

LAPORAN PRAKTIKUM

MODUL IV LINKED LIST CIRCULAR AND NON-CIRCULAR



**Disusun oleh:
Ilhan Sahal Mansiz
2311102029**

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 dapat memahami linked list circular dan non-circular
- 2.** Mahasiswa dapat membedakan circular dengan non-circular linked list

BAB II

DASAR TEORI

Circular linked list adalah jenis linked list di mana node terakhir terhubung kembali ke node pertama, sehingga membentuk sebuah lingkaran tertutup. Dalam circular linked list, traversal dari node pertama akan terus berlanjut hingga kembali ke node pertama lagi. Dalam linked list circular, node tail tidak menunjuk ke 'NULL', melainkan mengarah kembali ke node head, sehingga membentuk lingkaran. Pada node tail dalam linked list circular, pointer menunjuk kembali ke node head. Linked list circular dapat digunakan untuk menyimpan data yang perlu diakses secara berulang, seperti daftar putar lagu dalam pemutar musik, daftar pesan dalam antrian, atau penggunaan memori yang berulang dalam suatu aplikasi. Sedangkan non-circular linked list adalah jenis linked list di mana node terakhir memiliki pointer NULL, sehingga traversal dari node terakhir akan menghasilkan nilai NULL. Kedua jenis linked list ini memiliki kelebihan dan kekurangan masing-masing. Kelebihan circular linked list adalah traversal dari node pertama hingga terakhir dapat dilakukan secara lebih efisien, karena tidak perlu melakukan pengecekan pada pointer NULL. Selain itu, circular linked list juga dapat digunakan untuk membuat algoritma dengan konsep "loop", seperti Round Robin scheduling pada sistem operasi. Namun, kekurangan dari circular linked list adalah lebih sulit dalam melakukan operasi penambahan dan penghapusan node di ujung linked list, karena harus menangani kasus saat menambahkan atau menghapus node pada ujung linked list yang menjadi pusat lingkaran. Sedangkan pada non-circular linked list, operasi penambahan dan penghapusan node pada ujung linked list lebih mudah dilakukan. Sementara itu, kelebihan non-circular linked list adalah lebih mudah untuk diimplementasikan dan dipahami. Selain itu, non-circular linked list juga lebih mudah dalam melakukan operasi penambahan dan penghapusan node pada ujung linked list. Namun, kekurangan dari non-circular linked list adalah traversal dari node pertama hingga terakhir memerlukan pengecekan pada pointer

NULL, sehingga operasi traversal lebih lambat jika dibandingkan dengan circular linked list.

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\ACER\Downloads\PRAKTIKUM STRUKTUR DATA\Modul4> cd 'c:\Users\ACER\Downloads\PRAKTIKUM STRUKTUR DATA\Modul4\output' & .\guided1
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\ACER\Downloads\PRAKTIKUM STRUKTUR DATA\Modul4\output>
```

Deskripsi program

Program yang diberikan merupakan implementasi dari struktur data linked list dengan menggunakan bahasa pemrograman C++. Program ini menghadirkan berbagai fungsi dasar untuk manipulasi linked list seperti penambahan elemen di depan, belakang, atau tengah, penghapusan elemen di depan, belakang, atau tengah, serta pengubahan nilai elemen di depan, belakang, atau tengah. Selain itu, terdapat fungsi-fungsi lain seperti inisialisasi, pengecekan kekosongan, menghitung jumlah elemen, dan menampilkan isi linked list. Dengan demikian, program ini memberikan kerangka dasar untuk memahami konsep dan implementasi dari struktur data linked list non-circular dalam bahasa pemrograman C++.

2. Guided 2

Surce 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() {
    if (head == NULL)
        return 1;
    else
        return 0;
}

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() == 1) {
        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() == 1) {
        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() == 1) {
        head = baru;
        tail = head;
        baru->next = head;
    } else {
        baru = new Node;
        baru->data = data;
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1 && bantu->next != head) {
            bantu = bantu->next;
            nomor++;
        }
        if (posisi == 1) {
            insertDepan(data);
        } else if (posisi <= hitungList()) {
            baru->next = bantu->next;
            bantu->next = baru;
        } else {
            cout << "Posisi diluar jangkauan" << endl;
        }
    }
}

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

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

void hapusTengah(int posisi) {
    if (isEmpty() == 0) {
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1 && bantu->next != head) {
            bantu = bantu->next;
            nomor++;
        }
        if (posisi == 1) {
            hapusDepan();
        } else if (posisi <= hitungList()) {
            hapus = bantu->next;
            bantu->next = hapus->next;
            delete hapus;
        }
    }
}

