

# **LAPORAN PRAKTIKUM STRUKTUR DATA**

## **MODUL VIII**

Laporan ini disusun untuk memenuhi Tugas Mata Kuliah  
Praktikum struktur data

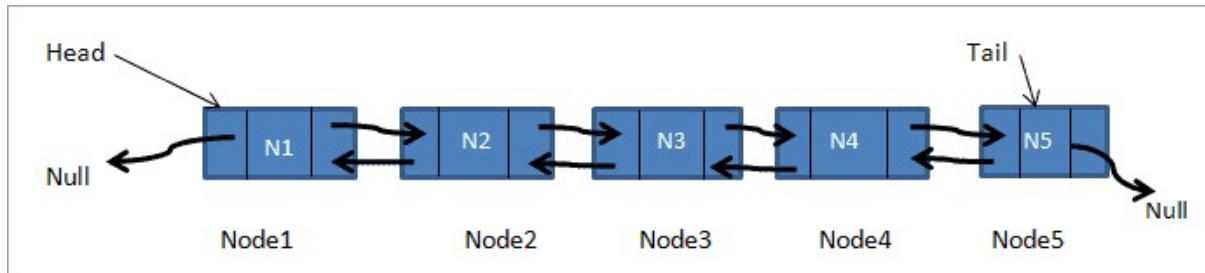


Disusun Oleh :  
M HAMKA ZAINUL ARDHI  
NIM: 2311103156

**PROGRAM STUDI S1 SISTEM INFORMASI**  
**FAKULTAS INFORMATIKA**  
**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**  
**2024**

## DASAR TEORI

Seperti halnya daftar singel linked list, daftar tertaut ganda juga memiliki kepala dan ekor. Penunjuk kepala sebelumnya disetel ke NULL karena ini adalah node pertama. Penunjuk berikutnya dari simpul ekor diatur ke NULL karena ini adalah simpul terakhir



Pada gambar di atas, kita melihat bahwa setiap node memiliki dua pointer, satu menunjuk ke node sebelumnya dan yang lainnya menunjuk ke node berikutnya. Hanya simpul pertama (kepala) yang simpul sebelumnya disetel ke nol dan simpul terakhir (ekor) yang penunjuk berikutnya disetel ke nol.

Karena daftar double linked list berisi dua petunjuk yaitu sebelumnya dan berikutnya, kita dapat menelusurinya ke arah maju dan mundur. Ini adalah keuntungan utama dari daftar double linked list dibandingkan daftar singel linked list.

Dalam deklarasi C++, sebuah simpul dari daftar double linked list direpresentasikan sebagai berikut:

```
simpul struct
{
    simpul struct *sebelumnya;
    ke dalam data;
    simpul struct *berikutnya;
};
```

simpul 'struct sebelumnya' ini adalah pointer yang menunjuk ke simpul sebelumnya dalam linked list. Dengan menggunakan pointer ini, setiap simpul dapat mengakses simpul sebelumnya dalam urutan linked list.

selanjutnya 'dalam data' ini adalah bagian di mana data sebenarnya disimpan. Data ini bisa berupa bilangan bulat, string, atau tipe data lainnya yang sesuai dengan kebutuhan aplikasi. Biasanya, ini adalah informasi yang ingin disimpan di dalam simpul linked list. simpul struct 'berikutnya' Ini adalah pointer yang

menunjuk ke simpul berikutnya dalam linked list. Pointer ini memungkinkan setiap simpul untuk mengakses simpul berikutnya dalam urutan linked list.

Dan ada beberapa operasi dasar dalam linked list :

### 1. Operasi Penambahan Data

```
void tambahDepan(string dataUser){
if(dLinkKosong() == true){
vertex = new dlinkedlist;
vertex->data = dataUser;
vertex->prev = NULL;
vertex->next = NULL;
head = vertex;
tail = vertex;
}else{
vertex = new dlinkedlist;
vertex->data = dataUser;
vertex->prev = NULL;
vertex->next = NULL;
vertex->next = head;
head->prev = vertex;
head = vertex;
}
}
```

### 2. Operasi penghapusan data

```
void hapusBelakang(){
if(dLinkKosong() == true){
cout<<"Kosong...!!!"<<endl;
}else{
if(head == tail){
dlinkedlist* helper;
helper = head;
head = NULL;
tail = NULL;
delete helper;
}else{
dlinkedlist* helper;
helper = tail;
tail = tail->prev;
tail->next = NULL;
helper->prev = NULL;
delete helper;
}
}
```

