`LAPORAN PRAKTIKUM

MODUL IV
LINKED LIST CIRCULAR DAN NON CIRCULAR

Disusun oleh:
Brian Farrel Evandhika
NIM: 2311102037

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

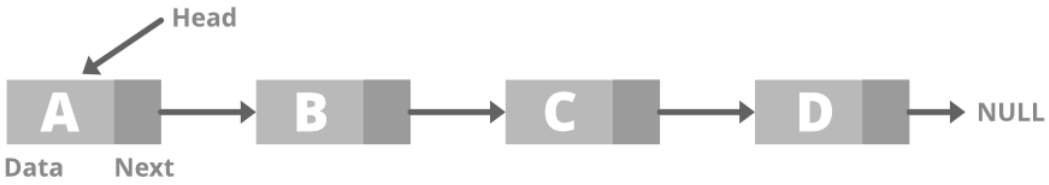
A. TUJUAN PRAKTIKUM

a. Praktikan dapat mengetahui dan memahami linked list circular dan non circular.
b. Praktikan dapat membuat linked list circular dan non circular.
c. Praktikan dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat.

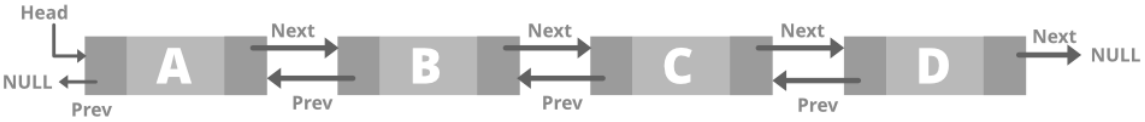
B. DASAR TEORI

Linked list adalah struktur data yang berisi kumpulan elemen data dimana (node) tiap elemen saling berkaitan atau dihubungkan dengan elemen lain melalui suatu pointer. Pointer adalah alamat elemen data yang tersimpan di mermori.Ada dua jenis pointer yaitu head dan tail. Head adalah simpul pertama yang digunakan sebagai awal akses ke seluruh data dalam linked list. Tail adalah simpul terakhir yang digunakan sebagai penanda akhir urutan simpul. Apabila linked list berisi elemen kosong, maka pointer dari head akan menunjuk ke NULL, begitu juga untuk pointer berikutnya dari tail.Berbeda dengan array, linked list memiliki ukuran elemen yang dapat berubah secara dinamis dan mudah dalam menyisipkan dan menghapus elemen. Selain itu, pada linked list penyimpanan memorinya tidak harus berurutan dan berdekatan.

Single linked list Non Circular adalah jenis linked list yang hanya memiliki 1 pointer saja. Pointer digunakan untuk menunjuk node selanjutnya (next), kecuali pada node tail atau node terakhir yang pointernya menunjuk ke NULL. Single Linked List



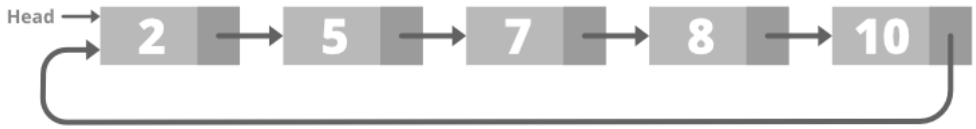
Double Linked List Non Circular adalah jenis linked list yang memiliki 2 pointer. 1 pointer menunjuk ke node selanjutnya (next) dan 1 lagi menunjuk ke node sebelumnya (prev). Pada node head, pointer prev akan bernilai NULL karena node Head adalah node pertama. Pada node Tail, pointer next akan menunjuk ke NULL. Double Linked List



Single : artinya field pointer-nya hanya satu buah saja dan satu arah.

Circular : artinya pointer next-nya akan menunjuk pada dirinya sendiri sehingga berputar.

Jadi, Single Circular Linked List adalah linked list yang pointer next-nya menunjuk ke dirinya sendiri, jadi node tail (node terakhir) akan menunjuk ke node head (node pertama).

