

**LAPORAN PRAKTIKUM STRUKTUR DATA DAN
ALGORITMA**

**MODUL 4
“LINKED LIST CIRCULAR DAN NON
CIRCULAR”**



DISUSUN OLEH :

M.Fathiakmal.L.A

18102096

DOSEN

WAHYU ANDI SAPUTRA, S.PD., M.PD.

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

2024

A. Dasar Teori

Linked list adalah struktur data linier berbentuk rantai simpul di mana setiap simpul menyimpan 2 item, yaitu nilai data dan pointer ke simpul elemen berikutnya. Berbeda dengan array, elemen linked list tidak ditempatkan dalam alamat memori yang berdekatan melainkan elemen ditautkan menggunakan pointer.

Simpul pertama dari linked list disebut sebagai head atau simpul kepala. Apabila linked list berisi elemen kosong, maka nilai pointer dari head menunjuk ke NULL. Begitu juga untuk pointer berikutnya dari simpul terakhir atau simpul ekor akan menunjuk ke NULL.

Ukuran elemen dari linked list dapat bertambah secara dinamis dan mudah untuk menyisipkan dan menghapus elemen karena tidak seperti array, kita hanya perlu mengubah pointer elemen sebelumnya dan elemen berikutnya untuk menyisipkan atau menghapus elemen.

Linked list biasanya digunakan untuk membuat file system, adjacency list, dan hash table.

JENIS JENIS LINKEDLIST YANG AKAN DI BAHAS PADA MODUL INI :

• LINKED LIST NON CIRCULAR

Linked list non circular merupakan linked list dengan node pertama (head) dan node terakhir (tail) yang tidak saling terhubung. Pointer terakhir (tail) pada Linked List ini selalu bernilai “NULL” sebagai pertanda data terakhir dalam list-nya. Linked list non circular dapat digambarkan sebagai berikut.

OPERASI PADA LINKED LIST NON CIRCULAR

1. Deklarasi Simpul (Node)

```
struct node { int data;  
  
node *next;  
  
};
```

2. Membuat dan Menginisialisasi Pointer Head dan Tail

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

3. Pengecekan Kondisi Linked List

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

```
}  
|_____|
```

4. Penambah Simpul (Node)

```
void insertBelakang(string  
dataUser) { if (isEmpty()  
== true)  
{ node *baru = new  
node; baru->data =  
dataUser; head = baru;  
tail = baru;  
baru->next = NULL;  
} else { node *baru =  
new node; baru->data =  
dataUser; baru->next =  
NULL;  
tail->next = baru;  
  
tail = baru;  
}  
};
```

5. Penghapusan Simpul (Node)

```

void hapusDepan()

{ if (isEmpty() ==

true)

{ cout << "List kosong!"

<< endl; } else { node

*helper; helper = head; if

(head == tail)

{

head = NULL; tail

= NULL; delete

helper;

} else head = head-

>next; helper->next =

NULL; delete helper;

}

}

}

```

6. Tampil Data Linked List

```
void tampil()
{ if (isEmpty() ==
true)
{ cout << "List kosong!" <<
endl;
} else { node *helper;
helper = head; while
(helper != NULL)
{ cout << helper->data <<
ends; helper = helper->next;
}
}
}
```

• LINKED LIST CIRCULAR

Linked list circular merupakan linked list yang tidak memiliki akhir karena node terakhir(tail) tidak bernilai 'NULL', tetapi terhubung dengan node pertama (head). 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 (head). Linked list circular dapat digunakan untuk menyimpan data yang perlu diakses secara berulang, seperti daftar putar lagu, daftar pesan dalam antrian, atau penggunaan memori berulang dalam suatu aplikasi.

OPERASI PADA LINKED LIST CIRCULAR

1. Deklarasi Simpul (Node)

```
struct Node  
{ string data;  
Node *next;  
};
```

2. Membuat dan Menginisialisasi pointer Head dan Tail

```
Node *head, *tail, *baru,  
*bantu, *hapus; void  
init()  
{  
head = NULL;  
tail = head;  
}
```

3. Pengecekan Kondisi Linked List

```
int isEmpty()

{ if (head ==

NULL)

return 1; // true

else return 0; //

false

}
```

4. Pembuatan Simpul (Node)

```
void buatNode(string data)

{ baru = new Node;

baru->data = data;

baru->next =

NULL;

}
```