## GUIDED

### 1. Guided 1

Membuat program dasar yang menerapkan struktur data double linkedlist dalam bahasa pemograman c++

```
#include <iostream>

using namespace std;

struct node
{
    string nama;
    node *front;
    node *back;
};

node *train;
node *head;
node *tail;
node *helperAF;
node *helperAM;

int main () {

    train = new node;
    train->nama = "afik";
    train->front = NULL;
    train->back = NULL;
    head = train;
    tail = train;

    train = new node;
    train->nama = "ani";
    train->back= head;
    train->front = NULL;
    head->front = train;
    head = train;

    train = new node;
    train->nama = "amelia";
    tail->back= train;
    train->front = tail;
    tail = train;
    train->back = NULL;

    train = new node;
    train->nama = "asbi";
    tail->front = train;
```

```
head->back->back = train;
train->back = tail;
train->front = head->back;
```

ini adalah program yang sangat dasar dalam struktur link list penulisan syntac manual tanpa menggunakan variabel apapun untuk saling menyambung data satu dengan data lain tentu nya cara ini efektif untuk memberikan pemahaman awal terhadap seseorang yang baru belajar link list namun nyatanya ini bukanlah program link list yang efektif, dengan penambahan helper yang berfungsi untuk untuk menambahkan sebuah simpul baru di antara dua simpul tertentu dalam linked list secara efektif tanpa harus membuat sytac manual nya Untuk menggunakan ini, jangan lupa untuk menambahkan pointer helper

```
helperAF = head;
while (helperAF->nama != "afiq" && helperAF != NULL)
{
    helperAF = helperAF->back;
}
if (helperAF != NULL && helperAF !=tail){
helperAM = helperAF->back;
train = new node;
train->nama = "asbi";
helperAF->back = train;
train->front =helperAF;
train->back = helperAM;
helperAM->front = train;
}
```

Syntac kode ini menjadikan program lebih efisien dengan cara kerja Program akan membuat simpul baru dengan nilai nama "asbi", kemudian menyisipkannya di antara simpul yang memiliki nilai nama "afiq" (simpul sebelumnya) dan simpul sebelumnya dari simpul "afiq" tersebut Dengan demikian, bagian ini bertujuan untuk menyisipkan simpul baru dengan nilai "asbi" di antara simpul yang memiliki nilai "afiq" dan simpul sebelumnya dari simpul "afiq" tersebut tanpa harus membuat manual untuk membuat code hubungan hubungan seperti code di awal tadi.

## 2. Guided 2

membuat program yang menerapkan struktur data double linkedlist circular, yang terdapat fungsi tambah depan, tambah belakang, hapus depan, hapus belakang, tampil data menggunakan bahasa pemrograman c++

Source code

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

struct dlinkedlist {
    dlinkedlist* prev;
    string data;
    dlinkedlist* next;
};

dlinkedlist* head;
dlinkedlist* tail;
dlinkedlist* vertex;

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

bool dLinkKosong() {
    if (head == NULL && tail == NULL) {
        return true;
    } else {
        return false;
    }
}

void tambahDepan(string dataUser) {
    if (dLinkKosong() == true) {
        vertex = new dlinkedlist;
        vertex->data = dataUser;
        vertex->prev = NULL;
        vertex->next = NULL;
        head = vertex;
        tail = vertex;
    } else {
        vertex = new dlinkedlist;
        vertex->data = dataUser;
        vertex->prev = NULL;
        vertex->next = head;
        head->prev = vertex;
    }
}
```

```

        head = vertex;
    }
}

void tambahBelakang(string dataUser) {
    if (dLinkKosong() == true) {
        vertex = new dlinkedlist;
        vertex->data = dataUser;
        vertex->prev = NULL;
        vertex->next = NULL;
        head = vertex;
        tail = vertex;
    } else {
        vertex = new dlinkedlist;
        vertex->data = dataUser;
        vertex->prev = tail;
        vertex->next = NULL;
        tail->next = vertex;
        tail = vertex;
    }
}

void hapusDepan() {
    if (dLinkKosong() == true) {
        cout << "Kosong...!!!" << endl;
    } else {
        dlinkedlist* helper;
        helper = head;
        if (head == tail) {
            head = NULL;
            tail = NULL;
            delete helper;
        } else {
            head = head->next;
            head->prev = NULL;
            helper->next = NULL;
            delete helper;
        }
    }
}

void hapusBelakang() {
    if (dLinkKosong() == true) {
        cout << "Kosong...!!!" << endl;
    } else {
        if (head == tail) {
            dlinkedlist* helper;
            helper = head;

```

