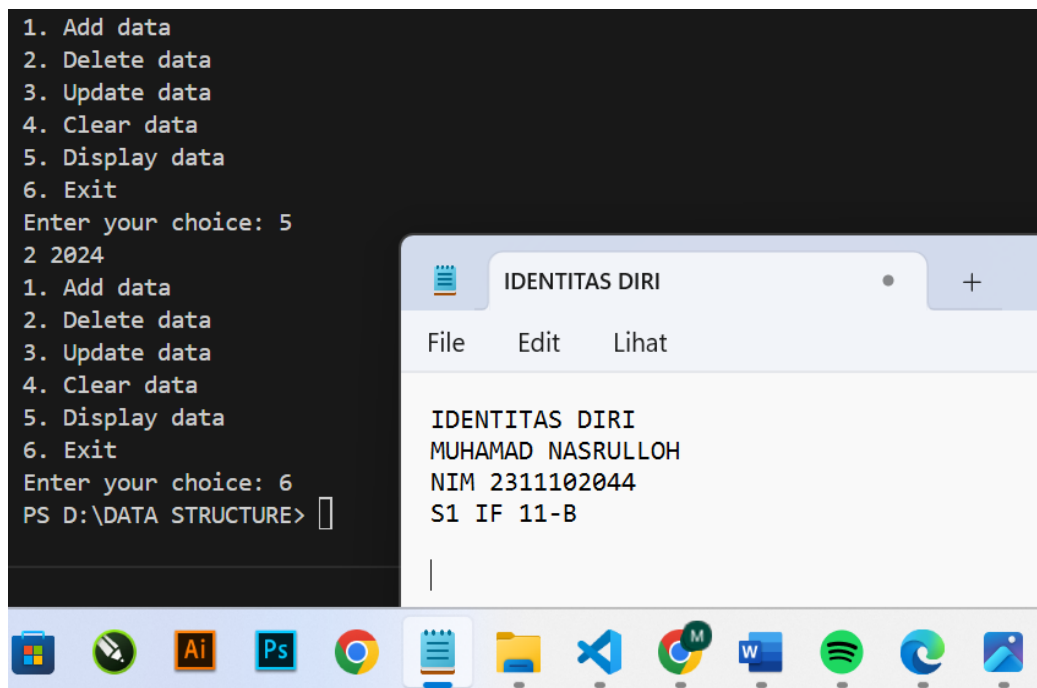
## Screenshot Output

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```

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: 2024
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: 02

```



### Deskripsi

Program diatas menampilkan sebuah menu sederhana dimana user bebas memilih pilihan, seperti menginputkan data(push), memperbarui data(update), menghapus data keseluruhan(deleteAll) atau menampilkannya(display). Program ini mengimplementasikan Double Linked List yang menyimpan sekumpulan elemen dengan urutan tertentu, dimana setiap elemen memiliki pointer ke elemen sebelumnya dan elemen berikutnya.

### C. UNGUIDED

**1. 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 Aketchi

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 Mahasiswa{
    string nama;
    int usia;
    Mahasiswa *next;
};

Mahasiswa *head = nullptr;

void insertDepan(string nama, int usia){
    Mahasiswa *node = new Mahasiswa;
    node->nama = nama;
    node->usia = usia;
    node->next = head;
    head = node;
}

void insertBelakang(string nama, int usia){
    Mahasiswa *node = new Mahasiswa;
```

```

        node->nama = nama;
        node->usia = usia;
        node->next = nullptr;
        if (head == nullptr){
            head = node;
            return;
        }

        Mahasiswa *curr = head;
        while (curr->next != nullptr){
            curr = curr->next;
        }
        curr->next = node;
    }

    void insertTengah(string nama, int usia, int pos){
        Mahasiswa *node = new Mahasiswa;
        node->nama = nama;
        node->usia = usia;
        Mahasiswa *curr = head;
        for (int i=0; i < pos - 1; i++){
            curr = curr->next;
        }
        node->next = curr->next;
        curr->next = node;
    }

    void hapusData(string nama){
        if (head == nullptr){
            return;
        }
        if (head->nama == nama){
            Mahasiswa *node = head;
            head = head->next;
            delete node;
            return;
        }
        Mahasiswa *curr = head;
        while (curr->next != nullptr && curr->next->nama != nama){
            curr = curr->next;
        }
        if (curr->next != nullptr){
            Mahasiswa *node = curr->next;
            curr->next = curr->next->next;
            delete node;
        }
    }

    void ubahData(string nama, int usia){
        if (head == nullptr){
            cout << "Tidak ada data yang bisa diubah" << endl;
            return;
        }
        Mahasiswa *node = head;
        string key;
        cout << "Masukkan nama yang ingin diubah: ";

