

**LAPORAN
STRUKTUR DATA DAN ALGORITMA**

**MODUL III
SINGLE AND DOUBLE LINKED LIST**



DISUSUN OLEH :

Nama: Ahmad Titana Nanda Pramudya
Nim : (2311102042)

Dosen

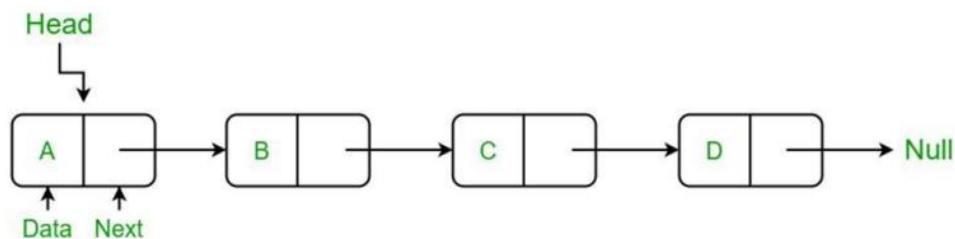
Wahyu Andi Saputra, S.Pd. , M.Eng

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

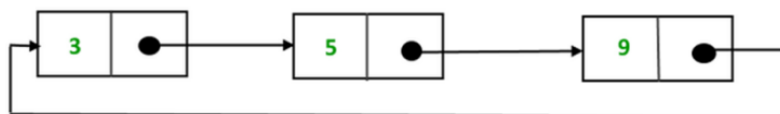
A. Dasar Teori

1. Single linked list

Single Linked List merupakan suatu bentuk struktur data yang berisi kumpulan data yang disebut sebagai node yang tersusun secara sekuensial, saling sambung menyambung, dinamis, dan terbatas. Setiap elemen dalam linked list dihubungkan ke elemen lain melalui pointer. Masing-masing komponen sering disebut dengan simpul atau node atau verteks. Pointer adalah alamat elemen. Setiap simpul pada dasarnya dibagi atas dua bagian pertama disebut bagian isi atau informasi atau data yang berisi nilai yang disimpan oleh simpul. Bagian kedua disebut bagian pointer yang berisi alamat dari node berikutnya atau sebelumnya. 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.

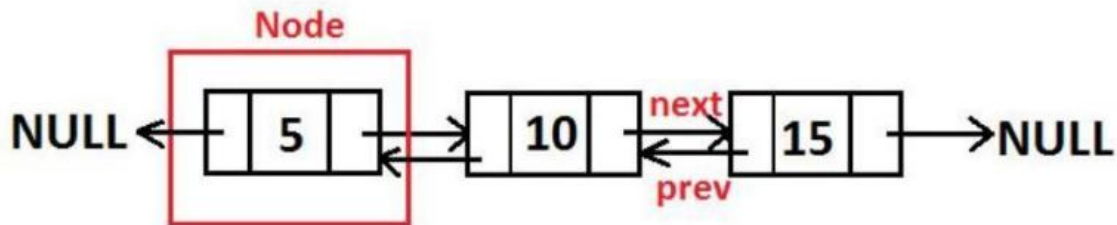


Dalam operasi Single Linked List, umumnya dilakukan operasi penambahan dan penghapusan simpul pada awal atau akhir daftar, serta pencarian dan pengambilan nilai pada simpul tertentu dalam daftar. Karena struktur data ini hanya memerlukan satu pointer untuk setiap simpul, maka Single Linked List umumnya lebih efisien dalam penggunaan memori dibandingkan dengan jenis Linked List lainnya, seperti Double Linked List dan Circular Linked List. Single linked list yang kedua adalah circular linked list. Perbedaan circular linked list dan non circular linked adalah penunjuk next pada node terakhir pada circular linked list akan selalu merujuk ke node pertama.



