

**LAPORAN PRAKTIKUM STRUKTUR  
DATA DAN ALGORITMA**

**MODUL III  
SINGLE AND DOUBLE LINKED LIST**



**Disusun Oleh :**

NAMA : FAISAL KHOIRUDDIN

NIM : 2311102046

**Dosen**

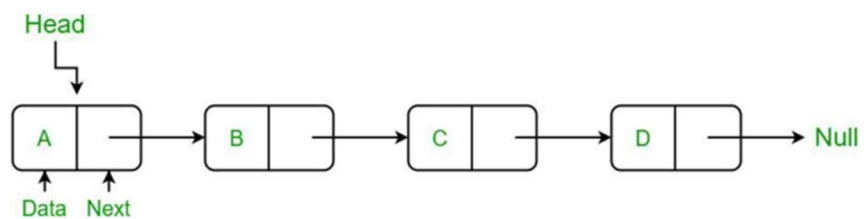
WAHYU ANDI SAPUTRA, S.Pd., M.Eng.

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

## A. Dasar Teori

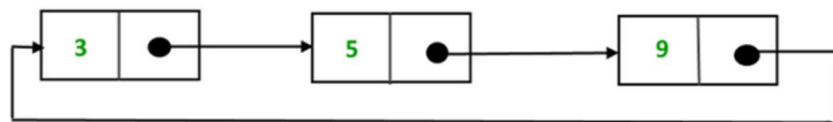
### 1) Single Linked List

Linked List merupakan 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

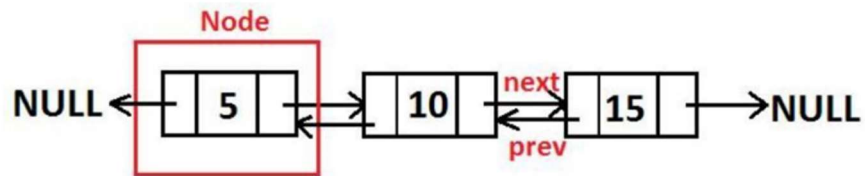
linked list dan non circular linked adalah penunjuk next pada node terakhir pada circular linked list akan selalu merujuk ke node pertama.



## 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

### 1. Guided 1 : Latihan Single Linked List

#### Source Code

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

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

Node* head;
Node* tail;

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

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

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

// Tambah Node di belakang
void insertBelakang(int nilai) {
    Node* baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
```

```

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

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

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

// Hapus Node di depan
void hapusDepan() {
    if (!isEmpty()) {
        Node* hapus = head;

```

```

        if (head->next != NULL) {
            head = head->next;
            delete hapus;
        } else {
            head = tail = NULL;
            delete hapus;
        }
    } else {
        cout << "List kosong!" << endl;
    }
}

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

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

```

```

        bantu = bantu->next;
    }
    hapus = bantu->next;
    bantu->next = hapus->next;
    delete hapus;
}

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

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

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

```



```

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

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

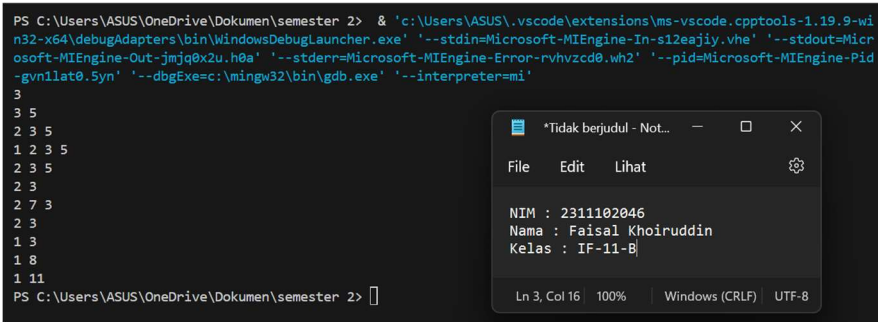
// Tampilkan semua data Node di list
void tampil() {
    if (!isEmpty()) {
        Node* bantu = head;
        while (bantu != NULL) {
            cout << bantu->data << " ";
            bantu = bantu->next;
        }
        cout << endl;
    } else {
        cout << "List masih kosong!" <<
endl;
 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

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

```

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

Screenshots Output



Deskripsi:

Program tersebut merupakan single linked list. Program tersebut melakukan operasi seperti menambahkan Node ke depan, belakang, atau posisi tertentu dalam list, menghapus Node dari depan, belakang, atau posisi tertentu dalam list, mengubah data Node di depan, belakang, atau posisi tertentu dalam list, dan menghapus semua Node dalam list. Program tersebut menampilkan output bilangan dari insert depan, insert belakang, hapusDepan, hapusBelakang, insertTengah, hapusTengah, ubahDepan, dan ubahBelakang

ubahTengah

- #include <iostream> ➔ merupakan input output stream header yang digunakan sebagai standar input output operasi yang digunakan di c++
- using namespace std; ➔ digunakan untuk mendeklarasikan/memberitahukan kepada compiler bahwa kita akan menggunakan semua fungsi/class/file yang terdapat dalam namespace std
- struct Node ➔ mendefinisikan struct node
- int data; ➔ deklarasi variabel data dengan nama tipe data integer
- Node\* next; ➔ pointer next untuk menunjuk ke node berikutnya

- Node\* head; ➔ pointer head untuk menunjuk ke node pertama dalam list
- Node\* tail; ➔ pointer tail untuk menunjuk ke node terakhir dalam list
- void init() ➔ prosedur init untuk menginisialisasi list
- head = NULL; ➔ mengatur head menjadi null
- tail = NULL; ➔ mengatur tail menjadi null
- bool isEmpty() ➔ fungsi untuk memeriksa apakah list kosong
- return head == NULL; ➔ mengembalikan true jika head yaitu null atau list kosong
- void insertDepan(int nilai) ➔ prosedur untuk memasukkan node di depan list
- Node\* baru = new Node; ➔ membuat node
- baru->data = nilai; ➔ mengatur data ke node baru
- baru->next = NULL; ➔ mengatur next dari node baru menjadi null
- if (isEmpty()) ➔ percabangan if, jika list kosong
- head = tail = baru; ➔ node baru menjadi head dan tail
- else ➔ jika list tidak kosong
- baru->next = head; ➔ node baru menjadi head dan tail
- head = baru; ➔ mengatur head menjadi node baru
- void insertBelakang(int nilai) ➔ prosedur untuk memasukkan node di belakang list
- Node\* baru = new Node; ➔ membuat node baru
- baru->data = nilai; ➔ mengatur data ke node baru
- baru->next = NULL; ➔ mengatur next dari node baru menjadi null
- if (isEmpty()) ➔ jika list kosong
- head = tail = baru; ➔ node baru menjadi head dan tail

- else { → jika list tidak kosong
- tail->next = baru; → mengatur next dari tail saat ini menjadi node baru
- tail = baru; → mengatur tail menjadi node baru
- int hitungList() → prosedur untuk menghitung jumlah node dalam list
- Node\* hitung = head; → membuat pointer hitung yang menunjuk head
- int jumlah = 0; → variabel untuk menyimpan jumlah node
- while (hitung != NULL) { → selama hitung bukan sama dengan null (selama ada node dalam list)
- jumlah++; → jumlah++ (increment) selama ada node dalam list
- hitung = hitung->next; → menggeser hitung ke node berikutnya
- return jumlah; → mengembalikan jumlah node dalam list
- void insertTengah(int data, int posisi) → prosedur untuk memasukkan node di Tengah list
- if (posisi < 1 || posisi > hitungList()) → jika posisi < 1 atau posisi lebih dari hitungList (posisi tidak valid)
- cout << "Posisi diluar jangkauan" << endl; → cout menampilkan statement Posisi diluar jangkauan
- else if (posisi == 1) { → jika posisi sama dengan 1
- cout << "Posisi bukan posisi tengah" << endl; → cout menampilkan statement Posisi bukan posisi tengah
- else → jika posisi valid
- Node\* baru = new Node(); → membuat node baru
- baru->data = data; → mengatur data ke node baru
- Node\* bantu = head; → membuat pointer bantu yang menunjuk ke head
- int nomor = 1; → nomor sama dengan 1
- while (nomor < posisi - 1) → menggeser bantu ke posisi

sebelum posisi di tuju

- `bantu = bantu->next;` ➔ bantu sama dengan bantu -> next
- `nomor++;` ➔ `nomor++` (increment)
- `baru->next = bantu->next;` ➔ mengatur next dari node baru menjadi node setelah bantu
- `bantu->next = baru;` ➔ mengatur next dari bantu menjadi node baru
- `void hapusDepan()` ➔ prosedur untuk menghapus node di depan
- `if (!isEmpty()) {` ➔ jika list tidak kosong
- `Node* hapus = head;` ➔ membuat pointer hapus yang menunjuk ke head
- `if (head->next != NULL) {` ➔ jika ada lebih dari satu node dalam list
- `head = head->next;` ➔ mengatur head menjadi node setelah head saat ini
- `delete hapus;` ➔ menghapus node yang ditunjuk oleh hapus
- `}` else { ➔ jika list kosong
- `head = tail = NULL;` ➔ mengatur head dan tail menjadi null
- `delete hapus;` ➔ menghapus node yang ditunjuk oleh hapus
- `}` else { ➔ jika list kosong
- `cout << "List kosong!" << endl;` ➔ cout menampilkan statement List kosong
- `void hapusBelakang()` { ➔ prosedur untuk menghapus node di belakang list
- `if (!isEmpty()) {` ➔ jika list tidak kosong
- `if (head != tail) {` ➔ jika ada lebih dari satu node dalam list
- `Node* hapus = tail;` ➔ membuat pointer hapus yang menunjuk ke tail
- `Node* bantu = head;` ➔ membuat pointer bantu yang menunjuk ke head

- while (bantu->next != tail) { ➔ Menggeser bantu ke node sebelum tail
- bantu = bantu->next; ➔ bantu sama dengan bantu->next
- tail = bantu; ➔ mengatur tail menjadi node yang ditunjuk oleh bantu
- tail->next = NULL; ➔ mengatur next dari tail menjadi NULL
- delete hapus; ➔ menghapus node yang ditunjuk oleh hapus
- } else { ➔ Jika hanya ada satu node dalam list
- head = tail = NULL; ➔ Mengatur head dan tail menjadi NULL
- } else { ➔ Jika list kosong
- cout << "List kosong!" << endl; ➔ cout menampilkan statement List kosong!
- void hapusTengah(int posisi) { ➔ prosedur untuk menghapus node di tengah list
- if (posisi < 1 || posisi > hitungList()) { ➔ jika posisi kurang dari 1 atau posisi lebih dari hitungList (Jika posisi tidak valid)
- cout << "Posisi diluar jangkauan" << endl; ➔ cout menampilkan statement Posisi diluar jangkauan
- } else if (posisi == 1) { ➔ jika posisi sama dengan 1 (depan list)
- cout << "Posisi bukan posisi tengah" << endl; ➔ cout menampilkan statement Posisi bukan posisi tengah
- } else { ➔ jika posisi valid
- Node\* hapus; ➔ pointer hapus node
- Node\* bantu = head; ➔ membuat pointer bantu yang menunjuk ke head
- for (int nomor = 1; nomor < posisi - 1; nomor++) { ➔ perulangan for untuk menggeser bantu ke posisi sebelum posisi yang dituju
- bantu = bantu->next; ➔ bantu sama dengan -> next
- hapus = bantu->next; ➔ membuat pointer hapus yang menunjuk ke node setelah bantu

- bantu->next = hapus->next; ➔ mengatur next dari bantu menjadi node setelah node yang ditunjuk oleh hapus
- delete hapus; ➔ menghapus node yang ditunjuk oleh hapus
- void ubahDepan(int data) { ➔ prosedur untuk mengubah data node di depan list
- if (!isEmpty()) { ➔ jika list tidak kosong
- head->data = data; ➔ mengubah data head menjadi data baru
- } else { ➔ jika list kosong
- cout << "List masih kosong!" << endl; ➔ cout menampilkan statement List masih kosong!
- void ubahTengah(int data, int posisi) { ➔ prosedur untuk mengubah data node di tengah list
- if (!isEmpty()) { ➔ jika list tidak kosong
- if (posisi < 1 || posisi > hitungList()) { ➔ jika posisi kurang dari 1 atau posisi lebih dari hitungList() (jika posisi tidak valid)
- cout << "Posisi di luar jangkauan" << endl; ➔ cout menampilkan statement Posisi di luar jangkauan
- } else if (posisi == 1) { ➔ jika posisi sama dengan 1 (depan list)
- cout << "Posisi bukan posisi tengah" << endl; ➔ cout menampilkan statement posisi bukan posisi tengah
- } else { ➔ Jika posisi valid
- Node\* bantu = head; ➔
- for (int nomor = 1; nomor < posisi; nomor++) { ➔ perulangan for untuk menggeser bantu ke posisi yang dituju
- bantu = bantu->next; ➔ bantu sama dengan bantu->next
- bantu->data = data; ➔ mengubah data node yang ditunjuk oleh bantu menjadi data baru
- } else { ➔ jika list kosong
- cout << "List masih kosong!" << endl; ➔ cout menampilkan statement List masih kosong!

- void ubahBelakang(int data) { ➔ prosedur untuk mengubah data node di belakang list
- if (!isEmpty()) { ➔ jika list tidak kosong
- tail->data = data; ➔ mengubah data tail menjadi data baru
- } else { ➔ Jika list kosong
- cout << "List masih kosong!" << endl; ➔ cout menampilkan statement List masih kosong!
- void clearList() { ➔ prosedur untuk menghapus semua node dalam list
- Node\* bantu = head; ➔ membuat pointer bantu yang menunjuk ke head
- while (bantu != NULL) { ➔ selama bantu bukan NULL (selama ada node dalam list)
- Node\* hapus = bantu; ➔ membuat pointer hapus yang menunjuk ke node yang ditunjuk oleh bantu
- bantu = bantu->next; ➔ menggeser bantu ke node berikutnya
- delete hapus; ➔ menghapus node yang ditunjuk oleh hapus
- head = tail = NULL; ➔ Mengatur head dan tail menjadi NULL
- cout << "List berhasil terhapus!" << endl; ➔ cout menampilkan statement List berhasil terhapus!
- void tampil() { ➔ prosedur untuk menampilkan semua node dalam list
- if (!isEmpty()) { ➔ jika list tidak kosong
- Node\* bantu = head; ➔ membuat pointer bantu yang menunjuk ke head
- while (bantu != NULL) { ➔ Selama bantu tidak sama dengan NULL (selama ada node dalam list)
- cout << bantu->data << " "; ➔ menampilkan data dari node yang ditunjuk oleh bantu
- bantu = bantu->next; ➔ menggeser bantu ke node berikutnya
- cout << endl; ➔ membuat baris baru setelah menampilkan



semua node

- } else { ➔ jika list kosong
- cout << "List masih kosong!" << endl; ➔ cout menampilkan statement List masih kosong!
- int main() { ➔ merupakan fungsi utama yang akan dieksekusi saat program dijalankan.
- init(); ➔ menginisialisasi list
- insertDepan(3); tampil(); ➔ memasukkan 3 di depan list dan menampilkan list
- insertBelakang(5); tampil(); ➔ memasukkan 5 di belakang list dan menampilkan list
- insertDepan(2); tampil(); ➔ memasukkan 2 di depan list dan menampilkan list
- insertDepan(1); tampil(); ➔ memasukkan 1 di depan list dan menampilkan list
- hapusDepan(); tampil(); ➔ menghapus node depan dan menampilkan list
- hapusBelakang(); tampil(); ➔ menghapus node belakang dan menampilkan list
- insertTengah(7, 2); tampil(); ➔ memasukkan 7 di posisi 2 dan menampilkan list
- hapusTengah(2); tampil(); ➔ menghapus node di posisi 2 dan menampilkan list
- ubahDepan(1); tampil(); ➔ Mengubah node depan menjadi 1 dan menampilkan list
- ubahBelakang(8); tampil(); ➔ Mengubah node belakang menjadi 8 dan menampilkan list
- ubahTengah(11, 2); tampil(); ➔ Mengubah node di posisi 2 menjadi 11 dan menampilkan list
- return 0; ➔ program akan mengembalikan (return) nilai 0 ke operating sistem yang menjalankan program tersebut

## Guided 2 :