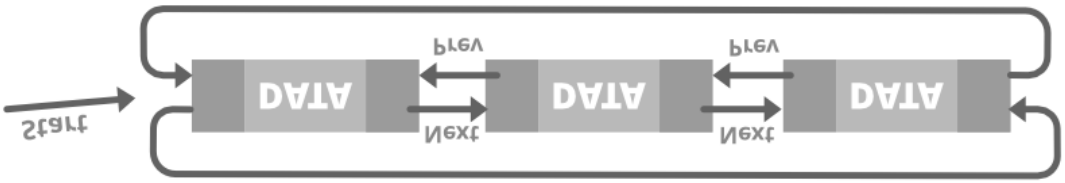


Double Circular Linked List

Double: artinya field pointer-nya terdiri dari dua buah dan dua arah, yaitu prev dan next

Circular : artinya pointer next-nya akan menunjuk pada dirinya sendiri sehingga berputar.

Jadi, Double Circular Linked List adalah linked list yang pointer next dan pointer prev-nya menunjuk ke dirinya sendiri secara circular.



Guided

Guided 1

```cpp
#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;
}

// Pengecekkan 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;
    }
```

Screenshot Program

```
PS C:\Users\MSI GAMING\Documents\Kuliah\SEME
STER 2\PRAKTIKUM STRUKTUR DATA DAN ALGORITME
\Praktikum\2311102037_Brian-Farrel-Evandhika
_IF-11-A_Praktikum-Struktur-Data>  & 'c:\Use
rs\MSI GAMING\.vscode\extensions\ms-vscode.c
pptools-1.19.9-win32-x64\debugAdapters\bin\W
indowsDebugLauncher.exe' '--stdin=Microsoft-
MIEngine-In-c5zskep5.auw' '--stdout=Microsof
t-MIEngine-Out-pyxispgz.tbr' '--stderr=Micro
soft-MIEngine-Error-rpjqikug.ris' '--pid=Mic
rosoft-MIEngine-Pid-mo1pdhot.biq' '--dbgExe=
C:\msys64\ucrt64\bin\gdb.exe' '--interpreter
=mi'
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\MSI GAMING\Documents\Kuliah\SEME
STER 2\PRAKTIKUM STRUKTUR DATA DAN ALGORITME
\Praktikum\2311102037_Brian-Farrel-Evandhika
_IF-11-A_Praktikum-Struktur-Data>
```

Deskripsi Program

Program di atas adalah program single linked list non-circular yang menggunakan bahasa C++. Pertama kita deklarasikan struct Node yang berisi data dan pointer next. Kemudian kita inisialisasi head dan tail dengan NULL. Kemudian kita membuat fungsi-fungsi untuk menambahkan data di depan, belakang, dan tengah. Kemudian kita membuat fungsi untuk menghitung jumlah list, menghapus data di depan, belakang, dan tengah, mengubah data di depan, belakang, dan tengah, menghapus semua data, dan menampilkan data.

Guided 2

```cpp
#include <iostream>

using namespace std;

// Deklarasi Struct Node


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

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

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

//Pengecekan isi list
int isEmpty(){
    if (head == NULL){
        return 1; // true
    } else {
        return 0; // false
    }
}

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

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

//Tambah Depan
void insertDepan(string data){
    // Buat Node baru
    buatNode(data);

    if (isEmpty() == 1){
        head = baru;
        tail = head;
        baru->next = head;
    } else {
        while (tail->next != head){
            tail = tail->next;
        }
        baru->next = head;
        head = baru;
        tail->next = head;
    }
}

//Tambah Belakang
void insertBelakang(string data){
    // Buat Node baru
    buatNode(data);

    if (isEmpty() == 1){
        head = baru;
        tail = head;
        baru->next = head;
    } else {
        while (tail->next != head){
            tail = tail->next;
        }
        tail->next = baru;
        baru->next = head;
    }
}
```

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

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

//Hapus Belakang
void hapusBelakang(){
    if (isEmpty() == 0){
        hapus = head;
```

```
        tail = head;
        if (hapus->next == head){
            head = NULL;
            tail = NULL;
            delete hapus;
        } else {
            while (hapus->next != head){
                hapus = hapus->next;
            }
            while (tail->next != hapus){
                tail = tail->next;
            }
            tail->next = head;
            hapus->next = NULL;
            delete hapus;
        }
    } else {
        cout << "List masih kosong!" << endl;
    }
}