5. Penambahan Simpul (Node)

```
// 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;
}
}
```

6. Penghapusan Simpul (Node)

```
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;  
}  
}
```

7. Menampilkan Data Linked List

```
void tampil()
{ if (isEmpty() ==
0)
{ tail = head; do { cout <<
tail->data << ends; tail =
tail->next; } while (tail !=
head); cout << endl;
}
}
```

B. Guided

Guided 1

Source Code :

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

struct Node
{
    int data;
    Node *next;
```

```
};

Node *head;
Node *tail;

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

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

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;
    }
}

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;
    }
}
```

```

}

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

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;
    }
}

void hapusDepan()
{
    if (!isEmpty())
    {
        Node *hapus = head;
        if (head->next != NULL)

```

```

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

```

```

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

```

```

void hapusTengah(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 *bantu = head;
        Node *hapus;
        Node *sebelum = NULL;
        int nomor = 1;
        while (nomor < posisi)
        {
            sebelum = bantu;
            bantu = bantu->next;
            nomor++;
        }
        hapus = bantu;
        if (sebelum != NULL)
        {
            sebelum->next = bantu->next;
        }
        else
        {
            head = bantu->next;
        }
        delete hapus;
    }
}

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

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

void ubahBelakang(int data)
{
    if (!isEmpty())
    {
        tail->data = data;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

void clearList()
{
    Node *bantu = head;
    Node *hapus;
    while (bantu != NULL)
    {
        hapus = bantu;
    }
}

```

```

        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}

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

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

```

```

    ubahBelakang(8);
    tampil();
    ubahTengah(11, 2);
    tampil();

    return 0;
}

```

Screenshots Output:

```

3 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\Administrator\Documents\GitHub\Laporan-Praktikum-SDA\Modul4> M.Fathiakmal 18102096

```

Deskripsi Program :

Fungsi `init()` bertanggung jawab untuk menginisialisasi pointer `head` dan `tail` menjadi `NULL`, menandakan bahwa linked list kosong. Fungsi `isEmpty()` digunakan untuk memeriksa apakah linked list kosong atau tidak. Kemudian, fungsi `insertDepan(int nilai)` dan `insertBelakang(int nilai)` menambahkan node baru di depan dan belakang linked list, berturut-turut. Fungsi `hitungList()` menghitung jumlah node dalam linked list. Fungsi `insertTengah(int data, int posisi)` menambahkan node baru di posisi tengah linked list. Sementara itu, fungsi `hapusDepan()` dan `hapusBelakang()` menghapus node pertama dan terakhir dari linked list.

Untuk mengubah nilai data dari node, ada fungsi `ubahDepan(int data)`, `ubahTengah(int data, int posisi)`, dan `ubahBelakang(int data)`. Fungsi-fungsi tersebut mengubah nilai data dari node pertama, node di posisi tengah, dan node terakhir linked list. Terakhir, fungsi `clearList()` menghapus semua node dari linked list, sementara `tampil()` menampilkan

semua nilai data dalam linked list. Fungsi main() digunakan untuk menguji implementasi linked list dengan memanggil fungsi-fungsi tersebut.

Guided 2

Source Code :

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

struct Node {
    string data;
    Node *next;
};

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

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

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

void buatNode(string data) {
    baru = new Node;
    baru->data = data;
    baru->next = NULL;
}

int hitungList() {
    bantu = head;
    int jumlah = 0;
    while (bantu != NULL) {
        jumlah++;
        bantu = bantu->next;
    }
    return jumlah;
}

void insertDepan(string data) {
    buatNode(data);
    if (isEmpty()) {
        head = baru;
        tail = head;
    }
```

```

        baru->next = head;
    } else {
        while (tail->next != head) {
            tail = tail->next;
        }
        baru->next = head;
        head = baru;
        tail->next = head;
    }
}

void insertBelakang(string data) {
    buatNode(data);
    if (isEmpty()) {
        head = baru;
        tail = head;
        baru->next = head;
    } else {
        while (tail->next != head) {
            tail = tail->next;
        }
        tail->next = baru;
        baru->next = head;
    }
}

void insertTengah(string data, int posisi) {
    if (isEmpty()) {
        head = baru;
        tail = head;
        baru->next = head;
    } else {
        baru->data = data;
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1) {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

void hapusDepan() {
    if (!isEmpty()) {

```

```

        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;
    }
}

```

```

void hapusBelakang() {
    if (!isEmpty()) {
        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;
    }
}

```

```

void hapusTengah(int posisi) {
    if (!isEmpty()) {

```

```

        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;
    }
}

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;
}

void tampil() {
    if (!isEmpty()) {
        tail = head;
        do {
            cout << tail->data << " ";
            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:

```

Ayam
Bebek Ayam
Bebek Ayam Cicak
Bebek Ayam Cicak Domba
Bebek Ayam Cicak
Ayam Cicak
Ayam Sapi Cicak
PS C:\Users\Administrator\Documents\GitHub\Laporan-Praktikum-SDA\Modul4> Fathiakmal 18102096

```

Deskripsi Program :

Program di atas adalah implementasi dari Circular Singly Linked List dalam C++ yang menyimpan data string. Struktur Node mendefinisikan node dengan atribut data dan next. Fungsi init menginisialisasi linked list, sementara isEmpty memeriksa apakah linked list kosong. Fungsi buatNode membuat node baru, dan hitungList menghitung jumlah node dalam linked list. Fungsi insertDepan, insertBelakang, dan insertTengah digunakan untuk menambahkan node baru di depan, belakang, dan posisi tertentu dalam linked list. Fungsi hapusDepan, hapusBelakang, dan hapusTengah digunakan untuk menghapus node dari depan, belakang, dan posisi tertentu. Fungsi clearList menghapus seluruh node dalam linked list, sementara tampil menampilkan semua data dalam linked list. Program utama (main) mendemonstrasikan penggunaan fungsi-fungsi tersebut dengan berbagai operasi pada linked list, seperti menambah, menghapus, dan menampilkan node.

C. Unguided

Source Code:

```
#include <iostream>
using namespace std;
/// PROGRAM SINGLE LINKED LIST NON-CIRCULAR
// Deklarasi Struct Node
struct Node
{
    string nama;
    string nim;
    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(string nama, string nim)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->nama = nama;
    baru->nim = nim;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}
```

```

    }
}
// Tambah Belakang
void insertBelakang(string nama, string nim)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->nama = nama;
    baru->nim = nim;
    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(string nama, string nim, 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->nama = nama;
    baru->nim = nim;
    // 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 << "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;
    }
    else
    {
        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(string nama, string nim)
{
    if (isEmpty() == 0)
    {
        head->nama = nama;
        head->nim = nim;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah Tengah
void ubahTengah(string nama, string nim, 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
        {
            bantu = head;
            int nomor = 1;
            while (nomor < posisi)
            {
                bantu = bantu->next;
                nomor++;
            }

```

```

        bantu->nama = nama;
        bantu->nim = nim;
    }
}
else
{
    cout << "List masih kosong!" << endl;
}
}

// Ubah Belakang
void ubahBelakang(string nama, string nim)
{
    if (isEmpty() == 0)
    {
        tail->nama = nama;
        tail->nim = nim;
    }
    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())
    {
        cout << endl << endl;
        cout << "\t\t\t\t\t Output Data" << endl;
        cout << "\t\t\t\t\t _____" << endl << endl;
    }
}

```

```

        cout << " Nama\t\t|  Nim\t\t|" << endl;
        while (bantu != NULL)
        {
            cout << bantu->nama << "\t\t|" << bantu->nim << "\t" << endl;
            bantu = bantu->next;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
        return;
    }
}

int main()
{

    string nama;
    string nim;
    int pilihan;
    int posisi;

    do
    {
        cout << endl;
        cout << "\t\tData Mahasiswa" << 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" << endl;
        cout << "12. Exit" << endl;
        cout << "input (1-12): ";
        cin >> pilihan;
        switch (pilihan)
        {
            case 1:
            {
                cout << "manu  depan\n\n";
                cout << "Masukkan Nama :";
                cin.ignore();
            }
        }
    }
}

```

```

        getline(cin, nama);
        cout << "Masukkan Nim: ";
        cin.ignore();
        cin >> nim;
        insertDepan(nama, nim);
        cout << "Data " << nama << " berhasil diinput!";

        break;
    }
    case 2:
    {
        cout << "manu  belakang\n\n";
        cout << "Masukkan Nama :";
        cin.ignore();
        getline(cin, nama);
        cout << "Masukkan Nim: ";
        cin >> nim;
        insertBelakang(nama, nim);
        cout << "Data " << nama << " berhasil diinput!";

        break;
    }
    case 3:
    {

        cout << "manu  tengah\n\n";
        cout << "Masukkan Nama :";
        cin.ignore();
        getline(cin, nama);
        cout << "Masukkan Nim: ";
        cin >> nim;
        cout << "Masukan Posisi: ";
        cin >> posisi;
        insertTengah(nama, nim, posisi);
        cout << "Data " << nama << " Berhasil diinput!" << endl;
        break;
    }
    case 4:
    {
        cout << "manu ubah depan\n\n";
        cout << "Masukkan Nama :";
        cin.ignore();
        getline(cin, nama);
        cout << "Masukkan Nim: ";
        cin >> nim;
        ubahDepan(nama, nim);
        cout << "data depan berhasil diubah";
    }
}

```



