

**LAPORAN PRAKTIKUM STRUKTUR  
DATA DAN ALGORITMA**

**MODUL III  
SINGLE AND DOUBLE LINKED LIST**



**Disusun oleh:  
Alfin Ilham Berlianto  
2311102047**

**Dosen pengampu:**  
Wahyu Andi Saputra, S.Pd., M.Eng.

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

## MODUL 3

### SINGLE AND DOUBLE LINKED LIST

#### A. Tujuan

1. Mahasiswa memahami perbedaan konsep Single Linked List dan Double Linked List
2. Mahasiswa mampu menerapkan Single dan Double Linked List dalam pemrograman

#### B. Dasar Teori

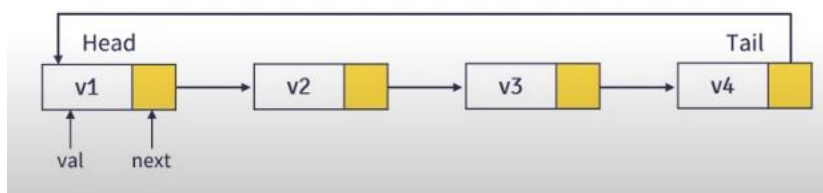
##### Single Linked List

linked list adalah struktur data yang terdiri dari sekelompok node yang terhubung satu sama lain secara berurutan. Setiap node terdiri dari dua bagian utama: data yang sebenarnya disimpan (biasanya disebut sebagai "elemen" atau "node"), dan referensi atau pointer ke simpul berikutnya dalam urutan tersebut.



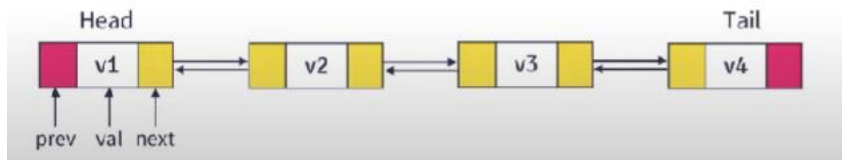
Lalu selain itu di dalam Linked List ini terdapat Circular Single Linked List, apa itu Circular Linked List?

Circular Linked List adalah sekumpulan node atau simpul yang tidak terdapat nilai NULL pada satupun nodenya. Bentuk node pada Circular Single Linked List sama dengan Single Linked List yang mempunyai data dan pointer next. Yang membedakan adalah jika pada Single Linked List node terakhir menunjuk pada ke NULL, tapi di Circular Single Linked List node terakhir menunjuk pada node pertama.

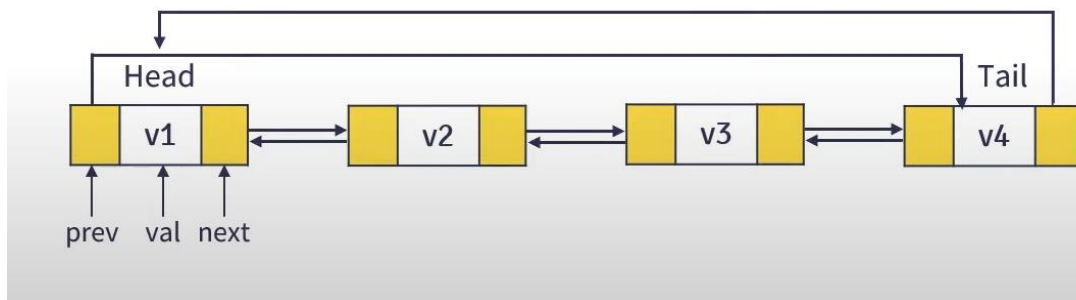


## Double Linked List

Double Linked List adalah struktur data linear yang terdiri dari sekelompok elemen, di mana setiap elemen memiliki dua bagian utama, yaitu data itu sendiri dan dua pointer (atau referensi) yang menunjuk ke elemen sebelumnya atau biasa kata kuncinya 'prev' dan sesudahnya dalam list 'next'. Dan pada tail bernilai NULL.



Lalu di dalam Double Linked List ini juga terdapat yang namanya Circular Double Linked List, yaitu struktur node sama seperti Double Linked List yang mana terdapat data, pointer prev dan pointer next. Tetapi karena Circular linked list maka pointer next pada tail menunjukan ke head dan pointer prev pada head menunjukan ke tail.



## B. Guided

### Guided 1

#### 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 Windows command prompt window with the following commands and output:

```

PS C:\Users\sjsjs\Desktop\code c++\modul linked list> cd "c:\Users\sjsjs\Desktop\code c++\modul linked li
st\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
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 C:\Users\sjsjs\Desktop\code c++\modul linked list>

```

Next to the command prompt is a Notepad window titled "Alfin Ilhar" containing the following text:

```

Alfin Ilham Berlianto
2311102047
IF-11-B

