

**LAPORAN PRAKTIKUM  
STRUKTUR DATA DAN  
ALGORITMA**

**MODUL IV  
LINKED LIST CIRCULAR DAN  
NON CIRCULAR**



**Disusun Oleh :**

Muhammad Rusdiyanto Asatman  
2311102053

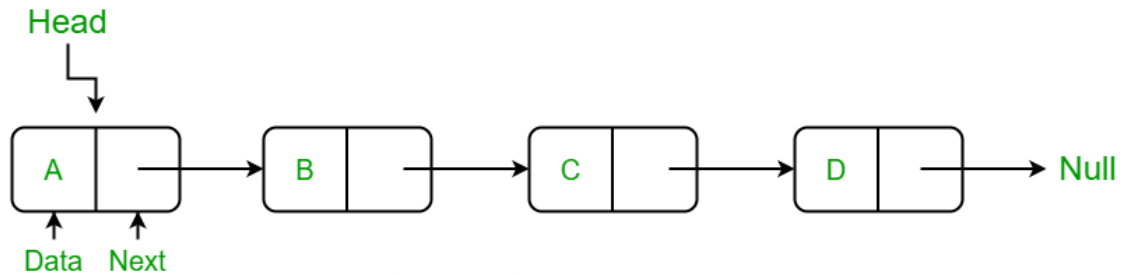
**Dosen :**

Wahyu Andi Saputra, S.Pd., M.Eng.

**PROGRAM STUDI S1 TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

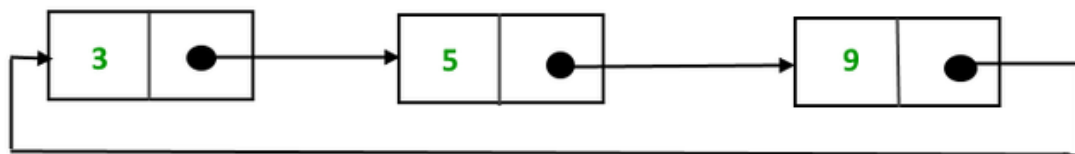
## A. Dasar Teori

### 1. Linked List Non Circular



Linked list adalah struktur data yang terdiri dari serangkaian elemen yang disusun dalam urutan linier. Dalam linked list non circular, setiap elemen (node) terdiri dari dua bagian utama: data dan referensi (pointer) ke node berikutnya dalam urutan. Pointer terakhir menunjuk ke null, menandakan akhir dari linked list. Linked list non circular memiliki node terakhir yang menunjuk ke null, sehingga tidak membentuk lingkaran tertutup. Dalam kata lain, looping melalui linked list non circular akan berhenti ketika mencapai node terakhir yang menunjuk ke null.

### 2. Linked List Circular



Linked list circular adalah struktur data yang mirip dengan linked list non circular, namun dengan perbedaan bahwa node terakhir dalam linked list circular tidak menunjuk ke null, melainkan kembali ke node pertama, membentuk suatu lingkaran tertutup. Artinya, looping melalui linked list circular akan terus berlanjut secara tak terbatas kecuali dihentikan secara eksplisit.