```
#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)
{ //prevview dan next
    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

```

PS C:\Users\ASUS\OneDrive\Dokumen\semester 2> & 'c:\Users\ASUS\.vscode\extensions\ms-vscode.cpptools-1.19.9-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-philnfga.dbq' '--stdout=Microsoft-MIEngine-Out-t01npalq.2yy' '--stderr=Microsoft-MIEngine-Error-3r2dkqv0.cdm' '--pid=Microsoft-MIEngine-Pid-dqbrsmxu.yxk' '--dbgExe=c:\mingw32\bin\gdb.exe' '--interpreter=mi'
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: 12938198
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: 23982398

```

\*Tidak berjudul - Not...

File Edit Lihat

NIM : 2311102046  
Nama : Faisal Khoiruddin  
Kelas : IF-11-B

Ln 3, Col 16 100% Windows (CRLF) 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: 12831298
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: 29389128
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: 99384098

```

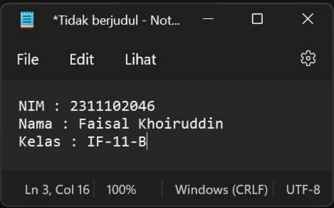
\*Tidak berjudul - Not...

File Edit Lihat

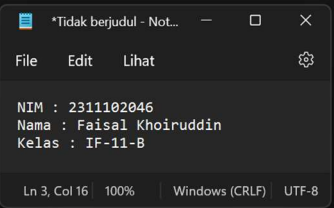
NIM : 2311102046  
Nama : Faisal Khoiruddin  
Kelas : IF-11-B

Ln 3, Col 16 100% Windows (CRLF) UTF-8

```
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: 3
Enter old data: 993844098
Enter new data: 977898179
Data not found
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
29389128 12831298 23982398 12938198
```



