

**LAPORAN PRAKTIKUM STRUKTUR DATA DAN
ALGORITMA
MODUL 4
LINKED LIST CIRCULAR DAN NON
CIRCULAR**



DISUSUN OLEH:

PRIESTY AMEILIANA MAULIDAH

2311102175

S1 IF-11-E

DOSEN:

Muhammad Afrizal Amrustian, S. Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO PURWOKERTO

2024

A.DASAR TEORI

1. Linked List Non Circular

Linked list non circular merupakan linked list dengan node pertama dan node terakhir yang tidak saling terhubung.

2. Linked List Circular

Linked list circular merupakan linked list yang tidak memiliki akhir karena node terakhir tidak bernilai 'NULL', tetapi terhubung dengan node pertama . Saat menggunakan linked list circular kita membutuhkan dummy node atau node pengecoh yang biasanya dinamakan dengan node current supaya program dapat berhenti menghitung data ketika node current mencapai node pertama .

B.Guided

Guided 1

```
// priesty ameiliana maulidah
// 2311102175

#include <iostream>
using namespace std;

/// PROGRAM SINGLE LINKED LIST NON-CIRCULAR
// Deklarasi Struct Node
struct Node
{
    int data;
    Node *next;
};
Node *head;
Node *tail;

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

// Pengecekan
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;  
        tail->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;
    tail->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 diluar 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;

        }
        baru
        nomor++;
    }
}
```

```

}

baru
nomor++;

->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;

head = tail = NULL;

else
{
cout << "List 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;

            head = tail = NULL;

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

// Hapus Tengah
void hapusTengah(int posisi)
```



```
{
Node *bantu, *hapus, *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 << "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)
{

}
else
{
cout << "Posisi bukan posisi tengah" << endl;

bantu = head;
int nomor = 1;
while (nomor < posisi)
{
bantu = bantu->next;
nomor++;
}
bantu->data = data;
}
}
else
{
cout << "List masih kosong!" << endl;
}
}
```

```
// Ubah Belakang
void ubahBelakang(int data)
{
    if (isEmpty() == 0)
    {
        tail->data = data;
    }
    else
    {
        cout << "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 tampil()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << ends;
            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:

- **Deklarasi Struct Node:** Mendefinisikan struktur data Node yang terdiri dari dua field: data (menyimpan nilai) dan next (pointer ke node berikutnya).
- **Variabel Global:** Mendeklarasikan variabel global head dan tail untuk menunjuk ke node pertama dan terakhir dalam list.
- **Fungsi-fungsi:**
 - `init()`: Mengatur head dan tail menjadi NULL untuk menunjukkan list kosong.
 - `isEmpty()`: Memeriksa apakah list kosong atau tidak.
 - `insertDepan(int nilai)`: Menambahkan node baru di awal list.
 - `insertBelakang(int nilai)`: Menambahkan node baru di akhir list.
 - `hitungList()`: Menghitung jumlah node dalam list.

- insertTengah(int data, int posisi):
Menambahkan node baru di posisi tertentu di tengah list.
- hapusDepan(): Menghapus node pertama dari list.
- hapusBelakang(): Menghapus node terakhir dari list.
- hapusTengah(int posisi): Menghapus node di posisi tertentu di tengah list.
- ubahDepan(int data): Mengubah data pada node pertama.
- ubahBelakang(int data): Mengubah data pada node terakhir.
- ubahTengah(int data, int posisi): Mengubah data pada node di posisi tertentu di tengah list.
- tampil(): Menampilkan data semua node dalam list.
- clearList(): Menghapus semua node dalam list.

Guided 2

```
// priesty ameiliana maulidah
// 2311102175

#include <iostream>

using namespace std;

/// PROGRAM SINGLE LINKED LIST CIRCULAR

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

Node *head, *tail, *baru, *bantu, *hapus;

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

// Pengecekan
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 (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 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);