2. Double Linked list

Double Linked List adalah struktur data Linked List yang mirip dengan Single Linked List, namun dengan tambahan satu pointer tambahan pada setiap simpul yaitu pointer prev yang menunjuk ke simpul sebelumnya. Dengan adanya pointer prev, Double Linked List memungkinkan untuk melakukan operasi penghapusan dan penambahan pada simpul mana saja secara efisien. Setiap simpul pada Double Linked List memiliki tiga elemen penting, yaitu elemen data (biasanya berupa nilai), pointer next yang menunjuk ke simpul berikutnya, dan pointer prev yang menunjuk ke simpul sebelumnya. Keuntungan dari Double Linked List adalah memungkinkan untuk melakukan operasi penghapusan dan penambahan pada simpul dimana saja dengan efisien, sehingga sangat berguna dalam implementasi beberapa algoritma yang membutuhkan operasi tersebut. Selain itu, Double Linked List juga memungkinkan kita untuk melakukan traversal pada list baik dari depan (head) maupun dari belakang (tail) dengan mudah. Namun, kekurangan dari Double Linked List adalah penggunaan memori yang lebih besar dibandingkan dengan Single Linked List, karena setiap simpul membutuhkan satu pointer tambahan. Selain itu, Double Linked List juga membutuhkan waktu eksekusi yang lebih lama dalam operasi penambahan dan penghapusan jika dibandingkan dengan Single Linked List. Representasi sebuah double linked list dapat dilihat pada gambar berikut ini:



Di dalam sebuah linked list, ada 2 pointer yang menjadi penunjuk utama, yakni pointer HEAD yang menunjuk pada node pertama di dalam linked list itu sendiri dan pointer TAIL yang menunjuk pada node paling akhir di dalam linked list. Sebuah linked list dikatakan kosong apabila isi pointer head adalah NULL. Selain itu, nilai pointer prev dari HEAD selalu NULL, karena merupakan data pertama. Begitu pula dengan pointer next dari TAIL yang selalu bernilai NULL sebagai penanda data terakhir.

B. Guided

Guided 1

Sourcode :

```
#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;
}

```

Output :

```
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:\contoh>
```

Rea

File Edit View

Nama : Ahmad Titana Nanda Pramudya
Nim : 2311102042
Kelas: IF-11-B

Ln 3, Col 15 | 67 characters | 100% | Window | UTF-8

Deskripsi :

Program diatas merupakan program Double Linked List Non-Circular (DLLNC). Pada program awal Mendeklarasikan Struct dinamain Node berisi 3 field yaitu field data bertipe data integer, field next bertipe pointer dan field prev bertipe pointer. Mendeklarasikan pointer head, tail baru, bantu, bantu2, dan hapus ke struct node. Pada program ini memiliki 15 fungsi, untuk fungsi pertama yaitu fungsi init yaitu untuk memberikan nilai awal node pada list kosong dengan memberikan nilai NULL pada node head dan tail., selanjutnya isEmpty Operasi untuk memeriksa apakah suatu linked list masih kosong , selanjutnya fungsi create untuk membuat node baru, Fungsi selanjutnya yaitu insertdepan, insertbelakang dan insert tengah Operasi untuk menambahkan satu node ke dalam linked list didepan, dibelakang dan ditengah. Fungsi delete Operasi untuk menghapus node pada linked list. Yang berada di depan, belakang ataupun tengah, selanjutnya fungsi countlist untuk menghitung jumlah list, fungsi selanjutnya merubah list yang berada didepan, dibelakang dan ditengah. Fungsi selanjutnya yaitu clearlist untuk menghapus semua data list yang sudah diinputkan oleh user. Dan fungsi terakhir yaitu display untk menampilkan list yang sudah diinputkan. Selanjutnya program utama dan pengaplikasian program tersebut, berawal memanggil fungsi init dan seterusnya seperti di dalam program yang di atas.