```

        break;
    }
    case 5:
    {
        cout << "manu ubah belakang\n\n";
        cout << "Masukkan Nama :";
        cin.ignore();
        getline(cin, nama);
        cout << "Masukkan Nim: ";
        cin >> nim;
        ubahBelakang(nama, nim);
        cout << "data belakang berhasil diubah";
        break;
    }
    case 6:
    {
        cout << "manu ubah tengah\n\n";
        cout << "Masukkan Nama :";
        cin.ignore();
        getline(cin, nama);
        cout << "Masukkan Nim: ";
        cin >> nim;
        cout << "Masukkan Posisi: ";
        cin >> posisi;
        ubahTengah(nama, nim, posisi);
        cout << "data tengah berhasil diubah";
        break;
    }

    case 7:
    {
        cout << "manu hapus depan\n\n";
        hapusDepan();
        cout << "Data Depan berhasil terhapus!";
        break;
    }
    case 8:
    {
        cout << "manu hapus belakang\n\n";
        hapusBelakang();
        cout << "Data Belakang berhasil terhapus!";
        break;
    }
    case 9:
    {
        cout << "manu hapus tengah\n\n";

```

```
        cout << "Masukkan Posisi: ";
        cin >> posisi;
        hapusTengah(posisi);
        cout << "Data Tengah berhasil terhapus!";
        break;
    }
    case 10:
    {
        clearList();
        break;
    }

    case 11:
    {
        tampil();
        break;
    }
    case 12:
    {
        cout << "Terima Kasih!" << endl;
    }

    default:
    {
        cout << "Pilihan tidak Valid!" << endl;
        break;
    }
}
} while (pilihan != 12);

return 0;
}
```

Screenshot output:

1. Buatlah menu untuk menambahkan, mengubah, menghapus, dan melihat Nama dan NIM Mahasiswa.

- Tampilan Menu

```
                Data Mahasiswa
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
12. Exit
input (1-12): M.Fathiakmal 18102096
```

- Tampilan Operasi Tambah

```
                Data Mahasiswa
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
12. Exit
input (1-12): 1
manu depan

Masukkan Nama :Fathiakmal
Masukkan Nim: 18102096
Data Fathiakmal berhasil diinput!
```

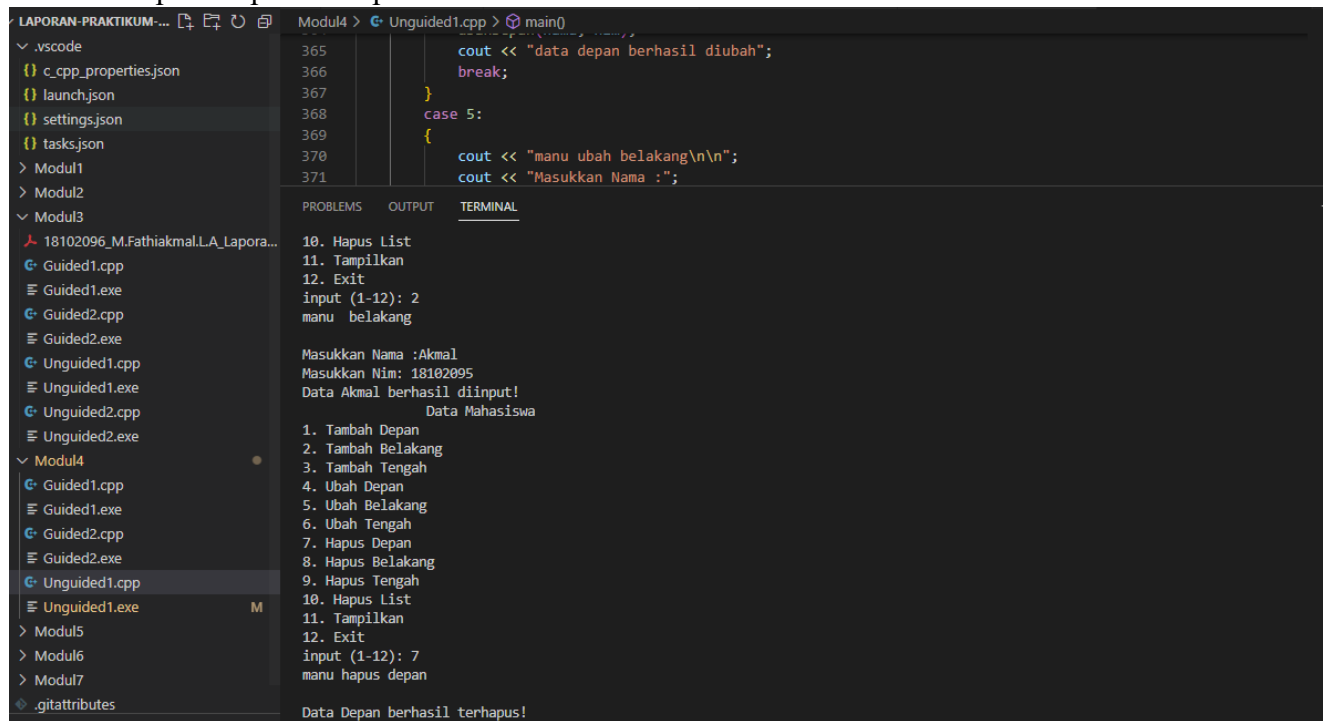
```