```

## DESKRIPSI PROGRAM

Pada program ini membuat data menggunakan Single Linked List, Yang mana di dalam program ini memiliki 14 Fungsi atau prosedur yang di setiap fungsi atau prosedurnya memiliki kegunaan masing-masing, Seperti contoh yang digunakan di dalam main program ada `init()`, void ini adalah untuk menginisialisasikan Node awal yang head dan tail nya masih bernilai NULL, Lalu ada `insertDepan()`, fungsi ini kegunaanya untuk menambahkan node di awal data, `insertBelakang()` kegunaannya untuk menambahkan data setelah node awal dibuat, `hapusBelakang()` fungsinya untuk menghapus node yang berada di belakang data. `insertTengah()` fungsinya menambahkan node di Tengah-tengah data yang sudah dibuat, `hapusTengah()` fungsinya untuk menghapus node yang berada di urutan Tengah atau setelah awal dan sebelum akhir, `ubahDepan()` berguna untuk mengedit node yang sudah



dibuat diawal data, ubahTengah() berfungsi untuk mengedit node yang telah dibuat di Tengah-tengah data atau setelah awal node sampai sebelum akhir node.ubahBelakang() berfungsi untuk mengubah data atau node yang berada di urutan belakang.

## Guided 2

### 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;
            }
        }
    }
}

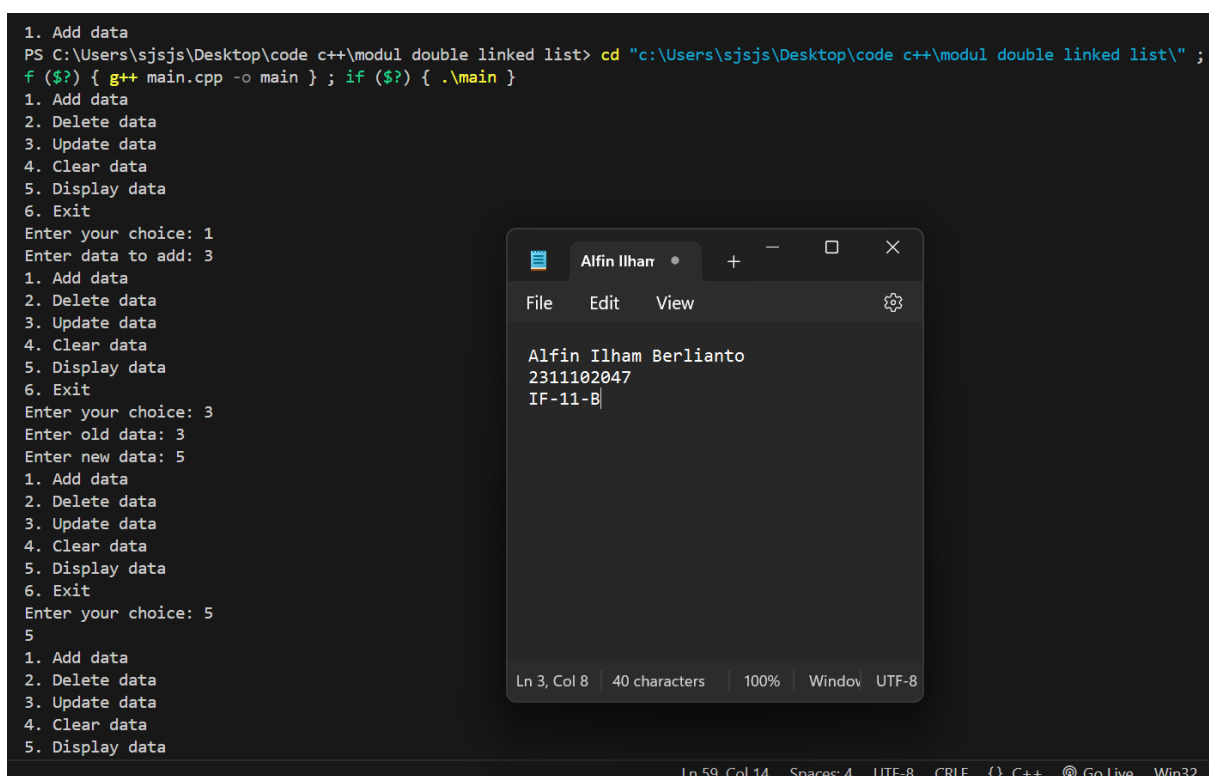
```

```

        default: {
            cout << "Invalid choice" << endl;
            break;
        }
    }
}
return 0;
}

```

## SCREENSHOT OUTPUT



```

1. Add data
PS C:\Users\sjsjs\Desktop\code c++\modul double linked list> cd "c:\Users\sjsjs\Desktop\code c++\modul double linked list\" ;
f ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
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: 3
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 3
Enter old data: 3
Enter new data: 5
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
5
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data

```

Alfin Ilham Berlianto  
2311102047  
IF-11-B

Ln 3, Col 8 | 40 characters | 100% | Window UTF-8

Ln 59, Col 14 | Spaces: 4 | UTF-8 | CRLF | {} C++ | Go Live Win32

## DESKRIPSI PROGRAM

Pada program ini membuat menu dengan menggunakan Double Linked List, yang dimana dalam program tersebut terdapat 5 menu, yang 1 ada add data yang bertujuan untuk menginput node awal/baru yang diisi oleh user, lalu ke 2 Delete data untuk menghapus node-node yang telah dibuat sebelumnya, yang ke 3 ada Update data yang bertujuan untuk mengedit data/node yang nantinya akan ditanya di posisi mana node yang ingin diubah, ke 4 ada clear data berguna untuk menghapus semua data yang telah dibuat, yang ke 5 ada display data yaitu untuk menampilkan dari menu-menu sebelumnya, seperti node yang telah dibuat, node yang telah diedit, atau node yang telah dihapus.

## C. Unguided

### Unguided 1

#### SOURCE CODE