```

        head = NULL;
        tail = NULL;
        delete helper;
    } else {
        dlinkedList* helper;
        helper = tail;
        tail = tail->prev;
        tail->next = NULL;
        helper->prev = NULL;
        delete helper;
    }
}

void tampilData() {
    if (dLinkKosong() == true) {
        cout << "Kosong...!!!" << endl;
    } else {
        dlinkedList* helper;
        helper = head;
        while (helper != NULL) {
            cout << helper->data << " ";
            helper = helper->next;
        }
    }
}

int main() {
    inisialisasi();
    tambahDepan("Bandung");
    tambahDepan("Aceh");
    tambahBelakang("Ciawi");
    hapusDepan();
    tambahDepan("Ambon");
    hapusBelakang();
    hapusDepan();
    hapusBelakang();
    hapusBelakang();
    tampilData();
    return 0;
}

```

Di awali dengan menerapkan struct dan diisi dengan pionter prev dan next yang artinya di double link list setiap simpul memiliki dua pointer, yaitu satu pointer yang menunjuk ke simpul sebelumnya dan satu pointer yang menunjuk ke simpul berikutnya dalam urutan inilah yang



disebut circulation. Ini lah yang membedakan double link list dan singel link list yang hanya mempunya simpul untuk menyambung kedepan

Program ini dilengkapi dengan penambahan data, penghapusan data dan menampilkan output data, dengan menggunakan fungsi bertipe data 'void' yang bertanggung jawab untuk mengatur fungsi fungsi yang ada di program Dalam program ini, fungsi 'tambahDepan()' digunakan untuk menambahkan data di depan linked list, 'hapusDepan()' digunakan untuk menghapus data di depan linked list, dan 'tampilData()' digunakan untuk menampilkan data dalam linked list. Semua fungsi ini menggunakan tipe data void karena mereka tidak mengembalikan nilai.

## Hasil Program

```
PS C:\Users\hamka\OneDrive\Documents\PRAKTIKUM STRUKTUR DATA\MODUL 8> cd "c:\Users\hamka\OneDrive\Documents\PRAKTIKUM STRUKTUR DATA\MODUL 8\" ; if ($?) { g++ GUIDED2.CPP -o GUIDED2 } ; if ($?) { .\GUIDED2 }  
Kosong...!!!  
Kosong...!!!  
PS C:\Users\hamka\OneDrive\Documents\PRAKTIKUM STRUKTUR DATA\MODUL 8>
```

Melihat dari perintah yang di minta untuk menghapus nilai nilai yang telah disini kan sehingga output yang diberikan adalah tampilan dari fungsi 'tampildata()' yang jika keadaan link list nya kosong bernilai 'true' maka akan di tampilkan 'kosong...!' sebagai indikator bahwa data data sudah di hapus dan menjadi kan link list kosong

### 3. Guided 3

Seperti hal nya guided 2 program ini bertujuan sama sama membuat double linked list namun dengan versi circular yang berarti link list nya head dan tail nya menyambung pointer nextnya menunjuk pada dirinya sendiri. Jika Single Linked List tersebut terdiri dari beberapa node, maka pointer next pada node terakhir akan menunjuk ke node terdepannya.

#### Source Code

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

struct dlinkedlist {
    dlinkedlist* prev;
    string data;
    dlinkedlist* next;
};

dlinkedlist* head;
dlinkedlist* tail;
dlinkedlist* vertex;

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

bool dLinkKosong() {
    if (head == NULL && tail == NULL) {
        return true;
    } else {
        return false;
    }
}

void tambahDepan(string dataUser) {
    if (dLinkKosong() == true) {
        vertex = new dlinkedlist;
        vertex->data = dataUser;
        head = vertex;
        tail = vertex;
        head->prev = tail;
        tail->next = head;
    } else {
        vertex = new dlinkedlist;
        vertex->data = dataUser;
        vertex->next = head;
```

```

        head->prev = vertex;
        head = vertex;
        head->prev = tail;
        tail->next = head;
    }
}

void tambahBelakang(string dataUser) {
    if (dLinkKosong() == true) {
        vertex = new dlinkedlist;
        vertex->data = dataUser;
        head = vertex;
        tail = vertex;
        head->prev = tail;
        tail->next = head;
    } else {
        vertex = new dlinkedlist;
        vertex->data = dataUser;
        vertex->prev = tail;
        tail->next = vertex;
        tail = vertex;
        tail->next = head;
        head->prev = tail;
    }
}

void hapusDepan() {
    if (dLinkKosong() == true) {
        cout << "Kosong...!!!" << endl;
    } else {
        dlinkedlist* helper;
        helper = head;
        if (head == tail) {
            tail->next = NULL;
            head->prev = NULL;
            head = NULL;
            tail = NULL;
            delete helper;
        } else {
            head = head->next;
            head->prev = tail;
            tail->next = head;
            helper->next = NULL;
            delete helper;
        }
    }
}

