

# **LAPORAN PRAKTIKUM**

## **MODUL IV CIRCULAR & NON CIRCULAR**



**Disusun oleh:  
Aryo Tegar Sukarno  
NIM: 2311102018**

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

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
PURWOKERTO  
2023**

## **BAB I**

### **TUJUAN PRAKTIKUM**

1. Mahasiswa memahami perbedaan konsep Linked List Circular dan NonCircular.
2. Mahasiswa mampu menerapkan Linked List Circular dan Non Circular kedalam pemrograman

## **BAB II**

### **DASAR TEORI**

#### **1. Linked List**

Linked list circular adalah struktur data yang digunakan yang terdiri dari kumpulan simpul (node) yang terhubung dengan pointer yang saling terkait membentuk sirkuit, memiliki simpul akhir yang kembali ke simpul pertama sehingga membuat suatu siklus(cycle) dan dapat digunakan dalam situasi di mana data perlu diolah secara berkelanjutan.

#### **2. Linked List Non Circular**

Linked list non circular adalah struktur data yang terdiri dari node dengan pointer yang menunjuk ke node berikutnya, kecuali pada node terakhir yang menunjuk ke nilai null sebagai penanda akhir dari linked list. Digunakan pada situasi di mana data perlu diolah secara berurutan dan tidak membentuk lingkaran atau sirkuit. Lebih efisien dalam penggunaannya karena mudah menambah atau menghapus node pada awal atau akhir linked list dan tidak perlu memperhatikan hubungan antar node yang saling terkait seperti pada linked list circular.

## BAB III

### GUIDED

#### 1. Guided 1

##### Source code

```
#include <iostream>

using namespace std;

// PROGRAM SINGLE LINKED LIST NON-CIRCULAR

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

Node *head; // Deklarasi head
Node *tail; // Deklarasi tail

// Inisialisasi Node
```

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

// Pengecekan apakah linked list kosong
bool isEmpty()
{
    if (head == NULL)
    {
        return true;
    }
    else
    {
        return false;
    }
}

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

```
        head = baru;
    }
}

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

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

```
        return jumlah;
    }

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

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

            baru->next = bantu->next;
            bantu->next = baru;
        }
    }
}
```

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

// Hapus belakang
void hapusBelakang()
{
    Node *hapus;
    Node *bantu;
    if (isEmpty() == false)
    {
        if (head != tail)
        {
            hapus = tail;
            bantu = head;
```



```

        while (bantu->next != tail)
        {
            bantu = bantu->next;
        }
        tail = bantu;
        tail->next = NULL;
        delete hapus;
    }
    else
    {
        head = tail = NULL;
    }
}
else
{
    cout << "Linked list masih kosong" << endl;
}
}
// Hapus tengah
void hapusTengah(int posisi)
{
    Node *hapus, *bantu, *sebelum;
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi di luar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        int nomor = 1;
        bantu = head;
    }
}

```

```

        while (nomor <= posisi)
        {
            if (nomor == posisi - 1)
            {
                sebelum = bantu;
            }
            if (nomor == posisi)
            {
                hapus = bantu;
            }
            bantu = bantu->next;
            nomor++;
        }
        sebelum->next = bantu;
        delete hapus;
    }
}

// ubah depan
void ubahDepan(int data)
{
    if (isEmpty() == 0)
    {
        head->data = data;
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

// ubah tengah
void ubahTengah(int data, int posisi)
{

```

```

Node *bantu;
if (isEmpty() == 0)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi di luar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        int nomor = 1;
        bantu = head;
        while (nomor < posisi)
        {
            bantu = bantu->next;
            nomor++;
        }
        bantu->data = data;
    }
}
else
{
    cout << "Linked list masih kosong" << endl;
}

// ubah belakang
void ubahBelakang(int data)
{
    if (isEmpty() == 0)
    {

```

```

        tail->data = data;
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

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

// Tampilkan list
void tampilList()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << " ";
            bantu = bantu->next;
        }
    }
}