```
#include <iostream>

using namespace std;

// deklarasi single linked list
struct Data{

    // komponen / member
    string nama;
    int umur;

    Data *next;

};

Data *head, *tail, *cur, *newNode, *del, *before;

// create single linked list
void createSingleLinkedList(string nama, int umur){
    head = new Data;
    head->nama = nama;
    head->umur = umur;
    head->next = NULL;
    tail = head;
}

// print single linked list
int countSingleLinkedList(){
    cur = head;
    int jumlah = 0;
    while( cur != NULL ){
        jumlah++;
        cur = cur->next;
    }
    return jumlah;
}

// tambahAwal Single linked list
void addFirst(string nama, int umur){
    newNode = new Data;
    newNode->nama = nama;
    newNode->umur = umur;
```

```

newNode->next = NULL;
if(head == NULL) {
    head = tail = newNode;
} else {
newNode->next = head;
    head = newNode;
}
}

// tambahAkhir Single linked list
void addLast(string nama, int umur){
    newNode = new Data;
    newNode->nama = nama;

    newNode->umur = umur;
    newNode->next = NULL;
    if(head == NULL) {
        head = tail = newNode;
    } else {
tail->next = newNode;
        tail = newNode;
    }

}

// tambah tengah single linked list
void addMiddle(string nama, int umur, int posisi){
    if( posisi < 1 || posisi > countSingleLinkedList() ){
        cout << "Posisi diluar jangkauan" << endl;
    }else if( posisi == 1){
        cout << "Posisi bukan posisi tengah" << endl;
    }else{
        newNode = new Data;
        newNode->nama = nama;

        newNode->umur = umur;

        // tranversing
        cur = head;
        int nomor = 1;
        while( nomor < posisi - 1 ){
            cur = cur->next;
            nomor++;
        }
        newNode->next = cur->next;
        cur->next = newNode;
    }
}

```

```

// Remove First
void removeFirst(){
    del = head;
    head = head->next;
    delete del;
}

// Remove Last
void removeLast(){
    del = tail;
    cur = head;
    while( cur->next != tail ){
        cur = cur->next;
    }
    tail = cur;
    tail->next = NULL;
    delete del;
}

// remove middle
void removeMiddle(int posisi){
    if( posisi < 1 || posisi > countSingleLinkedList() ){
        cout << "Posisi diluar jangkauan" << endl;
    }else if( posisi == 1){
        cout << "Posisi bukan posisi tengah" << endl;
    }else{
        int nomor = 1;
        cur = head;
        while( nomor <= posisi ){
            if( nomor == posisi-1 ){
                before = cur;
            }
            if( nomor == posisi ){
                del = cur;
            }
            cur = cur->next;
            nomor++;
        }
        before->next = cur;
        delete del;
    }
}

// ubahAwal Single linked list
void changeFirst(string nama, int umur){
    head->nama = nama;
    head->umur = umur;
}

// ubahAkhir Single linked list

```

```

void changeLast(string nama, int umur){
    tail->nama = nama;

    tail->umur = umur;
}

// ubah Tengah Single linked list
void changeMiddle(string nama, int umur, int posisi){
    if( posisi < 1 || posisi > countSingleLinkedList() ){
        cout << "Posisi diluar jangkauan" << endl;
    }else if( posisi == 1 || posisi == countSingleLinkedList() ){
        cout << "Posisi bukan posisi tengah" << endl;
    }else{
        cur = head;
        int nomor = 1;
        while( nomor < posisi ){
            cur = cur->next;
            nomor++;
        }
        cur->nama = nama;

        cur->umur = umur;
    }
}

// print single linked list
void printSingleLinkedList(){
    if(head != NULL) {
        cur = head;
        while( cur != NULL ){
            cout << cur->nama << "\t";

            cout << cur->umur << endl;

            cur = cur->next;
        }
        cout << endl;
    }
}

int main(){
    int choose, umur, posisi;
    string nama;
    cout << "Data Mahasiswa" << endl;
    do
    {
        cout << " 1. Tambah Depan" << endl;
        cout << " 2. Tambah Belakang" << endl;
        cout << " 3. Tambah Tengah" << endl;
        cout << " 4. Hapus Depan" << endl;
    }
}

```