                                Data Mahasiswa
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
12. Exit
input (1-12): 2
manu belakang

Masukkan Nama :Akmal
Masukkan Nim: 18102095
Data Akmal berhasil diinput!

```

- Tampilan Operasi Hapus



- Tampilan Operasi Ubah

```

Data Mahasiswa
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
12. Exit
input (1-12): 5
manu ubah belakang

Masukkan Nama :Fathiakmal
Masukkan Nim: 18102096
data belakang berhasil diubah

```

- Tampilan Operasi Tampil Data

```

Data Mahasiswa
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
12. Exit
input (1-12): 11

Output Data
-----
Nama      | Nim      |
Fathiakmal | 18102096 |

```

2. Setelah membuat menu tersebut, masukkan data sesuai urutan berikut, lalu tampilkan data yang telah dimasukkan. (Gunakan insert depan, belakang atau tengah)

The screenshot shows a C++ IDE with a file explorer on the left and a console window on the right. The file explorer shows a project named '18102096_M.Fathiakmal.LA_Lapora...' with files 'Guided1.cpp', 'Guided1.exe', 'Guided2.cpp', 'Guided2.exe', 'Unguided1.cpp', 'Unguided1.exe', and 'Unguided2.cpp'. The 'Modul4' folder is expanded, showing 'Guided1.cpp', 'Guided1.exe', 'Guided2.cpp', 'Guided2.exe', and 'Unguided1.cpp'. The 'Unguided1.cpp' file is selected, and its output is displayed in the console window.

```
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
12. Exit
input (1-12): 11
```

Output Data

Nama	Nim
jawad	3300001
Bintang Yudistira	2311102052
farrel	23300003
denis	23300005
anis	23300008
bowo	23300015
gahar	23300040
udin	23300048
ucok	23300050
budi	23300099

3. a. Tambahkan data berikut diantara Farrel dan Denis: Wati 23300004

```

18102096_M.Fathiakmal.LA_Lapora...
Guided1.cpp
Guided1.exe
Guided2.cpp
Guided2.exe
Unguided1.cpp
Unguided1.exe
Unguided2.cpp
Unguided2.exe
Modul4
  Guided1.cpp
  Guided1.exe
  Guided2.cpp
  Guided2.exe
  Unguided1.cpp
  Unguided1.exe
Modul5
Modul6
Modul7
.gitattributes

```

```

gahar      |23300040
udin       |23300048
ucok       |23300050
budi       |23300099

Data Mahasiswa
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
12. Exit
input (1-12): 3
manu tengah

Masukkan Nama :wati
Masukkan Nim: 2330004
Masukan Posisi: 4
Data wati Berhasil diinput!

```

b. Hapus data Denis

```

18102096_M.Fathiakmal.LA_Lapora...
Guided1.cpp
Guided1.exe
Guided2.cpp
Guided2.exe
Unguided1.cpp
Unguided1.exe
Unguided2.cpp
Unguided2.exe
Modul4
  Guided1.cpp
  Guided1.exe
  Guided2.cpp
  Guided2.exe
  Unguided1.cpp
  Unguided1.exe
Modul5
Modul6
Modul7
.gitattributes

```

```

denis      |23300005
anis       |23300008
bowo       |23300015
gahar      |23300040
udin       |23300048
ucok       |23300050
budi       |23300099

Data Mahasiswa
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
12. Exit
input (1-12): 9
manu hapus tengah

Masukkan Posisi: 5
Data Tengah berhasil terhapus!

```

c. Tambahkan data berikut di awal: Owi 2330000

```

Data Mahasiswa
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
12. Exit
input (1-12): 1
manu depan

Masukkan Nama :owi
Masukkan Nim: 2330000
Data owi berhasil diinput!

```

d. Tambahkan data berikut di akhir: David 23300100

```

Data Mahasiswa
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
12. Exit
input (1-12): 2
manu belakang

Masukkan Nama :David
Masukkan Nim: 23300100
Data David berhasil diinput!