```

```

        }
        cout << endl;
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

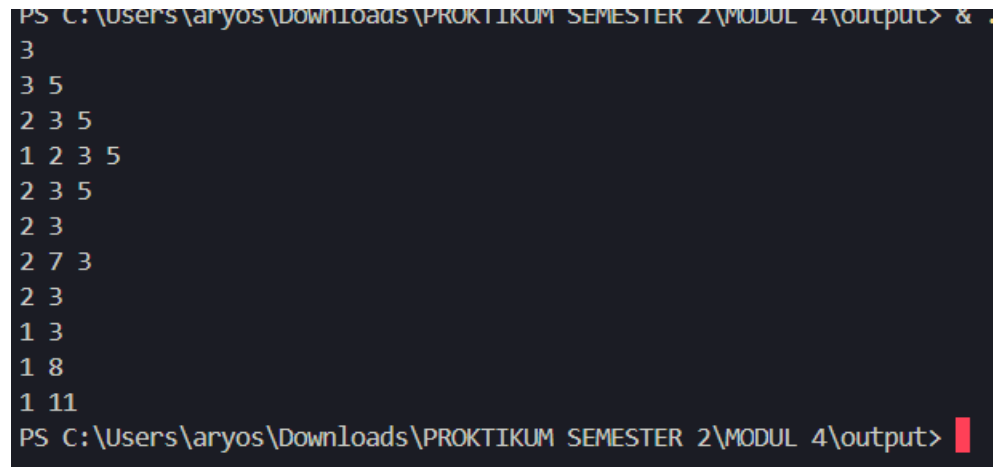
int main()
{
    init();
    insertDepan(3);
    tampilList();
    insertBelakang(5);
    tampilList();
    insertDepan(2);
    tampilList();
    insertDepan(1);
    tampilList();
    hapusDepan();
    tampilList();
    hapusBelakang();
    tampilList();
    insertTengah(7, 2);
    tampilList();
    hapusTengah(2);
    tampilList();
    ubahDepan(1);
    tampilList();
    ubahBelakang(8);
    tampilList();
    ubahTengah(11, 2);
    tampilList();
}

```

```
return 0;
```

```
}
```

### Screenshoot program



```
PS C:\Users\aryos\Downloads\PROKTIKUM SEMESTER 2\MODUL 4\output> & .
3
3 5
2 3 5
1 2 3 5
2 3 5
2 3
2 7 3
2 3
1 3
1 8
1 11
PS C:\Users\aryos\Downloads\PROKTIKUM SEMESTER 2\MODUL 4\output> █
```

### Deskripsi program

Program tersebut menggunakan linked list non-circular dan memiliki fitur untuk menambah, menghapus, mengubah. Program tersebut juga menggunakan beberapa fungsi if, else if, dan while do. Selain itu, program tersebut juga menggunakan Struct

## 2. Guided 2

### Source code

```
#include <iostream>
using namespace std;
// Deklarasi Struct Node
struct Node
{
    string data;
    Node *next;
};
Node *head, *tail, *baru, *bantu, *hapus;
// Inisialisasi node head & tail
void init()
{
    head = NULL;
    tail = head;
```

```
}
// Pengecekan isi list
int isEmpty()
{
    if (head == NULL)
    {
        return 1; // true
    }
    else
    {
        return 0; // false
    }
}
// Buat Node Baru
void buatNode(string data)
{
    baru = new Node;
    baru->data = data;
    baru->next = NULL;
}
// Hitung List
int hitungList()
{
    bantu = head;
    int jumlah = 0;
    while (bantu != NULL)
    {
        jumlah++;
        bantu = bantu->next;
    }
    return jumlah;
}
// Tambah Depan
void insertDepan(string data)
{
    // Buat Node baru
    buatNode(data);
    if (isEmpty() == 1)
    {
        head = baru;
        tail = head;
        baru->next = head;
    }
    else
    {
        while (tail->next != head)
        {
            tail = tail->next;
        }
        baru->next = head;
        head = baru;
        tail->next = head;
    }
}
```

```

}
}
// Tambah Belakang
void insertBelakang(string data)
{
    // Buat Node baru
    buatNode(data);
    if (isEmpty() == 1)
    {
        head = baru;
        tail = head;
        baru->next = head;
    }
    else
    {
        while (tail->next != head)
        {
            tail = tail->next;
        }
        tail->next = baru;
        baru->next = head;
    }
}
// Tambah Tengah
void insertTengah(string data, int posisi)
{
    if (isEmpty() == 1)
    {
        head = baru;
        tail = head;
        baru->next = head;
    }
    else
    {
        baru->data = data;
        // transversing
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}
// Hapus Depan
void hapusDepan()
{
    if (isEmpty() == 0)
    {