```

cout << " 5. Hapus Belakang"<< endl;
cout << " 6. Hapus Tengah"<< endl;
cout << " 7. Ubah Depan"<< endl;
cout << " 8. Ubah Belakang"<< endl;
cout << " 9. Ubah Tengah"<< endl;
cout << " 10. Tampilkan"<< endl;
cout << " 0. Keluar Program"<< endl;
cout << " Pilihan : ";
cin >> choose;
switch (choose)
{
case 1:
    cout << "Masukkan Nama : ";
    cin >> nama;
    cout << "Masukkan umur : ";
    cin >> umur;
    addFirst(nama,umur);
    cout << endl;

    break;
case 2:
    cout << "Masukkan Nama : ";
    cin >> nama;
    cout << "Masukkan umur : ";
    cin >> umur;
    addLast(nama,umur);
    cout << endl;

    break;
case 3:
    cout << "Masukkan Posisi : ";
    cin >> posisi;
    cout << "Masukkan Nama : ";
    cin >> nama;
    cout << "Masukkan umur : ";
    cin >> umur;
    addMiddle(nama,umur,posisi);
    cout << endl;

    break;
case 4:
    removeFirst();
    cout << endl;

    break;
case 5:
    removeLast();
    cout << endl;

    break;

```

```

case 6:
    cout << "Masukkan Posisi : ";
    cin >> posisi;
    removeMiddle(posisi);
    cout << endl;

    break;
case 7:
    cout << "Masukkan Nama : ";
    cin >> nama;
    cout << "Masukkan umur : ";
    cin >> umur;
    changeFirst(nama,umur);
    cout << endl;

    break;
case 8:
    cout << "Masukkan Nama : ";
    cin >> nama;
    cout << "Masukkan umur : ";
    cin >> umur;
    changeLast(nama,umur);
    cout << endl;

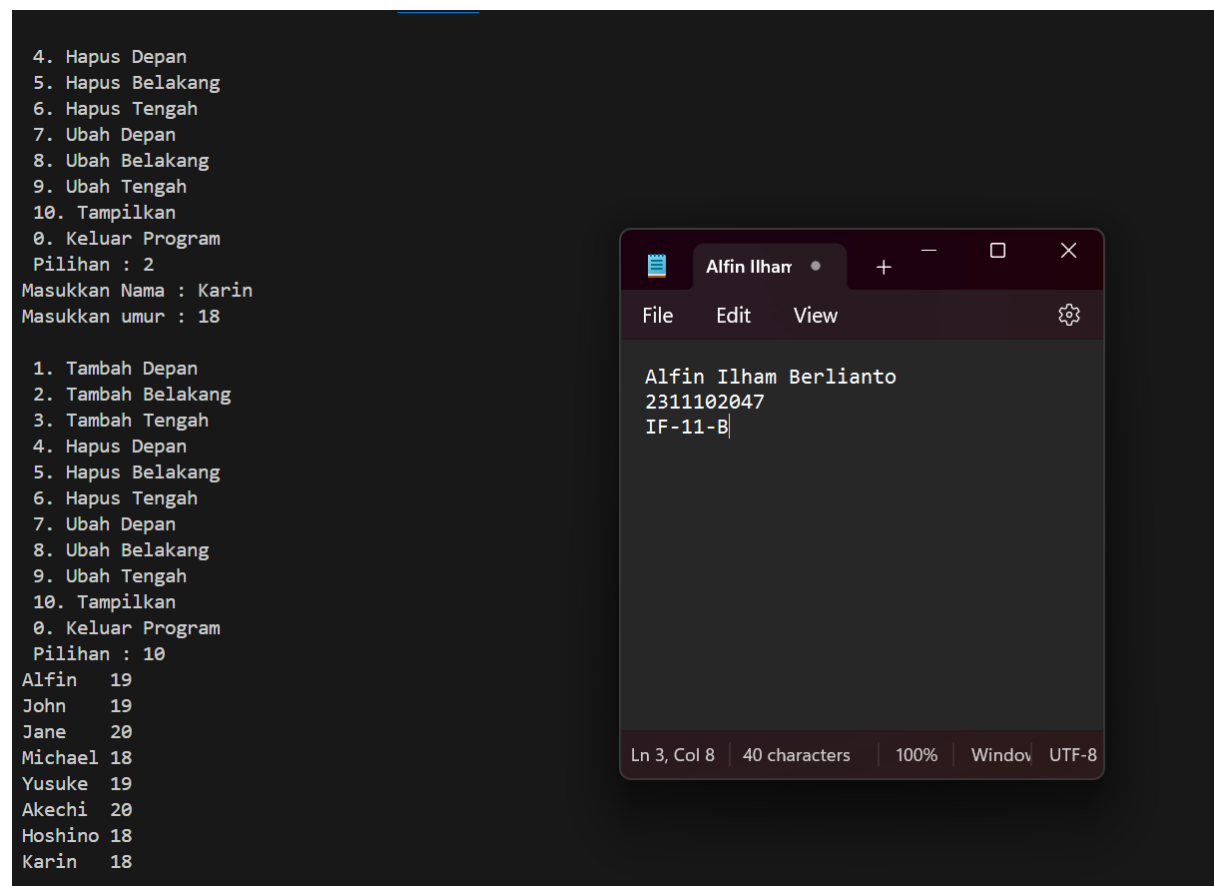
    break;
case 9:
    cout << "Masukkan Posisi : ";
    cin >> posisi;
    cout << "Masukkan Nama : ";
    cin >> nama;
    cout << "Masukkan umur : ";
    cin >> umur;
    changeMiddle(nama,umur,posisi);
    cout << endl;