## B. Guided

### Guided 1

```
#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

```

PS D:\Codes\strukdat\4> & 'c:\Users\Roesdi\.vscode\extensions\ms-vscode.cpptools-1.19.9-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-dm2mxzru.ned' '--stdout=Microsoft-MIEngine-Out-i2xg0opi.egs' '--stderr=Microsoft-MIEngine-Error-rsq0v4z4.00m' '--pid=Microsoft-MIEngine-Pid-cdfrzsiv.wnj' '--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 D:\Codes\strukdat\4>

```

### Deskripsi:

Program di atas adalah program yang mendemonstrasikan penggunaan dari linked list, lebih tepatnya single linked list non circular. Setiap node dalam linked list tersebut menyimpan 2 informasi berupa data angka (integer) dan alamat memori node berikutnya. Selain itu, program ini memiliki 14 fungsi yang terdiri dari :

1. *init* : merupakan fungsi yang pertama kali dipanggil dalam program untuk menginisialisasi nilai pointer head dan tail menjadi NULL. Hal ini dilakukan untuk menghindari error.
2. *isEmpty* : fungsi yang berguna untuk mengecek apakah list kosong dengan melihat nilai pointer head. Jika masih NULL, maka list masih kosong dan akan dikembalikan true. Jika tidak NULL, maka list sudah memiliki data dan akan dikembalikan false.
3. *insertDepan* : fungsi yang digunakan untuk menambahkan node/data ke posisi depan linked list, sehingga data yang dimasukkan akan menjadi head baru, dan head yang lama akan diletakkan setelah data tadi.
4. *insertBelakang* : fungsi ini memiliki kurang lebih memiliki cara kerja yang sama dengan *insertDepan*. Hanya saja, fungsi ini melakukan penambahan data ke tail dari linked list.



5. *hitungList* : menghitung jumlah/banyaknya node yang ada dalam linked list dengan cara looping. Loop akan berhenti jika node sekarang sudah bernilai null atau dalam kata lain, pointer sudah mencapai akhir dari list (malah kelewatan sebenarnya).
6. *insertTengah* : digunakan untuk memasukkan nilai di tengah list atau dalam kata lain di antara 2 node dalam list. Cara kerjanya adalah fungsi ini akan looping terus hingga sampai sebelum posisi yang diinputkan (misal, jika input 5, maka berhenti di 4, jika input 2, maka berhenti di 1, dsb.). Setelah itu, data yang diinputkan baru akan ditambahkan.
7. *hapusDepan* : fungsi ini digunakan untuk menghapus node depan atau head dari linked list. Setelah dihapus, node selanjutnya akan menjadi head (jika tidak NULL). Jika NULL, maka tail dan head akan menjadi NULL.
8. *hapusBelakang* : kurang lebih memiliki cara kerja yang sama dengan *hapusDepan* tetapi digunakan untuk mengubah tail dari list.
9. *hapusTengah* : menghapus node dalam posisi tertentu dengan cara looping ke posisi tersebut dan mengubah pointer next dari node sebelumnya ke node berikutnya. Setelah itu, node baru bisa dihapus.
10. *ubahDepan* : mengubah data di node head.
11. *ubahTengah* : mengubah data di tengah list sesuai dengan posisi yang diinputkan.
12. *ubahBelakang* : mengubah data di node tail.
13. *clearList* : menghapus semua data linked list.
14. *tampil* : menampilkan semua data dari list.

Program tersebut tidak menerima input dari pengguna, melainkan nilai/data yang digunakan sudah disertakan dalam script/code program.

## Guided 2

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

```

PS D:\Codes\strukdat\4> & 'c:\Users\Roesdi\.vscode\extensions\ms-vscode.cpptools-1.19.9-win32-x64\debugAdapte
rs\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-ka05b4xt.vd5' '--stdout=Microsoft-MIEngine-Out
-5mk0ltqv.iug' '--stderr=Microsoft-MIEngine-Error-1yqjlbzw.kfn' '--pid=Microsoft-MIEngine-Pid-5ut1lptu.vgq' '-
dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Ayam
Bebek Ayam
Bebek Ayam Cicak
Bebek Ayam Cicak Domba
Bebek Ayam Cicak
Ayam Cicak
Ayam Sapi Cicak
Ayam Cicak
PS D:\Codes\strukdat\4>

```

### Deskripsi:

Program tersebut adalah program yang mendemonstrasikan penggunaan dari circular linked list. Berbeda dari program sebelumnya, program ini hanya memiliki 12 fungsi, sehingga terdapat beberapa fitur yang tidak tersedia seperti ubah (update) data. Di samping itu, ada juga fungsi yang berbeda dari sebelumnya seperti fungsi *buatNode* yang seperti namanya, digunakan untuk membuat node baru. Salah satu perbedaan yang dapat dilihat dalam kode program ini dengan kode program Guided 1 adalah ketika looping, program mengecek apakah data sekarang (pointer) tidak sama dengan head. Ketika tidak sama, maka looping akan berjalan terus, hal ini menandakan bahwa pointer belum “mengunjungi” semua node dalam list. Ketika sama, maka looping akan berhenti, karena pointer sudah mengunjungi semua node dan kembali ke titik awal (karena node tail selalu menunjuk ke head).

## C. Unguided

### Unguided 1

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

struct Node {
    string nama;
    string nim;
    Node *next;
};

Node *head;
Node *tail;

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

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

void insertDepan(string nama, string nim) {
    Node *baru = new Node;
    baru->nama = nama;
    baru->nim = nim;
    baru->next = NULL;
    if (isEmpty()) {
        head = tail = baru;
    } else {
        baru->next = head;
        head = baru;
    }
    cout << "\nData telah ditambahkan\n";
}

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

```

        baru->next = NULL;
        if (isEmpty()) {
            head = tail = baru;
        } else {
            tail->next = baru;
            tail = baru;
        }
        cout << "\nData telah ditambahkan\n";
    }