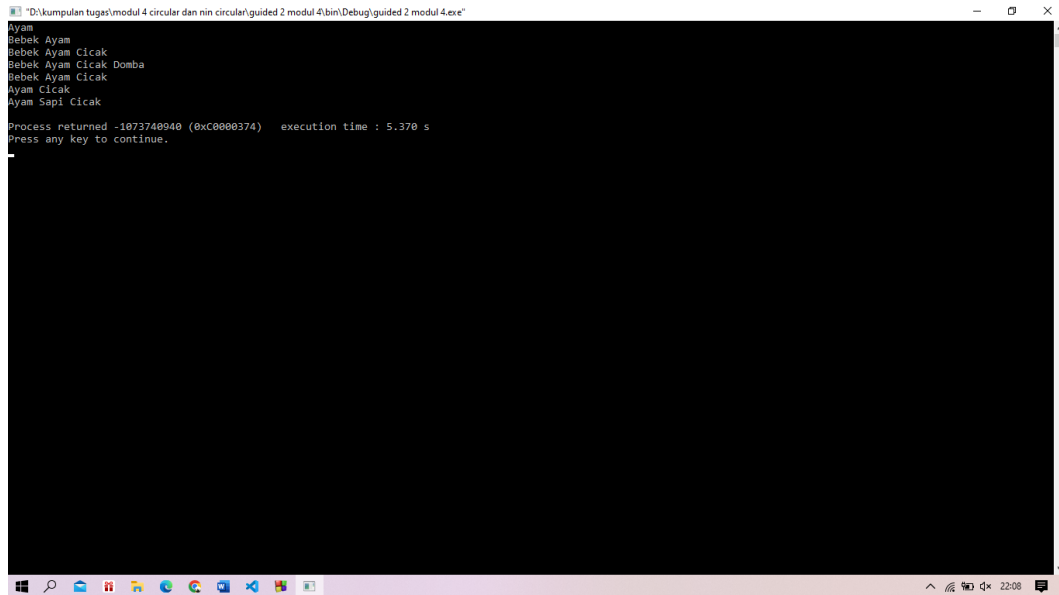
    tampil();

    hapusTengah(2);

    tampil();

    return 0;
}
```

Screenshots output



```
"D:\kumpulan tugas\modul 4 circular dan linier\circular\guided 2 modul 4\bin\Debug\guided 2 modul 4.exe"
Ayam
Bebek Ayam
Bebek Ayam Cicak
Bebek Ayam Cicak Domba
Bebek Ayam Cicak
Ayam Cicak
Ayam Sapi Cicak

Process returned -1073740940 (0xC0000374)   execution time : 5.370 s
Press any key to continue.
```

Deskripsi :

Deklarasi Struct Node: Mendefinisikan struktur data Node yang terdiri dari dua field: data (menyimpan nilai) dan next (pointer ke node berikutnya).

Variabel Global: Mendeklarasikan variabel global head dan tail untuk menunjuk ke node pertama dan terakhir dalam list.

Fungsi-fungsi:

init(): Mengatur head dan tail menjadi NULL untuk menunjukkan list kosong.

isEmpty(): Memeriksa apakah list kosong atau tidak.

buatNode(string data): Membuat node baru dengan data yang diberikan.

hitungList(): Menghitung jumlah node dalam list.

insertDepan(string data): Menambahkan node baru di awal list.

insertBelakang(string data): Menambahkan node baru di akhir list.

insertTengah(string data, int posisi): Menambahkan node baru di posisi tertentu di tengah list.

hapusDepan(): Menghapus node pertama dari list.

hapusBelakang(): Menghapus node terakhir dari list.

hapusTengah(int posisi): Menghapus node di posisi tertentu di tengah list.

tampil(): Menampilkan data semua node dalam list.

clearList(): Menghapus semua node dalam list.

c. unguided/tugas

unguided 1