    break;
case 10:
    printSingleLinkedList();
    break;
case 0:
    cout << "terimakasih";
    break;
default:
    cout << "Pilihan Salah" << endl;
    break;
}
} while (choose != 0);
}

```

## SCREENSHOT OUTPUT

Data awal



The screenshot displays a terminal window on the left and a text editor window on the right. The terminal window shows a menu-driven program with options to add, delete, or change data. The user has selected option 2 to add data, entered the name 'Karin' and age '18'. The program then displays a list of data entries. The text editor window, titled 'Alfin Ilham', shows the text 'Alfin Ilham Berlianto', '2311102047', and 'IF-11-B'.

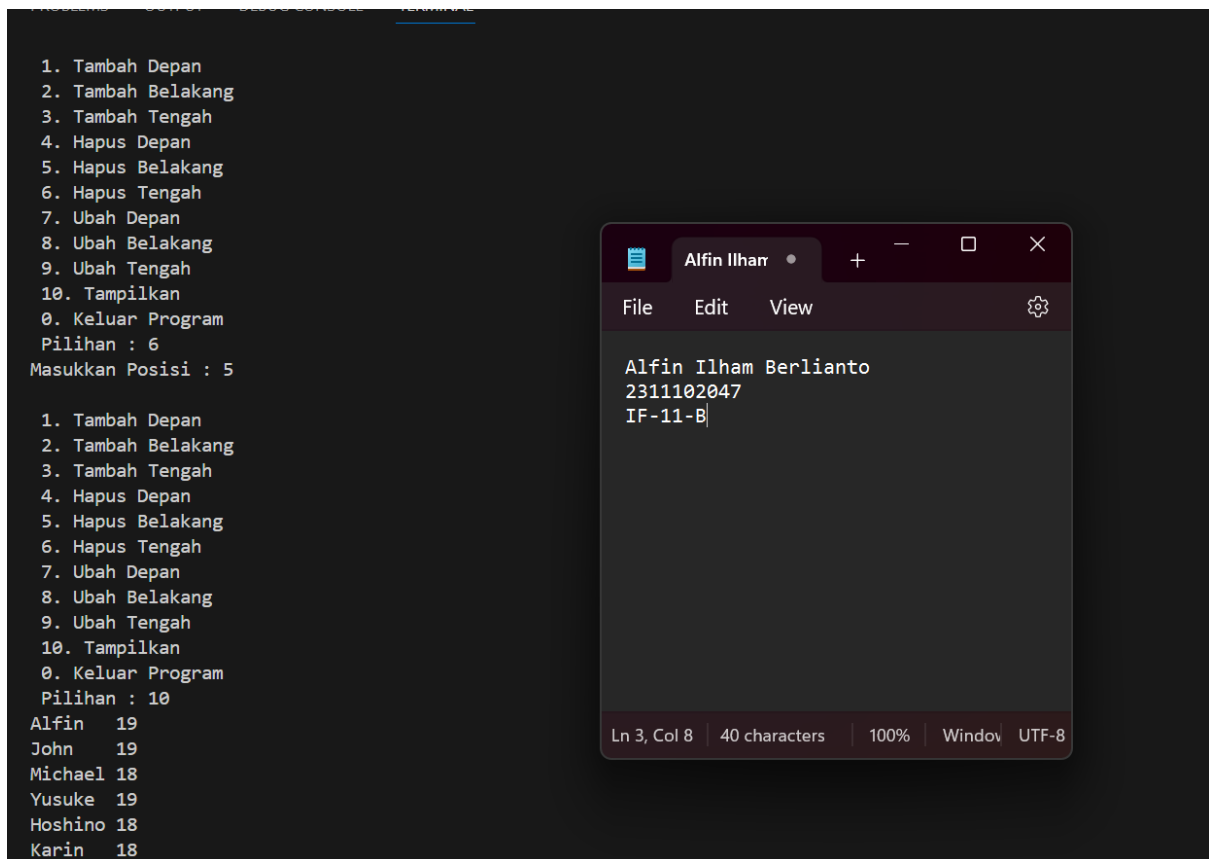
```
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 2
Masukkan Nama : Karin
Masukkan umur : 18