```
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 4
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 6
PS C:\Users\ASUS\OneDrive\Dokumen\semester 2>
```



### Deskripsi:

Program tersebut merupakan contoh dari double linked list. Program tersebut meminta pengguna untuk menginputkan dari menu yang dipilih, seperti memilih tambah data maka menginputkan angka. Program tersebut menampilkan output dari menu pilihan pengguna .

- `#include <iostream>` ➔ merupakan input output stream header yang digunakan sebagai standar input output operasi yang digunakan di c++
- `using namespace std;` ➔ digunakan untuk mendeklarasikan/ memberitahukan kepada compiler bahwa kita akan menggunakan semua fungsi/class/file yang terdapat dalam namespace std
- `class Node {` ➔ mendefinisikan class node



- `public:` ➔ berfungsi bahwa anggota kelas yang dideklarasikan setelah `public` dapat diakses dari mana saja
- `int data;` ➔ deklarasi variabel data dengan nama tipe data integer untuk menyimpan nilai node
- `Node* prev;` ➔ pointer `prev` untuk menunjuk ke node sebelumnya
- `Node* next;` ➔ Pointer `next` untuk menunjuk ke node berikutnya
- `class DoublyLinkedList {` ➔ Mendefinisikan kelas `DoublyLinkedList`
- `public` ➔ berfungsi bahwa anggota kelas yang dideklarasikan setelah `public` dapat diakses dari mana saja
- `Node* head;` ➔ Pointer `head` untuk menunjuk ke node pertama dalam list
- `Node* tail;` ➔ Pointer `tail` untuk menunjuk ke node terakhir dalam list
- `DoublyLinkedList()` { ➔ Konstruktor untuk menginisialisasi list
- `head = nullptr;` ➔ mengatur `head` menjadi `nullptr`
- `tail = nullptr;` ➔ mengatur `tail` menjadi `nullptr`
- `void push(int data) {` ➔ prosedur untuk memasukkan node di depan list dengan parameter data nama tipe data integer
- `Node* newNode = new Node;` ➔ membuat node baru
- `newNode->data = data;` ➔ mengatur data node baru
- `newNode->prev = nullptr;` ➔ mengatur `prev` dari node baru menjadi `nullptr`
- `newNode->next = head;` ➔ mengatur `next` dari node baru menjadi `head` saat ini
- `if (head != nullptr) {` ➔ jika `head` tidak sama dengan `nullptr` (list tidak kosong)
- `head->prev = newNode;` ➔ mengatur `prev` dari `head` saat ini menjadi node baru

- } else { ➔ jika head adalah nullptr (list kosong)
- tail = newNode; ➔ mengatur tail menjadi node baru
- head = newNode; ➔ mengatur head menjadi node baru
- void pop() { ➔ prosedur untuk menghapus node di depan list
- if (head == nullptr) { ➔ jika head sama dengan nullptr (list kosong)
- return; ➔ digunakan untuk keluar dari fungsi lebih awal jika list kosong
- Node\* temp = head; ➔ membuat pointer temp yang menunjuk ke head
- head = head->next; ➔ mengatur head menjadi node setelah head saat ini
- if (head != nullptr) { ➔ Jika head tidak sama dengan nullptr (list tidak kosong setelah penghapusan)
- head->prev = nullptr; ➔ Mengatur prev dari head menjadi nullptr
- } else { ➔ jika head adalah nullptr (list kosong setelah penghapusan)
- tail = nullptr; ➔ mengatur tail menjadi nullptr
- delete temp; ➔ menghapus node yang ditunjuk oleh temp
- bool update(int oldData, int newData) { ➔ fungsi untuk mengubah data dalam list dengan parameter int oldData, int newData
- Node\* current = head; ➔ membuat pointer current yang menunjuk ke head
- while (current != nullptr) { ➔ selama current bukan nullptr (selama ada node dalam list)
- if (current->data == oldData) { ➔ jika data dalam node yang ditunjuk oleh current sama dengan oldData
- current->data = newData; ➔ mengubah data dalam node yang ditunjuk oleh current menjadi newData

- `return true;` ➔ mengembalikan true (data berhasil diubah)
- `current = current->next;` ➔ menggeser current ke node berikutnya
- `return false;` ➔ mengembalikan false (data tidak ditemukan dalam list)
- `void deleteAll() {` ➔ prosedur untuk menghapus semua node dalam list
- `Node* current = head;` ➔ membuat pointer current yang menunjuk ke head
- `while (current != nullptr) {` ➔ selama current tidak sama dengan nullptr (selama ada node dalam list)
- `Node* temp = current;` ➔ membuat pointer temp yang menunjuk ke node yang ditunjuk oleh current
- `current = current->next;` ➔ menggeser current ke node berikutnya
- `delete temp;` ➔ menghapus node yang ditunjuk oleh temp
- `head = nullptr;` ➔ mengatur head menjadi nullptr
- `tail = nullptr;` ➔ mengatur tail menjadi nullptr
- `void display() {` ➔ prosedur untuk menampilkan semua node dalam list
- `Node* current = head;` ➔ membuat pointer current yang menunjuk ke head
- `while (current != nullptr) {` ➔ selama current tidak sama dengan nullptr (selama ada node dalam list)
- `cout << current->data << " ";` ➔ cout menampilkan data dari node yang ditunjuk oleh current
- `current = current->next;` ➔ menggeser current ke node berikutnya
- `cout << endl;` ➔ membuat baris baru
- `int main() {` ➔ merupakan fungsi utama yang akan dieksekusi saat program dijalankan.

- `DoublyLinkedList list;` ➔ membuat objek list dari kelas `DoublyLinkedList`
- `while (true) {` ➔ perulangan while selama dalam keadaan true loop tak terbatas sampai pengguna memilih untuk keluar
- `cout << "1. Add data" << endl;` ➔ cout menampilkan menu ke- 1 yaitu Add data
- `cout << "2. Delete data" << endl;` ➔ cout menampilkan menu ke- 2 yaitu Delete data
- `cout << "3. Update data" << endl;` ➔ cout menampilkan menu ke- 3 yaitu Update data
- `cout << "4. Clear data" << endl;` ➔ cout menampilkan menu ke- 4 yaitu Clear data
- `cout << "5. Display data" << endl;` ➔ cout menampilkan menu ke- 5 yaitu Display data
- `cout << "6. Exit" << endl;` ➔ cout menampilkan menu ke- 6 yaitu Exit
- `int choice;` ➔ deklarasi variabel untuk menyimpan pilihan pengguna dengan nama tipe data integer
- `cout << "Enter your choice: ";` ➔ cout menampilkan statement untuk memasukkan pilihan menu
- `cin >> choice;` ➔ cin menerima input pilihan pengguna dan menyimpannya ke dalam variabel choice
- `switch (choice) {` ➔ menjalankan blok kode berdasarkan pilihan pengguna
- `case 1: {` ➔ Jika pengguna memilih 1 yaitu Tambah data
- `int data;` ➔ deklarasi variabel data dengan nama tipe data integer
- `cout << "Enter data to add: ";` ➔ cout menampilkan statement memasukkan data yang ingin ditambahkan
- `cin >> data;` ➔ cin menerima input pilihan pengguna dan menyimpannya ke dalam variabel data
- `list.push(data);` ➔ memasukkan data ke dalam list

- `break;` ➔ keluar dari switch statement
- case 2: { ➔ jika pengguna memilih 2 (Hapus data)
- `list.pop();` ➔ menghapus node dari depan list
- `break;` ➔ keluar dari switch statement
- case 3: { ➔ jika pengguna memilih 3 (Perbarui data)
- `int oldData, newData;` ➔ deklarasi variabel `int oldData, newData`
- `cout << "Enter old data: ";` ➔ `cout` menampilkan statement untuk memasukkan data yang lama
- `cin >> oldData;` ➔ menerima input data lama dari pengguna dan menyimpan ke dalam variabel `oldData`
- `cout << "Enter new data: ";` ➔ `cout` menampilkan statement untuk memasukkan data yang baru
- `cin >> newData;` ➔ menerima input data dari pengguna dan menyimpan ke dalam variabel `newData`
- `bool updated = list.update(oldData, newData);` ➔ mengubah data lama menjadi data baru dalam list
- `if (!updated) {` ➔ jika data lama tidak ditemukan dalam list
- `cout << "Data not found" << endl;` ➔ `cout` menampilkan statement data not found
- `break;` ➔ keluar dari switch statement
- case 4: { ➔ jika pengguna memilih 4 (Hapus semua data)
- `list.deleteAll();` ➔ menghapus semua node dalam list
- `break;` ➔ keluar dari switch statement
- case 5: { ➔ jika pengguna memilih 5 (Tampilkan data)
- `list.display();` ➔ menampilkan semua node dalam list
- `break;` ➔ keluar dari switch statement
- case 6: { ➔ jika pengguna memilih 6 (Keluar)
- `return 0;` ➔ mengakhiri program

- default: { → jika pengguna memasukkan pilihan yang tidak valid (diluar dari 1 sampai 6)
- cout << "Invalid choice" << endl; → cout menampilkan statement Invalid choice
- break; → keluar dari switch statement
- return 0; → program akan mengembalikan (return) nilai 0 ke operating sistem yang menjalankan program tersebut

### C. Unguided/Tugas

#### 1. Unguided 1 : Soal mengenai Single Linked Lis

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

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

// Deklarasi Struct Node
struct Node {
    string nama;
    int usia;
    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(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;
    }
}

// Tambah Node di belakang
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;
}
}

// 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(string nama, int usia,
int posisi) {
    if (posisi < 1 || posisi >
hitungList() + 1) {
        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;
    }
}
}

```



```

// 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(string nama, int usia) {
    if (!isEmpty()) {
        head->nama = nama;
    }
}

```

```

        head->usia = usia;
    } else {
        cout << "List masih kosong!" <<
endl;
    }
}

// Ubah data Node di posisi tengah
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;
    }
}

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

void tampil() {
    if (!isEmpty()) {
        Node* bantu = head;
        cout << "-----\n";
        cout << "| No | Nama\n";
        cout << "| Usia |\n";
        cout << "-----\n";
        int no = 1;
        while (bantu != NULL) {
            cout << "| " << left <<
setw(2) << no << " | " << left << setw(17)
<< bantu->nama << " | " << left << setw(3)
<< bantu->usia << " |\n";
            bantu = bantu->next;
            no++;
        }
        cout << "-----\n";
    } else {
        cout << "List masih kosong!" <<
endl;
    }
}

int main() {
    init();
    int pilihan;
    do {
        cout << "===== | Menu |
===== \n";
        cout << "1. Tambah Data\n";
        cout << "2. Hapus Data\n";
        cout << "3. Update Data\n";
        cout << "4. Tambah Data Urutan

```

```

Tertentu\n";
    cout << "5. Hapus Data Urutan
Tertentu\n";
    cout << "6. Hapus Seluruh Data\n";
    cout << "7. Tampilkan Data\n";
    cout << "8. Exit\n";
    cout << "Masukkan pilihan Anda :
";
    cin >> pilihan;
    switch (pilihan) {
        case 1: {
            string nama;
            int usia;
            cout << "Masukkan nama :
";
            cin >> nama;
            cout << "Masukkan usia :
";
            cin >> usia;
            int posisi;
            cout << "Tambah data di 1
depan atau 2 belakang? ";
            cin >> posisi;
            if (posisi == 1) {
                insertDepan(nama,
usia);
            } else if (posisi == 2) {
                insertBelakang(nama,
usia);
            } else {
                cout << "Pilihan tidak
valid. Data akan ditambahkan di depan.\n";
                insertDepan(nama,
usia);
            }
            break;
        }
        case 2: {
            if (!isEmpty()) {
                int posisi;
                cout << "Masukkan
posisi data yang ingin dihapus : ";
                cin >> posisi;
                hapusTengah(posisi);
                cout << "Data berhasil
dihapus.\n";
            }
        }
    }
}

```

```

        } else {
            cout << "List masih
kosong!\n";
        }
        break;
    }
    case 3: {
        if (!isEmpty()) {
            int posisi;
            string nama;
            int usia;
            cout << "Masukkan
posisi data yang ingin diperbarui : ";
            cin >> posisi;
            cout << "Masukkan nama
baru : ";
            cin >> nama;
            cout << "Masukkan usia
baru : ";
            cin >> usia;
            ubahTengah(nama, usia,
posisi);
            cout << "Data berhasil
diupdate.\n";
        } else {
            cout << "List masih
kosong!\n";
        }
        break;
    }
    case 4: {
        int posisi;
        string nama;
        int usia;
        cout << "Masukkan posisi
data yang ingin ditambahkan : ";
        cin >> posisi;
        cout << "Masukkan nama :
";
        cin >> nama;
        cout << "Masukkan usia :
";
        cin >> usia;
        insertTengah(nama, usia,
posisi);
        cout << "Data berhasil

```

```

ditambahkan.\n";
        break;
    }
    case 5: {
        if (!isEmpty()) {
            int posisi;
            cout << "Masukkan
posisi data yang ingin dihapus : ";
            cin >> posisi;
            hapusTengah(posisi);
            cout << "Data berhasil
dihapus.\n";
        } else {
            cout << "List masih
kosong!\n";
        }
        break;
    }
    case 6: {
        if (!isEmpty())
{clearList();
            cout << "Seluruh data
berhasil dihapus.\n";
        } else {
            cout << "List masih
kosong!\n";
        }
        break;
    }
    case 7: {
        cout << "Data dalam list:
\n";
        tampil();
        break;
    }
    case 8:
        cout << "Keluar dari
program.\n";
        break;
    default:
        cout << "Pilihan tidak
valid. Coba lagi.\n";
        break;
    }
} while (pilihan != 8);
return 0;