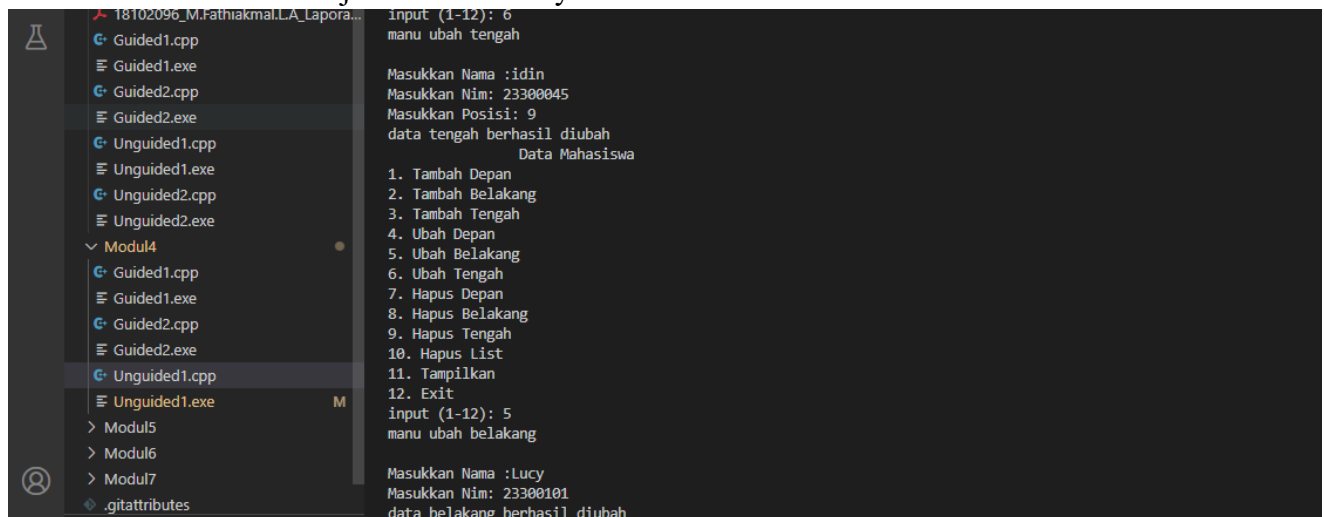
```

e. Ubah data Udin menjadi data berikut: Idin 23300045


```
Data Mahasiswa
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
12. Exit
input (1-12): 6
manu ubah tengah

Masukkan Nama :idin
Masukkan Nim: 23300045
Masukkan Posisi: 9
data tengah berhasil diubah
```

f. Ubah data terkahir menjadi berikut: Lucy 23300101



```
input (1-12): 6
manu ubah tengah

Masukkan Nama :idin
Masukkan Nim: 23300045
Masukkan Posisi: 9
data tengah berhasil diubah
Data Mahasiswa
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
12. Exit
input (1-12): 5
manu ubah belakang

Masukkan Nama :Lucy
Masukkan Nim: 23300101
data belakang berhasil diubah
```

g. Hapus Data Awal

```
18102096_M.Fathiakmal.LA_Lapora... 10. Hapus List
Guided1.cpp 11. Tampilkan
Guided1.exe 12. Exit
Guided2.cpp input (1-12): 5
Guided2.exe manu ubah belakang

Masukkan Nama :Lucy
Masukkan Nim: 23300101
data belakang berhasil diubah
Data Mahasiswa

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
12. Exit
input (1-12): 7
manu hapus depan