```

```

void hapusBelakang() {
    if (dLinkKosong() == true) {
        cout << "Kosong...!!!" << endl;
    } else {
        if (head == tail) {
            dlinkedlist* helper;
            helper = head;
            head->prev = NULL;
            tail->next = NULL;
            head = NULL;
            tail = NULL;
            delete helper;
        } else {
            dlinkedlist* helper;
            helper = tail;
            tail = tail->prev;
            tail->next = head;
            head->prev = tail;
            helper->prev = NULL;
            helper->next = NULL;
            delete helper;
        }
    }
}

void tampilData() {
    if (dLinkKosong() == true) {
        cout << "Kosong...!!!" << endl;
    } else {
        dlinkedlist* helper;
        helper = head;
        do {
            cout << helper->data << " ";
            helper = helper->next;
        } while (helper != head);
    }
}

int main() {
    inisialisasi();
    tambahDepan("Bandung");
    tambahDepan("Aceh");
    tambahBelakang("Ciamis");
    hapusDepan();
    hapusBelakang();
    hapusBelakang();
    tampilData();
    return 0;
}

```

```
}
```

Sekilas untuk tujuan dan pengoprasian program di guided 2 dan guided 3 adalah sama karena sama sama menerapkan struktur data double linklist yang terdapat fungsi tambah depan, tambah belakang, hapus depan, hapus belakang, tampil data menggunakan bahasa pemograman c++.

Namun perbedaan dari kedua nya adalah jika guided 2 adalah linkedlist non circular dan guided 3 adalah linkedlist circular yang artinya perbedaan nya terletak pada struktur kode nya masing masing dan mengatur dan memanipulasi node mudah nya link list circular ini yang berputar simpul nya sehingga simpul terakhir terhubung dengan simpul pertama . Dalam double linked list circular, setiap simpul memiliki dua pointer atau tautan, yaitu satu untuk menunjuk ke simpul sebelumnya biasanya disebut sebagai prev atau back, dan satu lagi untuk menunjuk ke simpul selanjutnya biasanya disebut sebagai next atau forward

## Hasil Program

```
PS C:\Users\hamka\OneDrive\Documents\PRAKTIKUM STRUKTUR DATA\MODUL 8>  
DATA\MODUL 8\" ; if ($?) { g++ GUIDED3.CPP -o GUIDED3 } ; if ($?) { .\GUIDED3 }  
Kosong...!!!  
PS C:\Users\hamka\OneDrive\Documents\PRAKTIKUM STRUKTUR DATA\MODUL 8>
```

Dari instruksi yang diminta untuk menghapus nilai-nilai yang telah disimpan di sini, output yang dihasilkan adalah tampilan dari fungsi 'tampildata()'. Ketika kondisi linked list kosong, nilai baliknya akan 'true', yang menandakan bahwa output yang ditampilkan adalah "kosong...!" untuk menunjukkan bahwa data telah dihapus dan menyebabkan linked list menjadi kosong.

## UNGUIDED

### 1. Unguided 1

Memodifikasi tugas guided 1 dan 2 sehingga menjadi program yang menyajikan tampilan pilihan menu, tambah depan, tambah belakang, hapus depan, hapus belakang, tampil data linkedlist, dan keluar, yang dapat dipilih oleh user/pengguna.

#### Source Code

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

struct dlinkedlist {
    dlinkedlist* prev;
    string data;
    dlinkedlist* next;
};

dlinkedlist* head;
dlinkedlist* tail;
dlinkedlist* vertex;

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

bool dLinkKosong() {
    if (head == NULL && tail == NULL) {
        return true;
    } else {
        return false;
    }
}

void tambahDepan(string dataUser) {
    if (dLinkKosong() == true) {
        vertex = new dlinkedlist;
        vertex->data = dataUser;
        vertex->prev = NULL;
        vertex->next = NULL;
        head = vertex;
        tail = vertex;
    } else {
        vertex = new dlinkedlist;
        vertex->data = dataUser;
        vertex->prev = NULL;
```

```

        vertex->next = head;
        head->prev = vertex;
        head = vertex;
    }
}

void tambahBelakang(string dataUser) {
    if (dLinkKosong() == true) {
        vertex = new dlinkedlist;
        vertex->data = dataUser;
        vertex->prev = NULL;
        vertex->next = NULL;
        head = vertex;
        tail = vertex;
    } else {
        vertex = new dlinkedlist;
        vertex->data = dataUser;
        vertex->prev = tail;
        vertex->next = NULL;
        tail->next = vertex;
        tail = vertex;
    }
}

void hapusDepan() {
    if (dLinkKosong() == true) {
        cout << "Kosong...!!!" << endl;
    } else {
        dlinkedlist* helper;
        helper = head;
        if (head == tail) {
            head = NULL;
            tail = NULL;
            delete helper;
        } else {
            head = head->next;
            head->prev = NULL;
            helper->next = NULL;
            delete helper;
        }
    }
}

void hapusBelakang() {
    if (dLinkKosong() == true) {
        cout << "Kosong...!!!" << endl;
    } else {
        if (head == tail) {

```