```

```

        cin >> key;
        while (node != nullptr){
            if (node->nama == key){
                cout << "Masukkan nama baru: ";
                cin >> node->nama;
                cout << "Masukkan usia baru: ";
                cin >> node->usia;
                return;
            }
            node = node->next;
        }
        cout << "Data berhasil diubah" << endl;
    }

    void tampilkanData(){
        Mahasiswa *curr = head;
        while (curr != nullptr){
            cout << curr->nama << " " << curr->usia << endl;
            curr = curr->next;
        }
    }

    int main(){
        int pilihan, usia, posisi;
        string nama;

        do
        {
            // Tampilkan menu
            cout << "Menu:" << endl;
            cout << "1. Insert Depan" << endl;
            cout << "2. Insert Belakang" << endl;
            cout << "3. Insert Tengah" << endl;
            cout << "4. Hapus Data" << endl;
            cout << "5. Ubah Data" << endl;
            cout << "6. Tampilkan Data" << endl;
            cout << "7. Keluar" << endl;
            cout << "Pilihan: ";
            cin >> pilihan;

            switch (pilihan){
                case 1:
                    cout << "Masukkan nama: ";
                    cin >> nama;
                    cout << "Masukkan usia: ";
                    cin >> usia;
                    insertDepan(nama, usia);
                    break;
                case 2:
                    cout << "Masukkan nama: ";
                    cin >> nama;
                    cout << "Masukkan usia: ";
                    cin >> usia;
                    insertBelakang(nama, usia);
                    break;
                case 3:

```

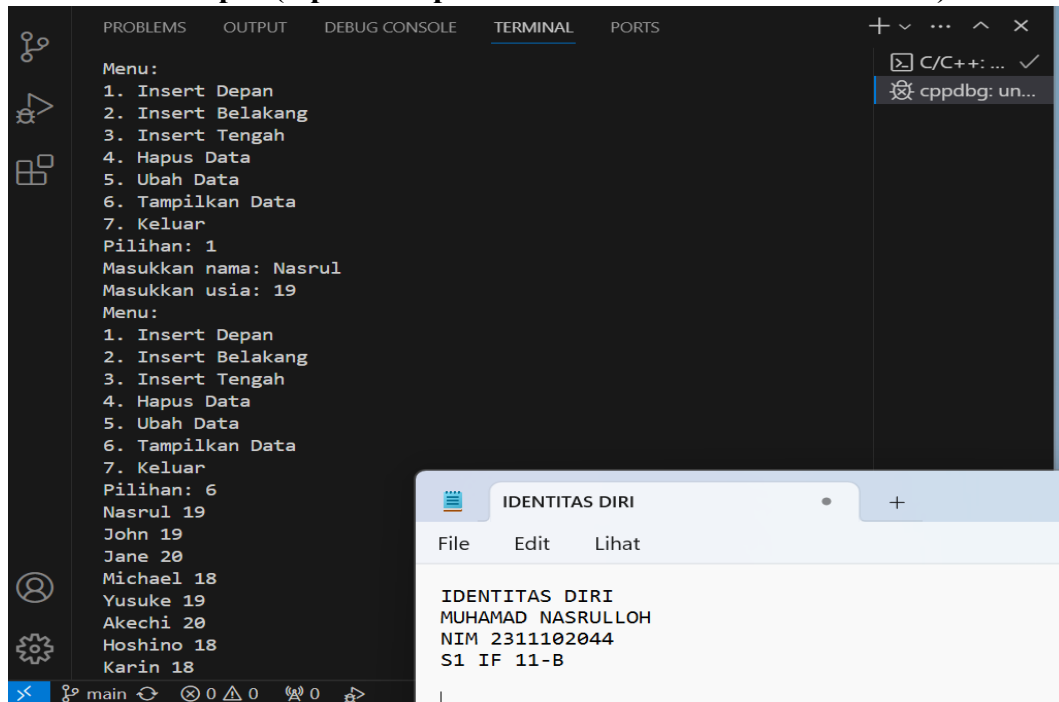
```

        cout << "Masukkan nama: ";
        cin >> nama;
        cout << "Masukkan usia: ";
        cin >> usia;
        cout << "Masukkan posisi: ";
        cin >> posisi;
        insertTengah(nama, usia, posisi);
        break;
    case 4:
        cout << "Masukkan nama: ";
        cin >> nama;
        hapusData(nama);
        break;
    case 5:
        ubahData(nama, usia);
        break;
    case 6:
        tampilkanData();
        break;
    case 7:
        cout << "Terima kasih." << endl;
        break;
    default:
        cout << "Pilihan tidak valid." << endl;
    }
} while (pilihan != 7);

return 0;
}

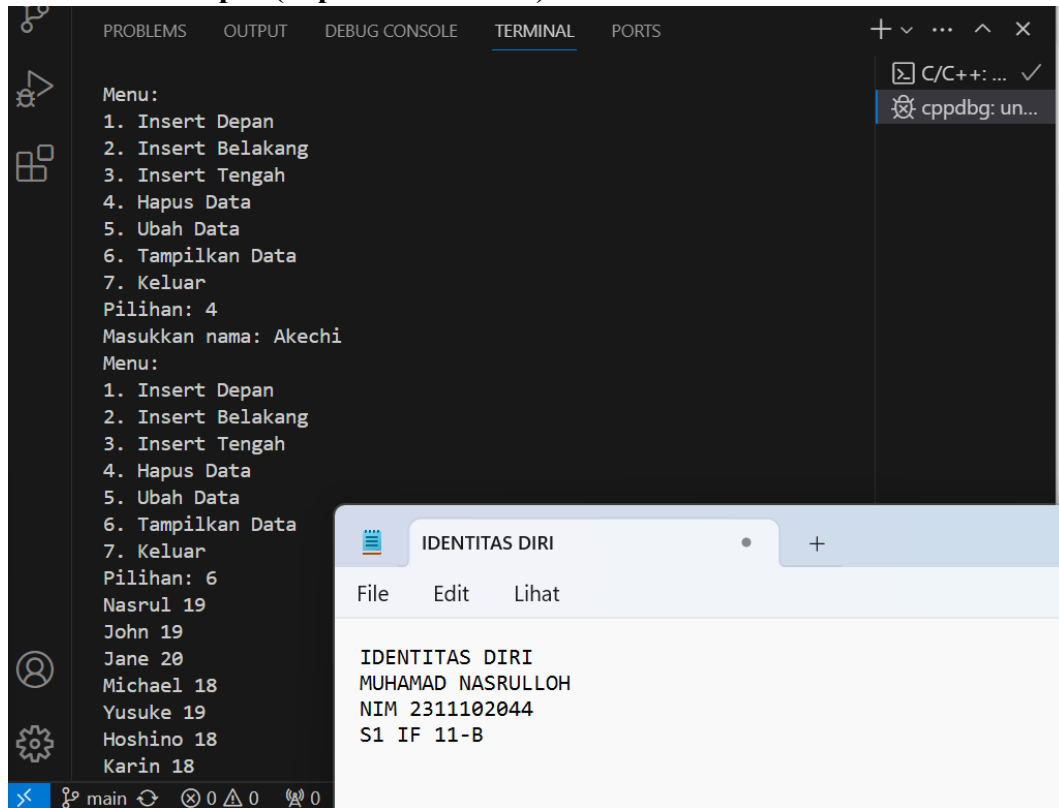
```