```
// priesty ameiliana mualidah
```

```
// 2311102175
```

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
struct Node {
```

```
    string nama;
```

```
    string nim;
```

```
    Node *next;
```

```
};
```

```
Node *head, *tail;
```

```
void init() {
```

```
    head = tail = NULL;
```

```
}
```

```
bool isEmpty() {
```

```
    return head == NULL;
```

```
}
```

```
void addDepan(string nama, string nim) {  
  
    Node *baru = new Node;  
  
    baru->nama = nama;  
  
    baru->nim = nim;  
  
    if (isEmpty()) {  
  
        head = tail = baru;  
  
        baru->next = head;  
  
    } else {  
  
        baru->next = head;  
  
        tail->next = baru;  
  
        head = baru;  
  
    }  
}  
  
void addBelakang(string nama, string nim) {  
  
    Node *baru = new Node;  
  
    baru->nama = nama;  
  
    baru->nim = nim;  
  
    if (isEmpty()) {  
  
        head = tail = baru;  
  
baru->next = head;  
  
    } else {  
  
        baru->next = head;  
  
        tail->next = baru;  
  
        tail = baru;  
  
    }  
}
```

```
void addTengah(string nama, string nim, int posisi) {
```

```
    if (isEmpty() || posisi <= 1) {
```

```
        addDepan(nama, nim);
```

```
        return;
```

```
    }
```

```
    Node *baru = new Node;
```

```
    baru->nama = nama;
```

```
    baru->nim = nim;
```

```
    Node *curr = head;
```

```
    for (int i = 1; i < posisi - 1; i++) {
```

```
        curr = curr->next;
```

```
    }
```

```
    baru->next = curr->next;
```

```
    curr->next = baru;
```

```
}
```

```
void hapusDepan() {
```

```
    if (isEmpty()) {
```

```
        cout << "List kosong!" << endl;
```

```
    return;
```

```
}
```

```
Node *hapus = head;
```

```
head = head->next;
```

```
tail->next = head;
```

```
delete hapus;
```

```
    cout << "Data " << hapus->nama << " (NIM: " << hapus->nim << ") berhasil  
    dihapus" << endl;
```

```
}
```

```
void hapusBelakang() {
```

```
    if (isEmpty()) {
```

```
        cout << "List kosong!" << endl;
```

```
        return;
```

```
    }
```

```
    Node *hapus = tail;
```

```
    Node *prev = head;
```

```
    while (prev->next != tail) {
```

```
        prev = prev->next;
```

```
    }
```

```
    tail = prev;
```

```
    prev->next = head;
```

```
    delete hapus;
```

```
    cout << "Data " << hapus->nama << " (NIM: " << hapus->nim << ") berhasil  
    dihapus" << endl;
```

```
}
```

```
void hapusTengah(int posisi) {
```



```
if (isEmpty() || posisi <= 1) {  
    hapusDepan();  
    return;  
}  
  
Node *hapus = head;  
Node *prev = head;  
for (int i = 1; i < posisi - 1; i++) {  
    prev = prev->next;  
}  
  
hapus = prev->next;  
prev->next = hapus->next;  
delete hapus;  
  
    cout << "Data " << hapus->nama << " (NIM: " << hapus->nim << ") pada posisi " <<  
posisi << " berhasil dihapus" << endl;  
}  
  
void ubahDepan(string namaBaru, string nimBaru) {  
    if (isEmpty()) {  
        cout << "List kosong!" << endl;  
        return;  
    }  
  
    head->nama = namaBaru;  
    head->nim = nimBaru;
```

```

cout << "Data " << head->nama << " (NIM: " << head->nim << ") telah diganti dengan
data " << namaBaru << " (NIM: " << nimBaru << ")" << endl;

}

void ubahBelakang(string namaBaru, string nimBaru) {
    if (isEmpty()) {
        cout << "List kosong!" << endl;
        return;
    }

    tail->nama = namaBaru;
    tail->nim = nimBaru;

    cout << "Data " << tail->nama << " (NIM: " << tail->nim << ") telah diganti dengan data
" << namaBaru << " (NIM: " << nimBaru << ")" << endl;
}

void ubahTengah(int posisi, string namaBaru, string nimBaru) {
    if (isEmpty() || posisi <= 1) {
        ubahDepan(namaBaru, nimBaru);
        return;
    }

    Node *curr = head;
    Node *prev = head;

    for (int i = 1; i < posisi - 1; i++) {
        prev = prev->next;
    }

    curr = prev->next;