```



```

hapus = head;
tail = head;
if (hapus->next == head)
{
head = NULL;
tail = NULL;
delete hapus;
}
else
{
while (tail->next != hapus)
{
tail = tail->next;
}
head = head->next;
tail->next = head;
hapus->next = NULL;
delete hapus;
}
}
else
{
cout << "List masih kosong!" << endl;
}
}
// Hapus Belakang
void hapusBelakang()
{
if (isEmpty() == 0)
{
hapus = head;
tail = head;
if (hapus->next == head)
{
head = NULL;
tail = NULL;
delete hapus;
}
else
{
while (hapus->next != head)
{
hapus = hapus->next;
}
while (tail->next != hapus)
{
tail = tail->next;
}
tail->next = head;
hapus->next = NULL;
delete hapus;
}
}
}

```

```

    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
// Hapus Tengah
void hapusTengah(int posisi)
{
    if (isEmpty() == 0)
    {
        // transversing
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
// Hapus List
void clearList()
{
    if (head != NULL)
    {
        hapus = head->next;
        while (hapus != head)
        {
            bantu = hapus->next;
            delete hapus;
            hapus = bantu;
        }
        delete head;
        head = NULL;
    }
    cout << "List berhasil terhapus!" << endl;
}
// Tampilkan List
void tampil()
{
    if (isEmpty() == 0)
    {
        tail = head;
        do

```

```

{
    cout << tail->data << ends;
    tail = tail->next;
} while (tail != head);
cout << endl;
}
else
{
    cout << "List masih kosong!" << endl;
}
}
int main()
{
    init();
    insertDepan("Ayam");
    tampil();
    insertDepan("Bebek");
    tampil();
    insertBelakang("Cicak");
    tampil();
    insertBelakang("Domba");
    tampil();
    hapusBelakang();
    tampil();
    hapusDepan();
    tampil();
    insertTengah("Sapi", 2);
    tampil();
    hapusTengah(2);
    tampil();
    return 0;
}

```

### Screenshoot program

```

PS C:\Users\aryos> cd 'c:\Users\aryos\Downloads\PROKTIKUM SEMESTER 2\MODUL 4\output'
PS C:\Users\aryos\Downloads\PROKTIKUM SEMESTER 2\MODUL 4\output> & .\'guided2.exe'
Ayam
BebekAyam
BebekAyamCicak
BebekAyamCicakDomba
BebekAyamCicak
AyamCicak
AyamSapiCicak

```

### Deskripsi program

Program tersebut merupakan program Linked List Circular, program tersebut dapat menambah dan menghaous pada depan dan belakang. Struktur datatersebut terdiri dari Struct Node yang berisi data dan pointer ke Node selanjutnya.