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

//Hapus List
void clearList(){
    if (head != NULL){
        hapus = head->next;
        while (hapus != head){
            bantu = hapus->next;
            delete hapus;
            hapus = bantu;
        }
```

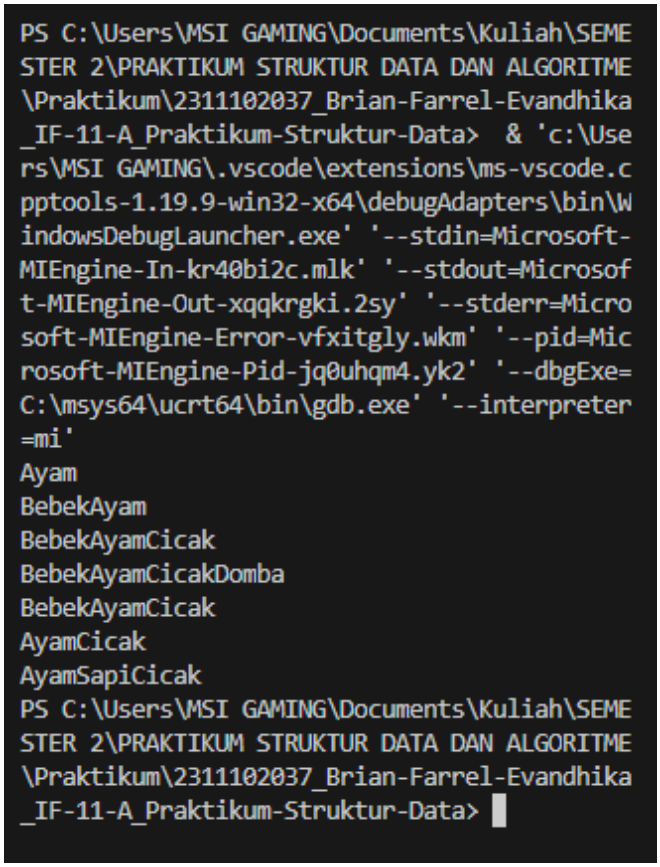
```
        delete head;
        head = NULL;
    }
    cout << "List berhasil terhapus!" << endl;
}

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

int main(){
    init();
    insertDepan("Ayam");
    tampil();
    insertDepan("Bebek");
    tampil();
    insertBelakang("Cicak");
    tampil();
    insertBelakang("Domba");
    tampil();
    hapusBelakang();
    tampil();
    hapusDepan();
    tampil();
    insertTengah("Sapi", 2);
    tampil();
    hapusTengah(2);
    tampil();

    return 0;
}
```

Screenshot Program



Deskripsi Program

Program di atas adalah program list data dengan menggunakan single linked list circular yang di dalamnya menggunakan struct Node yang berisi string data dan pointer next. Pertama kita deklarasikan struct Node yang berisi string data dan pointer next. Lalu kita deklarasikan variabel head, tail, baru, bantu, dan hapus yang bertipe pointer Node. Lalu kita membuat fungsi menambahkan data di depan, belakang, dan tengah; menghapus data di depan, belakang, dan tengah; menghitung jumlah data; menampilkan data; dan menghapus semua data.

Unguided

Unguided 1

```cpp
#include <iostream>
#include <iomanip>

using namespace std;

struct Node
{
    string nama23;
    long long int nim23;
    Node *next;
};

Node *head;
Node *tail;

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

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

void insertDepan(string nama, long long int nim)
{
    Node *baru = new Node();
    baru->nama23 = nama;
    baru->nim23 = nim;
    baru->next = NULL;
```

```
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}

void insertBelakang(string nama, long long int nim)
{
    Node *baru = new Node();
    baru->nama23 = nama;
    baru->nim23 = nim;
    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(string nama, long long int nim, 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 = new Node();
        baru->nama23 = nama;
        baru->nim23 = nim;

        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 << "Linked list masih 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 << "Linked list masih 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 *hapus, *bantu, *sebelum;
        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;
    }
}

void ubahDepan(string nama, long long int nim)
{
    if (!isEmpty())
    {
        head->nama23 = nama;
        head->nim23 = nim;
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

void ubahTengah(string nama, long long int nim, 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->nama23 = nama;
            bantu->nim23 = nim;
        }
    }
    else
    {
```