Guided 2

Sourcode :

```
#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();
            }
        }
    }
}

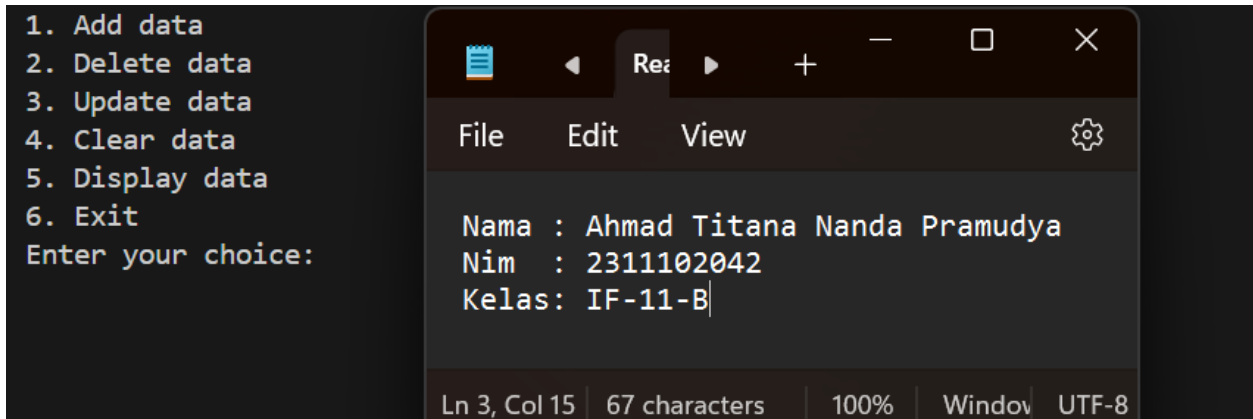
```

```
        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;
}
```

Output:



```
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5

Nama : Ahmad Titana Nanda Pramudya
Nim : 2311102042
Kelas: IF-11-B
```

Deskripsi :

Program ini adalah implementasi dari Doubly Linked List (Daftar Berantai Ganda) dalam C++. Doubly Linked List adalah struktur data berantai di mana setiap node memiliki dua pointer, yaitu pointer ke node sebelumnya (prev) dan pointer ke node selanjutnya (next). Program ini memiliki fitur-fitur untuk menambah data ke awal daftar (push), menghapus data dari awal daftar (pop), mengupdate data, menghapus semua data, dan menampilkan semua data yang ada dalam daftar. Program ini menggunakan menu sederhana untuk memilih operasi yang ingin dilakukan pada daftar berantai ganda.

C. Unguided

Unguided 1

Sourcode

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

struct Node
{
    string nama;
    int usia;
    Node *next;
};

Node *head;
Node *tail;

void init()
{
    head = NULL;
    tail = NULL;
}

bool isEmpty()
{
    return head == NULL;
}

void insertDepan(string nama, int usia)
{
    Node *baru = new Node;
    baru->nama = nama;
    baru->usia = usia;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}

void insertBelakang(string nama, int usia)
{
    Node *baru = new Node;
```

```

    baru->nama = nama;
    baru->usia = usia;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}

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

void insertTengah(string nama, int usia, 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->nama = nama;
        baru->usia = usia;
        Node *bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
    }
}

```

```

        bantu->next = baru;
    }
}

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;
    }
}

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;
    }
}

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;
    }
}

void ubahDepan(string nama, int usia)
{
    if (!isEmpty())
    {
        head->nama = nama;
        head->usia = usia;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

void ubahTengah(string nama, int usia, 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->nama = nama;
            bantu->usia = usia;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

void ubahBelakang(string nama, int usia)
{
    if (!isEmpty())
    {
        tail->nama = nama;
        tail->usia = usia;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

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

void tampil()
{

```

```

    if (!isEmpty())
    {
        Node *bantu = head;
        while (bantu != NULL)
        {
            cout << bantu->nama << " ";
            cout << bantu->usia << " , ";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

int main()
{
    init();
    int menu, usia, posisi;
    string nama;
    cout << "\n# Menu Linked List Mahasiswa #" << endl;
    do
    {
        cout << "\n 1. Insert Depan"
              << "\n 2. Insert Belakang"
              << "\n 3. Insert Tengah"
              << "\n 4. Hapus Depan"
              << "\n 5. Hapus Belakang"
              << "\n 6. Hapus Tengah"
              << "\n 7. Ubah Depan"
              << "\n 8. Ubah Belakang"
              << "\n 9. Ubah Tengah"
              << "\n 10. Tampilkan"
              << "\n 0. Keluar Program"
              << "\n Pilihan : ";

        cin >> menu;
        switch (menu)
        {
            case 1:
                cout << "Masukkan Nama : ";
                cin >> nama;
                cout << "Masukkan Usia : ";
                cin >> usia;
                insertDepan(nama, usia);
                cout << endl;
                tampil();
                break;

```

