

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL III
SINGLE AND DOUBLE LINKED LIST**



Disusun Oleh :

Nama : D'sharlendita Febianda Aurelia
NIM : 2311102069

Dosen :

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

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

A. Dasar Teori

Linked list adalah suatu struktur data yang penting dalam pemrograman yang digunakan untuk menyimpan dan mengorganisir data. Dalam bahasa C++, linked list dapat diimplementasikan dengan menggunakan pointer dan alokasi dinamis. Simpul pertama dari linked list disebut sebagai head atau simpul kepala yang berfungsi sebagai titik awal akses ke seluruh data dalam Linked List. Apabila linked list berisi elemen kosong, maka nilai pointer dari head menunjuk ke NULL. Begitu juga untuk pointer berikutnya dari simpul terakhir atau simpul ekor yang menjadi penanda akhir dari urutan simpul. Simpul ekor akan menunjuk ke NULL. Ukuran elemen dari linked list dapat bertambah secara dinamis dan mudah untuk menyisipkan dan menghapus elemen karena tidak seperti array, kita hanya perlu mengubah pointer elemen sebelumnya dan elemen berikutnya untuk menyisipkan atau menghapus elemen.

1. Single Linked List

Single linked list adalah apabila hanya ada satu pointer yang menghubungkan setiap node (satu arah “next”)

2. Double Linked List

Double Linked List adalah linked list dengan node yang memiliki data dan dua buah reference link (biasanya disebut next dan prev) yang menunjuk ke node sebelum dan node sesudahnya.

Linked List memiliki beberapa fungsi penting, antara lain:

1. Menyimpan dan mengelola data dalam urutan tertentu.
2. Memudahkan penambahan dan penghapusan data secara dinamis tanpa harus menggeser data lain.
3. Digunakan dalam implementasi berbagai algoritma dan struktur data lain seperti stack dan queue.

B. Guided

Guided 1

Latihan single linked list

Source Code:

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

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

Node* head;
Node* tail;

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

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

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

// Tambah Node di belakang
```

```

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

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

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

```

```

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

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

// Hapus Node di posisi tengah
void hapusTengah(int posisi) {
    if (posisi < 1 || posisi > hitungList()) {
        cout << "Posisi diluar jangkauan" << endl;
    } else if (posisi == 1) {
        cout << "Posisi bukan posisi tengah" << endl;
    } else {

```

```

        Node* hapus;
        Node* bantu = head;
        for (int nomor = 1; nomor < posisi - 1; nomor++) {
            bantu = bantu->next;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    }
}

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

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

// Ubah data Node di belakang
void ubahBelakang(int data) {
    if (!isEmpty()) {
        tail->data = data;
    }
}

```

```

    } else {
        cout << "List masih kosong!" << endl;
    }
}

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

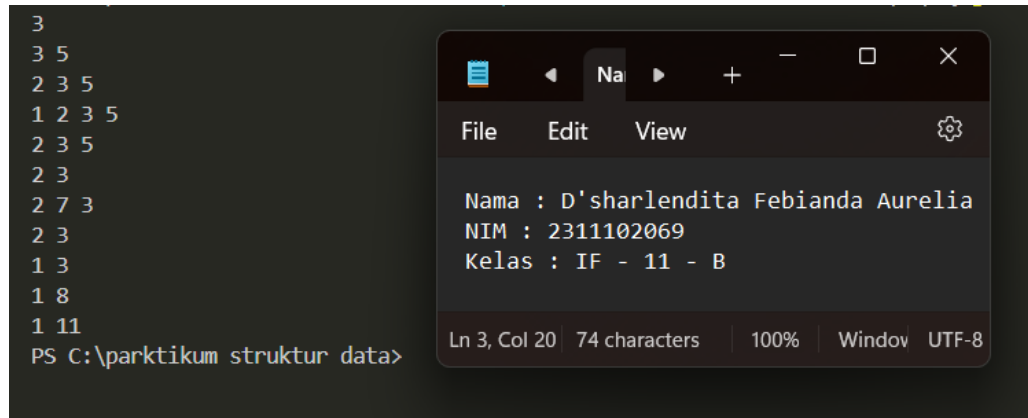
// Tampilkan semua data Node di list
void tampil() {
    if (!isEmpty()) {
        Node* bantu = head;
        while (bantu != NULL) {
            cout << bantu->data << " ";
            bantu = bantu->next;
        }
        cout << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