1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 10
Alfin 19
John 19
Jane 20
Michael 18
Yusuke 19
Akechi 20
Hoshino 18
Karin 18
```

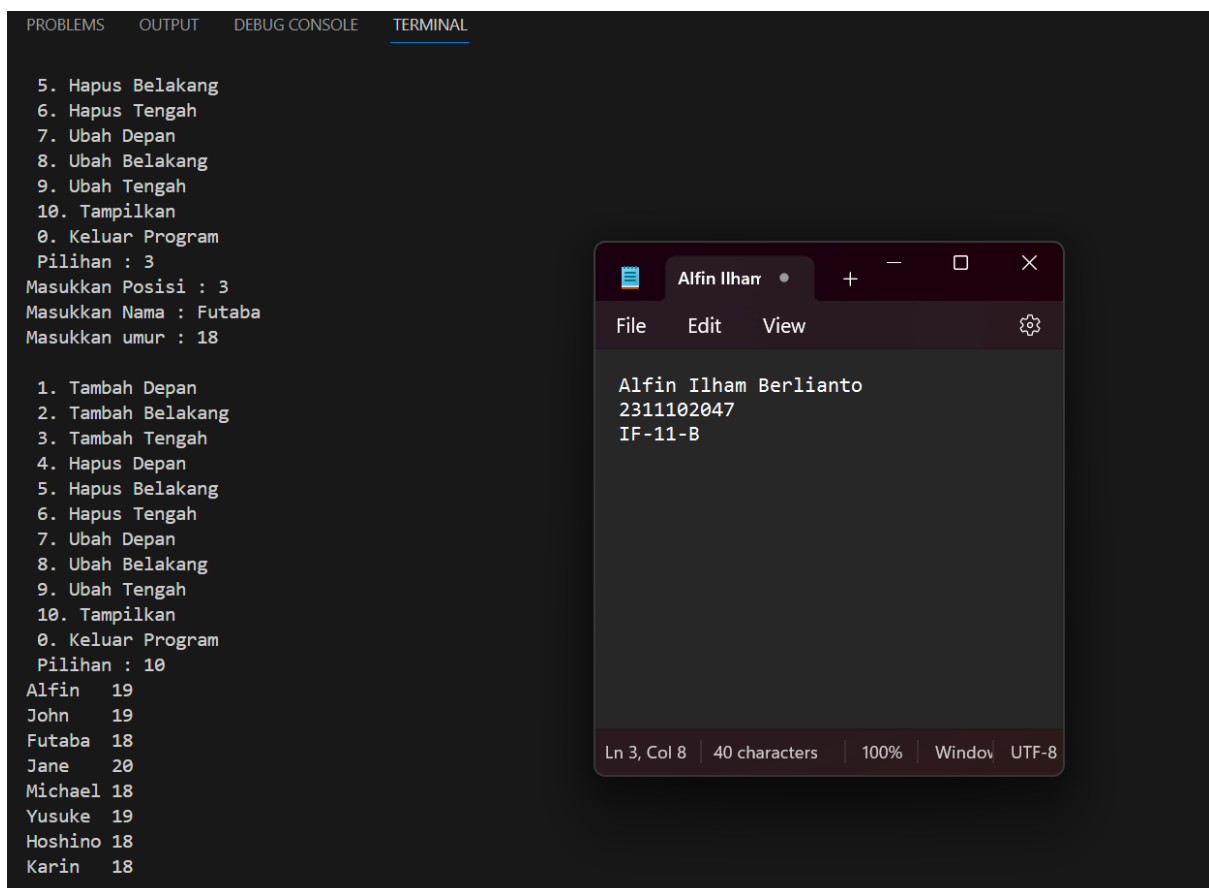
Alfin Ilham Berlianto  
2311102047  
IF-11-B

Ln 3, Col 8 | 40 characters | 100% | Window | UTF-8

Hapus data Akechi



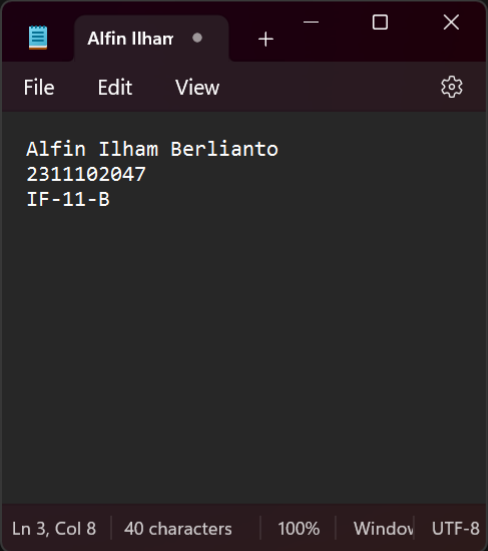
## Tambah data Tengah



## Tambah Data Awal

```
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 1
Masukkan Nama : Igor
Masukkan umur : 20

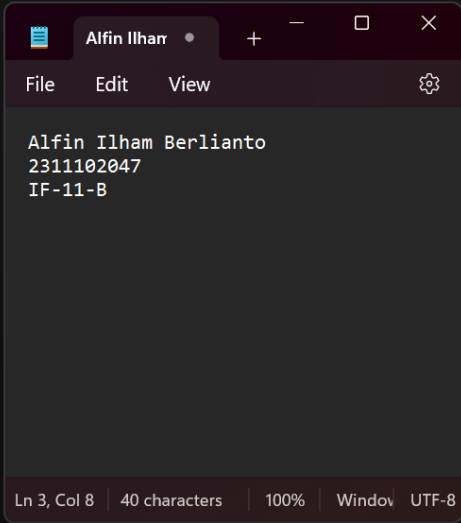
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 10
Igor    20
Alfin   19
John    19
Futaba  18
Jane    20
Michael 18
Yusuke  19
Hoshino 18
Karin   18
```



## Ubah data dan tampilkan semua data

```
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 9
Masukkan Posisi : 6
Masukkan Nama : Reyn
Masukkan umur : 18