```
        cout << "Linked list masih kosong" << endl;
    }
}

void ubahBelakang(string nama, long long int nim)
{
    if (!isEmpty())
    {
        tail->nama23 = nama;
        tail->nim23 = nim;
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

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

void tampilList()
{
    Node *bantu = head;
    cout << "============ TAMPILKAN LIST ============" << endl;
    cout << "----------------------------------------" << endl;
    cout << left << setw(20) << "| Nama"
         << "| NIM" << endl;
    cout << "----------------------------------------" << endl;
    while (bantu != NULL)
    {
        cout << left << setw(20) << "| " + bantu->nama23 << "| " << bantu-
>nim23 << endl;
        bantu = bantu->next;
    }
    cout << "----------------------------------------" << endl;
}

int main()
```

```
{
    string nama;
    long long int nim;
    int posisi;
    init();

    cout << "========== PROGRAM SINGLE LINKED LIST NON-CIRCULAR ==========" <<
endl;
    while (true)
    {
        cout << "1.  Tambah Depan" << endl;
        cout << "2.  Tambah Belakang" << endl;
        cout << "3.  Tambah Tengah" << endl;
        cout << "4.  Ubah Depan" << endl;
        cout << "5.  Ubah Belakang" << endl;
        cout << "6.  Ubah Tengah" << endl;
        cout << "7.  Hapus Depan" << endl;
        cout << "8.  Hapus Belakang" << endl;
        cout << "9.  Hapus Tengah" << endl;
        cout << "10. Hapus List" << endl;
        cout << "11. Tampilkan" << endl;
        cout << "0. Keluar" << endl;
        int pilihan;
        cout << "Masukkan pilihan: ";
        cin >> pilihan;

        switch (pilihan)
        {
        case 1:
            cout << "========== TAMBAH DEPAN ==========" << endl;
            cout << "Masukkan Nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            insertDepan(nama, nim);
            cout << "Data [" << nama << "] berhasil ditambahkan di depan! " <<
endl;
            break;
        case 2:
            cout << "========== TAMBAH BELAKANG ==========" << endl;
            cout << "Masukkan Nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            insertBelakang(nama, nim);
            cout << "Data [" << nama << "] berhasil ditambahkan di belakang! "
<< endl;
            break;
```

```
        case 3:
            cout << "========== TAMBAH TENGAH ==========" << endl;
            cout << "Masukkan posisi: ";
            cin >> posisi;
            cout << "Masukkan nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            insertTengah(nama, nim, posisi);
            cout << "Data [" << nama << "] berhasil ditambahkan di posisi ke-"
<< posisi << "!" << endl;
            break;
        case 4:
            cout << "========== UBAH DEPAN ==========" << endl;
            cout << "Masukkan nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            ubahDepan(nama, nim);
            cout << "Data depan berhasil diubah! " << endl;
            break;
        case 5:
            cout << "========== UBAH BELAKANG ==========" << endl;
            cout << "Masukkan nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            ubahBelakang(nama, nim);
            cout << "Data belakang berhasil diubah! " << endl;
            break;
        case 6:
            cout << "========== UBAH TENGAH ==========" << endl;
            cout << "Masukkan posisi: ";
            cin >> posisi;
            cout << "Masukkan nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            ubahTengah(nama, nim, posisi);
            cout << "Data di posisi ke-" << posisi << " berhasil diubah! " <<
endl;
            break;
        case 7:
            cout << "========== HAPUS DEPAN ==========" << endl;
            hapusDepan();
            cout << "Data depan berhasil dihapus! " << endl;
            break;
        case 8:
```

```
            cout << "========== HAPUS BELAKANG ==========" << endl;
            hapusBelakang();
            cout << "Data belakang berhasil dihapus! " << endl;
            break;
        case 9:
            cout << "========== HAPUS TENGAH ==========" << endl;
            cout << "Masukkan posisi: ";
            cin >> posisi;
            hapusTengah(posisi);
            cout << "Data di posisi ke-" << posisi << " berhasil dihapus! " <<
endl;
            break;
        case 10:
            cout << "========== HAPUS LIST ==========" << endl;
            clearList();
            cout << "Semua data berhasil dihapus! " << endl;
            break;
        case 11:
            cout << "========== TAMPILKAN LIST ==========" << endl;
            tampilList();
            break;
        case 0:
            cout << "Program selesai!" << endl;
            return 0;
        default:
            cout << "Pilihan tidak tersedia!" << endl;
            break;
        }
    }
    return 0;
}
```

Screenshot Program