```

```
curr->nama = namaBaru;

curr->nim = nimBaru;

cout << "Data " << curr->nama << " (NIM: " << curr->nim << ") pada posisi " << posisi
<< " telah diganti dengan data " << namaBaru << " (NIM: " << nimBaru << ")" << endl;
}

void hapusList() {
if (isEmpty()) {
    cout << "List kosong!" << endl;
    return;
}

Node *curr = head;
Node *next;

while (curr != tail) {
    next = curr->next;
    delete curr;
    curr = next;
}

delete tail;

head = tail = NULL;

cout << "List data mahasiswa telah dihapus" << endl;
}
```

```
void tampilData() {
    if (isEmpty()) {
        cout << "List kosong!" << endl;
        return;
    }

    cout << "DATA MAHASISWA" << endl;
    cout << "-----" << endl;

    Node *curr = head;
    do {
        cout << "| " << curr->nama << " | " << curr->nim << " |" << endl;
        curr = curr->next;
    } while (curr != head);

    cout << "-----" << endl;
}

int main() {
    init();

    int pilihan;
    string nama, nim;
    int posisi;

    do {
```

```
// Menampilkan menu

cout << "PROGRAM CIRCULAR LINKED LIST" << endl;

cout << "-----" << endl;

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 Data" << endl;
cout << "0. Keluar" << endl;


// Memilih operasi

cout << "Pilih Operasi: ";

cin >> pilihan; switch (pilihan) {

    case 1:

        cout << "- Tambah Depan -" << endl;

        cout << "Masukkan Nama: ";

        cin >> nama;

        cout << "Masukkan NIM: ";

        cin >> nim;
```

```
addDepan(nama, nim);

    break;

    case 2:

        cout << "- Tambah Belakang -" << endl;

        cout << "Masukkan Nama: ";

        cin >> nama;

        cout << "Masukkan NIM: ";

        cin >> nim;

        addBelakang(nama, nim);

        break;

    case 3:

        cout << "- Tambah Tengah -" << endl;

        cout << "Masukkan Nama: ";

        cin >> nama;

        cout << "Masukkan NIM: ";

        cin >> nim;

        cout << "Masukkan Posisi: ";

        cin >> posisi;

        addTengah(nama, nim, posisi);

        break;

    case 4:

        cout << "- Ubah Depan -" << endl;

        cout << "Masukkan Nama: ";

        cin >> nama;

        cout << "Masukkan NIM: ";

        cin >> nim;

        ubahDepan(nama, nim);
```

```
break;

case 5:

    cout << "- Ubah Belakang -" << endl;

    cout << "Masukkan Nama: ";

    cin >> nama;

    cout << "Masukkan NIM: ";

    cin >> nim;

    ubahBelakang(nama, nim);

    break;

case 6:

    cout << "- Ubah Tengah -" << endl;

    cout << "Masukkan Posisi: ";

    cin >> posisi;

    cout << "Masukkan Nama: ";

    cin >> nama;

    cout << "Masukkan NIM: ";

    cin >> nim;

    ubahTengah(posisi, nama, nim);

    break;

case 7:

    cout << "- Hapus Depan -" << endl;

    hapusDepan();

    break;

case 8:

    cout << "- Hapus Belakang -" << endl;
```

```
hapusBelakang();

    break;

case 9:

    cout << "- Hapus Tengah -" << endl;

    cout << "Masukkan Posisi: ";

    cin >> posisi;

    hapusTengah(posisi);

    break;

case 10:

    hapusList();

    break;

case 11:

    tampilData();

    break;

}

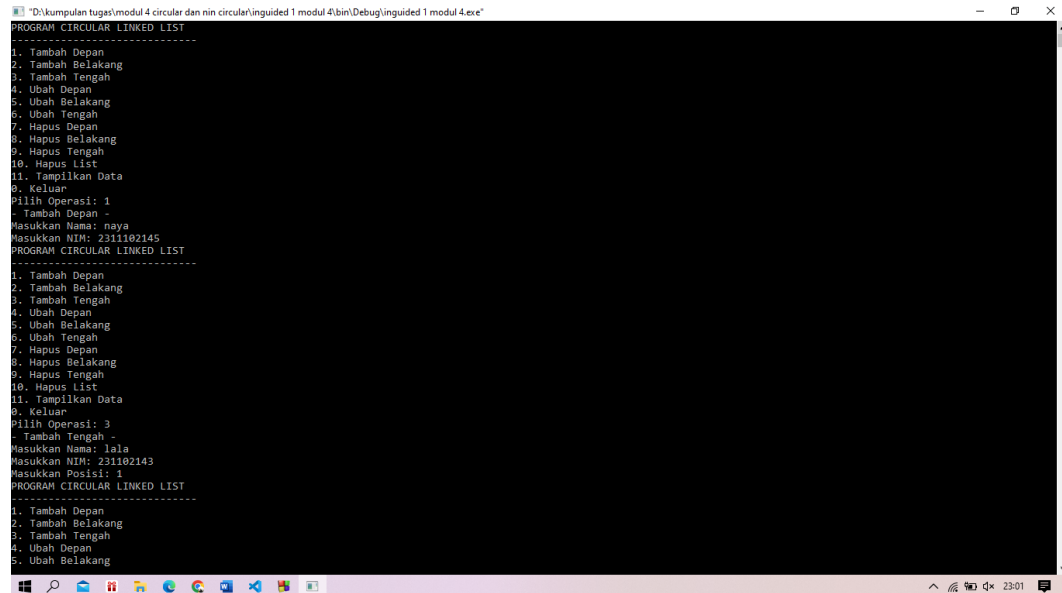
} while (pilihan != 0);

cout << "Terima kasih telah menggunakan program ini!" << endl;

return 0;

}
```