```

        dlinkedList* helper;
        helper = head;
        head = NULL;
        tail = NULL;
        delete helper;
    } else {
        dlinkedList* helper;
        helper = tail;
        tail = tail->prev;
        tail->next = NULL;
        helper->prev = NULL;
        delete helper;
    }
}

void tampilData() {
    if (dLinkKosong() == true) {
        cout << "Kosong...!!!" << endl;
    } else {
        dlinkedList* helper;
        helper = head;
        while (helper != NULL) {
            cout << helper->data << " ";
            helper = helper->next;
        }
    }
}

```

Ini adalah bentuk code awal yang ada pada guided 2 yang mana hanya berikan pengoprasian kode tanpa menu untuk memilih pengorasian nya sehinga mustahil untuk bisa di tampilkan kepada user, oleh karana itu butuh nya nama nya opsi menu untuk memudah kan user memilih oprasi yang ingin ia lakukan disini penulis akan menggunakan struktur ‘switch case’ (control flow statement). Berikut adalah penerpan nya di dalam ‘int main()’

```

int main() {
    int pilih;
    string data_user;
    bool lanjut = true;
    char keluar;
    do {
        cout << endl;
        cout << "MENU" <<endl;
        cout << "1. tambah_belakang" <<endl;

```



```
cout << "2. tambah_depan" <<endl;
cout << "3. hapus_belakang" <<endl;
cout << "4. hapus_depan" <<endl;
cout << "5. tampilkan list" <<endl;
cout << "6. keluar" <<endl;
cout << "pilih : ";
cin >> pilih ;
cout <<endl;

switch (pilih) {

    case 1:
        cout << "tambah_belakang : " ;
        cin >> data_user;
        tambahBelakang(data_user);
        break;

    case 2:
        cout << "tambah_depan : " ;
        cin >> data_user;
        tambahDepan(data_user);
        break;

    case 3:
        hapusBelakang();
        cout << "data belakang telah terhapus" << endl;
        break;

    case 4:
        hapusDepan();
        cout << "data depan telah terhapus" << endl;
        break;

    case 5:
        cout << "daftar list" <<endl;
        tampilData();
        break;

    case 6:
        cout << "apakah anda ingin benar benar keluar ? (y/n) " ;
        cin >> keluar;

        if (keluar == 'y') {
            cout << "terima kasih" << endl;
            lanjut = false;
        } else {
            lanjut = true;
        }
}
```

```

        break;

        default:
            cout << "Pilihan tidak valid. Silakan pilih antara 1 hingga 6." <<
endl;
            break;

    }
} while (lanjut);

return 0;
}

```

Pertama tama yang paling utama adalah pendeklarasian varibel pilih dengan tipe data 'int' ini memungkinkan user memilih menu berdasar kan nomor nya saja, selanjut nya adalah pendeklarasian 'string data user' berguna untuk menyimpan data yang dimasukkan oleh pengguna yang kemudian akan ditambahkan ke dalam linked list yang ada di fungsi.

Variabel lanjut bertipe data bool bernilai true yang berguna untuk mematikan looping 'do while' bisa berjalan terus menerus kecuali jika nilai nya di ubah ke false sehingga tidak akan terjadi looping lagi seperti hal nya pada case exit No. 6

Variabel keluar bertipe data char berguna untuk mengkompirmasi kembali jika user ingin exit jika user input 'y' maka program akan berakhir dengan menampilkan ucapan terimakasih, dan jika menginput selain nya maka program akan kembali terlooping.

Struktur 'swieth case' berguna untuk melakukan pengambilan keputusan berdasarkan nilai dari yang di inputkan pengguna. switch-case memungkinkan user untuk memeriksa nilai dari suatu ekspresi dan menjalankan blok kode operasi yang sesuai dengan nilai tersebut.