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

```

```
    ubahTengah(11, 2); tampil();  
    return 0;  
}
```

Screenshots 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 C:\parktikum struktur data>
```

Nama : D'sharlendita Febianda Aurelia
NIM : 2311102069
Kelas : IF - 11 - B

Ln 3, Col 20 | 74 characters | 100% | Window UTF-8

Deskripsi:

Program ini adalah implementasi dari single linked list yang akan memberikan output untuk menampilkan data yang ada di linked list, menambahkan data baik di depan, di belakang, atau di tengah, menghapus data di depan, di belakang, atau di tengah, mengubah data di depan, di belakang, atau di tengah, menghapus semua data di linked list, dan menghitung jumlah data di linked list. Program ini juga akan mengecek apakah linked list kosong atau tidak.

Guided 2

Latihan double linked list

Source Code:

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

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

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

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

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

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

        head = newNode;
    }

    void pop() {
        if (head == nullptr) {
            return;
        }
    }
}
```

```

    Node* temp = head;
    head = head->next;

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

    delete temp;
}

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

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

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

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

```

```

    }
};

int main() {
    DoublyLinkedList list;
    while (true) {
        cout << "1. Add data" << endl;
        cout << "2. Delete data" << endl;
        cout << "3. Update data" << endl;
        cout << "4. Clear data" << endl;
        cout << "5. Display data" << endl;
        cout << "6. Exit" << endl;

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

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

```

```

        break;
    }
    case 5: {
        list.display();
        break;
    }
    case 6: {
        return 0;
    }
    default: {
        cout << "Invalid choice" << endl;
        break;
    }
}
}
return 0;
}

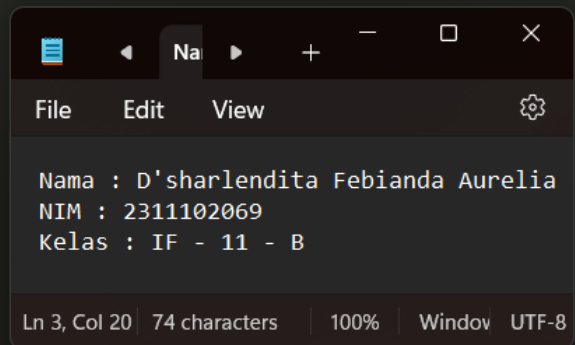
```

Screenshots Output :

```

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: 9
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: 8
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: 7
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: 6

```



```

File Edit View
Nama : D'sharlendita Febianda Aurelia
NIM : 2311102069
Kelas : IF - 11 - B
Ln 3, Col 20 74 characters 100% Window UTF-8

```

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: 5

1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit

Enter your choice: 5

5 6 7 8 9

1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit

Enter your choice: 3

Enter old data: 8

Enter new data: 7

1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit

Enter your choice: 5

5 6 7 7 9

1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit

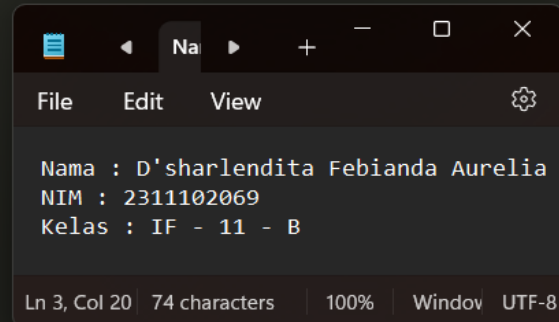
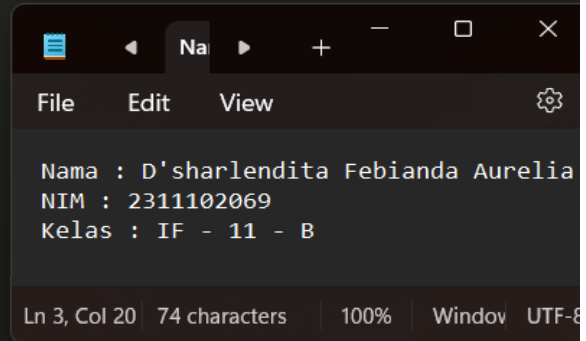
Enter your choice: 2

1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit

Enter your choice: 5

6 7 7 9

1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit



```
Enter your choice: 6
PS C:\parktikum struktur data> 
```

Deskripsi:

Program ini adalah implementasi dari double linked list yang dapat menambahkan data di awal linked list, menghapus data di awal linked list, mengupdate data yang ada di linked list, menghapus semua data di linked list, dan menampilkan data yang ada di linked list. Program ini juga akan mengecek apakah linked list kosong atau tidak.

C. Unguided

Unguided 1

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

- 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

- Hapus data Akechi
- Tambahkan data berikut diantara John dan Jane : Futaba 18
- 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;

class Node {
public:
    string nama;
    int usia;
    Node* next;

    Node(string nama, int usia) {
        this->nama = nama;
        this->usia = usia;
        next = nullptr;
    }
};

class LinkedList {
public:
    Node* head;

    LinkedList() {
        head = nullptr;
    }

    void tambah_depan(string nama, int usia) {
        Node* new_node = new Node(nama, usia);
        new_node->next = head;
        head = new_node;
    }

    void tambah_belakang(string nama, int usia) {
        Node* new_node = new Node(nama, usia);
        if (head == nullptr) {
            head = new_node;
            return;
        }
        Node* last = head;
```

```

        while (last->next != nullptr) {
            last = last->next;
        }
        last->next = new_node;
    }

    void tambah_tengah(string nama, int usia, int posisi) {
        if (posisi < 1) {
            cout << "Posisi harus lebih besar dari 0." << endl;
            return;
        }
        if (posisi == 1) {
            tambah_depan(nama, usia);
            return;
        }
        Node* new_node = new Node(nama, usia);
        Node* current = head;
        int count = 1;
        while (current != nullptr && count < posisi - 1) {
            current = current->next;
            count++;
        }
        if (current == nullptr) {
            cout << "Posisi melebihi panjang linked list." <<
endl;
            return;
        }
        new_node->next = current->next;
        current->next = new_node;
    }

    void hapus(string nama) {
        Node* current = head;
        if (current != nullptr && current->nama == nama) {
            head = current->next;
            delete current;
            return;
        }
        Node* prev = nullptr;
        while (current != nullptr && current->nama != nama) {
            prev = current;
            current = current->next;
        }
    }

```



```

        if (current == nullptr) {
            cout << nama << " tidak ditemukan dalam linked list."
<< endl;
            return;
        }
        prev->next = current->next;
        delete current;
    }

    void tampilkan() {
        Node* current = head;
        cout << "Data dalam linked list:" << endl;
        while (current != nullptr) {
            cout << current->nama << "\t" << current->usia <<
endl;
            current = current->next;
        }
    }
};

int main() {
    LinkedList linked_list;

    char choice;
    do {
        cout << "\nPilih operasi yang ingin dilakukan:" << endl;
        cout << "a. Tambahkan data" << endl;
        cout << "b. Hapus data" << endl;
        cout << "c. Tambahkan data di tengah" << endl;
        cout << "d. Tambahkan data di awal" << endl;
        cout << "e. Ubah data" << endl;
        cout << "f. Tampilkan seluruh data" << endl;
        cout << "g. keluar" << endl;
        cout << "Pilihan: ";
        cin >> choice;

        switch (choice) {
            case 'a': {
                string nama_list[8];
                int usia_list[8];
                for (int i = 0; i < 8; ++i) {
                    cout << "Masukkan nama data ke-" << i + 1 <<
" : ";

```

```

        cin >> nama_list[i];
        cout << "Masukkan umur data ke-" << i + 1 <<
" : ";

        cin >> usia_list[i];

        linked_list.tambah_belakang(nama_list[i],
usia_list[i]);
    }
    break;
}
case 'b': {
    string nama_hapus;
    cout << "Masukkan nama yang ingin dihapus: ";
    cin >> nama_hapus;
    linked_list.hapus(nama_hapus);
    break;
}
case 'c': {
    string nama_tambah;
    int usia_tambah, posisi;
    cout << "Masukkan nama yang ingin ditambahkan: ";
    cin >> nama_tambah;
    cout << "Masukkan usia yang ingin ditambahkan: ";
    cin >> usia_tambah;
    cout << "Masukkan posisi (indeks) tempat
penambahan: ";
    cin >> posisi;
    linked_list.tambah_tengah(nama_tambah,
usia_tambah, posisi);
    break;
}
case 'd': {
    string nama_tambah_awal;
    int usia_tambah_awal;
    cout << "Masukkan nama yang ingin ditambahkan di
awal: ";
    cin >> nama_tambah_awal;
    cout << "Masukkan usia yang ingin ditambahkan di
awal: ";
    cin >> usia_tambah_awal;
    linked_list.tambah_depan(nama_tambah_awal,
usia_tambah_awal);
    break;
}