### Screenshot Output (input data pertama adalah nama dan usia Anda)

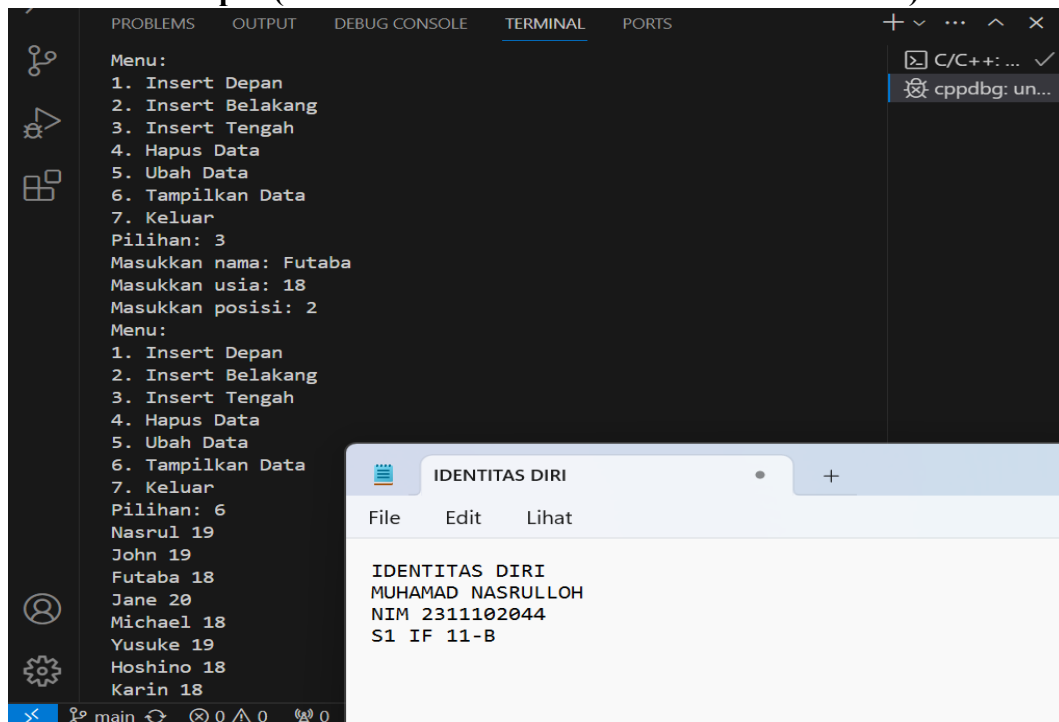




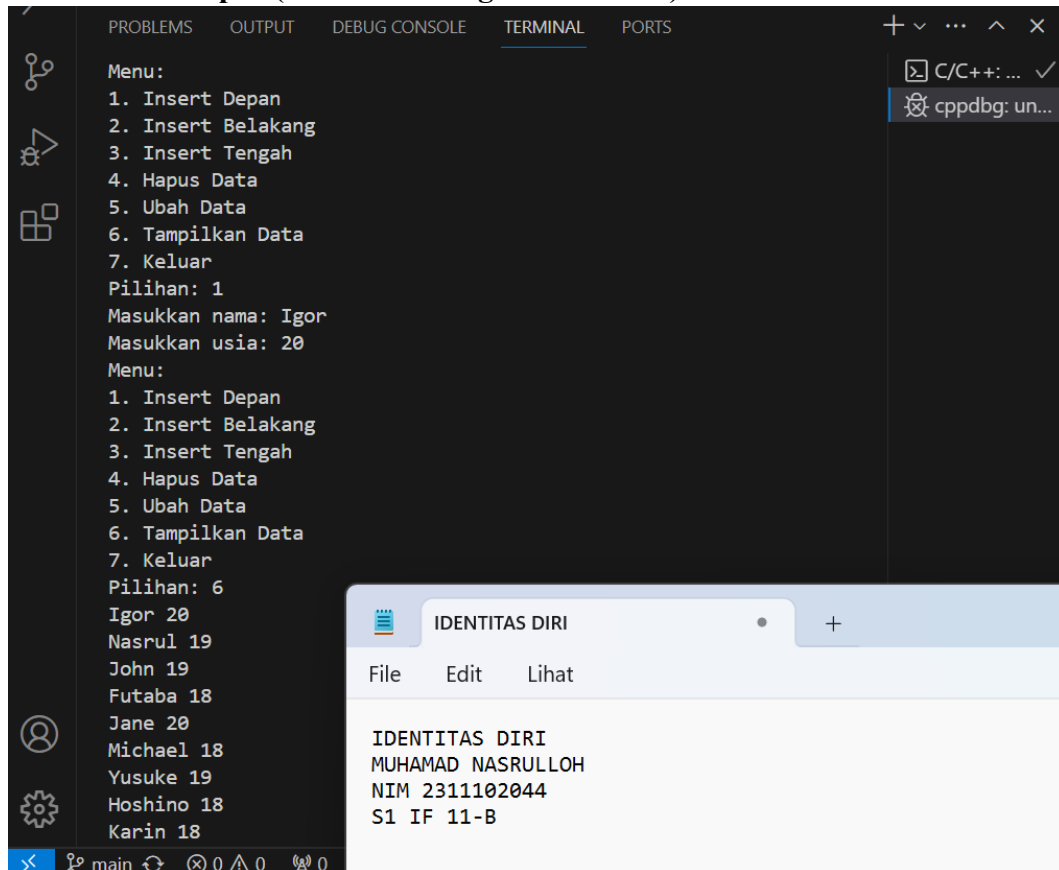