```

```
}
```

## Screenshots Output

```
PS C:\Users\ASUS\OneDrive\Dokumen\semester 2> & 'c:\Users\ASUS\.vscode\extensions\ms-vscode.cpptools-1.19.9-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-z4vgtw5n.o5e' '--stdout=Microsoft-MIEngine-Out-02s1r-s1l.czj' '--stderr=Microsoft-MIEngine-Error-p342ehjn.lkh' '--pid=Microsoft-MIEngine-Pid-z43a150a.bnf' '--dbgExe=c:\mingw32\bin\gdb.exe' '--interpreter=mi'
===== | 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
Masukkan pilihan Anda : 1
Masukkan nama : Faisal
Masukkan usia : 19
Tambah data di 1 depan atau 2 belakang? 1
===== | 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
Masukkan pilihan Anda : 1
Masukkan nama : John
```

```
Masukkan usia : 19
Tambah data di 1 depan atau 2 belakang? 2
===== | 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
Masukkan pilihan Anda : 1
Masukkan nama : Jane
Masukkan usia : 20
Tambah data di 1 depan atau 2 belakang? 2
===== | 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
Masukkan pilihan Anda : 1
Masukkan nama : Michael
Masukkan usia : 18
Tambah data di 1 depan atau 2 belakang? 2
```

```
===== | 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
Masukkan pilihan Anda : 1
Masukkan nama : Yusuke
Masukkan usia : 19
Tambah data di 1 depan atau 2 belakang? 2
===== | 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
Masukkan pilihan Anda : 1
Masukkan nama : Akechi
Masukkan usia : 20
Tambah data di 1 depan atau 2 belakang? 2
===== | 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
Masukkan pilihan Anda : 1
Masukkan nama : Hoshino
Masukkan usia : 18
Tambah data di 1 depan atau 2 belakang? 2
===== | 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
Masukkan pilihan Anda : 1
Masukkan nama : Karin
Masukkan usia : 18
Tambah data di 1 depan atau 2 belakang? 2
===== | Menu | =====
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
```

\*Tidak berjudul - Note... — □ ×

File Edit Lihat ⚙

NIM : 2311102046  
Nama : Faisal Khoiruddin  
Kelas : IF-11-B

Ln 3, Col 16 100% Windows (CRLF) UTF-8

```
6. Hapus Seluruh Data
7. Tampilkan Data
8. Exit
Masukkan pilihan Anda : 7
Data dalam list:
-----
| No | Nama      | Usia |
-----
| 1  | Faisal    | 19   |
| 2  | John      | 19   |
| 3  | Jane      | 20   |
| 4  | Michael   | 18   |
| 5  | Yusuke    | 19   |
| 6  | Akechi    | 20   |
| 7  | Hoshino   | 18   |
| 8  | Karin     | 18   |
-----
===== | 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
Masukkan pilihan Anda : 5
Masukkan posisi data yang ingin dihapus : 6
Data berhasil dihapus.
```

\*Tidak berjudul - Note... — □ ×

File Edit Lihat ⚙

NIM : 2311102046  
Nama : Faisal Khoiruddin  
Kelas : IF-11-B

Ln 3, Col 16 100% Windows (CRLF) UTF-8

```
===== | 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
Masukkan pilihan Anda : 7
Data dalam list:
-----
| No | Nama      | Usia |
-----
| 1  | Faisal    | 19   |
| 2  | John      | 19   |
| 3  | Jane      | 20   |
| 4  | Michael   | 18   |
| 5  | Yusuke    | 19   |
| 6  | Hoshino   | 18   |
| 7  | Karin     | 18   |
-----
===== | Menu | =====
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
```

\*Tidak berjudul - Note... — □ ×

File Edit Lihat ⚙

NIM : 2311102046  
Nama : Faisal Khoiruddin  
Kelas : IF-11-B

Ln 3, Col 16 100% Windows (CRLF) UTF-8



```
7. Tampilkan Data
8. Exit
Masukkan pilihan Anda : 4
Masukkan posisi data yang ingin ditambahkan : 3
Masukkan nama : Futaba
Masukkan usia : 18
Data berhasil ditambahkan.
===== | 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
Masukkan pilihan Anda : 7
Data dalam list:
```

No	Nama	Usia
1	Faisal	19
2	John	19
3	Futaba	18
4	Jane	20
5	Michael	18
6	Yusuke	19
7	Hoshino	18
8	Karin	18

\*Tidak berjudul - Note...

File Edit Lihat

NIM : 2311102046  
Nama : Faisal Khoiruddin  
Kelas : IF-11-B

Ln 3, Col 16 100% Windows (CRLF) UTF-8

```
===== | 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
Masukkan pilihan Anda : 1
Masukkan nama : Igor
Masukkan usia : 20
Tambah data di 1 depan atau 2 belakang? 1
===== | 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
Masukkan pilihan Anda : 7
```

\*Tidak berjudul - Note...

File Edit Lihat

NIM : 2311102046  
Nama : Faisal Khoiruddin  
Kelas : IF-11-B

Ln 3, Col 16 100% Windows (CRLF) UTF-8

```
Data dalam list:
```

No	Nama	Usia
1	Igor	20
2	Faisal	19
3	John	19
4	Futaba	18
5	Jane	20
6	Michael	18
7	Yusuke	19
8	Hoshino	18
9	Karin	18

```
===== | 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
Masukkan pilihan Anda : 3
Masukkan posisi data yang ingin diperbarui : 6
Masukkan nama baru : Reyn
Masukkan usia baru : 18
Data berhasil diupdate.
```

\*Tidak berjudul - Note...

File Edit Lihat

NIM : 2311102046  
Nama : Faisal Khoiruddin  
Kelas : IF-11-B

Ln 3, Col 16 100% Windows (CRLF) UTF-8

```
===== 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
Masukkan pilihan Anda : 7
Data dalam list:
-----
| No | Nama      | Usia |
-----
| 1  | Igor      | 20   |
| 2  | Faisal    | 19   |
| 3  | John      | 19   |
| 4  | Futaba    | 18   |
| 5  | Jane      | 20   |
| 6  | Reyn      | 18   |
| 7  | Yusuke    | 19   |
| 8  | Hoshino   | 18   |
| 9  | Karin     | 18   |
-----
```

\*Tidak berjudul - Note...

File Edit Lihat

NIM : 2311102046  
Nama : Faisal Khoiruddin  
Kelas : IF-11-B

Ln 3, Col 16 100% Windows (CRLF) UTF-8

### Deskripsi:

Program tersebut program program menyimpan Nama dan usia mahasiswa, dengan menggunakan inputan dari user. Contoh program tersebut meminta user menginputkan nama dan usia pada menu tambah data, update data, tambah data urutan tertentu, dan hapus data uruan tertentu. Program tersebut menampilkan nama dan usia.

- `#include <iostream>` ➔ merupakan input output stream header yang digunakan sebagai standar input output operasi yang digunakan di c++
- `#include <iomanip>` ➔ standar untuk menentukan beberapa manipulator yang masing-masing mengambil satu argumen
- `using namespace std;` ➔ digunakan untuk mendeklarasikan/memberitahukan kepada compiler bahwa kita akan menggunakan semua fungsi/class/file yang terdapat dalam namespace std
- `struct Node {` ➔ mendefinisikan struct node
- `string nama;` ➔ deklarasi variabel nama dengan nama tipe data string
- `int usia;` ➔ deklarasi variabel data dengan usia tipe data integer
- `Node* next;` ➔ pointer next untuk menunjuk ke node berikutnya
- `Node* head;` ➔ pointer head untuk menunjuk ke node

pertama dalam list

- `Node* tail;` ➔ pointer tail untuk menunjuk ke node terakhir dalam list
- `void init()` ➔ prosedur init untuk menginisialisasi list
- `head = NULL;` ➔ mengatur head menjadi null
- `tail = NULL;` ➔ mengatur tail menjadi null
- `bool isEmpty()` ➔ fungsi untuk memeriksa apakah list kosong
- `return head == NULL;` ➔ mengembalikan true jika head yaitu null atau list kosong
- `void insertDepan(string nama, int usia) {` ➔ prosedur untuk memasukkan node di depan list dengan parameter string nama, int usia
- `Node* baru = new Node;` ➔ membuat node
- `baru->nama = nama;` ➔ mengatur nama Node baru
- `baru->usia = usia;` ➔ mengatur usia Node baru
- `baru->next = NULL;` ➔ mengatur next Node baru menjadi NULL
- `if (isEmpty()) {` ➔ percabangan if jika list kosong
- `head = tail = baru;` ➔ mengatur head dan tail menjadi Node baru
- `} else {` ➔ jika list tidak kosong
- `baru->next = head;` ➔ mengatur next Node baru menjadi head saat ini
- `head = baru;` ➔ mengatur head menjadi Node baru
- `void insertBelakang(string nama, int usia) {` ➔ prosedur tambah Node di belakang
- `Node* baru = new Node;` ➔ membuat Node baru
- `baru->nama = nama;` ➔ mengatur nama Node baru
- `baru->usia = usia;` ➔ mengatur usia Node baru

- `baru->next = NULL;` ➔ mengatur next Node baru menjadi NULL
- `if (isEmpty()) {` ➔ percabangan if jika list kosong
- `head = tail = baru;` ➔ mengatur head dan tail menjadi Node baru
- `}` else { ➔ jika list tidak kosong
- `tail->next = baru;` ➔ mengatur next Node tail saat ini menjadi Node baru
- `tail = baru;` ➔ mengatur tail menjadi Node baru
- `int hitungList() {` ➔ fungsi untuk menghitung jumlah Node dalam list
- `Node* hitung = head;` ➔ membuat pointer hitung yang menunjuk ke head
- `int jumlah = 0;` ➔ membuat variabel jumlah dan mengatur nilainya menjadi 0
- `while (hitung != NULL) {` ➔ perulangan while selama hitung bukan NULL
- `jumlah++;` ➔ `jumlah++(increment)` menambahkan 1 ke jumlah
- `hitung = hitung->next;` ➔ mengatur hitung menjadi Node berikutnya
- `return jumlah;` ➔ mengembalikan jumlah Node dalam list
- `void insertTengah(string nama, int usia, int posisi) {` ➔ prosedur menambah Node di posisi tengah
- `if (posisi < 1 || posisi > hitungList() + 1) {` ➔ jika posisi kurang dari 1 atau lebih dari jumlah Node + 1
- `cout << "Posisi diluar jangkauan" << endl;` ➔ mencetak pesan error
- `} else if (posisi == 1) {` ➔ jika posisi sama dengan 1
- `cout << "Posisi bukan posisi tengah" << endl;` ➔ cout menampilkan pesan error