```

```

    }
    case 'e':
    {
        string nama_lama, nama_baru;
        int usia_baru;
        cout << "Masukkan nama yang ingin diubah: ";
        cin >> nama_lama;
        cout << "Masukkan nama baru: ";
        cin >> nama_baru;
        cout << "Masukkan usia baru: ";
        cin >> usia_baru;
        linked_list.hapus(nama_lama);
        linked_list.tambah_belakang(nama_baru,
usia_baru);
        break;
    }
    case 'f':
    {
        linked_list.tampilkan();
        break;
    }
    case 'g': {
        cout << "Keluar dari program." << endl;
        break;
    }
    default:
        cout << "Pilihan tidak valid!" << endl;
    }
} while(choice != 'g');

return 0;
}

```

Screenshots Output

Pilih operasi yang ingin dilakukan:

- a. Tambahkan data
- b. Hapus data
- c. Tambahkan data di tengah
- d. Tambahkan data di awal
- e. Ubah data
- f. Tampilkan seluruh data
- g. keluar

Pilihan: a

Masukkan nama data ke-1 : Dita

Masukkan umur data ke-1 : 19

Masukkan nama data ke-2 : John

Masukkan umur data ke-2 : 19

Masukkan nama data ke-3 : Jane

Masukkan umur data ke-3 : 20

Masukkan nama data ke-4 : Michael

Masukkan umur data ke-4 : 18

Masukkan nama data ke-5 : Yusuke

Masukkan umur data ke-5 : 19

Masukkan nama data ke-6 : Akechi

Masukkan umur data ke-6 : 20

Masukkan nama data ke-7 : Hoshino

Masukkan umur data ke-7 : 18

Masukkan nama data ke-8 : Karin

Masukkan umur data ke-8 : 18

Pilih operasi yang ingin dilakukan:

- a. Tambahkan data
- b. Hapus data
- c. Tambahkan data di tengah
- d. Tambahkan data di awal
- e. Ubah data
- f. Tampilkan seluruh data
- g. keluar

Pilihan: f

Data dalam linked list:

Dita 19

John 19

Jane 20

Michael 18

Yusuke 19

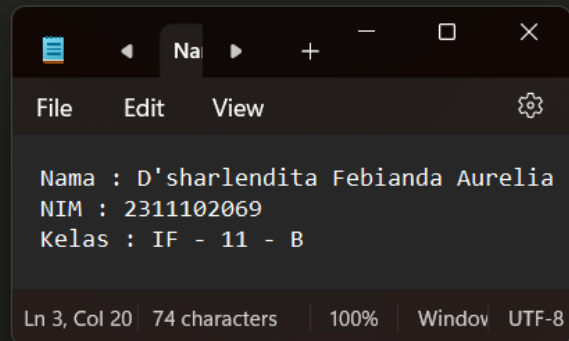
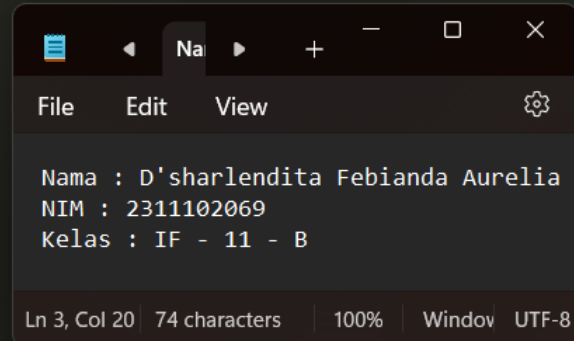
Akechi 20

Hoshino 18

Karin 18

Pilih operasi yang ingin dilakukan:

- a. Tambahkan data
- b. Hapus data
- c. Tambahkan data di tengah
- d. Tambahkan data di awal
- e. Ubah data
- f. Tampilkan seluruh data



g. keluar
Pilihan: b
Masukkan nama yang ingin dihapus: Akechi

Pilih operasi yang ingin dilakukan:

- a. Tambahkan data
 - b. Hapus data
 - c. Tambahkan data di tengah
 - d. Tambahkan data di awal
 - e. Ubah data
 - f. Tampilkan seluruh data
 - g. keluar
- Pilihan: f

Data dalam linked list:

Dita 19
John 19
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18

Pilih operasi yang ingin dilakukan:

- a. Tambahkan data
 - b. Hapus data
 - c. Tambahkan data di tengah
 - d. Tambahkan data di awal
 - e. Ubah data
 - f. Tampilkan seluruh data
 - g. keluar
- Pilihan: c

Masukkan nama yang ingin ditambahkan: Futaba

Masukkan usia yang ingin ditambahkan: 18

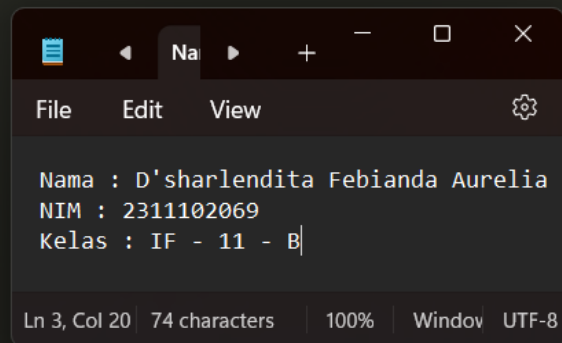
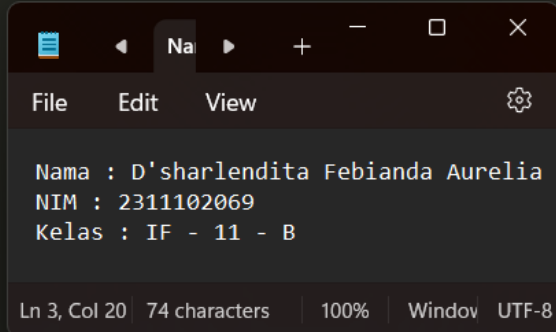
Masukkan posisi (indeks) tempat penambahan: 3

Pilih operasi yang ingin dilakukan:

- a. Tambahkan data
 - b. Hapus data
 - c. Tambahkan data di tengah
 - d. Tambahkan data di awal
 - e. Ubah data
 - f. Tampilkan seluruh data
 - g. keluar
- Pilihan: f

Data dalam linked list:

Dita 19
John 19
Futaba 18
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18



Pilih operasi yang ingin dilakukan:

- a. Tambahkan data
- b. Hapus data
- c. Tambahkan data di tengah
- d. Tambahkan data di awal
- e. Ubah data
- f. Tampilkan seluruh data
- g. keluar

Pilihan: d

Masukkan nama yang ingin ditambahkan di awal: Igor

Masukkan usia yang ingin ditambahkan di awal: 20

Pilih operasi yang ingin dilakukan:

- a. Tambahkan data
- b. Hapus data
- c. Tambahkan data di tengah
- d. Tambahkan data di awal
- e. Ubah data
- f. Tampilkan seluruh data
- g. keluar

Pilihan: f

Data dalam linked list:

Igor 20

Dita 19

John 19

Futaba 18

Jane 20

Michael 18

Yusuke 19

Hoshino 18

Karin 18

Pilih operasi yang ingin dilakukan:

- a. Tambahkan data
- b. Hapus data
- c. Tambahkan data di tengah
- d. Tambahkan data di awal
- e. Ubah data
- f. Tampilkan seluruh data
- g. keluar