### Screenshot Output (hapus data Akechi)



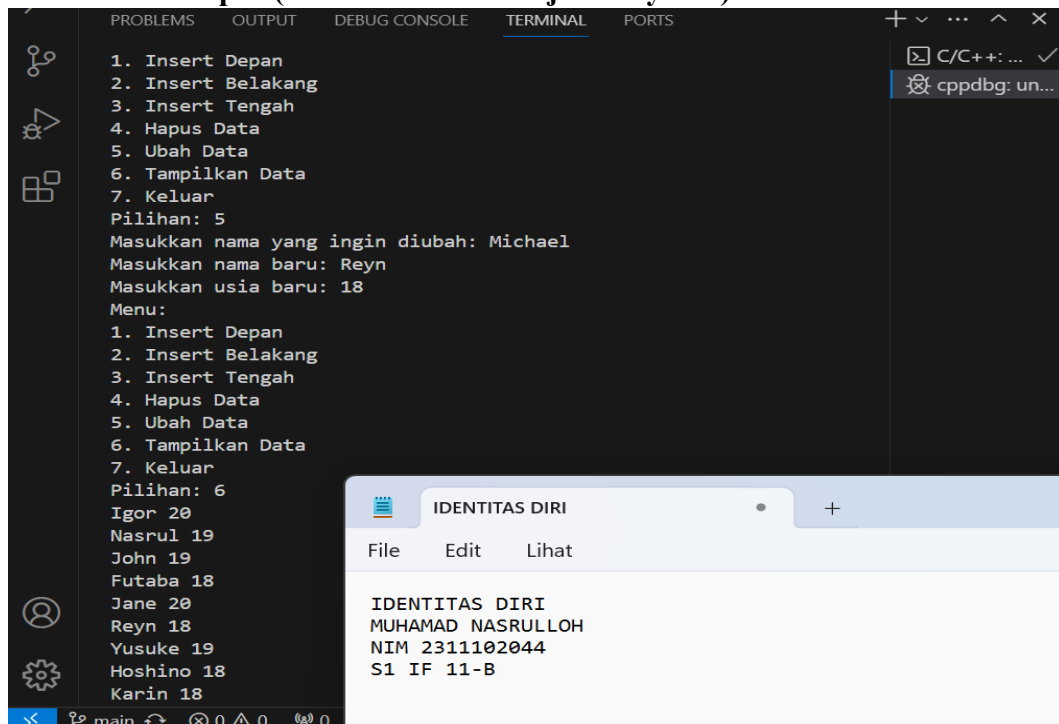
### Screenshot Output (tambah data Futaba 18 antara John dan Jane)