```

```

        } else {
            cout << "Posisi diluar jangkauan" << endl;
        }
    } 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;
}

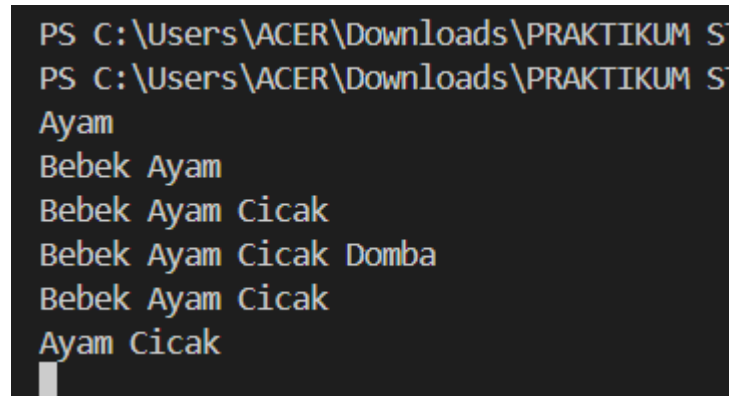
void tampil() {
    if (isEmpty() == 0) {
        tail = head;
        do {
            cout << tail->data << " ";
            tail = tail->next;
        } while (tail != head);
        cout << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

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;  
}
```

Screenshot Program



```
PS C:\Users\ACER\Downloads\PRAKTIKUM S  
PS C:\Users\ACER\Downloads\PRAKTIKUM S  
Ayam  
Bebek Ayam  
Bebek Ayam Cicak  
Bebek Ayam Cicak Domba  
Bebek Ayam Cicak  
Ayam Cicak
```

Deskripsi Program

Program yang diberikan merupakan implementasi dari struktur data linked list sirkular dalam bahasa pemrograman C++. Program ini menyediakan fungsi-fungsi dasar untuk manipulasi linked list, termasuk penambahan elemen di depan, belakang, atau tengah, penghapusan elemen di depan, belakang, atau tengah, serta menampilkan isi linked list. Struktur linked list yang sirkular memungkinkan tail untuk mengarah kembali ke head, menciptakan sirkuit tertutup. Dengan demikian, operasi penambahan atau penghapusan elemen di bagian depan atau belakang dapat dilakukan dengan memperbarui referensi head dan tail secara tepat, sementara penambahan atau penghapusan elemen di tengah memerlukan transversal melalui linked list. Program ini memberikan kerangka dasar untuk memahami konsep dan implementasi dari struktur data linked list sirkular dalam konteks pemrograman C++.

LATIHAN KELAS - UNGUIDED

1. Buatlah program menu Linked List Non Circular untuk menyimpan Nama dan NIM mahasiswa, dengan menggunakan input dari user.

Buatlah menu untuk menambahkan, mengubah, menghapus, dan melihat Nama dan NIM mahasiswa, berikut contoh tampilan output dari nomor 1:

Source code

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

class Node {
public:
    string name;
    string nim;
    Node* next;
};

class LinkedList {
public:
    LinkedList() {
        head = nullptr;
    }

    void tambahAwal(string name, string nim) {
        Node* node = new Node();
        node->name = name;
        node->nim = nim;
        node->next = head;
        head = node;
    }

    void tambahAkhir(string name, string nim) {
```

```

Node* node = new Node();
node->name = name;
node->nim = nim;
node->next = nullptr;

if (head == nullptr) {
    head = node;
}
else {
    Node* curr = head;
    while (curr->next != nullptr) {
        curr = curr->next;
    }
    curr->next = node;
}

}

void tambahTengah(string name, string nim, int pos) {
    Node* node = new Node();
    node->name = name;
    node->nim = nim;

    Node* curr = head;
    for (int i = 1; i < pos - 1; i++) {
        curr = curr->next;
    }
    node->next = curr->next;
    curr->next = node;
}

void ubahAwal(string name, string nim) {
    head->name = name;
    head->nim = nim;
}

```