- } else { ➔ jika posisi valid
- Node\* baru = new Node(); ➔ membuat Node baru
- baru->nama = nama; ➔ mengatur nama Node baru
- baru->usia = usia; ➔ mengatur usia Node baru
- Node\* bantu = head; ➔ membuat pointer bantu yang menunjuk ke head
- int nomor = 1; ➔ membuat variabel nomor dan mengatur nilainya menjadi 1
- while (nomor < posisi - 1) { ➔ perulangan while selama nomor kurang dari posisi - 1
- bantu = bantu->next; ➔ mengatur bantu menjadi Node berikutnya
- nomor++; ➔ nomor++(increment) menambahkan 1 ke nomor
- baru->next = bantu->next; ➔ mengatur next Node baru menjadi Node setelah bantu
- bantu->next = baru; ➔ Mengatur next Node bantu menjadi Node baru
- void hapusBelakang() { ➔ prosedur untuk menghapus Node di belakang
- if (!isEmpty()) { ➔ percabangan if jika list tidak kosong
- if (head != tail) { ➔ percabangan if jika head bukan tail (lebih dari satu Node dalam list)
- Node\* hapus = tail; ➔ membuat pointer hapus yang menunjuk ke tail
- Node\* bantu = head; ➔ membuat pointer bantu yang menunjuk ke head
- while (bantu->next != tail) { ➔ perulangan while selama next Node bantu bukan tail
- bantu = bantu->next; ➔ mengatur bantu menjadi Node berikutnya

- `tail = bantu;` ➔ mengatur tail menjadi Node bantu
- `tail->next = NULL;` ➔ mengatur next Node tail menjadi NUL
- `delete hapus;` ➔ menghapus Node hapus
- `} else {` ➔ jika hanya ada satu Node dalam list
- `head = tail = NULL;` ➔ mengatur head dan tail menjadi NUL
- `} else {` ➔ Jjika list kosong
- `cout << "List kosong!" << endl;` ➔ cout menampilkan pesan error
- `void hapusTengah(int posisi) {` ➔ fungsi untuk menghapus Node di posisi tengah list
- `if (posisi < 1 || posisi > hitungList()) {` ➔ percabangan jika ika posisi kurang dari 1 atau lebih dari jumlah Node
- `cout << "Posisi diluar jangkauan" << endl;` ➔ cout menampilkan pesan error
- `} else if (posisi == 1) {` ➔ jika posisi adalah 1
- `cout << "Posisi bukan posisi tengah" << endl;` ➔ cout menampilkan pesan error
- `} else {` ➔ mencetak pesan error
- `Node* hapus;` ➔ membuat pointer hapu
- `Node* bantu = head;` ➔ membuat pointer bantu yang menunjuk ke head
- `for (int nomor = 1; nomor < posisi - 1; nomor++) {` ➔ perulangan for untuk nomor dari 1 hingga posisi - 1
- `bantu = bantu->next;` ➔ Mengatur bantu menjadi Node berikutnya
- `hapus = bantu->next;` ➔ mengatur hapus menjadi Node setelah bantu
- `bantu->next = hapus->next;` ➔ mengatur next Node bantu menjadi Node setelah hapus
- `delete hapus;` ➔ m enghapus Node hapus

- `void ubahDepan(string nama, int usia) {` ➔ prosedur untuk mengubah untuk mengubah data Node di depan list
- `if (!isEmpty()) {` ➔ percabangan if jika list tidak kosong
- `head->nama = nama;` ➔ mengatur nama Node head
- `head->usia = usia;` ➔ mengatur usia Node head
- `} else {` ➔ jika list kosong
- `cout << "List masih kosong!" << endl;` ➔ cout menampilkan pesan error
- `void ubahTengah(string nama, int usia, int posisi) {` ➔ prosedur untuk mengubah data Node di posisi tengah list
- `if (!isEmpty()) {` ➔ jika list tidak kosong
- `if (posisi < 1 || posisi > hitungList()) {` ➔ percabangan if jika posisi kurang dari 1 atau lebih dari jumlah Node
- `cout << "Posisi di luar jangkauan" << endl;` ➔ cout menampilkan pesan error
- `} else if (posisi == 1) {` ➔ jika posisi adalah 1
- `cout << "Posisi bukan posisi tengah" << endl;` ➔ mencetak pesan error
- `} else {` ➔ jika posisi valid
- `Node* bantu = head;` ➔ membuat pointer bantu yang menunjuk ke head
- `for (int nomor = 1; nomor < posisi; nomor++) {` ➔ perulangan for untuk nomor dari 1 hingga posisi - 1
- `bantu = bantu->next;` ➔ mengatur bantu menjadi Node berikutnya
- `bantu->nama = nama;` ➔ mengatur nama Node bantu
- `bantu->usia = usia;` ➔ mengatur usia Node bantu
- `else {` ➔ jika list kosong
- `cout << "List masih kosong!" << endl;` ➔ cout menampilkan mencetak pesan error

- `void ubahBelakang(string nama, int usia) {` ➔ prosedur untuk mengubah data Node di belakang list
- `if (!isEmpty()) {` ➔ jika list tidak kosong
- `tail->nama = nama;` ➔ mengatur nama Node tail
- `tail->usia = usia;` ➔ mengatur usia Node tail
- `}` else { ➔ jika list kosong
- `cout << "List masih kosong!" << endl;` ➔ cout menampilkan pesan error
- `void clearList() {` ➔ prosedur untuk menghapus semua Node dalam list
- `Node* bantu = head;` ➔ membuat pointer bantu yang menunjuk ke head
- `while (bantu != NULL) {` ➔ perulangan while selama bantu bukan NULL
- `Node* hapus = bantu;` ➔ membuat pointer hapus yang menunjuk ke Node yang ditunjuk oleh bantu
- `bantu = bantu->next;` ➔ mengatur bantu menjadi Node berikutnya
- `delete hapus;` ➔ menghapus Node yang ditunjuk oleh hapus
- `head = tail = NULL;` ➔ mengatur head dan tail menjadi NULL
- `cout << "List berhasil terhapus!" << endl;` ➔ mencetak pesan bahwa list telah berhasil dihapus
- `void tampil() {` ➔ prosedur untuk menampilkan semua Node dalam list
- `if (!isEmpty()) {` ➔ jika list tidak kosong
- `Node* bantu = head;` ➔ membuat pointer bantu yang menunjuk ke head
- `cout << "-----\n";` ➔ cout mencetak garis
- `cout << "| No | Nama | Usia | \n";` ➔ cout



menampilkan header tabel

- `cout << "-----\n";` ➔ `cout` mencetak garis
- `int no = 1;` ➔ deklarasi membuat variabel `no` dan mengatur nilainya menjadi 1
- `while (bantu != NULL) {` ➔ selama `bantu` bukan `NULL`
- `cout << "| " << left << setw(2) << no << " | " << left << setw(17) << bantu->nama << " | " << left << setw(3) << bantu->usia << " \n";` ➔ mencetak nomor, nama, dan usia Node
- `bantu = bantu->next;` ➔ mengatur `bantu` menjadi Node berikutnya
- `no++;` ➔ `no++(increment)` menambahkan 1 ke `no`
- `cout << "-----\n";` ➔ `cout` mencetak garis
- `} else {` ➔ jika list kosong
- `cout << "List masih kosong!" << endl;` ➔ `cout` menampilkan pesan bahwa list masih kosong
- `int main() {` ➔ merupakan fungsi utama yang akan dieksekusi saat program dijalankan.
- `init();` ➔ memanggil fungsi `init()` untuk menginisialisasi list
- `int pilihan;` ➔ membuat variabel `pilihan` untuk menyimpan pilihan pengguna
- `do {` ➔ memulai loop `do-while`
- `cout << "===== | Menu | =====\n";` ➔ `cout` menampilkan menu
- `cout << "1. Tambah Data\n";` ➔ `cout` menampilkan pilihan menu 1. Tambah Data
- `cout << "2. Hapus Data\n";` ➔ `cout` menampilkan pilihan menu 2. Hapus Data
- `cout << "3. Update Data\n";` ➔ `cout` menampilkan pilihan menu 2. Hapus Data

- `cout << "4. Tambah Data Urutan Tertentu\n";` ➔ `cout` menampilkan pilihan menu 4. Tambah Data Urutan Tertentu
- `cout << "5. Hapus Data Urutan Tertentu\n";` ➔ `cout` menampilkan pilihan menu 5. Hapus Data Urutan Tertentu
- `cout << "6. Hapus Seluruh Data\n";` ➔ `cout` menampilkan pilihan menu 6. Hapus Seluruh Data
- `cout << "7. Tampilkan Data\n";` ➔ `cout` menampilkan pilihan menu 7. Tampilkan Data
- `cout << "8. Exit\n";` ➔ `cout` menampilkan pilihan menu 8. Exit
- `cout << "Masukkan pilihan Anda : ";` ➔ `cout` menampilkan masukkan pilihan Anda
- `cin >> pilihan;` ➔ menerima input pilihan dari pengguna
- `switch (pilihan) {` ➔ memulai switch case berdasarkan pilihan
- `case 1: {` ➔ jika pengguna memilih 1
- `string nama;` ➔ deklarasi membuat variabel nama untuk menyimpan nama
- `int usia;` ➔ deklarasi membuat variabel usia untuk menyimpan usia
- `cout << "Masukkan nama : ";` ➔ `cout` menampilkan masukkan nama
- `cin >> nama;` ➔ menerima input nama dari pengguna
- `cout << "Masukkan usia : ";` ➔ `cout` menampilkan masukkan usia
- `cin >> usia;` ➔ menerima input usia dari pengguna
- `int posisi;` ➔ membuat variabel posisi untuk menyimpan posisi
- `cout << "Tambah data di 1 depan atau 2 belakang? ";` ➔ `cout` menampilkan Tambah data di 1 depan atau 2 belakang?
- `cin >> posisi;` ➔ menerima input posisi dari pengguna