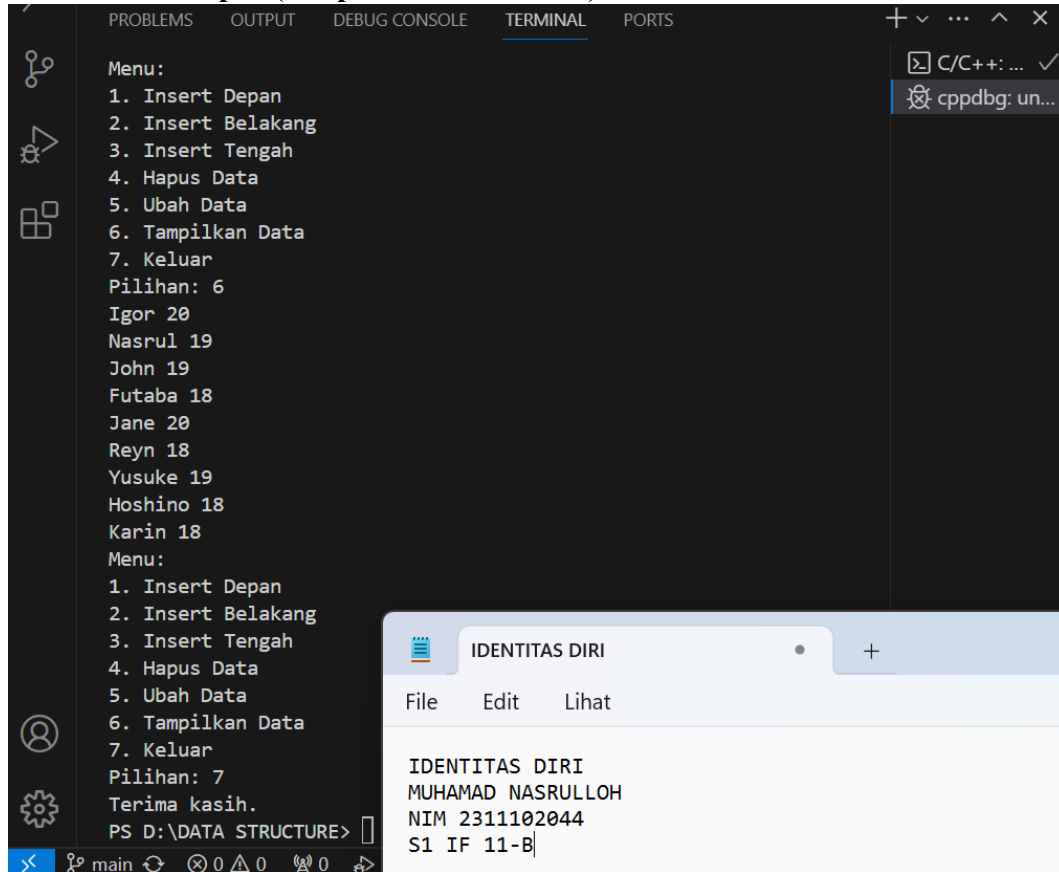
### Screenshot Output (tambah data Igor 20 di awal)



### Screenshot Output (ubah data Michael jadi Reyn 18)



### Screenshot Output (tampilkan semua data)



### Deskripsi

Program diatas menampilkan menu dengan 7 opsi/pilihan : (insert) menambah data di depan, (insert) menambah data di belakang, (insert) menambah data di tengah, hapus data, ubah data, tampilkan data, dan keluar. Dengan menu tersebut memudahkan user dalam memilih operasi yang diinginkan. Setiap user memilih operasi, program akan menampilkan kata yang meminta user tersebut menginput sesuatu lalu akan dipanggil fungsi yang sesuai. Jika user telah selesai operasi lalu memilih keluar maka program akan berhenti.

### 2. Modifikasi guided Double Linked List dengan penambahan operasi untuk menambah data, menghapus, dan update di tengah/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 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 ke7, tampilan seperti 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>
using namespace std;

class Node{
public:
    string namaProduk;
    int harga;
    Node *prev, *next;
};

class DoublyLinkedList{
public:
    Node *head;
    Node *tail;
    DoublyLinkedList(){
        head = nullptr;
        tail = nullptr;
    }

    void push(string namaProduk, int harga){
        Node *newNode = new Node;
```

```

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

void pushCenter(string namaProduk, int harga, int posisi){
    if (posisi < 0){
        cout << "Posisi harus bernilai non-negatif." << endl;
        return;
    }

    Node *newNode = new Node;
    newNode->namaProduk = namaProduk;
    newNode->harga = harga;
    if (posisi == 0 || head == nullptr){
        newNode->prev = nullptr;
        newNode->next = head;
    }
    if (head != nullptr){
        head->prev = newNode;
    }
    else{
        tail = newNode;
    }
    head = newNode;
}
else{
    Node *temp = head;
    int count = 0;
    while (temp != nullptr && count < posisi){
        temp = temp->next;
        count++;
    }
    if (temp == nullptr){
        newNode->prev = tail;
        newNode->next = nullptr;
        tail->next = newNode;
        tail = newNode;
    }
    else{
        newNode->prev = temp->prev;
        newNode->next = temp;
        temp->prev->next = newNode;
        temp->prev = newNode;
    }
}
}

void pop() {

```

```

        if (head == nullptr){
            return;
        }

        Node *temp = head;
        head = head->next;
        if (head != nullptr){
            head->prev = nullptr;
        }
        else{
            tail = nullptr;
        }
        delete temp;
    }