Screenshots output:



```
"D:\kumpulan tugas\modul 4 circular dan nin circular\inguided 1 modul 4\bin\Debug\inguided 1 modul 4.exe"
PROGRAM CIRCULAR LINKED LIST
-----
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan Data
0. Keluar
Pilih Operasi: 1
- Tambah Depan -
Masukkan Nama: naya
Masukkan NIM: 231102145
PROGRAM CIRCULAR LINKED LIST
-----
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan Data
0. Keluar
Pilih Operasi: 3
- Tambah Tengah -
Masukkan Nama: lala
Masukkan NIM: 231102143
Masukkan Posisi: 1
PROGRAM CIRCULAR LINKED LIST
-----
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
```

Deskripsi :

- **Deklarasi Struct Node:** Mendefinisikan struktur data Node yang terdiri dari dua field: nama (menyimpan nama), nim (menyimpan NIM), dan next (pointer ke node berikutnya).
- **Variabel Global:** Mendeklarasikan variabel global head dan tail untuk menunjuk ke node pertama dan terakhir dalam list.
- **Fungsi-fungsi:**
 - `init()`: Mengatur head dan tail menjadi NULL untuk menunjukkan list kosong.
 - `isEmpty()`: Memeriksa apakah list kosong atau tidak.

- addDepan(string nama, string nim): Menambahkan node baru di awal list.
- addBelakang(string nama, string nim): Menambahkan node baru di akhir list.
- addTengah(string nama, string nim, int posisi): Menambahkan node baru di posisi tertentu di tengah list.
- ubahDepan(string namaBaru, string nimBaru): Mengubah data pada node pertama.
- ubahBelakang(string namaBaru, string nimBaru): Mengubah data pada node terakhir.
- ubahTengah(int posisi, string namaBaru, string nimBaru): Mengubah data pada node di posisi tertentu di tengah list.
- hapusDepan(): Menghapus node pertama dari list.
- hapusBelakang(): Menghapus node terakhir dari list.
- hapusTengah(int posisi): Menghapus node di posisi tertentu di tengah list.
- hapusList(): Menghapus semua node dalam list.

- tampilData(): Menampilkan data semua node dalam list.