- `if (posisi == 1) {` ➔ jika posisi adalah 1
- `insertDepan(nama, usia);` ➔ memanggil fungsi `insertDepan()` untuk menambahkan Node di depan list
- `} else if (posisi == 2) {` ➔ jika posisi sama dengan 2
- `insertBelakang(nama, usia);` ➔ memanggil fungsi `insertBelakang()` untuk menambahkan Node di belakang list
- `} else {` ➔ jika posisi bukan 1 atau 2
- `cout << "Pilihan tidak valid. Data akan ditambahkan di depan.\n";` ➔ mencetak pesan bahwa pilihan tidak valid
- `insertDepan(nama, usia);` ➔ memanggil fungsi `insertDepan()` untuk menambahkan Node di depan list
- `break;` ➔ mengakhiri case ini
- `case 2: {` ➔ jika pengguna memilih 2
- `if (!isEmpty()) {` ➔ percabangan jika list tidak kosong
- `int posisi;` ➔ membuat variabel posisi untuk menyimpan posisi
- `cout << "Masukkan posisi data yang ingin dihapus : ";` ➔ cout menampilkan m asukkan posisi data yang ingin dihapus
- `cin >> posisi;` ➔ menerima input posisi dari pengguna
- `hapusTengah(posisi);` ➔ memanggil fungsi `hapusTengah()` untuk menghapus Node di posisi tengah list
- `cout << "Data berhasil dihapus.\n";` ➔ cout menampilkan data berhasil dihapus
- `} else {` ➔ jika list kosong
- `cout << "List masih kosong!\n";` ➔ cout menampilkan list masih kosong!
- `break;` ➔ mengakhiri case ini
- `case 3: {` ➔ jika pengguna memilih 3
- `if (!isEmpty()) {` ➔ jika list tidak kosong
- `int posisi;` ➔ membuat variabel posisi untuk menyimpan

posisi

- `string nama;` ➔ deklarasi membuat variabel nama untuk menyimpan nama
- `int usia;` ➔ deklarasi membuat variabel usia untuk menyimpan usia
- `cout << "Masukkan posisi data yang ingin diperbarui : ";` ➔ `cout` menampilkan masukkan posisi data yang ingin diperbarui
- `cin >> posisi;` ➔ menerima input posisi dari pengguna
- `cout << "Masukkan nama baru : ";` ➔ `cout` menampilkan masukkan nama baru :
- `cin >> nama;` ➔ menerima input nama baru dari pengguna
- `cout << "Masukkan usia baru : ";` ➔ `cout` menampilkan masukkan usia baru :
- `cin >> usia;` ➔ menerima input usia baru dari pengguna
- `ubahTengah(nama, usia, posisi);` ➔ memanggil fungsi `ubahTengah()` untuk mengubah data Node di posisi tengah list
- `cout << "Data berhasil diupdate.\n";` ➔ `cout` menampilkan data berhasil diupdate
- `} else {` ➔ jika list kosong
- `cout << "List masih kosong!\n";` ➔ `cout` menampilkan pesan bahwa list masih kosong
- `break;` ➔ mengakhiri case ini
- `case 4: {` ➔ jika pengguna memilih 4
- `int posisi;` ➔ deklarasi membuat variabel posisi untuk menyimpan posisi
- `string nama;` ➔ membuat variabel nama untuk menyimpan nama
- `int usia;` ➔ membuat variabel usia untuk menyimpan usia
- `cout << "Masukkan posisi data yang ingin ditambahkan : ";`

➔ masukkan posisi data yang ingin ditambahkan :

- `cin >> posisi;` ➔ menerima input posisi dari pengguna
- `cout << "Masukkan nama : ";` ➔ `cout` menampilkan Masukkan nama :
- `cin >> nama;` ➔ menerima input nama dari pengguna
- `cout << "Masukkan usia : ";` ➔ `cout` menampilkan masukkan usia
- `cin >> usia;` ➔ menerima input usia dari pengguna
- `insertTengah(nama, usia, posisi);` ➔ memanggil fungsi `insertTengah()` untuk menambahkan Node di posisi tengah list
- `cout << "Data berhasil ditambahkan.\n";` ➔ `cout` menampilkan pesan bahwa data telah berhasil ditambahkan
- `break;` ➔ mengakhiri case in
- `case 5: {` ➔ jika pengguna memilih 5
- `if (!isEmpty()) {` ➔ percabangan if jika list tidak kosong
- `int posisi;` ➔ membuat variabel posisi untuk menyimpan posisi
- `cout << "Masukkan posisi data yang ingin dihapus : ";` ➔ `cout` menampilkan masukkan posisi data yang ingin dihapus
- `cin >> posisi;` ➔ menerima input posisi dari pengguna
- `hapusTengah(posisi);` ➔ memanggil fungsi `hapusTengah()` untuk menghapus Node di posisi tengah list
- `cout << "Data berhasil dihapus.\n";` ➔ `cout` menampilkan data berhasil dihapus
- `} else {` ➔ jika list kosong
- `cout << "List masih kosong!\n";` ➔ `cout` menampilkan bahwa list masih kosong
- `break;` ➔ mengakhiri case ini
- `case 6: {` ➔ jika pengguna memilih 6

- `if (!isEmpty()) {` → Jika list tidak kosong
- `clearList();` → memanggil fungsi `clearList()` untuk menghapus semua Node dalam list
- `cout << "Seluruh data berhasil dihapus.\n";` → `cout` menampilkan seluruh data berhasil dihapus
- `}` `else {` → jika list kosong
- `cout << "List masih kosong!\n";` → `cout` menampilkan bahwa list masih kosong
- `break;` → mengakhiri case ini
- `case 7: {` → jika pengguna memilih 7
- `cout << "Data dalam list: \n";` → `cout` menampilkan data dalam list:
- `tampil();` → memanggil fungsi `tampil()` untuk menampilkan semua Node dalam list
- `break;` → mengakhiri case ini
- `case 8:` → jika pengguna memilih 8
- `cout << "Keluar dari program.\n";` → `cout` menampilkan keluar dari program
- `break;` → mengakhiri case ini
- `default:` → jika pengguna memasukkan pilihan yang tidak valid (diluar dari 1 sampai 8)
- `cout << "Pilihan tidak valid. Coba lagi.\n";` → `cout` menampilkan pilihan tidak valid. Coba lagi
- `break;` → mengakhiri case ini
- `} while (pilihan != 8);` → melakukan perulangan selama pilihan bukan 8 (exit)
- `return 0;` → program akan mengembalikan (return) nilai 0 ke operating sistem yang menjalankan program tersebut

## 2. Unguided 2 : Soal mengenai Double Linked List

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

Nama Produk	Harga
Originote	60.000
Somethinc	150.000
Skintific	100.000

Wardah	50.000
Hanasui	30.000

Case:

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

Toko Skincare Purwokerto

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

## 8. Exit

Pada menu 7, tampilan akhirnya akan menjadi seperti dibawah

ini :

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

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

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

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

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

    void push_back(string namaProduk,
double harga) {
        Node* newNode = new Node;
        newNode->namaProduk = namaProduk;
        newNode->harga = harga;
        newNode->next = nullptr;
        if (tail != nullptr) {
```



```

        tail->next = newNode;
        newNode->prev = tail;
    }
    else {
        head = newNode;
    }
    tail = 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(string oldNamaProduk,
string newNamaProduk, double newHarga) {
    Node* current = head;
    while (current != nullptr) {
        if (current->namaProduk ==
oldNamaProduk) {
            current->namaProduk =
newNamaProduk;
            current->harga =
newHarga;
            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 insertAt(int position, string
namaProduk, double harga) {
    Node* newNode = new Node;
    newNode->namaProduk = namaProduk;
    newNode->harga = harga;
    if (position == 0) {
        newNode->next = head;
        newNode->prev = nullptr;
        if (head != nullptr) {
            head->prev = newNode;
        }
        head = newNode;
    }
    else {
        Node* current = head;
        for (int i = 0; i < position - 1;
i++) { //aslinya - 1
            if (current != nullptr) {
                current = current->next;
            }
        }
        if (current != nullptr &&
current->next != nullptr) {
            newNode->next = current->
>next;
            newNode->prev = current;
            current->next->prev =
newNode;
            current->next = newNode;
        }
    }
}

void deleteAt(int position) {
    if (head == nullptr) {
        return;
    }
    Node* current = head;
    if (position == 0) {
        head = head->next;
    }
}

```

```

        if (head != nullptr) {
            head->prev = nullptr;
        }
        delete current;
    }
    else {
        for (int i = 0; i < position;
i++) {
            if (current != nullptr) {
                current = current-
>next;
            }
        }
        if (current != nullptr) {
            if (current->next !=
nullptr) {
                current->next->prev =
current->prev;
            }
            if (current->prev !=
nullptr) {
                current->prev->next =
current->next;
            }
            delete current;
        }
    }
}

void display() {
    Node* current = head;
    cout << "-----
-----" << endl;
    cout << "| " << left << setw(19) <<
"Nama Produk" << "| " << setw(10) <<
"Harga" << " |" << endl;
    cout << "-----
-----" << endl;
    while (current != nullptr) {
        cout << "| " << left << setw(19)
<< current->namaProduk << "| " <<
setw(10) << current->harga << " |" <<
endl;
        cout << "-----
-----" << endl; // Tambahkan garis
ini

```

```

        current = current->next;
    }
}
};

int main() {
    DoublyLinkedList list;
    list.push_back("Originote", 60000);
    list.push_back("Somethinc", 150000);
    list.push_back("Skintific", 100000);
    list.push_back("Wardah", 50000);
    list.push_back("Hanasui", 30000);
    list.display();

    cout << "\nToko Perawatan Kulit
Purwokerto\n" << endl;
    while (true) {
        cout << "\n1. Tambah data" <<
endl;
        cout << "2. Hapus data" << endl;
        cout << "3. Perbarui data" <<
endl;
        cout << "4. Tambah Data Urutan
Tertentu" << endl;
        cout << "5. Hapus Data Urutan
Tertentu" << endl;
        cout << "6. Hapus semua data" <<
endl;
        cout << "7. Tampilkan data" <<
endl;
        cout << "8. Keluar\n" << endl;
        int choice;
        cout << "Masukkan pilihan Anda
: ";
        cin >> choice;
        switch (choice) {
            case 1: {
                string namaProduk;
                double harga;
                cout << "Masukkan nama produk
: ";
                cin >> namaProduk;
                cout << "Masukkan harga
: ";
                cin >> harga;
                list.push_back(namaProduk,

```