    void popCenter(int posisi){
        if (head == nullptr){
            cout << "List kosong. Tidak ada yang bisa dihapus."
<< endl;
            return;
        }
        if (posisi < 0){
            cout << "Posisi harus bernilai non-negatif." << endl;
            return;
        }
        if (posisi == 0){
            Node *temp = head;
            head = head->next;
            if (head != nullptr){
                head->prev = nullptr;
            }
            else{
                tail = nullptr;
            }
            delete temp;
        }
        else{
            Node *temp = head;
            int count = 0;
            while (temp != nullptr && count < posisi){
                temp = temp->next;
                count++;
            }
            if (temp == nullptr){
                cout << "Posisi melebihi ukuran list. Tidak ada yang
dihapus." << endl;
                return;
            }
            if (temp == tail){
                tail = tail->prev;
                tail->next = nullptr;
                delete temp;
            }
            else{
                temp->prev->next = temp->next;
                temp->next->prev = temp->prev;
            }
        }
    }

```

```

        delete temp;
    }
}

bool update(string oldNamaProduk, string newNamaProduk, int
newHarga){
    Node *current = head;
    while (current != nullptr){
        if (current->namaProduk == oldNamaProduk){
            current->namaProduk = newNamaProduk;
            current->harga = newHarga;
            return true;
        }
        current = current->next;
    }
    return false;
}

bool updateCenter(string newNamaProduk, int newHarga, int
posisi){
    if (head == nullptr){
        cout << "List kosong. Tidak ada yang dapat diperbarui."
<< endl;
        return false;
    }
    if (posisi < 0){
        cout << "Posisi harus bernilai non-negatif." << endl;
        return false;
    }
    Node *current = head;
    int count = 0;
    while (current != nullptr && count < posisi){
        current = current->next;
        count++;
    }
    if (current == nullptr){
        cout << "Posisi melebihi ukuran list. Tidak ada yang
diperbarui." << endl;
        return false;
    }
    current->namaProduk = newNamaProduk;
    current->harga = newHarga;
    return true;
}

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

```

```

void display(){
    if (head == nullptr){
        cout << "List kosong." << endl;
        return;
    }
    Node *current = head;
    cout << setw(37) << setfill('-') << "-" << setfill(' ')
<< endl;
    cout << "| " << setw(20) << left << "Nama Produk" << " |
" << setw(10) << "Harga" << " |" << endl;
    cout << setw(37) << setfill('-') << "-" << setfill(' ')
<< endl;
    while (current != nullptr){
        cout << "| " << setw(20) << left << current-
>namaProduk << " | " << setw(10) << current->harga << " |" <<
endl;
        current = current->next;
    }
    cout << setw(37) << setfill('-') << "-" << setfill(' ')
<< endl;
}
};

int main(){
    DoublyLinkedList list;
    int choice;
    cout << endl
        << "Toko Skincare Purwokerto" << endl;
    do{
        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;
        cout << "Pilihan : ";
        cin >> choice;
        switch (choice){
            case 1: {
                string namaProduk;
                int harga;
                cout << "Masukkan nama produk: ";
                cin.ignore();
                getline(cin, namaProduk);
                cout << "Masukkan harga produk: ";
                cin >> harga;
                list.push(namaProduk, harga);
                break;
            }
            case 2: {
                list.pop();
                break;
            }

```

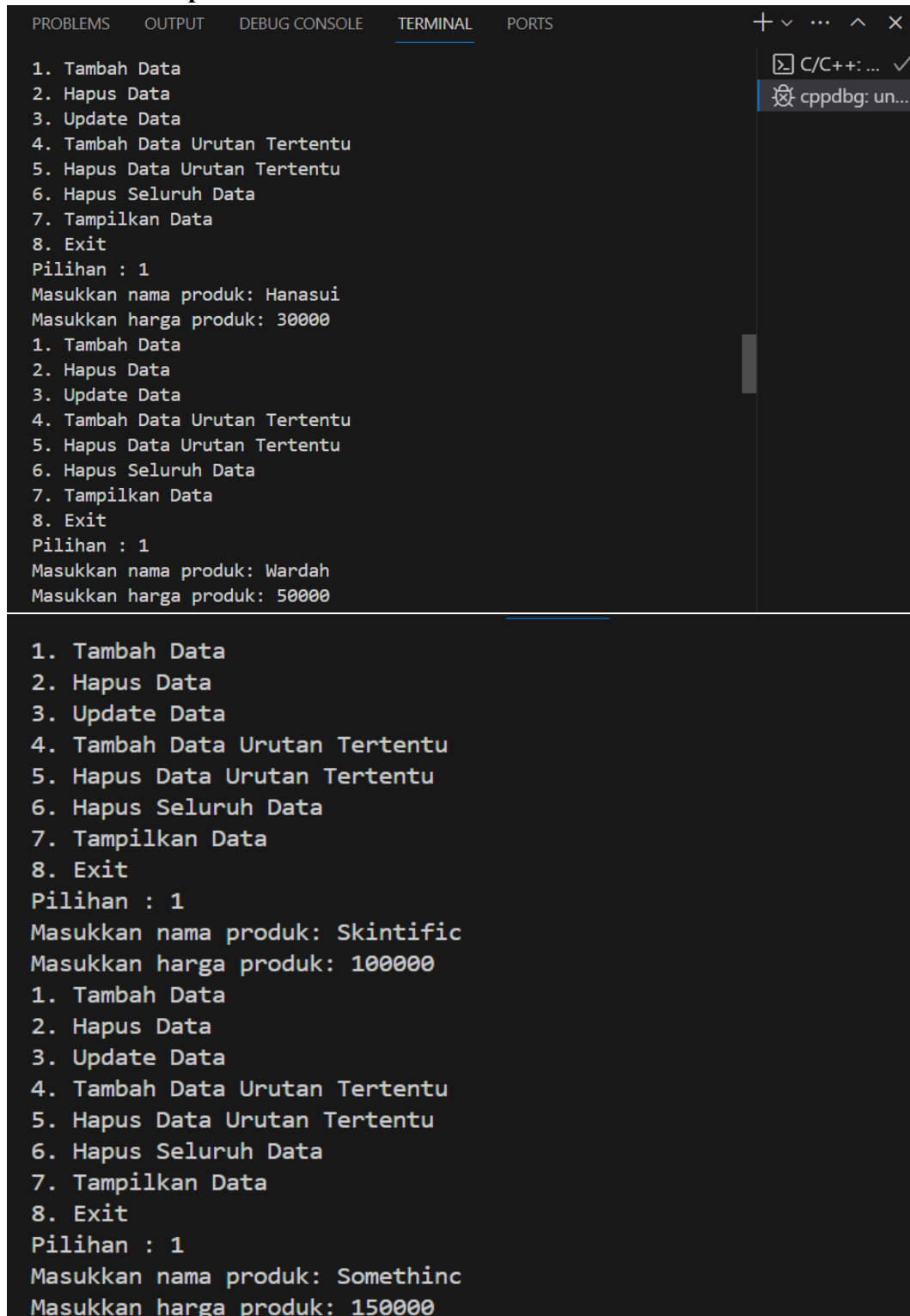