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

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 = new Node();
        baru->nama = nama;
        baru->nim = 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;

```



```

        string namaLama = head->nama;
        if (head->next != NULL) {
            head = head->next;
        } else {
            head = tail = NULL;
        }
        delete hapus;
        cout << "\nData " << namaLama << " berhasil dihapus\n";
    } else {
        cout << "List kosong!" << endl;
    }
}

void hapusBelakang() {
    if (!isEmpty()) {
        Node *hapus = tail;
        string namaLama = tail->nama;
        if (head != tail) {
            Node *bantu = head;
            while (bantu->next != tail) {
                bantu = bantu->next;
            }
            tail = bantu;
            tail->next = NULL;
        } else {
            head = tail = NULL;
        }
        delete hapus;
        cout << "\nData " << namaLama << " berhasil dihapus\n";
    } else {
        cout << "List kosong!" << endl;
    }
}

void hapusTengah(string nama, string nim) {
    if (!isEmpty()) {
        Node *bantu = head;
        Node *hapus;
        Node *sebelum = NULL;
        while (bantu != NULL) {
            if (bantu->nama == nama && bantu->nim == nim) {
                hapus = bantu;
                if (sebelum != NULL) {

```

```

        sebelum->next = bantu->next;
    } else {
        head = bantu->next;
    }
    delete hapus;
    cout << "\nData " << nama << " berhasil
dihapus\n";

    return;
}
sebelum = bantu;
bantu = bantu->next;
}
cout << "\nData tidak ditemukan\n";
}
else {
    cout << "List kosong!\n";
}
}

void ubahDepan(string nama, string nim) {
    if (!isEmpty()) {
        string namaLama = head->nama;
        head->nama = nama;
        head->nim = nim;
        cout << "\nData " << namaLama << " telah diganti dengan
data " << nama << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

void ubahTengah(string nama, string nim, string namaLama, string
nimLama) {
    if (!isEmpty()) {
        Node *bantu = head;
        while (bantu != NULL) {
            if (bantu->nama == namaLama && bantu->nim == nimLama)
{
                bantu->nama = nama;
                bantu->nim = nim;
                cout << "\nData " << namaLama << " telah diganti
dengan data " << nama << endl;
            }
        }
    }
}

```

```

        return;
    }
    bantu = bantu->next;
}
cout << "\nData tidak ditemukan\n";
} else {
    cout << "List masih kosong!" << endl;
}
}

void ubahBelakang(string nama, string nim) {
    if (!isEmpty()) {
        string namaLama = tail->nama;
        tail->nama = nama;
        tail->nim = nim;
        cout << "\nData " << namaLama << " telah diganti dengan\nData " << nama << endl;
    } 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;
    cout << "\nData MAHASISWA\n\n" << setw(32) << left << "NAMA"
    << setw(32) << left << "NIM";
    cout << endl;
    if (!isEmpty()) {
        while (bantu != NULL) {
            cout << setw(32) << left << bantu->nama << setw(32) <<
            left << bantu->nim;

```

```

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

string inputNama(bool baru = true) {
    string nama;
    if (baru) {
        cout << "Masukkan nama : ";
    }
    else {
        cout << "Masukkan nama lama : ";
    }
    cin >> nama;
    return nama;
}

string inputNIM(bool baru = true) {
    string nim;
    if (baru) {
        cout << "Masukkan NIM : ";
    }
    else {
        cout << "Masukkan NIM lama : ";
    }
    cin >> nim;
    return nim;
}

int inputPosisi() {
    int posisi;
    cout << "Masukkan posisi : ";
    cin >> posisi;
    return posisi;
}

int main() {
    init();
    int pilih;
    while (true) {

```

```

cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR\n\n";
cout << "1. Tambah Depan\n";
cout << "2. Tambah Belakang\n";
cout << "3. Tambah Tengah\n";
cout << "4. Ubah Depan\n";
cout << "5. Ubah Belakang\n";
cout << "6. Ubah Tengah\n";
cout << "7. Hapus Depan\n";
cout << "8. Hapus Belakang\n";
cout << "9. Hapus Tengah\n";
cout << "10. Hapus List\n";
cout << "11. TAMPILKAN\n";
cout << "0. Keluar\n\n";
cout << "Pilih Operasi : ";
cin >> pilih;

switch (pilih) {
    case 0: {
        return 0;
    }
    case 1: {
        cout << "\n- Tambah Depan\n\n";
        string nama = inputNama();
        string nim = inputNIM();
        insertDepan(nama, nim);
        break;
    }
    case 2: {
        cout << "\n- Tambah Belakang\n\n";
        string nama = inputNama();
        string nim = inputNIM();
        insertBelakang(nama, nim);
        break;
    }
    case 3: {
        cout << "\n- Tambah Tengah\n\n";
        string nama = inputNama();
        string nim = inputNIM();
        int posisi = inputPosisi();
        insertTengah(nama, nim, posisi);
        break;
    }
    case 4: {

```