1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 10
Igor 20
Alfin 19
John 19
Futaba 18
Jane 20
Reyn 18
Yusuke 19
Hoshino 18
Karin 18
```



## DESKRIPSI PROGRAM

Pada program ini membuat data nama dan usia mahasiswa dengan menggunakan Single Linked List non-circular, dimana di dalam program tersebut terdapat 10 menu yang di mana masing-masing menu tersebut memiliki eksekusinya tersendiri, untuk menu yang ke 1 tambah depan yaitu membuat node jika sebelumnya belum dibuat, dan jika sudah dibuat, akan menaruh node yang baru dibuat itu ke depan data, yang ke 2 ada tambah belakang yaitu untuk menambahkan node setelah node awal dibuat, yang ke 3 ada tambah Tengah yang berfungsi untuk menambahkan node di Tengah-tengah spesifiknya interval setelah awal node sampai sebelum akhir node, yang ke 4 ada hapus depan yaitu untuk menghapus node yang berada di awal data, yang ke 5 ada hapus belakang yaitu untuk menghapus node yang telah dibuat yang urutannya paling belakang dari node yang ada, yang ke 6 ada hapus Tengah yang berguna untuk menghapus node yang sudah dibuat intervalnya dari setelah node pertama sampai sebelum node akhir. Yang ke 7 ada ubah depan yang berfungsi untuk mengedit node yang telah dibuat yang eksekusi editnya ini di awal data, yang ke 8 ubah belakang berguna untuk mengubah node yang telah dibuat yang pengeksekusiannya dilakukan di akhir data, yang ke 9 ada ubah Tengah yaitu berfungsi untuk mengedit data atau node di Tengah-tengah antara setelah awal node sampai sebelum akhir node yang telah dibuat. Yang 10 ada menu tampilkan, menu ini berfungsi untuk menampilkan dari menu-menu yang sebelumnya, pada awalnya outputnya berisi data yang sesuai di screenshot, akan tetapi terdapat manipulasi data yang telah dilakukan sampai hasil akhirnya yang seperti di screenshot akhir,

## Unguided 2

### SOURCE CODE

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

class Node
{
public:
    string namaProduk;
    int harga;
    Node *prev;
    Node *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 pushMiddle(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 popMiddle(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 updateMiddle(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 bukan 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 print()
    {
        if (head == nullptr)
        {
            cout << "List kosong." << endl;
            return;
        }

        Node *current = head;

        cout << setw(20) << left << "Nama Produk " << setw(10) << left << "Harga Produk" <<
endl;

        while (current != nullptr)
        {
            cout << setw(20) << left << current->namaProduk << setw(10) << left << current-
>harga << endl;
            current = current->next;
        }
        cout << endl;
    }
};

int main()
{
    DoublyLinkedList list;
    int choose;
    cout << endl
        << "Toko Ambatukam" << 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 >> choose;

        switch (choose)
        {
            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 p1osisi produk: ";
    cin >> posisi;
    cout << "Masukkan nama baru produk: ";
    cin >> newNamaProduk;
    cout << "Masukkan harga baru produk: ";
    cin >> newHarga;
    bool updatedMiddle = list.updateMiddle(newNamaProduk, newHarga, posisi);
    if (!updatedMiddle)
    {
        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.pushMiddle(namaProduk, harga, posisi);
    break;
}
case 5:
{
    int posisi;
    cout << "Masukkan posisi data produk: ";
    cin >> posisi;

```

```
        list.popMiddle(posisi);
        break;
    }
    case 6:
    {
        list.deleteAll();
        break;
    }
    case 7:
    {
        list.print();
        break;
    }
    case 8:
    {
        return 0;
    }
    default:
    {
        cout << "Invalid choice" << endl;
        break;
    }
}
} while (choose != 8);

return 0;
}
```

## SCREENSHOT OUTPUT

Data awal

```

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: Originate
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 Produk
Originate	60000
Somethinc	150000
Skintific	100000
Wardah	50000
Hanasui	30000

Tambah data tengah

```

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: 3
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

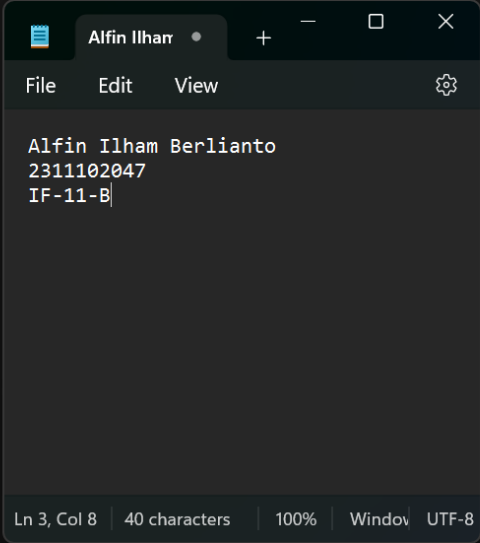
```

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

## Hapus produk