```

case 3: {
    string newNamaProduk;
    int newHarga, posisi;
    cout << "Masukkan posisi produk: ";
    cin >> posisi;
    cout << "Masukkan nama baru produk: ";
    cin >> newNamaProduk;
    cout << "Masukkan harga baru produk: ";
    cin >> newHarga;
    bool updatedCenter =
    list.updateCenter(newNamaProduk, newHarga, posisi);
    if (!updatedCenter){
        cout << "Data not found" << endl;
    }
    break;
}
case 4: {
    string namaProduk;
    int harga, posisi;
    cout << "Masukkan posisi data produk: ";
    cin >> posisi;
    cout << "Masukkan nama produk: ";
    cin.ignore();
    getline(cin, namaProduk);
    cout << "Masukkan harga produk: ";
    cin >> harga;
    list.pushCenter(namaProduk, harga, posisi);
    break;
}
case 5: {
    int posisi;
    cout << "Masukkan posisi data produk: ";
    cin >> posisi;
    list.popCenter(posisi);
    break;
}
case 6: {
    list.deleteAll();
    break;
}
case 7: {
    list.display();
    break;
}
case 8: {
    return 0;
}
default: {
    cout << "Invalid choice" << endl;
    break;
}
} while (choice != 8);
return 0;
}

```

## Screenshot Output



The screenshot shows a C++ IDE with a terminal window. The terminal displays a menu-driven program for product management. The menu options are: 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. The program prompts the user to select an option and then to enter the product name and price. The user has entered 'Hanasui' with a price of 30000, 'Wardah' with a price of 50000, 'Skintific' with a price of 100000, and 'Somethinc' with a price of 150000. The program is currently displaying the menu options again.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
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
Pilihan : 1
Masukkan nama produk: Hanasui
Masukkan harga produk: 30000
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
Pilihan : 1
Masukkan nama produk: Wardah
Masukkan harga produk: 50000
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
Pilihan : 1
Masukkan nama produk: Skintific
Masukkan harga produk: 100000
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
Pilihan : 1
Masukkan nama produk: Somethinc
Masukkan harga produk: 150000
```

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

Pilihan : 1

Masukkan nama produk: Originote

Masukkan harga produk: 60000

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

Pilihan : 7

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



#### IDENTITAS DIRI

File Edit Lihat

IDENTITAS DIRI  
MUHAMAD NASRULLOH  
NIM 2311102044  
S1 IF 11-B|

**Screenshot Output (tambah Azarine 65000 antara Somethinc dan Skintific)**

```
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
Pilihan : 4
Masukkan posisi data produk: 2
Masukkan nama produk: Azarine
Masukkan harga produk: 65000
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
```

Pilihan : 7

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



IDENTITAS DIRI


File Edit Lihat

IDENTITAS DIRI  
MUHAMAD NASRULLOH  
NIM 2311102044  
S1 IF 11-B|

### Screenshot Output (hapus Wardah)

```
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
Pilihan : 5
Masukkan posisi data produk: 4
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
Pilihan : 7
```