```

harga);
        break;
    }
    case 2: {
        list.pop();
        break;
    }
    case 3: {
        string oldNamaProduk,
newNamaProduk;
        double newHarga;
        cout << "Masukkan nama produk
lama: ";

        cin >> oldNamaProduk;
        cout << "Masukkan nama produk
baru: ";

        cin >> newNamaProduk;
        cout << "Masukkan harga baru
: ";

        cin >> newHarga;
        bool updated =
list.update(oldNamaProduk, newNamaProduk,
newHarga);
        if (!updated) {
            cout << "Data tidak
ditemukan" << endl;
        }
        break;
    }
    case 4: {
        // Tambahkan implementasi
untuk "Tambah Data Urutan Tertentu" di
sini

        string namaProduk;
        double harga;
        int pos;
        cout << "Masukkan posisi
: ";

        cin >> pos;
        cout << "Masukkan nama produk
: ";

        cin >> namaProduk;
        cout << "Masukkan harga
: ";

        cin >> harga;
        list.insertAt(pos,

```

```

namaProduk, harga);
    break;
}
case 5: {
    // Tambahkan implementasi
    untuk "Hapus Data Urutan Tertentu" di
    sini
        int pos;
        cout << "Masukkan posisi
: ";
        cin >> pos;
        list.deleteAt(pos);
        break;
}
case 6: {
    list.deleteAll();
    break;
}
case 7: {
    list.display();
    break;
}
case 8: {
    return 0;
}
default: {
    cout << "Pilihan tidak valid"
<< endl;
    break;
}
}
return 0;
}

```

Screenshots Output

```
PS C:\Users\ASUS\OneDrive\Dokumen\semester 2> & 'c:\Users\ASUS\.vscode\extensions\ms-vscode.cpptools-1.19.9-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-ap01mbon.gfz' '--stdout=Microsoft-MIEngine-Out-agpk0xnx.f34' '--stderr=Microsoft-MIEngine-Error-adwzon0t.3ik' '--pid=Microsoft-MIEngine-Pid-ii1xuny.de3' '--dbgExe=c:\mingw32\bin\gdb.exe' '--interpreter=mi'
```

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

Toko Perawatan Kulit Purwokerto

1. Tambah data
2. Hapus data
3. Perbarui data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus semua data
7. Tampilkan data
8. Keluar

\*Tidak berjudul - Notepad

File Edit Lihat

NIM : 2311102046  
Nama : Faisal Khoiruddin  
Kelas : IF-11-B

Ln 3, Col 16 100% Windows (CRLF) UTF-8

```
Masukkan pilihan Anda : 4  
Masukkan posisi : 2  
Masukkan nama produk : Azarine  
Masukkan harga : 65000
```

1. Tambah data
2. Hapus data
3. Perbarui data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus semua data
7. Tampilkan data
8. Keluar

```
Masukkan pilihan Anda : 5  
Masukkan posisi : 4
```

1. Tambah data
2. Hapus data
3. Perbarui data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus semua data
7. Tampilkan data
8. Keluar

\*Tidak berjudul - Notepad

File Edit Lihat

NIM : 2311102046  
Nama : Faisal Khoiruddin  
Kelas : IF-11-B

Ln 3, Col 16 100% Windows (CRLF) UTF-8

```
Masukkan pilihan Anda : 3  
Masukkan nama produk lama: Hanasui  
Masukkan nama produk baru: Cleora  
Masukkan harga baru : 55000
```

1. Tambah data
2. Hapus data
3. Perbarui data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus semua data
7. Tampilkan data
8. Keluar

```
Masukkan pilihan Anda : 7
```

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

\*Tidak berjudul - Notepad

File Edit Lihat

NIM : 2311102046  
Nama : Faisal Khoiruddin  
Kelas : IF-11-B

Ln 3, Col 16 100% Windows (CRLF) UTF-8

```
Masukkan pilihan Anda : 7
-----
| Nama Produk | Harga |
-----
| Originote   | 60000 |
-----
| Somethinc   | 150000|
-----
| Azarine     | 65000 |
-----
| Skintific   | 100000|
-----
| Cleora      | 55000 |
-----

1. Tambah data
2. Hapus data
3. Perbarui data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus semua data
7. Tampilkan data
8. Keluar