Unguided 2

```
// priesty ameiliana maulidah
// 2311102175

#include <iostream>
#include <string>

using namespace std;

struct Node {
    string nama;
    string nim;
    Node *next;
};
Node *head, *tail;

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

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

void addDepan(string nama, string nim) {
    Node *baru = new Node;
    baru->nama = nama;
    baru->nim = nim;
```

```
if (isEmpty()) {  
    head = tail = baru;  
    baru->next = head;  
} else {  
    baru->next = head;  
    tail->next = baru;  
    head = baru;  
}  
}  
  
void addBelakang(string nama, string nim) {  
    Node *baru = new Node;  
    baru->nama = nama;  
    baru->nim = nim;  
    if (isEmpty()) {  
        head = tail = baru;  
        baru->next = head;  
    } else {  
        baru->next = head;  
        tail->next = baru;  
    }  
    tail = baru;  
}  
}  
  
void addTengah(string nama, string nim, int posisi) {  
    if (isEmpty() || posisi <= 1) {  
        addDepan(nama, nim);  
        return;  
    }  
    Node *baru = new Node;  
    baru->nama = nama;
```

```
void hapusBelakang() {
    if (isEmpty()) {
        cout << "List kosong!" << endl;
        return;
    }

    Node *hapus = tail;
    Node *prev = head;

    while (prev->next != tail) {
        prev = prev->next;
    }

    tail = prev;
    prev->next = head;

    delete hapus;

    cout << "Data " << hapus->nama << " (NIM: " << hapus->nim << ") berhasil dihapus" <<
    endl;
}

void hapusTengah(int posisi) {
    if (isEmpty() || posisi <= 1) {
        hapusDepan();
        return;
    }
}
```

```
Node *hapus = head;

Node *prev = head;

for (int i = 1; i < posisi - 1; i++) {
    prev = prev->next;
}

hapus = prev->next;
prev->next = hapus->next;

delete hapus;

cout << "Data " << hapus->nama << " (NIM: " << hapus->nim << ") pada posisi " <<
posisi << " berhasil dihapus" << endl;
}

void ubahDepan(string namaBaru, string nimBaru) {
    if (isEmpty()) {
        cout << "List kosong!" << endl;
        return;
    }

    head->nama = namaBaru;
    head->nim = nimBaru;

    cout << "Data " << head->nama << " (NIM: " << head->nim << ") telah diganti
dengan data " << namaBaru << " (NIM: " << nimBaru << ")" << endl;
}
```

```

void ubahBelakang(string namaBaru, string nimBaru) {
    if (isEmpty()) {
        cout << "List kosong!" << endl;
        return;
    }

    tail->nama = namaBaru;
    tail->nim = nimBaru;

    cout << "Data " << tail->nama << " (NIM: " << tail->nim << ") telah diganti dengan
    data " << namaBaru << " (NIM: " << nimBaru << ")" << endl;
}

void ubahTengah(int posisi, string namaBaru, string nimBaru) {
    if (isEmpty() || posisi <= 1) {
        ubahDepan(namaBaru, nimBaru);
        return;
    }

    Node *curr = head;
    Node *prev = head;

    for (int i = 1; i < posisi - 1; i++) {
        prev = prev->next;
    }

    curr = prev->next;
    curr->nama = namaBaru;
    curr->nim = nimBaru;
}