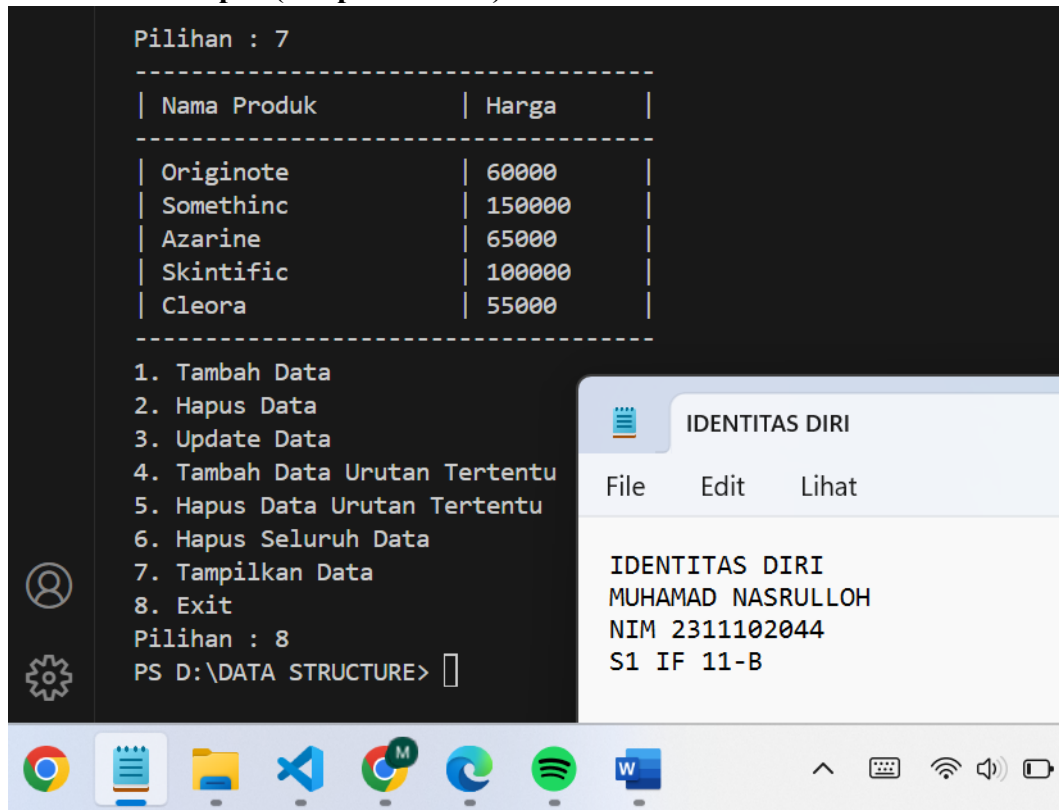
Nama Produk	Harga
Originote	60000
Somethinc	150000
Azarine	65000
Skintific	100000
Hanasui	30000

**IDENTITAS DIRI**  
File Edit Lihat  
  
IDENTITAS DIRI  
MUHAMAD NASRULLOH  
NIM 2311102044  
S1 IF 11-B|

### Screenshot Output (update Hanasui jadi Cleora 55000)

```
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
Pilihan : 3
Masukkan posisi produk: 4
Masukkan nama baru produk: Cleora
Masukkan harga baru produk: 55000
```

### Screenshot Output (tampilkan data)



### Deskripsi

Program diatas menampilkan menu untuk mengelola daftar skincare yang ada di sebuah toko. Berisi daftar nama produk beserta harganya. Dengan program Double Linked List yang memiliki 2 penunjuk, yaitu penunjuk ke node sebelumnya (prev) dan node berikutnya (next). Dalam program ini terdapat fungsi-fungsi seperti : menambah data di depan list (push), menambah data pada posisi tertentu (pushCenter), menghapus data di depan list (pop), menghapus data pada posisi tertentu (popCenter), mengubah data (update), mengubah data pada posisi tertentu (updateCenter), menghapus seluruh data (deleteAll), dan menampilkan data (display). User nantinya akan memilih pilihan dari menu tersebut, lalu program akan menampilkan hasil sesuai inputan user. Program akan terus berjalan sampai user memilih pilihan 8. Exit yang berarti program telah selesai atau berhenti.

#### **D. KESIMPULAN**

Baik Single Linked List maupun Double Linked List semuanya merupakan struktur data linear. Keduanya sama-sama memiliki kelebihan maupun kekurangan, contohnya Single Linked List lebih sederhana dan hemat memori cocok digunakan untuk iterasi maju pada list. Sedangkan Double Linked List lebih kompleks dan boros memori namun, memungkinkan iterasi maju dan mundur karena memiliki 2 pointer : next dan prev, serta lebih efisien untuk operasi pencarian dan penyisipan di tengah list. Dengan memahami karakteristik keduanya diharapkan dapat memilih yang tepat sesuai kebutuhan program yang akan dibuat.

#### **E. REFERENSI**

- [1] Rini Wongso, S.Kom., M.T.I (2017), Single Linked List. BINUS UNIVERSITY: School of Computer Science.  
<https://socs.binus.ac.id/2017/03/15/single-linked-list>. Diakses pada 2 April 2024
- [2] Rini Wongso, S.Kom., M.T.I (2017), Doubly Linked List. BINUS UNIVERSITY: School of Computer Science.  
<https://socs.binus.ac.id/2017/03/15/doubly-linked-list>. Diakses pada 2 April 2024