```
void ubahAkhir(string name, string nim) {
    Node* curr = head;
    while (curr->next != nullptr) {
        curr = curr->next;
    }
    curr->name = name;
    curr->nim = nim;
}

void ubahTengah(string name, string nim, int pos) {
    Node* curr = head;
    for (int i = 1; i < pos; i++) {
        curr = curr->next;
    }
    curr->name = name;
    curr->nim = nim;
}

void hapusDepan() {
    Node* temp = head;
    head = head->next;
    delete temp;
}

void hapusAkhir() {
    Node* curr = head;
    while (curr->next->next != nullptr) {
        curr = curr->next;
    }
    Node* temp = curr->next;
    curr->next = nullptr;
    delete temp;
}
```

```

void hapusTengah(int pos) {
    Node* curr = head;
    for (int i = 1; i < pos - 1; i++) {
        curr = curr->next;
    }
    Node* temp = curr->next;
    curr->next = curr->next->next;
    delete temp;
}

void hapusList() {
    Node* curr = head;
    while (head != nullptr) {
        head = head->next;
        delete curr;
        curr = head;
    }
}

void tampilkan() {
    Node* curr = head;
    while (curr != nullptr) {
        cout << "Nama: " << curr->name << ", NIM: " << curr->nim << endl;
        curr = curr->next;
    }
}

private:
    Node* head;
};

void menu() {

```



```

        cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR" << 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 isi data" << endl;
        cout << "0. Keluar" << endl;
        cout << "      " << endl;
        cout << "Masukkan pilihan Anda: ";
    }

int main() {
    LinkedList list;
    int choice;
    do {
        menu();
        cin >> choice;

        switch (choice) {
            case 1: {
                string name;
                string nim;
                cout << "Masukkan nama dan NIM mahasiswa: ";
                cin >> name >> nim;
                list.tambahAwal(name, nim);
                break;
            }

            case 2: {

```

```

        string name;
        string nim;
        cout << "Masukkan nama dan NIM mahasiswa: ";
        cin >> name >> nim;
        list.tambahAkhir(name, nim);
        break;
    }
    case 3: {
        string name;
        string nim;
        int pos;
        cout << "Masukkan nama dan NIM mahasiswa: ";
        cin >> name >> nim;
        cout << "Masukkan posisi node: ";
        cin >> pos;
        list.tambahTengah(name, nim, pos);
        break;
    }
    case 4: {
        string name;
        string nim;
        cout << "Masukkan nama dan NIM mahasiswa baru: ";
        cin >> name >> nim;
        list.ubahAwal(name, nim);
        break;
    }
    case 5: {
        string name;
        string nim;
        cout << "Masukkan nama dan NIM mahasiswa baru: ";
        cin >> name >> nim;
        list.ubahAkhir(name, nim);
        break;
    }
}

```

```
case 6: {
    string name;
    string nim;
    int pos;
    cout << "Masukkan nama dan NIM mahasiswa baru: ";
    cin >> name >> nim;
    cout << "Masukkan posisi node: ";
    cin >> pos;
    list.ubahTengah(name, nim, pos);
    break;
}
case 7: {
    list.hapusDepan();
    break;
}
case 8: {
    list.hapusAkhir();
    break;
}
case 9: {
    int pos;
    cout << "Masukkan posisi node: ";
    cin >> pos;
    list.hapusTengah(pos);
    break;
}
case 10: {
    list.hapusList();
    break;
}
case 11: {
    list.tampilkan();
    break;
}
```

```

        case 0: {
            cout << "Terima kasih telah menggunakan program ini!"
<< endl;
            break;
        }
        default: {
            cout << "Pilihan tidak valid, silahkan coba lagi."
<< endl;
            break;
        }
    }

    cout << endl;
} while (choice != 0);
return 0;
}

```

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

1. Masukkan Data Sesuai Urutan

```

11. Tampilkan isi data
0. Keluar

Masukkan pilihan Anda: 11
Nama: Jawad, NIM: 23300001
Nama: Ilhan, NIM: 2311102029
Nama: Farrel, NIM: 23300003
Nama: Denis, NIM: 23300005
Nama: Anis, NIM: 23300008
Nama: Bowo, NIM: 23300015
Nama: Gahar, NIM: 23300040
Nama: Udin, NIM: 23300048
Nama: Ucok, NIM: 23300050
Nama: Budi, NIM: 23300099

```

2. Tambahkan data Wati 2330004 diantara Farrel dan Denis

```
Masukkan pilihan Anda: 3
Masukkan nama dan NIM mahasiswa: Wati 2330004
Masukkan posisi node: 4
```