Pilihan: e

Masukkan nama yang ingin diubah: Michael

Masukkan nama baru: Reyn

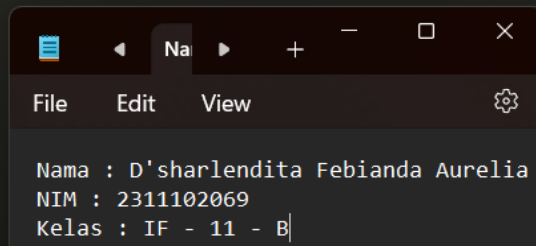
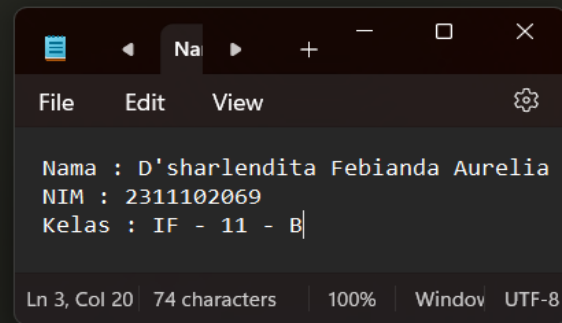
Masukkan usia baru: 18

Pilih operasi yang ingin dilakukan:

- a. Tambahkan data
- b. Hapus data
- c. Tambahkan data di tengah
- d. Tambahkan data di awal
- e. Ubah data
- f. Tampilkan seluruh data
- g. keluar

Pilihan: f

Data dalam linked list:



```
Igor    20
Dita    19
John    19
Futaba  18
Jane    20
Yusuke  19
Hoshino 18
Karin   18
Reyn    18

Pilih operasi yang ingin dilakukan:
a. Tambahkan data
b. Hapus data
c. Tambahkan data di tengah
d. Tambahkan data di awal
e. Ubah data
f. Tampilkan seluruh data
g. keluar
Pilihan: g
Keluar dari program.
PS C:\parktikum struktur data> 
```

Ln 3, Col 20 74 characters 100% Window UTF-8

File Edit View

Nama : D'sharlendita Febianda Aurelia
NIM : 2311102069
Kelas : IF - 11 - B

Ln 3, Col 20 74 characters 100% Window UTF-8

Deskripsi:

Program ini akan menampilkan menu pilihan untuk mengelola data menggunakan single linked list. Pengguna dapat memilih menu untuk menambahkan data di awal, di tengah, di akhir, menghapus data, mengubah data, dan menampilkan seluruh data. Jika pengguna memilih untuk menambahkan data, program akan menambahkan node baru di awal, di tengah, atau di akhir linked list. Jika pengguna memilih untuk menghapus data, program akan menghapus node dengan nama tertentu dari linked list. Jika pengguna memilih untuk mengubah data, program akan menghapus node dengan nama tertentu dan menambahkan node baru dengan nama dan usia yang diinputkan oleh pengguna. Jika pengguna memilih untuk menampilkan seluruh data, program akan menampilkan semua data yang ada dalam linked list. Jika pengguna memilih untuk keluar dari program, program akan mengakhiri proses dan menutup program.

Unguided 2

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

Nama Produk	Harga
Originote	60.000
Somethinc	150.000
Skintific	100.000
Wardah	50.000
Hanasui	30.000

Case:

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

Toko Skincare Purwokerto

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

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

Nama Produk	Harga
Originote	60.000
Somethinc	150.000

Azarine	65.000
Skintific	100.000
Cleora	55.000

Source Code :

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

using namespace std;

class TokoSkincarePurwokerto {
public:
    string nama_produk;
    int harga;
    TokoSkincarePurwokerto* prev;
    TokoSkincarePurwokerto* next;

    TokoSkincarePurwokerto(string nama_produk, int harga)
    {
        this->nama_produk = nama_produk;
        this->harga = harga;
        prev = nullptr;
        next = nullptr;
    }
};

class DoubleLinkedList {
public:
    TokoSkincarePurwokerto* head;
```

```

    DoubleLinkedList() {
        head = nullptr;
    }

    void tambah_belakang(string nama_produk, int harga) {
        TokoSkincarePurwokerto* new_node = new
TokoSkincarePurwokerto(nama_produk, harga);
        if (head == nullptr) {
            head = new_node;
            return;
        }
        TokoSkincarePurwokerto* last = head;
        while (last->next != nullptr) {
            last = last->next;
        }
        last->next = new_node;
        new_node->prev = last;
    }

    void tambah_posisi(string nama_produk, int harga, int
posisi) {
        if (posisi < 1) {
            cout << "Posisi harus lebih besar dari 0." <<
endl;
            return;
        }
        TokoSkincarePurwokerto* new_node = new
TokoSkincarePurwokerto(nama_produk, harga);

```