```

MENU
1. tambah_belakang
2. tambah_depan
3. hapus_belakang
4. hapus_depan
5. tampilkan list
6. keluar
pilih : 1

tambah_belakang : rabu

```

```

MENU
1. tambah_belakang
2. tambah_depan
3. hapus_belakang
4. hapus_depan
5. tampilkan list
6. keluar
pilih : 2

tambah_depan : Kamis

```

```

MENU
1. tambah_belakang
2. tambah_depan
3. hapus_belakang
4. hapus_depan
5. tampilkan list
6. keluar
pilih : 5

daftar list
Kamis rabu

```

```
MENU
1. tambah_belakang
2. tambah_depan
3. hapus_belakang
4. hapus_depan
5. tampilkan list
6. keluar
pilih : 5
```

```
daftar list
Tidak ada data
```

```
MENU
1. tambah_belakang
2. tambah_depan
3. hapus_belakang
4. hapus_depan
5. tampilkan list
6. keluar
pilih : 4
```

```
data depan telah terhapus
```

```
MENU
1. tambah_belakang
2. tambah_depan
3. hapus_belakang
4. hapus_depan
5. tampilkan list
6. keluar
pilih : 6
```

```
apakah anda ingin benar benar keluar ? (y/n) n
```

```
MENU
1. tambah_belakang
2. tambah_depan
3. hapus_belakang
4. hapus_depan
5. tampilkan list
6. keluar
pilih : █
```

```
MENU
1. tambah_belakang
2. tambah_depan
3. hapus_belakang
4. hapus_depan
5. tampilkan list
6. keluar
pilih : 3
```

```
data belakang telah terhapus
```

## 2. Unguided 1 dari modifikasi double linked list circulation

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

struct dlinkedlist {
    dlinkedlist* prev;
    string data;
    dlinkedlist* next;
};

dlinkedlist* head;
dlinkedlist* tail;
dlinkedlist* vertex;

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

bool dLinkKosong() {
    if (head == NULL && tail == NULL) {
        return true;
    } else {
        return false;
    }
}

void tambahDepan(string dataUser) {
    if (dLinkKosong() == true) {
        vertex = new dlinkedlist;
        vertex->data = dataUser;
        head = vertex;
        tail = vertex;
        head->prev = tail;
        tail->next = head;
    } else {
        vertex = new dlinkedlist;
        vertex->data = dataUser;
        vertex->next = head;
        head->prev = vertex;
        head = vertex;
        head->prev = tail;
        tail->next = head;
    }
}

void tambahBelakang(string dataUser) {
```

```

    if (dLinkKosong() == true) {
        vertex = new dlinkedlist;
        vertex->data = dataUser;
        head = vertex;
        tail = vertex;
        head->prev = tail;
        tail->next = head;
    } else {
        vertex = new dlinkedlist;
        vertex->data = dataUser;
        vertex->prev = tail;
        tail->next = vertex;
        tail = vertex;
        tail->next = head;
        head->prev = tail;
    }
}

void hapusDepan() {
    if (dLinkKosong() == true) {
        cout << "Kosong...!!!" << endl;
    } else {
        dlinkedlist* helper;
        helper = head;
        if (head == tail) {
            tail->next = NULL;
            head->prev = NULL;
            head = NULL;
            tail = NULL;
            delete helper;
        } else {
            head = head->next;
            head->prev = tail;
            tail->next = head;
            helper->next = NULL;
            delete helper;
        }
    }
}

void hapusBelakang() {
    if (dLinkKosong() == true) {
        cout << "Kosong...!!!" << endl;
    } else {
        if (head == tail) {
            dlinkedlist* helper;
            helper = head;
            head->prev = NULL;

```

```

        tail->next = NULL;
        head = NULL;
        tail = NULL;
        delete helper;
    } else {
        dlinkedList* helper;
        helper = tail;
        tail = tail->prev;
        tail->next = head;
        head->prev = tail;
        helper->prev = NULL;
        helper->next = NULL;
        delete helper;
    }
}

void tampilData() {
    if (dLinkKosong() == true) {
        cout << "Kosong...!!!" << endl;
    } else {
        dlinkedList* helper;
        helper = head;
        do {
            cout << helper->data << " ";
            helper = helper->next;
        } while (helper != head);
    }
}

```

Ini adalah bentuk awal dari kode dalam GUIDED 3 yang hanya memberikan operasi kode tanpa adanya menu untuk memilih operasinya, sehingga mustahil bagi pengguna untuk dapat menjalankan operasi yang diinginkan. Oleh karena itu, diperlukan opsi menu untuk memudahkan pengguna memilih operasi yang ingin mereka lakukan. Di sini, penulis akan menggunakan struktur 'IF ELSE' (pernyataan alur kontrol). Berikut adalah penerapannya dalam fungsi di int main()

### Source Code

```

int main() {
    int pilih;
    string data_user;
    bool lanjut = true;
    char keluar;
    do {
        cout << endl;
        cout << "MENU" << endl;
        cout << "1. tambah_belakang" << endl;
    }
}