```
case 2:
    cout << "Masukkan Nama : ";
    cin >> nama;
    cout << "Masukkan Usia : ";
    cin >> usia;
    insertBelakang(nama, usia);
    cout << endl;
    tampil();
    break;
case 3:
    cout << "Masukkan Posisi : ";
    cin >> posisi;
    cout << "Masukkan Nama : ";
    cin >> nama;
    cout << "Masukkan Usia : ";
    cin >> usia;
    insertTengah(nama, usia, posisi);
    cout << endl;
    tampil();
    break;
case 4:
    hapusDepan();
    cout << endl;
    tampil();
    break;
case 5:
    hapusBelakang();
    cout << endl;
    tampil();
    break;
case 6:
    cout << "Masukkan Posisi : ";
    cin >> posisi;
    hapusTengah(posisi);
    cout << endl;
    tampil();
    break;
case 7:
    cout << "Masukkan Nama : ";
    cin >> nama;
    cout << "Masukkan Usia : ";
    cin >> usia;
    ubahDepan(nama, usia);
    cout << endl;
    tampil();
    break;
case 8:
    cout << "Masukkan Nama : ";
    cin >> nama;
```

```
        cout << "Masukkan Usia : ";
        cin >> usia;
        ubahBelakang(nama, usia);
        cout << endl;
        tampil();
        break;
    case 9:
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan Usia : ";
        cin >> usia;
        ubahTengah(nama, usia, posisi);
        cout << endl;
        tampil();
        break;
    case 10:
        tampil();
        break;

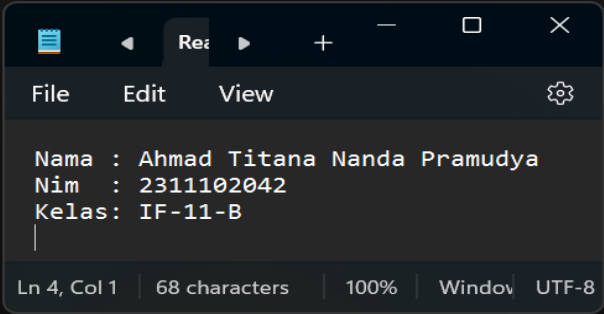
    default:
        cout << "Pilihan Salah" << endl;
        break;
    }
} while (menu != 0);
return 0;
}
```


Output :

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

```
# Menu Linked List Mahasiswa #

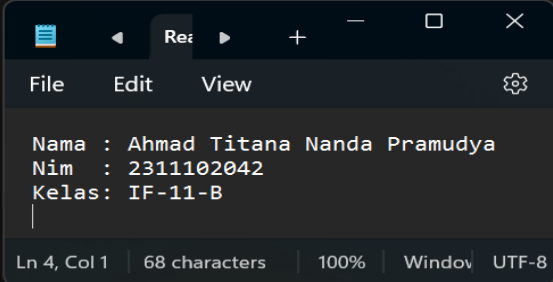
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 1
Masukkan Nama : Ahmad
Masukkan Usia : 18
```



2. Hapus data akechi

```
1. Insert Depan
2. Insert Belakang
3. Insert 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 : 6
Masukkan Posisi : 6

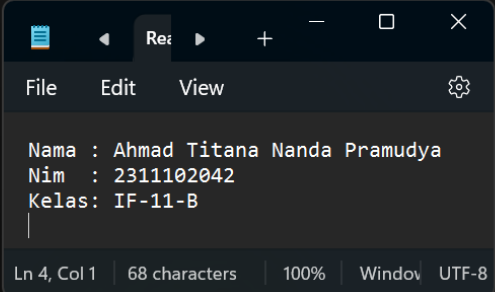
Ahmad 18 , john 19 , jane 20 , michael 19 , yusuke 18 , hoshino 19 , karin 18 ,
```



3. Tambahkan data berikut diantara John dan Jane : Futaba 18

```
1. Insert Depan
2. Insert Belakang
3. Insert 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 : 3
Masukkan Posisi : 3
Masukkan Nama : futaba
Masukkan Usia : 18

ahmad 18 , john 19 , futaba 18 , jane 20 , michael 18 , yusuke 19 , hoshino 18 , karin 18 ,
```



4. Tambahkan data igor 20 di awal