```

        if (posisi == 1 || head == nullptr) {
            new_node->next = head;
            if (head != nullptr) {
                head->prev = new_node;
            }
            head = new_node;
            return;
        }
        TokoSkincarePurwokerto* current = head;
        int count = 1;
        while (current != nullptr && count < posisi - 1)
    {
        current = current->next;
        count++;
    }
    if (current == nullptr) {
        cout << "Posisi melebihi panjang linked
list." << endl;
        return;
    }
    new_node->next = current->next;
    new_node->prev = current;
    if (current->next != nullptr) {
        current->next->prev = new_node;
    }
    current->next = new_node;
}

void hapus(string nama_produk) {

```

```

        TokoSkincarePurwokerto* current = head;
        while (current != nullptr && current->nama_produk
!= nama_produk) {
            current = current->next;
        }
        if (current == nullptr) {
            cout << "Produk " << nama_produk << " tidak
ditemukan." << endl;
            return;
        }
        if (current->prev != nullptr) {
            current->prev->next = current->next;
        } else {
            head = current->next;
        }
        if (current->next != nullptr) {
            current->next->prev = current->prev;
        }
        delete current;
    }

    void update(string nama_produk_Lama, string
nama_produk_baru, int harga_baru) {
        TokoSkincarePurwokerto* current = head;
        while (current != nullptr && current->nama_produk
!= nama_produk_Lama) {
            current = current->next;
        }
        if (current == nullptr) {

```

```

        cout << "Produk " << nama_produk_lama << "
tidak ditemukan." << endl;

        return;

    }

    current->nama_produk = nama_produk_baru;
    current->harga = harga_baru;

    cout << "Produk " << nama_produk_lama << "
berhasil diupdate menjadi " << nama_produk_baru << "
dengan harga " << harga_baru << endl;

}

void tampilkan() {
    TokoSkincarePurwokerto* current = head;
    cout << "_____ "
<<endl;
    cout << "| Nama Produk\t|Harga\t\t|" << endl;
    while (current != nullptr) {
        cout << "_____ "
<<endl;
        cout << "| " << current->nama_produk << "\t|"
<< current->harga << "\t\t" << "|" << endl;
        current = current->next;
    }
    cout << "_____ "
<<endl;
}

void hapus_seluruh_data() {
    TokoSkincarePurwokerto* current = head;

```

```

        while (current != nullptr) {
            TokoSkincarePurwokerto* next_node = current->next;

            delete current;
            current = next_node;
        }
        head = nullptr;
    }
};

int main() {
    DoubleLinkedList linked_list;

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

    int choice;
    do {
        cout << "\nToko Skincare Purwokerto" << endl;
        cout << "1. Tambah Data" << endl;
        cout << "2. Hapus Data" << endl;
        cout << "3. Update Data" << endl;
        cout << "4. Tambah Data Urutan Tertentu" << endl;
        cout << "5. Hapus Data Urutan Tertentu" << endl;
        cout << "6. Hapus Seluruh Data" << endl;
        cout << "7. Tampilkan Data" << endl;
    }
}

```

```
cout << "8. Exit" << endl;
cout << "Pilihan: ";
cin >> choice;

switch (choice) {
    case 1: {
        string nama_produk;
        int harga;
        cout << "Masukkan nama produk: ";
        cin >> nama_produk;
        cout << "Masukkan harga: ";
        cin >> harga;
        linked_list.tambah_belakang(nama_produk,
harga);

        break;
    }
    case 2: {
        string nama_produk_hapus;
        cout << "Masukkan nama produk yang ingin
dihapus: ";

        cin >> nama_produk_hapus;
        linked_list.hapus(nama_produk_hapus);
        break;
    }
    case 3: {
        string nama_produk_lama,
nama_produk_baru;
        int harga_baru;
        cout << "Masukkan nama produk yang ingin
```