```
6.  Ubah Tengah
7.  Hapus Depan
8.  Hapus Belakang
9.  Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar
Masukkan pilihan: 1
========== TAMBAH DEPAN ==========
Masukkan Nama: Bagas 2330002
Masukkan NIM: Data [Bagas] berhasil ditambahkan di depan!
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
0. Keluar
Masukkan pilihan: 11
========== TAMPILKAN LIST ==========
============ TAMPILKAN LIST ============
---------------------------------------
| Nama              | NIM
---------------------------------------
| Bagas             | 2330002
| Brian             | 2311102037
| Farrel            | 23300003
| Wati              | 2330004
| Denis             | 23300005
| Anis              | 23300008
| Bowo              | 23300015
| Gahar             | 23300040
| Idin              | 23300045
| Ucok              | 23300050
| Budi              | 23300099
---------------------------------------
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
0. Keluar
Masukkan pilihan:
```

Deskripsi Program

program di atas merupakan program list nama dan nim Mahasiswa yang menggunakan single linked list non-circular dengan menggunakan struct Node yang berisi string nama23, long long int nim, dan Node *next. Alasan saya menggunakan long long int untuk nim adalah karena nim memiliki 10 digit angka dimana long memiliki ukuran 4 byte dan int memiliki ukuran 2 byte. Program ini memiliki beberapa fungsi yaitu:

insertDepan(string nama, long long int nim) : Menambahkan data di depan linked list

insertBelakang(string nama, long long int nim) : Menambahkan data di belakang linked list

insertTengah(string nama, long long int nim, int posisi) : Menambahkan data di tengah linked list

hapusDepan() : Menghapus data di depan linked list

hapusBelakang() : Menghapus data di belakang linked list

hapusTengah(int posisi) : Menghapus data di tengah linked list

ubahDepan(string nama, long long int nim) : Mengubah data di depan linked list

ubahBelakang(string nama, long long int nim) : Mengubah data di belakang linked list

ubahTengah(string nama, long long int nim, int posisi) : Mengubah data di tengah linked list

clearList() : Menghapus semua data di linked list

tampilList() : Menampilkan data di linked list

Kesimpulan

Linked List adalah struktur data yang menyimpan elemen (node) secara berurutan dan terhubung melalui pointer yang disebut head dan tail. Berbeda dengan array yang memiliki batas data statis, linked list memiliki batas data yang dinamis. Ada beberapa jenis linked list, termasuk Single Linked List, Double Linked List, Single Circular Linked List, dan Double Circular Linked List.

Circular Single Linked List adalah jenis Single Linked List di mana pointer pada node terakhir menunjuk kembali ke node pertama. Dengan kata lain, linked list ini membentuk lingkaran, di mana pointer next pada node terakhir mengarah ke node pertama. Konsep ini dapat diuraikan dari dua kata, yaitu "single" yang berarti setiap node memiliki satu pointer dan arah yang sama, dan "circular" yang berarti pointer next dari node terakhir mengarah kembali ke node pertama, menciptakan lingkaran.

Perbedaan utama antara Circular Single Linked List dan Single Linked List adalah bahwa pada Circular Single Linked List, tail menunjuk kembali ke head, sehingga tidak ada pointer yang menunjuk ke NULL. Sementara pada Single Linked List biasa, tail menunjuk ke NULL setelah node terakhirnya.

Daftar Pustaka

[1] Abu Sara, M. R., Klaib, M. F. J., & Hasan, M. (2021). Hybrid Array List: An Efficient Dynamic Array with Linked List Structure. Indonesia Journal on Computing (Indo-JC), 5(3), 47-62.

[2] Anita Sindar, R. M. S. (2019). Struktur Data Dan Algoritma Dengan C++ (Vol. 1). CV. AA. RIZKY.