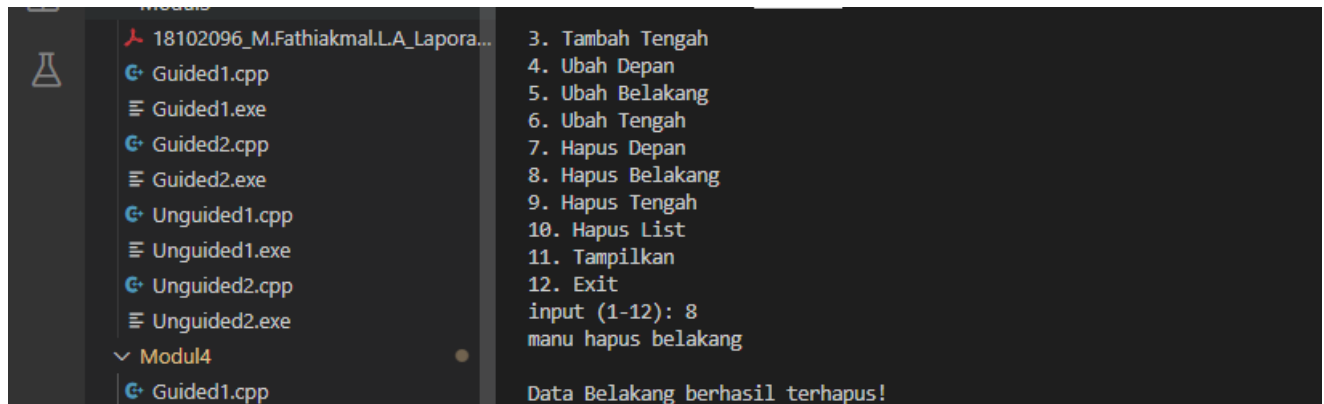
Data Depan berhasil terhapus!
```

h. Ubah data awal menjadi berikut: Bagus 2330002

```
18102096_M.Fathiakmal.LA_Lapora... 5. Ubah Belakang
Guided1.cpp 6. Ubah Tengah
Guided1.exe 7. Hapus Depan
Guided2.cpp 8. Hapus Belakang
Guided2.exe 9. Hapus Tengah
Unguided1.cpp 10. Hapus List
Unguided1.exe 11. Tampilkan
Unguided2.cpp 12. Exit
Unguided2.exe input (1-12): 4
manu ubah depan

Masukkan Nama :Bagas
Masukkan Nim: 2330002
data depan berhasil diubah
```

i. Hapus Data Akhir

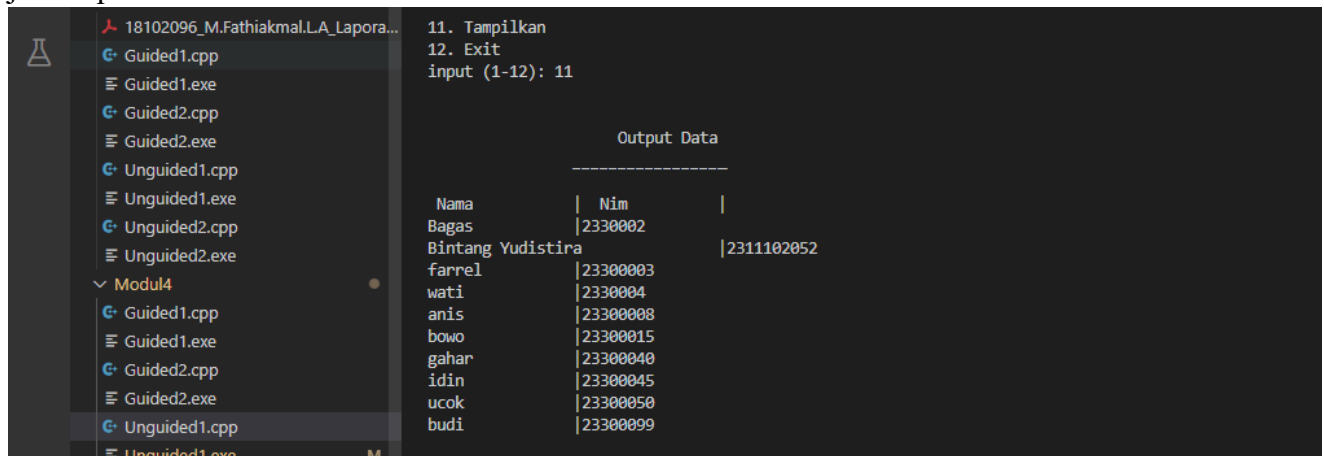


```
18102096_M.Fathiakmal.LA_Lapora...
Guided1.cpp
Guided1.exe
Guided2.cpp
Guided2.exe
Unguided1.cpp
Unguided1.exe
Unguided2.cpp
Unguided2.exe
Modul4
  Guided1.cpp