```

diupdate: ";

        cin >> nama_produk_lama;
        cout << "Masukkan nama produk baru: ";
        cin >> nama_produk_baru;
        cout << "Masukkan harga baru: ";
        cin >> harga_baru;
        linked_list.update(nama_produk_lama,
nama_produk_baru, harga_baru);
        break;
    }
    case 4: {
        string nama_produk_tambah;
        int harga_tambah, posisi;
        cout << "Masukkan nama produk yang ingin
ditambahkan: ";
        cin >> nama_produk_tambah;
        cout << "Masukkan harga: ";
        cin >> harga_tambah;
        cout << "Masukkan posisi untuk
menambahkan data: ";
        cin >> posisi;

        linked_list.tambah_posisi(nama_produk_tambah,
harga_tambah, posisi);
        break;
    }
    case 5: {

        break;
    }
}

```



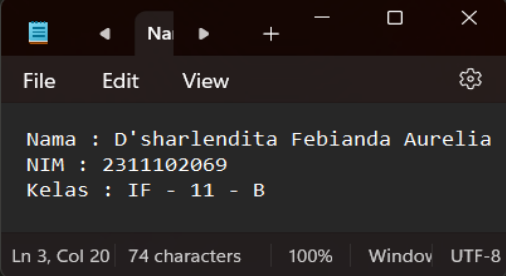
```
    }  
    case 6: {  
        linked_list.hapus_seluruh_data();  
        cout << "Semua data telah dihapus." <<  
endl;  
        break;  
    }  
    case 7: {  
        linked_list.tampilkan();  
        break;  
    }  
    case 8: {  
        cout << "Keluar dari program." << endl;  
        break;  
    }  
    default:  
        cout << "Pilihan tidak valid!" << endl;  
    }  
} while (choice != 8);  
  
return 0;  
}
```

Screenshots Output :

```
7. Tampilkan Data
8. Exit
Pilihan: 7

| Nama Produk | Harga |
|-----|-----|
| Originote   | 60000 |
| Somethinc   | 150000|
| Azarine     | 65000 |
| Skintific   | 100000|
| Cleora      | 55000 |
|-----|-----|

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
Pilihan: 8
Keluar dari program.
```



The screenshot shows a Notepad++ window with the following text:

```
Nama : D'sharlendita Febianda Aurelia
NIM : 2311102069
Kelas : IF - 11 - B
```

The window title is "Na" and it has a menu bar with "File", "Edit", and "View". The status bar at the bottom indicates "Ln 3, Col 20", "74 characters", "100%", "Window", and "UTF-8".

Deskripsi:

Program ini akan menampilkan menu pilihan untuk mengelola data produk skincare dalam bentuk double linked list. Pengguna dapat memilih untuk menambahkan, menghapus, mengupdate, atau menampilkan data produk. Jika pengguna memilih untuk menambahkan data, program akan menambahkan node baru di akhir linked list. Jika pengguna memilih untuk menghapus data, program akan menghapus node dengan nama produk tertentu dari linked list. Jika pengguna memilih untuk mengupdate data, program akan mengubah nama dan harga produk tertentu dalam linked list. Jika pengguna memilih untuk menampilkan data, program akan menampilkan semua data produk yang ada dalam linked list. Jika pengguna memilih untuk mengakhiri program, program akan mengakhiri proses dan menutup program.

D. Kesimpulan

Setelah mempelajari materi single linked list dan double linked list, dapat disimpulkan bahwa kedua jenis linked list ini digunakan untuk menyimpan koleksi data yang terurut. Single linked list hanya memiliki referensi ke node berikutnya, sedangkan double linked list memiliki referensi ke node sebelumnya dan node berikutnya.

E. Referensi

- [1] fikti.umsu.ac.id. 25 Juli 2023. Pengertian Linked List : Struktur Data dalam pemrograman. Diakses pada 02 April 2024, dari <https://fikti.umsu.ac.id/pengertian-linked-list-struktur-data-dalam-pemrograman/#:~:text=Keuntungan%20Double%20Linked%20List%20adalah,simpul%20harus%20menyimpan%20dua%20referensi.>
- [2] trivusi.web.id. 16 September 2022. Struktur Data Linked List: Pengertian, Karakteristik, dan jenis-jenisnya. Diakses pada 02 April 2024, dari <https://www.trivusi.web.id/2022/07/struktur-data-linked-list.html>
- [3] Saputra, Rendi. 2022. Linked List : Single & Double. Diakses pada 02 April 2024, dari https://id.scribd.com/embeds/142590303/content?start_page=1&view_mode=scroll&access_key=key-fFexxf7r1bzEfWu3HKwf