Masukkan pilihan Anda : 8
PS C:\Users\ASUS\OneDrive\Dokumen\semester 2>
```

```
*Tidak berjudul - Notepad
File Edit Lihat
NIM : 2311102046
Nama : Faisal Khoiruddin
Kelas : IF-11-B
Ln 3, Col 16 100% Windows (CRLF) UTF-8
```

Deskripsi:

Program tersebut merupakan program double linked list. Program tersebut penambahan operasi untuk menambah data, menghapus, dan update di tengah / di urutan tertentu yang diminta. Program tersebut menampilkan output nama produk dan harga.

- `#include <iostream>` ➔ merupakan input output stream header yang digunakan sebagai standar input output operasi yang digunakan di c++
- `#include <iomanip>` ➔
- `using namespace std;` ➔ digunakan untuk mendeklarasikan/ memberitahukan kepada compiler bahwa kita akan menggunakan semua fungsi/class/file yang terdapat dalam namespace std
- `class Node {` ➔ mendefinisikan kelas Node untuk menyimpan data produk
- `public:` ➔ mengatur class ke public
- `string namaProduk;` ➔ deklarasi variabel untuk menyimpan nama produk
- `double harga;` ➔ deklarasi variabel untuk menyimpan harga produk dengan tipe data double
- `Node* prev;` ➔ pointer ke node sebelumnya dalam daftar
- `Node* next;` ➔ pointer ke node berikutnya dalam daftar



- `class DoublyLinkedList {` ➔ mendefinisikan kelas `DoublyLinkedList` untuk menyimpan dan mengelola daftar produk
- `public:` ➔ mengatur class ke public
- `Node* head;` ➔ pointer ke node pertama dalam daftar
- `Node* tail;` ➔ pointer ke node terakhir dalam daftar
- `DoublyLinkedList() {` ➔ konstruktor untuk menginisialisasi daftar kosong
- `head = nullptr;` ➔ head diatur ke nullptr (list tersebut kosong dan tidak ada elemen pertama)
- `tail = nullptr;` ➔ tail diatur ke nullptr (list tersebut kosong dan tidak ada elemen terakhir)
- `void push_back(string namaProduk, double harga) {` ➔ prosedur untuk menambahkan produk baru ke akhir daftar dengan parameter string `namaProduk`, double `harga`
- `Node* newNode = new Node;` ➔ membuat node baru
- `newNode->namaProduk = namaProduk;` ➔ menetapkan nama produk
- `newNode->harga = harga;` ➔ menetapkan harga
- `newNode->next = nullptr;` ➔ menetapkan pointer berikutnya ke null
- `if (tail != nullptr) {` ➔ jika ada node dalam daftar
- `tail->next = newNode;` ➔ Menetapkan node baru sebagai node berikutnya dari node terakhir
- `newNode->prev = tail;` ➔ Menetapkan node terakhir sebagai node sebelumnya dari node baru
- `else {` ➔ jika tidak ada node dalam daftar
- `head = newNode;` ➔ Menetapkan node baru sebagai node pertama
- `tail = newNode;` ➔ menetapkan node baru sebagai node terakhir

- `void pop() {` ➔ prosedur untuk menghapus produk pertama dari daftar
- `if (head == nullptr) {` ➔ Jika tidak ada node dalam daftar
- `return;` ➔ keluar dari fungsi
- `Node* temp = head;` ➔ menyimpan node pertama dalam variabel sementara
- `head = head->next;` ➔ menetapkan node kedua sebagai node pertama
- `if (head != nullptr) {` ➔ jika ada node kedua
- `head->prev = nullptr;` ➔ menetapkan node sebelumnya dari node pertama ke null
- `else {` ➔ jika tidak ada node kedua
- `tail = nullptr;` ➔ menetapkan node terakhir ke null
- `delete temp;` ➔ menghapus node pertama
- `bool update(string oldNamaProduk, string newNamaProduk, double newHarga) {` ➔ Fungsi untuk memperbarui nama dan harga produk dengan parameter `string oldNamaProduk, string newNamaProduk, double newHarga`
- `Node* current = head;` ➔ mulai dari node pertama
- `while (current != nullptr) {` ➔ selama belum mencapai akhir daftar
- `if (current->namaProduk == oldNamaProduk) {` ➔ jika menemukan produk dengan nama lama
- `current->namaProduk = newNamaProduk;` ➔ perbarui nama produk
- `current->harga = newHarga;` ➔ memperbarui harga
- `return true;` ➔ mengembalikan true untuk menunjukkan bahwa pembaruan berhasil
- `current = current->next;` ➔ pindah ke node berikutnya
- `return false;` ➔ jika tidak menemukan produk dengan

nama lama, kembalikan false

- `void deleteAll() {` ➔ prosedur untuk menghapus semua produk dari daftar
- `Node* current = head;` ➔ mulai dari node pertama
- `while (current != nullptr) {` ➔ perulangan while selama belum mencapai akhir daftar
- `Node* temp = current;` ➔ menyimpan node saat ini dalam variabel sementara
- `current = current->next;` ➔ pindah ke node berikutnya
- `delete temp;` ➔ menghapus node saat ini
- `head = nullptr;` ➔ menetapkan node pertama ke null
- `tail = nullptr;` ➔ menetapkan node terakhir ke null
- `void insertAt(int position, string namaProduk, double harga) {` ➔ prosedur untuk memasukkan produk baru pada posisi tertentu dalam daftar
- `Node* newNode = new Node;` ➔ membuat node baru
- `newNode->namaProduk = namaProduk;` ➔ menetapkan nama produk
- `newNode->harga = harga;` ➔ untuk menetapkan harga
- `if (position == 0) {` ➔ jika posisi sama dengan 0
- `newNode->next = head;` ➔ menetapkan node pertama sebagai node berikutnya dari node baru
- `newNode->prev = nullptr;` ➔ menetapkan node sebelumnya dari node baru ke null
- `if (head != nullptr) {` ➔ percabangan if jika node head tidak sama dengan null
- `head->prev = newNode;` ➔ menetapkan node baru sebagai node sebelumnya dari node pertama
- `head = newNode;` ➔ menetapkan node baru sebagai node pertama
- `else {` ➔ jika posisi bukan 0

- `Node* current = head;` ➔ mulai dari node pertama
- `for (int i = 0; i < position - 1; i++) {` ➔ perulangan for dengan kondisi `i = 0 ; i < position - 1; i++(increment)`
- `if (current != nullptr) {` ➔ jika belum mencapai akhir daftar
- `current = current->next;` ➔ pindah ke node berikutnya
- `if (current != nullptr && current->next != nullptr) {` ➔ Jika posisi yang ditargetkan ada dalam daftar
- `newNode->next = current->next;` ➔ menetapkan node setelah posisi yang ditargetkan sebagai node berikutnya dari node baru
- `newNode->prev = current;` ➔ menetapkan node sebelum posisi yang ditargetkan sebagai node sebelumnya dari node baru
- `current->next->prev = newNode;` ➔ menetapkan node baru sebagai node sebelumnya dari node setelah posisi yang ditargetkan
- `current->next = newNode;` ➔ menetapkan node baru sebagai node berikutnya dari node sebelum posisi yang ditargetkan
- `void deleteAt(int position) {` ➔ prosedur untuk menghapus produk pada posisi tertentu dalam daftar
- `if (head == nullptr) {` ➔ jika tidak ada node dalam daftar
- `return;` ➔ keluar dari fungsi
- `Node* current = head;` ➔ mulai dari node pertama
- `if (position == 0) {` ➔ jika posisi sama dengan 0
- `head = head->next;` ➔ menetapkan node kedua sebagai node pertama
- `if (head != nullptr) {` ➔ jika ada node kedua
- `head->prev = nullptr;` ➔ menetapkan node sebelumnya dari node pertama ke null
- `delete current;` ➔ menghapus node pertama

- `else {` ➔ jika posisi bukan 0
- `for (int i = 0; i < position; i++) {` ➔ pindah ke node pada posisi yang ditargetkan
- `if (current != nullptr) {` ➔ jika belum mencapai akhir daftar
- `current = current->next;` ➔ pindah ke node berikutnya
- `if (current != nullptr) {` ➔ jika posisi yang ditargetkan ada dalam daftar
- `if (current->next != nullptr) {` ➔ jika ada node setelah posisi yang ditargetkan
- `current->next->prev = current->prev;` ➔ menetapkan node sebelum posisi yang ditargetkan sebagai node sebelumnya dari node setelah posisi yang ditargetkan
- `if (current->prev != nullptr) {` ➔ jika ada node sebelum posisi yang ditargetkan
- `current->prev->next = current->next;` ➔ menetapkan node setelah posisi yang ditargetkan sebagai node berikutnya dari node sebelum posisi yang ditargetkan
- `delete current;` ➔ menghapus node pada posisi yang ditargetkan
- `void display() {` ➔ prosedur untuk menampilkan semua produk dalam daftar
- `Node* current = head;` ➔ mulai dari node pertama
- `cout << "-----" << endl;` ➔ cout menampilkan garis
- `cout << "| " << left << setw(19) << "Nama Produk" << "| "`  
`<< setw(10) << "Harga" << " |" << endl;` ➔ cout menampilkan nama produk dan harga
- `cout << "-----" << endl;` ➔ cout menampilkan garis
- `while (current != nullptr) {` ➔ selama belum mencapai akhir daftar

- `cout << "|" << left << setw(19) << current->namaProduk << "|" << setw(10) << current->harga << " |" << endl;` ➔ `cout` menampilkan nama produk dan harga
- `cout << "-----" << endl;` ➔ `cout` menampilkan garis
- `current = current->next;` ➔ pindah ke node berikutnya
- `int main() {` ➔ merupakan fungsi utama yang akan dieksekusi saat program dijalankan.
- `DoublyLinkedList list;` ➔ membuat daftar produk
- `list.push_back("Originote", 60000);` ➔ menambahkan produk baru ke daftar dengan nama "Originote" dan harga 60000
- `list.push_back("Somethinc", 150000);` ➔ menambahkan produk baru ke daftar dengan nama "Somethinc" dan harga 150000
- `list.push_back("Skintific", 100000);` ➔ menambahkan produk baru ke daftar dengan nama "Skintific" dan harga 100000
- `list.push_back("Wardah", 50000);` ➔ menambahkan produk baru ke daftar dengan nama "Wardah" dan harga 50000
- `list.push_back("Hanasui", 30000);` ➔ menambahkan produk baru ke daftar dengan nama "Hanasui" dan harga 30000
- `list.display();` ➔ menampilkan semua produk dalam daftar
- `cout << "\nToko Perawatan Kulit Purwokerto\n" << endl;` ➔ `cout` menampilkan judul program yaitu Toko Perawatan Kulit Purwokerto
- `while (true) {` ➔ perulangan `while` akan terus berjalan selama kondisi `true` dan akan berhenti jika ada perintah `break`
- `cout << "\n1. Tambah data" << endl;` ➔ `cout` menampilkan pilihan menu menambahkan data

- `cout << "2. Hapus data" << endl;` ➔ `cout` menampilkan pilihan menu menghapus data
- `cout << "3. Perbarui data" << endl;` ➔ `cout` menampilkan menu memperbarui data
- `cout << "4. Tambah Data Urutan Tertentu" << endl;` ➔ `cout` menampilkan menu menambahkan data pada urutan tertentu
- `cout << "5. Hapus Data Urutan Tertentu" << endl;` ➔ `cout` menampilkan pilihan menu Hapus Data Urutan Tertentu
- `cout << "6. Hapus semua data" << endl;` ➔ `cout` menampilkan pilihan menu Hapus semua data
- `cout << "7. Tampilkan data" << endl;` ➔ `cout` menampilkan pilihan menu Tampilkan data
- `cout << "8. Keluar\n" << endl;` ➔ `cout` menampilkan menu keluar
- `int choice;` ➔ deklarasi variabel untuk menyimpan pilihan pengguna
- `cout << "Masukkan pilihan Anda : ";` ➔ `cout` menampilkan statement masukkan pilihan anda
- `cin >> choice;` ➔ menerima inputan dari pengguna dan menyimpan dalam variabel `choice`
- `switch (choice) {` ➔ memilih operasi berdasarkan pilihan pengguna
- `case 1: {` ➔ jika pengguna memilih untuk menambahkan data
- `string namaProduk;` ➔ deklarasi variabel untuk menyimpan nama produk
- `double harga;` ➔ variabel untuk menyimpan harga
- `cout << "Masukkan nama produk : ";` ➔ `cout` menampilkan pernyataan masukkan nama produk
- `cin >> namaProduk;` ➔ `cin` menerima inputan dari pengguna dan menyimpan dalam variabel `namaProduk`

- `cout << "Masukkan harga : "; ➔` `cout` menampilkan pernyataan masukkan harga
- `cin >> harga; ➔` menerima inputan dari pengguna dan menyimpan dalam variabel harga
- `list.push_back(namaProduk, harga); ➔` menambahkan produk ke daftar
- `break; ➔` keluar dari switch
- `case 2: { ➔` jika pengguna memilih untuk menghapus data
- `list.pop(); ➔` menghapus produk pertama dari daftar
- `break; ➔` keluar dari switch
- `case 3: { ➔` jika pengguna memilih untuk memperbarui data
- `string oldNamaProduk, newNamaProduk; ➔` variabel untuk menyimpan nama produk lama dan baru
- `double newHarga; ➔` deklarasi variabel untuk menyimpan harga baru
- `cout << "Masukkan nama produk lama: "; ➔` `cout` menampilkan pernyataan masukkan nama produk lama
- `cin >> oldNamaProduk; ➔` menerima inputan dari pengguna dan menyimpan pada variabel `oldNamaProduk`
- `cout << "Masukkan nama produk baru: "; ➔` `cout` menampilkan pernyataan masukkan nama produk baru
- `cin >> newNamaProduk; ➔` menerima inputan dari pengguna dan menyimpan pada variabel `newNamaProduk`
- `cout << "Masukkan harga baru : "; ➔` `cout` menampilkan pernyataan masukkan harga baru
- `cin >> newHarga; ➔` menerima inputan dari pengguna dan menyimpan pada variabel `newHarga`
- `bool updated = list.update(oldNamaProduk, newNamaProduk, newHarga);`



- `if (!updated) {` ➔ Memperbarui produk dalam daftar
- `cout << "Data tidak ditemukan" << endl;` ➔ `cout` menampilkan pesan kesalahan
- `break;` ➔ Keluar dari switch
- `case 4: {` ➔ jika pengguna memilih untuk menambahkan data pada urutan tertentu
- `string namaProduk;` ➔ variabel untuk menyimpan nama produk
- `double harga;` ➔ deklarasi variabel untuk menyimpan harga
- `int pos;` ➔ deklarasi variabel untuk menyimpan posisi
- `cout << "Masukkan posisi : ";` ➔ `cout` menampilkan pernyataan masukkan posisi
- `cin >> pos;` ➔ menerima inputan dari pengguna dan menyimpan pada variabel `pos`
- `cout << "Masukkan nama produk : ";` ➔ `cout` menampilkan pernyataan masukkan posisi
- `cin >> namaProduk;` ➔ menerima inputan dari pengguna dan menyimpan pada variabel `namaProduk`
- `cout << "Masukkan harga : ";` ➔ `cout` menampilkan pernyataan masukkan posisi
- `cin >> harga;` ➔ menerima inputan dari pengguna dan menyimpan pada variabel `harga`
- `list.insertAt(pos, namaProduk, harga);` ➔ menambahkan produk pada posisi tertentu dalam daftar
- `break;` ➔ keluar dari switch
- `case 5: {` ➔ jika pengguna memilih untuk menghapus data pada urutan tertentu
- `int pos;` ➔ deklarasi variabel untuk menyimpan posisi
- `cout << "Masukkan posisi : ";` ➔ `cout` menampilkan pernyataan masukkan posisi

- `cin >> pos;` ➔ menerima inputan dari pengguna dan menyimpan pada variabel `pos`
- `list.deleteAt(pos);` ➔ menghapus produk pada posisi tertentu dalam daftar
- `break;` ➔ keluar dari switch
- `case 6: {` ➔ jika pengguna memilih untuk menghapus semua data
- `list.deleteAll();` ➔ menghapus semua produk dari daftar
- `break;` ➔ keluar dari switch
- `case 7: {` ➔ jika pengguna memilih untuk menampilkan data
- `list.display();` ➔ menampilkan semua produk dalam daftar
- `break;` ➔ keluar dari switch
- `case 8: {` ➔ jika pengguna memilih untuk keluar
- `return 0;` ➔ program akan mengembalikan (return) nilai 0 ke operating sistem yang menjalankan program tersebut
- `default: {` ➔ jika pengguna memilih opsi yang tidak valid yaitu diluar 1 sampai 8
- `cout << "Pilihan tidak valid" << endl;` ➔ `cout` menampilkan pernyataan tidak valid
- `break;` ➔ keluar dari switch
- `return 0;` ➔ program akan mengembalikan (return) nilai 0 ke operating sistem yang menjalankan program tersebut

## **D. Kesimpulan**

### **1) Single Linked List**

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

### **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.

## **E. Referensi**

[1] Asisten Praktikum, Struktur Data "ARRAY", WhatsApp, 2024.

[2] BobTylerMSFT. (n.d.).<iomanip>. diakses dari

<https://learn.microsoft.com/id-id/cpp/standard-library/iomanip?view=msvc-170>

[3] Rini Wongso,S.Kom.,M.T.I. Single Linked List. diakses dari

<https://socs.binus.ac.id/2017/03/15/single-linked-list/>