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
12. Exit
input (1-12): 8
manu hapus belakang

Data Belakang berhasil terhapus!
```

j. Tampilkan seluruh data



```
18102096_M.Fathiakmal.LA_Lapora...
Guided1.cpp
Guided1.exe
Guided2.cpp
Guided2.exe
Unguided1.cpp
Unguided1.exe
Unguided2.cpp
Unguided2.exe
Modul4
  Guided1.cpp
  Guided2.cpp
  Unguided1.cpp
  Unguided2.cpp

11. Tampilkan
12. Exit
input (1-12): 11

Output Data
-----
Nama      | Nim      |
Bagas     | 2330002  |
Bintang Yudistira | 2311102052 |
farrel    | 23300003 |
wati      | 23300004 |
anis      | 23300008 |
bowo      | 23300015 |
gahar     | 23300040 |
idin      | 23300045 |
ucok      | 23300050 |
budi      | 23300099 |
```

Deskripsi Program :

Linked list tersebut menyimpan data mahasiswa, yang terdiri dari nama dan NIM. Struktur data linked list diimplementasikan dengan menggunakan dua struktur, yaitu mahasiswa yang merepresentasikan data mahasiswa, dan node yang merepresentasikan simpul (node) dalam linked list. Pada awalnya, terdapat inisialisasi dari pointer head dan tail dalam fungsi init(), yang diatur menjadi NULL untuk menandakan bahwa linked list masih kosong. Kemudian, terdapat fungsi isEmpty() yang digunakan untuk memeriksa apakah linked list kosong atau tidak. Operasi-operasi dasar pada linked list seperti insertDepan(), insertBelakang(), dan insertTengah() digunakan untuk menyisipkan data mahasiswa ke dalam linked list, baik di depan, di belakang, maupun di posisi tertentu.

Selain itu, terdapat fungsi-fungsi lain seperti `hapusDepan()`, `hapusBelakang()`, dan `hapusTengah()` untuk menghapus data dari linked list. Terdapat juga fungsi-fungsi `tampil()` untuk menampilkan seluruh data mahasiswa yang ada dalam linked list, serta fungsi-fungsi `ubahDepan()`, `ubahBelakang()`, dan `ubahTengah()` untuk mengubah data mahasiswa yang ada dalam linked list. Di dalam fungsi `main()`, terdapat implementasi dari menu operasi-operasi yang dapat dilakukan pada linked list, seperti penambahan data, penghapusan data, perubahan data, dan penampilan data. Program akan berjalan secara iteratif sehingga pengguna dapat melakukan operasi-operasi tersebut secara berulang hingga memilih untuk keluar dari program.

D. Kesimpulan

Linked list circular dan non-circular merupakan dua variasi struktur data sering digunakan dalam pengembangan perangkat lunak untuk mengatur dan mengelola kumpulan data. Perbedaan signifikan antara keduanya terletak pada cara simpul terakhir dalam linked list dihubungkan kembali ke simpul pertama dalam linked list circular, sementara dalam linked list non-circular, simpul terakhir hanya menunjuk ke NULL, menandakan akhir dari linked list. Linked list circular memiliki kelebihan dalam manajemen memori karena tidak ada simpul yang "mati", yang dapat mengoptimalkan penggunaan memori dalam beberapa kasus, terutama ketika operasi traverse (penelusuran) linked list secara terus-menerus diperlukan. Namun, implementasi linked list circular memerlukan operasi tambahan seperti menentukan apakah linked list kosong atau menghitung jumlah elemen, karena harus mempertimbangkan sirkularitas struktur. Pemilihan antara kedua jenis linked list harus didasarkan pada kebutuhan spesifik dari aplikasi atau masalah yang dihadapi. Jika navigasi berulang dari awal hingga akhir linked list sering terjadi, linked list circular mungkin lebih sesuai, namun jika tidak ada kebutuhan khusus seperti itu, linked list non-circular mungkin lebih mudah diimplementasikan dan dikelola..

E. Referensi

[1] Asisten Pratikum “Modul 4 Linkedlist Circular dan non Circular”, Learning Management System, 2024.

[2] Educative. Singly linked list in C++. Diakses pada 31 Maret 2024.

Diakses Pada 7 April 2024, dari

<https://www.educative.io/answers/singly-linked-list-in-cpp>

[3] Taufikkipo (2012, juli). “Single Linked List Non Circular”.

Diakses pada 7 April 2024, dari

<https://www.trivusi.web.id/2022/07/struktur-data-linked-list.html>