```
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: 3
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 Produk
Originate        60000
Somethinc        150000
Skintific        100000
Azarine          65000
Hanasui          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
```



The screenshot shows a text editor window with the title 'Alfin Ilham'. The window contains a list of names and IDs, with the cursor positioned at the end of the third line. The status bar at the bottom indicates 'Ln 3, Col 8 | 40 characters | 100% | Window | UTF-8'.

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

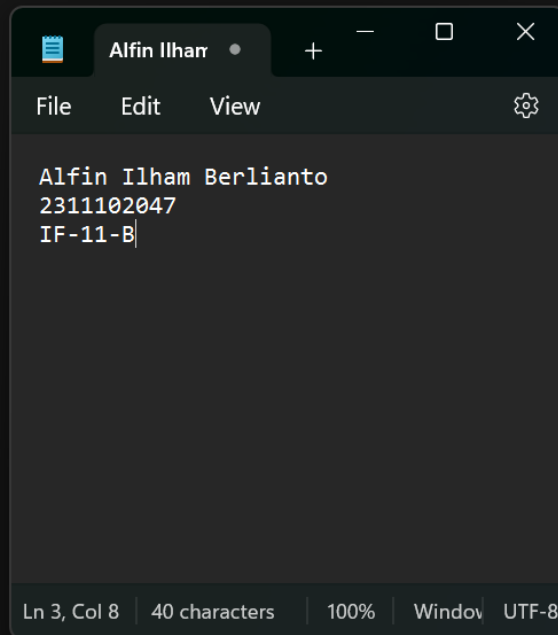
## Update Produk



```

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 plosisi produk: 3
Masukkan nama baru produk: Skintific
Masukkan harga baru 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 : 7
Nama Produk      Harga Produk
Originate        60000
Somethinc        150000
Azarine          55000
Skintific        100000
Cleora           55000

```



## DESKRIPSI PROGRAM

Pada program ini adalah modifikasi dari guided yang telah dibuat menggunakan Double Linked List, terdapat 7 menu yang ada di program ini, pada menu 1 ada tambah data, menu ini berfungsi untuk menambah node yang datanya diisi oleh user yang tidak ada batasannya, menu tambah data ini sudah termasuk seperti menu yang ada di unguided sebelumnya yaitu menu tambah belakang, lalu menu ke 2 ada hapus data, menu ini berfungsi menghapus semua data yang sudah dibuat, yang ke 3 ada menu update data, yaitu menu yang bertujuan mengedit node yang telah dibuat yang updatenya tergantung posisi mana yang diinginkan oleh user, lalu menu ke 4 ada menu tambah data urutan tertentu, menu ini sama saja seperti tambah data Tengah, yang penambahannya itu tergantung di posisi mana user ingin menaruh, yang ke 5 ada hapus data urutan tertentu, menu ini untuk menghapus di Tengah-tengah data tergantung posisi mana yang diinginkan oleh user, menu ke 6 ada hapus seluruh data, menu ini berfungsi untuk menghapus semua node yang telah dibuat, yang ke 7 ada menu tampilkan data, menu ini fungsinya adalah menampilkan dari menu-menu sebelumnya yang sudah diinput.

## **Kesimpulan**

### **1. Single Linked List:**

- Merupakan struktur data yang terdiri dari sekelompok node, di mana setiap node memiliki dua bagian: data dan pointer ke node berikutnya dalam urutan.
- Penambahan dan penghapusan elemen biasanya lebih cepat di bagian depan list karena hanya perlu mengubah pointer pertama.
- Akses ke elemen berdasarkan indeks tidak langsung didukung; untuk mengakses elemen, harus dilakukan pencarian dari awal.

### **Double Linked List:**

#### **2. Double Linked List**

- Sama seperti single linked list, tetapi setiap node memiliki dua pointer: satu menuju node berikutnya dan satu menuju node sebelumnya.
- Memungkinkan traversal maju dan mundur, sehingga akses ke elemen berdasarkan indeks bisa lebih cepat dibandingkan dengan single linked list.
- Operasi penambahan dan penghapusan elemen dapat dilakukan dengan cepat di bagian depan atau belakang list, serta di tengah list dengan sedikit modifikasi.

## Referensi

1. Konsep dan Implementasi Double Linked List Dasar di C++ | STRUKTUR DATA  
<https://www.youtube.com/watch?v=HoubOPoC44s&t=1470s>
2. Konsep dan Implementasi Single Linked List Dasar di C++ | STRUKTUR DATA  
<https://www.youtube.com/watch?v=VVemCxif9vg&t=466s>
3. geeksforgeeks (2014. 29 Juli) Introduction to Singly Linked List. Diakses pada 31 Maret 2024, dari  
[https://www.google.com/search?q=single+linked+list&rlz=1C1KNTJ\\_enID1075ID1075&oq=single+link&gs\\_lcrp=EgZjaHJvbWUqBwgAEAAAYgAQyBwgAEAAAYgAQyBggBEEUYO TIJCAIQABgKGIAEMgcIAxAAAGIAEMgcIBBAAGIAEMgYIBRBFGDwyBggGEEUYPTI GCAcQRRg9qAIAsAIA&sourceid=chrome&ie=UTF-8](https://www.google.com/search?q=single+linked+list&rlz=1C1KNTJ_enID1075ID1075&oq=single+link&gs_lcrp=EgZjaHJvbWUqBwgAEAAAYgAQyBwgAEAAAYgAQyBggBEEUYO TIJCAIQABgKGIAEMgcIAxAAAGIAEMgcIBBAAGIAEMgYIBRBFGDwyBggGEEUYPTI GCAcQRRg9qAIAsAIA&sourceid=chrome&ie=UTF-8)