```
1. Insert Depan
2. Insert Belakang
3. Insert 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 : 1
Masukkan Nama : igor
Masukkan Usia : 20

igor 20 , ahmad 18 , john 19 , futuba 18 , jane 20 , michael 18 , yusuke 19 , hoshino 18 , karin 18 ,
```

2. Ubah data Michael menjadi ryen : 18

```
1. Insert Depan
2. Insert Belakang
3. Insert 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 : 9
Masukkan Posisi : 6
Masukkan Nama : ryen
Masukkan Usia : 18

igor 20 , ahmad 18 , john 19 , futuba 18 , jane 20 , ryen 18 , yusuke 19 , hoshino 18 , karin 18 ,
```

3. Tampilkan semua data

```
1. Insert Depan
2. Insert Belakang
3. Insert 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 , ahmad 18 , john 19 , futuba 18 , jane 20 , ryen 18 , yusuke 19 , hoshino 18 , karin 18 ,
```

Deskripsi :

Program di atas adalah implementasi linked list sederhana untuk menyimpan data mahasiswa. Program ini memiliki fungsi-fungsi dasar seperti menambahkan data di depan, belakang, atau di tengah linked list, menghapus data di depan, belakang, atau di tengah linked list, mengubah data di depan, belakang, atau di tengah linked list, menghitung jumlah data dalam linked list, dan menampilkan seluruh data dalam linked list.

Dalam program ini, setiap node dalam linked list memiliki dua data, yaitu nama dan usia mahasiswa. Setiap node juga memiliki pointer ke node berikutnya dalam linked list.

Program ini menggunakan menu untuk memilih operasi yang akan dilakukan pada linked list, dan program akan terus berjalan hingga pengguna memilih untuk keluar (memasukkan angka 0).

Kesimpulan dari program ini adalah sebagai berikut:

- Program ini dapat digunakan untuk menyimpan data mahasiswa dalam linked list.
- Program ini memiliki fungsi-fungsi dasar untuk menambah, menghapus, mengubah, dan menampilkan data dalam linked list.
- Program ini menggunakan konsep linked list untuk mengelola data, sehingga lebih fleksibel dalam penambahan dan penghapusan data dibandingkan dengan struktur data array.

Unguided 2

Sourcode

```
#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 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);

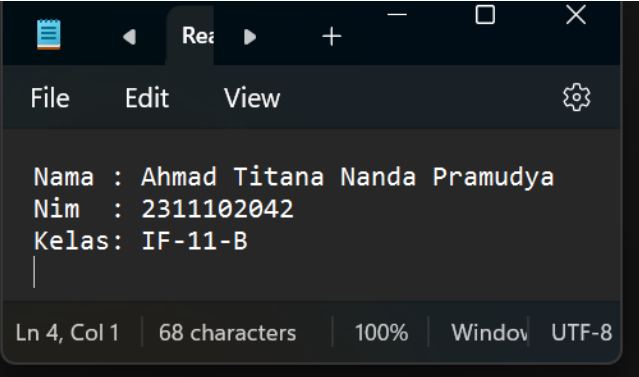
return 0;
}

```

Output :

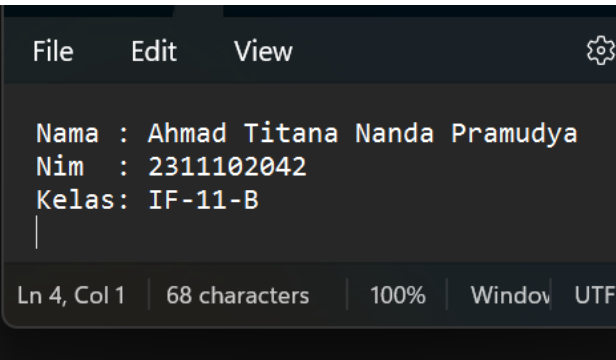
1. Tambahkan produk Azarine dengan harga 65000 diantara Somethinc dan Skintific