```



```
        cout << "Data " << curr->nama << " (NIM: " << curr->nim << ") pada posisi  
" << posisi << " telah diganti dengan data " << namaBaru << " (NIM: " << nimBaru <<  
")" << endl;  
    }  
  
void hapusList() {  
    if (isEmpty()) {  
        cout << "List kosong!" << endl;  
        return;  
    }  
  
    Node *curr = head;  
    Node *next;  
  
    while (curr != tail) {  
        next = curr->next;  
        delete curr;  
        curr = next;  
    }  
  
    delete tail;  
  
    head = tail = NULL;  
  
    cout << "List data mahasiswa telah dihapus" << endl;  
}
```

```
void tampilData() {  
    if (isEmpty()) {  
        cout << "List kosong!" << endl;  
        return;  
    }  
  
    cout << "DATA MAHASISWA" << endl;  
    cout << "-----" << endl;  
  
    Node *curr = head;  
    do {  
        cout << "| " << curr->nama << " | " << curr->nim << " |" << endl;  
        curr = curr->next;  
    } while (curr != head);  
  
    cout << "-----" << endl;  
}  
  
int main() {  
    init();  
  
    int pilihan;  
    string nama, nim;  
    int posisi;  
  
    do {
```

```
// Menampilkan menu

cout << "PROGRAM CIRCULAR LINKED LIST" << endl;

cout << "-----" << endl;

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 Data" << endl;
cout << "0. Keluar" << endl;


// Memilih operasi

cout << "Pilih Operasi: ";

cin >> pilihan;
```

```
switch (pilihan) {  
    case 1:  
        cout << "- Tambah Depan -" << endl;  
        cout << "Masukkan Nama: ";  
        cin >> nama;  
        cout << "Masukkan NIM: ";  
        cin >> nim;  
        addDepan(nama, nim);  
        break;  
  
    case 2:  
        cout << "- Tambah Belakang -" << endl;  
        cout << "Masukkan Nama: ";  
        cin >> nama;  
        cout << "Masukkan NIM: ";  
        cin >> nim;  
        addBelakang(nama, nim);  
        break;  
  
    case 3:  
        cout << "- Tambah Tengah -" << endl;  
        cout << "Masukkan Nama: ";  
        cin >> nama;  
        cout << "Masukkan NIM: ";  
        cin >> nim;
```

```
cout << "Masukkan Posisi: ";

    cin >> posisi;

    addTengah(nama, nim, posisi);

    break;

case 4:

    cout << "- Ubah Depan -" << endl;

    cout << "Masukkan Nama: ";

    cin >> nama;

    cout << "Masukkan NIM: ";

    cin >> nim;

    ubahDepan(nama, nim);

    break;

case 5:

    cout << "- Ubah Belakang -" << endl;

    cout << "Masukkan Nama: ";

    cin >> nama;

    cout << "Masukkan NIM: ";

    cin >> nim;

    ubahBelakang(nama, nim);

    break;

case 6:

    cout << "- Ubah Tengah -" << endl;

    cout << "Masukkan Posisi: ";

    cin >> posisi;

    cout << "Masukkan Nama: ";

    cin >> nama;

    cout << "Masukkan NIM: ";

    cin >> nim;

    ubahTengah(posisi, nama, nim);

    break;
```

case 7:

```
cout << "- Hapus Depan -" << endl;
```

```
hapusDepan();
```

```
break;
```

case 8:

```
cout << "- Hapus Belakang -" << endl;
```

```
hapusBelakang();
```

```
break;
```

case 9:

```
cout << "- Hapus Tengah -" << endl;
```

```
cout << "Masukkan Posisi: ";
```

```
cin >> posisi;
```

```
hapusTengah(posisi);
```

```
break;
```

case 10:

```
hapusList();
```

```
break;
```

case 11:

```
tampilData();
```

```
break;
```

```
}
```

```
} while (pilihan != 0);
```

```
cout << "Terima kasih telah menggunakan program ini!" << endl;
```

```
return 0;
```

```
}
```

Screenshots output

Deskripsi :

- **Menambah data:**

- addDepan(nama, nim): Menambahkan data di depan list.
- addBelakang(nama, nim): Menambahkan data di belakang list.
- addTengah(nama, nim, posisi): Menambahkan data di tengah list pada posisi tertentu.

- **Mengubah data:**

- ubahDepan(namaBaru, nimBaru): Mengubah data di depan list.
- ubahBelakang(namaBaru, nimBaru): Mengubah data di belakang list.
- ubahTengah(posisi, namaBaru, nimBaru): Mengubah data di tengah list pada posisi tertentu.

- **Menghapus data:**

- hapusDepan(): Menghapus data di depan list.
- hapusBelakang(): Menghapus data di belakang list.
- hapusTengah(posisi): Menghapus data di tengah list pada posisi tertentu.
- **Lain-lain:**
 - hapusList(): Menghapus seluruh data dalam list.
 - tampilData(): Menampilkan seluruh data dalam list.

E. Referensi

<https://osf.io/preprints/osf/u6qf7>