```

```

cout << "2. tambah_depan" <<endl;
cout << "3. hapus_belakang" <<endl;
cout << "4. hapus_depan" <<endl;
cout << "5. tampilkan list" <<endl;
cout << "6. keluar" <<endl;
cout << "pilih : ";
cin >> pilih ;
cout <<endl;

if (pilih == 1) {
    cout << "tambahkan belakang :";
    cin >>data_user;
    tambahBelakang(data_user);
}
if (pilih == 2) {
    cout << "tambahkan depan :";
    cin >>data_user;
    tambahDepan(data_user);
}
if (pilih == 3) {
    hapusBelakang();
    cout << "data belakang telah terhapus" << endl;
}
if (pilih == 4) {
    hapusDepan();
    cout << "data depan telah terhapus" << endl;
}
if (pilih == 5) {
    tampilData();
}
if (pilih == 6) {
    cout << " apakah anda ingin benar benar keluar ? (y/n)";
    cin >> keluar;

    if (keluar == 'y') {
        lanjut = false;
    } else {
        lanjut = true;
    }

} if (pilih > 6) {
    cout << " pilihan tidak valid silahkan memilih 1 sampai 6" <<
endl;
}

} while (lanjut);

```

```
return 0;
}
```

Berbeda dengan UNGUIDED 1 MODIFIKASI GUIDED 1 yang menggunakan struktur yang menggunakan Struktur 'switch case' untuk melakukan pengambilan keputusan berdasarkan nilai dari yang di inputkan penggunanya.

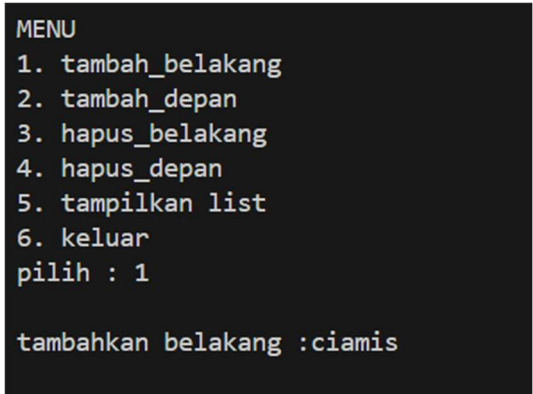
Disini di UNGUIDED 1 MODIFIKASI GUIDED 2 penulis menggunakan struktur yang serupa yang berguna untuk melakukan pengambilan keputusan berdasarkan nilai dari yang di inputkan penggunanya yaitu dengan menggunakan struktur IF Else dalam program nya sehingga menjadikan nya berbeda dengan unguided 1 di atas. Walaupun secara penggunaan if else ini tidak lebih efektif di banding dengan 'switch case'

Selebih nya sama, penggunaan variabel 'lanjut' dengan tipe data 'bool' dan nilai awal 'true' mengontrol perulangan dalam struktur 'do-while'. Nilai 'false' menandakan akhir dari perulangan.

Variabel 'keluar' dengan tipe data 'char' digunakan untuk konfirmasi apakah pengguna ingin keluar. Jika pengguna memasukkan karakter 'y', program berakhir dengan ucapan terima kasih; karakter lainnya akan membuat program tetap berjalan.

Struktur 'switch-case' memungkinkan pengambilan keputusan berdasarkan input pengguna. Setiap kasus dalam 'switch-case' mewakili operasi yang akan dijalankan pada linked list ganda sesuai dengan pilihan pengguna.

Dengan demikian, 'lanjut', 'keluar', dan 'switch-case' secara bersama-sama mengatur alur program dan interaksi dengan pengguna.

NO	Hasil Program	screenshot
1.	Hasil penambahan data belakang	 <pre> MENU 1. tambah_belakang 2. tambah_depan 3. hapus_belakang 4. hapus_depan 5. tampilkan list 6. keluar pilih : 1  tambahkan belakang :ciamis </pre>



2.	Hasil penambahan data depan	<pre> MENU 1. tambah_belakang 2. tambah_depan 3. hapus_belakang 4. hapus_depan 5. tampilkan list 6. keluar pilih : 2  tambahkan depan :aceh </pre>
3.	Hasil menampilkan data setelah di tambahkan	<pre> MENU 1. tambah_belakang 2. tambah_depan 3. hapus_belakang 4. hapus_depan 5. tampilkan list 6. keluar pilih : 5  bandung aceh ciamis </pre>
4.	Hasil menghapus data belakang	<pre> bandung aceh ciamis MENU 1. tambah_belakang 2. tambah_depan 3. hapus_belakang 4. hapus_depan 5. tampilkan list 6. keluar pilih : 3  data belakang telah terhapus </pre>
5.	Hasil menghapus data depan	<pre> MENU 1. tambah_belakang 2. tambah_depan 3. hapus_belakang 4. hapus_depan 5. tampilkan list 6. keluar pilih : 4  data depan telah terhapus </pre>