```
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
```



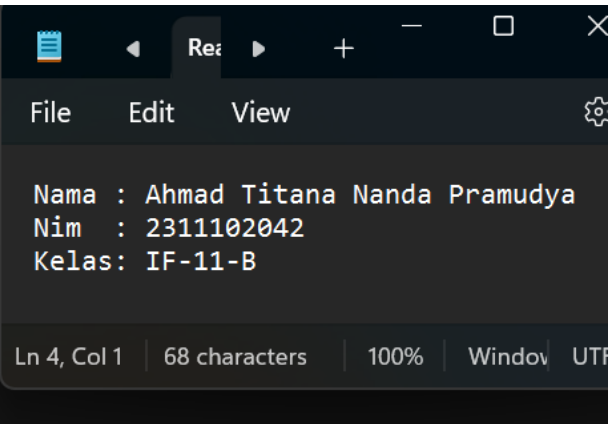
2. Hapus produl wardah

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



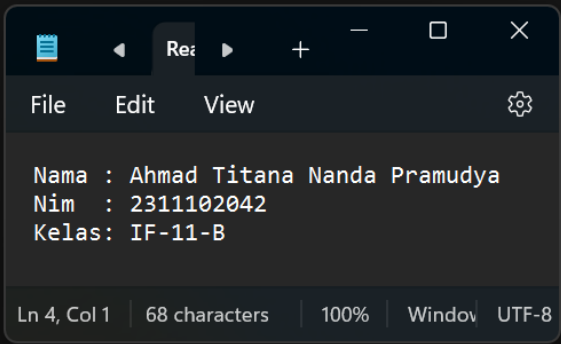
3. Update produk Hanasui menjadi Cleora dengan harga 55.000

```
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
```



4. Tampilkan menu

```
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 |
-----
| hanasui          | 30000 |
| wardah           | 50000 |
| azarine          | 65000 |
| skintific        | 100000|
| cleora           | 55000 |
-----
```



The screenshot shows a Notepad++ window with a dark theme. The title bar says 'Rea'. The menu bar includes 'File', 'Edit', 'View', and a settings icon. The text content is as follows:

```
Nama : Ahmad Titana Nanda Pramudya
Nim : 2311102042
Kelas: IF-11-B
```

The status bar at the bottom indicates 'Ln 4, Col 1', '68 characters', '100%', 'Window', and 'UTF-8'.

Deskripsi :

Program di atas adalah implementasi dari Doubly Linked List untuk menyimpan data produk skincare. Program ini memiliki beberapa fungsi dasar seperti menambah data, menghapus data, mengupdate data, menambah data pada posisi tertentu, menghapus data pada posisi tertentu, menghapus seluruh data, dan menampilkan data.

- Program ini menggunakan Doubly Linked List untuk mengelola data produk skincare.
- Setiap node dalam Doubly Linked List memiliki dua pointer, yaitu pointer ke node sebelumnya (prev) dan pointer ke node berikutnya (next).
- Program ini memiliki fungsi push untuk menambah data di depan linked list, pushCenter untuk menambah data pada posisi tertentu, pop untuk menghapus data di depan linked list, popCenter untuk menghapus data pada posisi tertentu, update untuk mengubah data, updateCenter untuk mengubah data pada posisi tertentu, deleteAll untuk menghapus seluruh data, dan display untuk menampilkan data.
- Program ini menggunakan menu untuk memilih operasi yang akan dilakukan pada Doubly Linked List, dan program akan terus berjalan hingga pengguna memilih untuk keluar.

D. Kesimpulan

Linked list adalah suatu simpul (node) yang dikaitkan dengan simpul yang lain dalam suatu urutan tertentu. Suatu linked list dikatakan single linked list apabila hanya ada satu pointer yang menghubungkan setiap node (satu arah “next”). list. Beberapa operasi pada linked list bisa diterapkan, pada praktikum ini terdapat 12 pengoperasian dengan menggunakan fungsi yaitu penginisialisasian, menambahkan, mengubah, mengurangi dan menampilkan.

E. Referensi

Sekolah Ilmu Komputer (SICS) Binus University. (2017, 15 Maret). Doubly Linked List. Diakses pada 29 Maret 2024, dari <https://socs.binus.ac.id/2017/03/15/doublylinked-list/>

Mikirin Kode. (n.d.). Single Linked List. Diakses pada 29 Maret 2024, dari <https://mikirinkode.com/single-linked-list/>