## LATIHAN KELAS - UNGUIDED

### 1. Unguided 1

#### Source code

```
#include <iostream>
#include <iomanip>
using namespace std;
struct mahasiswa
{
    string nama;
    string nim;
};
struct node
{
    mahasiswa identitas;
    node *prev;
    node *next;
};
node *head, *tail;
void init()
{
    head = NULL;
```

```
tail = NULL;
}
bool isEmpty()
{
return head == NULL;
}
mahasiswa mintaData()
{
system("cls");
mahasiswa identitas;
cout << "\nMasukkan Nama\t: ";
cin.ignore();
getline(cin, identitas.nama);
cout << "Masukkan NIM\t: ";
cin >> identitas.nim;
return identitas;
}
void insertDepan(mahasiswa identitas)
{
node *baru = new node;
baru->identitas = identitas;
baru->next = head;
baru->prev = NULL;
if (isEmpty())
{
tail = baru;
}
else
{
head->prev = baru;
}
head = baru;
}
void insertBelakang(mahasiswa identitas)
```

```
{
node *baru = new node;
baru->identitas = identitas;
baru->next = NULL;
baru->prev = tail;
if (isEmpty())
{
head = baru;
}
else
{
tail->next = baru;
}
tail = baru;
}
void insertTengah(mahasiswa identitas, int posisi)
{
if (posisi <= 1)
{
insertDepan(identitas);
return;
}
node *baru = new node;
baru->identitas = identitas;
node *bantu = head;
for (int i = 1; i < posisi - 1 && bantu != NULL; i++)
{
bantu = bantu->next;
}
if (bantu == NULL)
{
cout << "Posisi diluar jangkauan" << endl;
return;
}
```

```
baru->next = bantu->next;
baru->prev = bantu;
if (bantu->next != NULL)
{
    bantu->next->prev = baru;
}
bantu->next = baru;
}

void ubahNode(node *target, mahasiswa data)
{
    target->identitas = data;
}

void ubahDepan(mahasiswa data)
{
    if (isEmpty())
    {
        cout << "List kosong" << endl;
        return;
    }
    ubahNode(head, data);
}

void ubahBelakang(mahasiswa data)
{
    if (isEmpty())
    {
        cout << "List kosong" << endl;
        return;
    }
    ubahNode(tail, data);
}

void ubahTengah(mahasiswa data, string nim)
{
    node *bantu = head;
    while (bantu != NULL)
```

```
{
if (bantu->identitas.nim == nim)
{
ubahNode(bantu, data);
return;
}
bantu = bantu->next;
}
cout << "Data dengan NIM " << nim << " tidak ditemukan" << endl;
}
void tampil()
{
system("cls");
node *bantu = head;
cout << "Nama "
<< " Nim\n";
while (bantu != NULL)
{
cout << bantu->identitas.nama << " " << bantu->identitas.nim <<
endl;
bantu = bantu->next;
}
}
void hapusDepan()
{
if (isEmpty())
{
cout << "List kosong" << endl;
return;
}
node *hapus = head;
if (head == tail)
{
head = tail = NULL;
```



```
}  
else  
{  
head = head->next;  
head->prev = NULL;  
}  
delete hapus;  
}  
void hapusBelakang()  
{  
if (isEmpty())  
{  
cout << "List kosong" << endl;  
return;  
}  
node *hapus = tail;  
if (head == tail)  
{  
head = tail = NULL;  
}  
else  
{  
tail = tail->prev;  
tail->next = NULL;  
}  
delete hapus;  
}  
void hapusTengah(string nim)  
{  
if (isEmpty())  
{  
cout << "List kosong" << endl;  
return;  
}
```

```
node *bantu = head;
while (bantu != NULL)
{
    if (bantu->identitas.nim == nim)
    {
        if (bantu == head)
        {
            hapusDepan();
        }
        else if (bantu == tail)
        {
            hapusBelakang();
        }
        else
        {
            bantu->prev->next = bantu->next;
            bantu->next->prev = bantu->prev;
            delete bantu;
        }
        return;
    }
    bantu = bantu->next;
}
cout << "Data dengan NIM " << nim << " tidak ditemukan" << endl;
}

void hapusList()
{
    while (!isEmpty())
    {
        hapusDepan();
    }
    cout << "Semua data berhasil dihapus" << endl;
}

int main()
```

```
{
init();
mahasiswa identitas;
int operasi, posisi;
do
{
cout << "PROGRAM DOUBLE LINKED LIST NON-CIRCULAR\n\n";
cout << "1.Tambah Depan" << endl;
cout << "2.Tambah Belakang" << endl;
cout << "3.Tambah Tengah" << endl;
cout << "4.Ubah Depan" << endl;
cout << "5.Ubah Belakang" << endl;
cout << "6.Ubah Tengah" << endl;
cout << "7.Hapus Depan" << endl;
cout << "8.Hapus Belakang" << endl;
cout << "9.Hapus Tengah" << endl;
cout << "10.Hapus List" << endl;
cout << "11.Tampilkan" << endl;
cout << "0.Exit" << endl;
cout << "\nPilih Operasi : ";
cin >> operasi;
switch (operasi)
{
case 1:
cout << "Tambah Depan\n";
insertDepan(mintaData());
cout << endl;
break;
case 2:
cout << "Tambah Belakang\n";
insertBelakang(mintaData());
cout << endl;
break;
case 3:
```

```
cout << "Tambah Tengah\n";
cout << "Nama : ";
cin >> identitas.nama;
cout << "NIM : ";
cin >> identitas.nim;
cout << "Posisi : ";
cin >> posisi;
insertTengah(identitas, posisi);
cout << endl;
break;
case 4:
cout << "Ubah Depan\n";
ubahDepan(mintaData());
cout << endl;
break;
case 5:
cout << "Ubah Belakang\n";
ubahBelakang(mintaData());
cout << endl;
break;
case 6:
cout << "Ubah Tengah\n";
cout << "NIM : ";
cin >> identitas.nim;
ubahTengah(mintaData(), identitas.nim);
cout << endl;
break;
case 7:
cout << "Hapus Depan\n";
hapusDepan();
cout << endl;
break;
case 8:
cout << "Hapus Belakang\n";
```

```
hapusBelakang();
cout << endl;
break;
case 9:
cout << "Hapus Tengah\n";
cout << "NIM : ";
cin >> identitas.nim;
hapusTengah(identitas.nim);
cout << endl;
break;
    case 10:
cout << "Hapus List\n";
hapusList();
cout << endl;
break;
case 11:
cout << "Tampilkan\n";
tampil();
cout << endl;
break;
case 0:
cout << "Exit Program\n";
break;
default:
cout << "Salah input operasi\n";
cout << endl;
break;
}
} while (operasi != 0);
return 0;
}
```