6.	Hasil setelah data depan dan belakang di hapus hanya menyisakan 'aceh'	<pre> MENU 1. tambah_belakang 2. tambah_depan 3. hapus_belakang 4. hapus_depan 5. tampilkan list 6. keluar pilih : 5  aceh </pre>
7.	Hasil ketika user ingin keluar	<pre> MENU 1. tambah_belakang 2. tambah_depan 3. hapus_belakang 4. hapus_depan 5. tampilkan list 6. keluar pilih : 6  apakah anda ingin benar benar keluar ? (y/n)y </pre>

### 3. UNGUIDED 2

Disini akan membuat fungsi untuk melakukan operasi sisip data pada double linkedlist circular. Data disisipkan diantara data tertentu atau sebelum data tertentu atau setelah data tertentu.

#### Source Code

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

struct Node {
    int data;
    Node* prev;
    Node* next;
};

Node* buatnode(int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

void buat_linkedlist(Node** head, int value) {
    *head = buatnode(value);
    (*head)->next = (*head);
    (*head)->prev = (*head);
}

void tambkan_sisipdepan(Node** head, Node* newNode, Node* target) {
    newNode->next = target;
    newNode->prev = target->prev;
    if (target->prev != NULL) {
        target->prev->next = newNode;
    } else {
        newNode->next = *head;
    }
    target->prev = newNode;
    if (newNode->next == *head) {
        (*head)->prev = newNode;
    }
}
```

```

void tambakan_sisipbelakang(Node** head, Node* newNode, Node* target) {
    newNode->prev = target;
    newNode->next = target->next;
    if (target->next != NULL) {
        target->next->prev = newNode;
    } else {
        newNode->next = *head;
    }
    target->next = newNode;
    if (newNode->prev == *head) {
        (*head)->next = newNode;
    }
}

```

```

void tambahkan_tail(Node** head, Node* newNode) {
    if (*head == NULL) {
        *head = newNode;
        newNode->prev = newNode;
        newNode->next = newNode;
    } else {
        Node* last = (*head)->prev;
        newNode->prev = last;
        newNode->next = *head;
        last->next = newNode;
        (*head)->prev = newNode;
    }
}

```

```

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

```

```

int main() {
    Node* head = NULL;
    buat_linkedlist(&head, 1);
    tambakan_sisipdepan(&head, buatnode(0), head);
    tambakan_sisipbelakang(&head, buatnode(2), head);
    tambahkan_tail(&head, buatnode(3));
    tampilkan_data(head);
    return 0;
}

```

```
}
```

Dalam contoh di atas, fungsi `tambahkan_sisipdepan`, `tambahkan_sisipdepan`, dan `tampil_data` digunakan untuk sisip data pada linked list circular. Fungsi `buat_linked list` digunakan untuk membuat linked list circular dengan node pertama. Fungsi `tampil_data` digunakan untuk mencetak isi linked list. Dalam contoh, data 0 disisip sebelum node pertama, data 2 disisip setelah node pertama, dan data 3 disisip di akhir linked list. Hasilnya adalah linked list circular yang berisi data 0, 1, 2, dan 3.

#### Hasil Program

```
DATA\MODUL 8\" ; if ($?) { g++ unguided2.cpp -o unguided2 } ; if ($?) { .\unguided2 }  
1 2 0 3  
PS C:\Users\hamka\OneDrive\Documents\PRAKTIKUM STRUKTUR DATA\MODUL 8>
```

## **KESIMPULAN**

Double linked list (linked list ganda) dan single linked list (linked list tunggal) adalah dua struktur data yang digunakan untuk menyimpan dan mengelola kumpulan data secara terurut. Perbedaan utama antara keduanya terletak pada kemampuan untuk menavigasi maju dan mundur dalam elemen-elemen linked list. Double linked list memiliki koneksi dua arah antara setiap simpul, memungkinkan untuk traversing maju dan mundur, sementara single linked list hanya memiliki koneksi satu arah, membatasi traversing hanya ke arah maju. Meskipun double linked list memiliki fleksibilitas yang lebih tinggi dalam hal navigasi, ini memerlukan alokasi memori tambahan untuk setiap simpul karena setiap simpul harus menyimpan dua pointer, yang dapat meningkatkan penggunaan memori.

## REFRENSI

1. MODUL 8 STRUKTUR DATA
2. <https://inzaghiposuma.blogspot.com/2021/12/pengertian-dan-contoh-dari-linked-list.html>
3. <https://www.slideshare.net/slideshow/linked-list-40738331/40738331>
4. [https://www.academia.edu/30939554/Buku\\_Struktur\\_data\\_Copy](https://www.academia.edu/30939554/Buku_Struktur_data_Copy)