```
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

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 isi data
0. Keluar

```
Masukkan pilihan Anda: 11
Nama: Jawad, NIM: 23300001
Nama: Ilhan, NIM: 2311102029
Nama: Farrel, NIM: 23300003
Nama: Wati, NIM: 2330004
Nama: Denis, NIM: 23300005
Nama: Anis, NIM: 23300008
Nama: Bowo, NIM: 23300015
Nama: Gahar, NIM: 23300040
Nama: Udin, NIM: 23300048
Nama: Ucok, NIM: 23300050
Nama: Budi, NIM: 23300099
```

3. Hapus Data Denis

Masukkan pilihan Anda: 9

Masukkan posisi node: 5

PROGRAM SINGLE LINKED LIST NON-CIRCULAR

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 isi data
0. Keluar

Masukkan pilihan Anda: 11

Nama: Jawad, NIM: 23300001

Nama: Ilhan, NIM: 2311102029

Nama: Farrel, NIM: 23300003

Nama: Wati, NIM: 23300004

Nama: Anis, NIM: 23300008

Nama: Bowo, NIM: 23300015

Nama: Gahar, NIM: 23300040

Nama: Udin, NIM: 23300048

Nama: Ucok, NIM: 23300050

Nama: Budi, NIM: 23300099

4. Tambahkan data Owi 2330000 di awal

```
Masukkan pilihan Anda: 1
Masukkan nama dan NIM mahasiswa: Owi 2330000
```

```
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

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 isi data
0. Keluar

```
Masukkan pilihan Anda: 11
Nama: Owi, NIM: 2330000
Nama: Jawad, NIM: 23300001
Nama: Ilhan, NIM: 2311102029
Nama: Farrel, NIM: 23300003
Nama: Wati, NIM: 23300004
Nama: Anis, NIM: 23300008
Nama: Bowo, NIM: 23300015
Nama: Gahar, NIM: 23300040
Nama: Udin, NIM: 23300048
Nama: Ucok, NIM: 23300050
Nama: Budi, NIM: 23300099
```

5. Tambahkan data David 23300100 di akhir

```
Masukkan pilihan Anda: 2
Masukkan nama dan NIM mahasiswa: David 23300100
```

```
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

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 isi data
0. Keluar

```
Masukkan pilihan Anda: 11
Nama: Owi, NIM: 2330000
Nama: Jawad, NIM: 23300001
Nama: Ilhan, NIM: 2311102029
Nama: Farrel, NIM: 23300003
Nama: Wati, NIM: 23300004
Nama: Anis, NIM: 23300008
Nama: Bowo, NIM: 23300015
Nama: Gahar, NIM: 23300040
Nama: Udin, NIM: 23300048
Nama: Ucok, NIM: 23300050
Nama: Budi, NIM: 23300099
Nama: David, NIM: 23300100
```


6. Ubah data Udin menjadi Idin 23300045

```
Masukkan pilihan Anda: 6
Masukkan nama dan NIM mahasiswa baru: Idin 23300045
Masukkan posisi node: 9
```

```
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

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 isi data
0. Keluar

```
Masukkan pilihan Anda: 11
Nama: Owi, NIM: 2330000
Nama: Jawad, NIM: 23300001
Nama: Ilhan, NIM: 2311102029
Nama: Farrel, NIM: 23300003
Nama: Wati, NIM: 2330004
Nama: Anis, NIM: 23300008
Nama: Bowo, NIM: 23300015
Nama: Gahar, NIM: 23300040
Nama: Idin, NIM: 23300045
Nama: Ucok, NIM: 23300050
Nama: Budi, NIM: 23300099
Nama: David, NIM: 23300100
```

7. Ubah data terakhir menjadi Lucy 23300101

```
Masukkan pilihan Anda: 5
Masukkan nama dan NIM mahasiswa baru: Lucy 23300101
```

```
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

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 isi data
0. Keluar

```
Masukkan pilihan Anda: 11
Nama: Owi, NIM: 23300000
Nama: Jawad, NIM: 23300001
Nama: Ilhan, NIM: 2311102029
Nama: Farrel, NIM: 23300003
Nama: Wati, NIM: 23300004
Nama: Anis, NIM: 23300008
Nama: Bowo, NIM: 23300015
Nama: Gahar, NIM: 23300040
Nama: Idin, NIM: 23300045
Nama: Ucok, NIM: 23300050
Nama: Budi, NIM: 23300099
Nama: Lucy, NIM: 23300101
```