## Screenshoot program

### a. Masukkan data sesuai urutan

```
Nama  Nim
ronsy 2311624
gerry 2312853863
via 231119883
jovi 231110728
Lexxy 23111078
```

### b. Hapus data Akhir

```
8.Hapus Belakang
Nama  Nim
ronsy 2311624
gerry 2312853863
via 231119883
jovi 231110728
```

### c. Tambahkan data

```
11.Tampilkan
Nama  Nim
baim 23110272
Lexxy2311102018 2311102018
ronsy 2311624
gerry 2312853863
via 231119883
jovi 231110728
bayu 23211726
```

### D. Tambahkan Data Berikut Diakhir

```
Masukkan Nama : cipung
Masukkan NIM : 62515112
```

```
Nama  Nim
baim 23110272
Lexxy2311102018 2311102018
ronsy 2311624
gerry 2312853863
via 231119883
jovi 231110728
cipung 62515112
```

#### **E Ubah data baim jadi joni**

```
Nama  Nim
Joni 2311102018
Lexxy2311102018 2311102018
ronsy 2311624
gerry 2312853863
via 231119883
jovi 231110728
cipung 62515112
```

#### **F. Ubah data terakhir cipung menjadi slamet**

```
Nama  Nim
Joni 2311102018
Lexxy2311102018 2311102018
ronsy 2311624
gerry 2312853863
via 231119883
jovi 231110728
slamet 232726151
```

**g. Hapus Depan**

```
Nama  Nim
Lexxy2311102018 2311102018
ronsy 2311624
gerry 2312853863
via 231119883
jovi 231110728
slamet 232726151
```

**h. Ubah data awal menjadi riva**

```
Nama  Nim
riva 27286343
ronsy 2311624
gerry 2312853863
via 231119883
jovi 231110728
slamet 232726151
```

**i. Hapus data akhir**

```
Pilih Operasi : 8
Hapus Belakang
```

**j. Tampilkan**

```
Nama  Nim
riva 27286343
ronsy 2311624
gerry 2312853863
via 231119883
jovi 231110728
```



**Deskripsi program**

Program ini merupakan double linked list non-circular menggunakan bahasa pemrograman C++. Program ini memungkinkan pengguna untuk melakukan operasi dasar seperti penambahan, penghapusan, dan pengubahan data mahasiswa. Operasi yang disediakan antara lain adalah penambahan data di depan, di belakang, atau di tengah list, pengubahan data di depan, di belakang, atau di tengah list berdasarkan NIM, penghapusan data di depan, di belakang, atau di tengah list berdasarkan NIM, dan juga penghapusan seluruh data.

## **BAB IV**

### **KESIMPULAN**

Dari praktikum Linked List Non-Circular, dapat disimpulkan bahwa struktur data ini memberikan keleluasaan dalam penyimpanan serta pengelolaan data, terutama ketika data yang dimasukkan memiliki panjang atau jenis yang bervariasi. Dengan menggunakan linked list, kita dapat dengan mudah menambah, mengubah, atau menghapus data tanpa harus terikat pada urutan tertentu, sehingga proses tersebut menjadi lebih efisien.