8. Hapus data awal

Masukkan pilihan Anda: 7

PROGRAM SINGLE LINKED LIST NON-CIRCULAR

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 isi data
0. Keluar

Masukkan pilihan Anda: 11

Nama: Jawad, NIM: 23300001
Nama: Ilhan, NIM: 2311102029
Nama: Farrel, NIM: 23300003
Nama: Wati, NIM: 23300004
Nama: Anis, NIM: 23300008
Nama: Bowo, NIM: 23300015
Nama: Gahar, NIM: 23300040
Nama: Idin, NIM: 23300045
Nama: Ucok, NIM: 23300050
Nama: Budi, NIM: 23300099
Nama: Lucy, NIM: 23300101

9. Ubah data awal menjadi Bagus 2330002

```
Masukkan pilihan Anda: 4
Masukkan nama dan NIM mahasiswa baru: Bagus 2330002
```

```
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

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 isi data
0. Keluar

```
Masukkan pilihan Anda: 11
Nama: Bagus, NIM: 2330002
Nama: Ilhan, NIM: 2311102029
Nama: Farrel, NIM: 23300003
Nama: Wati, NIM: 23300004
Nama: Anis, NIM: 23300008
Nama: Bowo, NIM: 23300015
Nama: Gahar, NIM: 23300040
Nama: Idin, NIM: 23300045
Nama: Ucok, NIM: 23300050
Nama: Budi, NIM: 23300099
Nama: Lucy, NIM: 23300101
```

10. Hapus data akhir

Masukkan pilihan Anda: 8

PROGRAM SINGLE LINKED LIST NON-CIRCULAR

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 isi data
0. Keluar

Masukkan pilihan Anda: 11

Nama: Bagus, NIM: 2330002

Nama: Ilhan, NIM: 2311102029

Nama: Farrel, NIM: 23300003

Nama: Wati, NIM: 2330004

Nama: Anis, NIM: 23300008

Nama: Bowo, NIM: 23300015

Nama: Gahar, NIM: 23300040

Nama: Idin, NIM: 23300045

Nama: Ucok, NIM: 23300050

Nama: Budi, NIM: 23300099

11. Tampilkan Seluruh data

```
Masukkan pilihan Anda: 11  
Nama: Bagus, NIM: 2330002  
Nama: Ilhan, NIM: 2311102029  
Nama: Farrel, NIM: 23300003  
Nama: Wati, NIM: 2330004  
Nama: Anis, NIM: 23300008  
Nama: Bowo, NIM: 23300015  
Nama: Gahar, NIM: 23300040  
Nama: Idin, NIM: 23300045  
Nama: Ucok, NIM: 23300050  
Nama: Budi, NIM: 23300099
```

Deskripsi Program

Program ini adalah implementasi dari struktur data linked list non-circular dalam bahasa pemrograman C++. Program memungkinkan pengguna untuk melakukan sejumlah operasi pada linked list, termasuk penambahan elemen di awal, akhir, atau posisi tertentu, penghapusan elemen di awal, akhir, atau posisi tertentu, serta pengubahan nilai elemen di awal, akhir, atau posisi tertentu. Setiap operasi diimplementasikan melalui metode yang sesuai dalam kelas LinkedList, dan pengguna dapat memilih operasi yang diinginkan melalui menu yang ditampilkan secara berulang. Program ini memberikan fleksibilitas dalam manipulasi data secara dinamis melalui linked list.

BAB IV

KESIMPULAN

Modul linked list non-circular merupakan struktur data yang memungkinkan penyimpanan dan manipulasi data secara dinamis melalui node-node yang saling terhubung. Setiap node memiliki dua bagian utama: data dan pointer yang menunjukkan ke node selanjutnya dalam urutan. Keunggulan dari modul ini adalah kemampuannya untuk menyimpan data dengan ukuran dinamis tanpa memerlukan alokasi memori statis, serta kemudahan dalam melakukan operasi penghapusan, penambahan, dan pencarian data. Meskipun demikian, perlu diingat bahwa penggunaan linked list non-circular juga memiliki kelemahan, seperti overhead memori yang diperlukan untuk menyimpan pointer tambahan dan kompleksitas operasi pencarian data yang lebih tinggi dibandingkan dengan struktur data lainnya seperti array.

BAB V

REFERENSI

- <https://taufikkipo.blogspot.com/2012/07/single-linked-list-non-circular.html>
- <https://www.studocu.com/id/document/institut-teknologi-sepuluh-nopember/algoritma-pemrograman-komputer-algorithm-and-computer-programming/apa-itu-single-linked-list-non-circular-dan-contoh-program/45206382>
- <https://www.geeksforgeeks.org/types-of-linked-list/>