```

        cout << "\n- Ubah Depan\n\n";
        string nama = inputNama();
        string nim = inputNIM();
        ubahDepan(nama, nim);
        break;
    }
    case 5: {
        cout << "\n- Ubah Depan\n\n";
        string nama = inputNama();
        string nim = inputNIM();
        ubahBelakang(nama, nim);
        break;
    }
    case 6: {
        cout << "\n- Ubah Tengah\n\n";
        string namaLama = inputNama(false);
        string nimLama = inputNIM(false);
        string nama = inputNama();
        string nim = inputNIM();
        ubahTengah(nama, nim, namaLama, nimLama);
        break;
    }
    case 7: {
        cout << "\n- Hapus Depan\n\n";
        hapusDepan();
        break;
    }
    case 8: {
        cout << "\n- Hapus Belakang\n\n";
        hapusBelakang();
        break;
    }
    case 9: {
        cout << "\n- Hapus Tengah\n\n";
        string nama = inputNama();
        string nim = inputNIM();
        hapusTengah(nama, nim);
        break;
    }
    case 10: {
        cout << "\n- Hapus List\n\n";
        clearList();
        break;
    }

```

```

    }
    case 11: {
        tampil();
        break;
    }
    default: cout << "Pilihan tidak valid\n\n";
}
}
return 0;
}

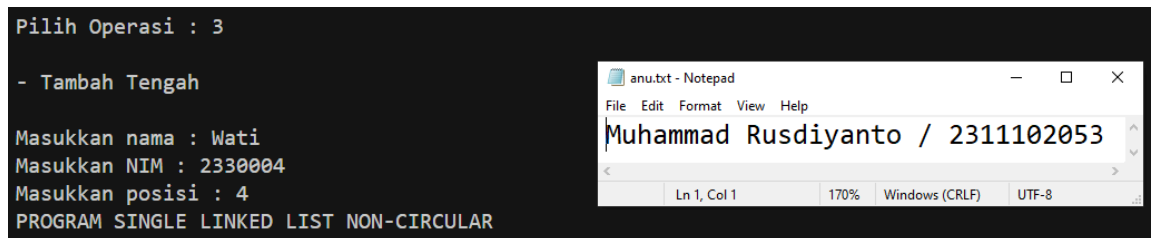
```

## Screenshots Output

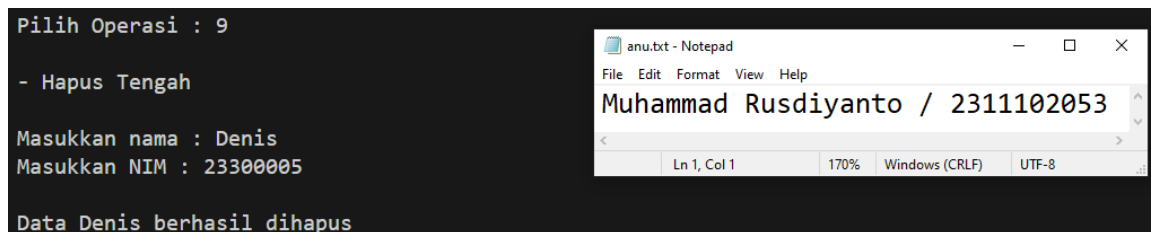
### - Menu



### - Tambah Wati



### - Hapus Denis



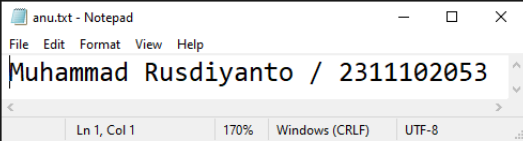
### - Tambah Owi

```
Pilih Operasi : 1

- Tambah Depan

Masukkan nama : Owi
Masukkan NIM : 2330000

Data telah ditambahkan
```

A screenshot of a Notepad window titled 'anu.txt - Notepad'. The text inside the Notepad is 'Muhammad Rusdiyanto / 2311102053'. The status bar at the bottom shows 'Ln 1, Col 1', '170%', 'Windows (CRLF)', and 'UTF-8'.

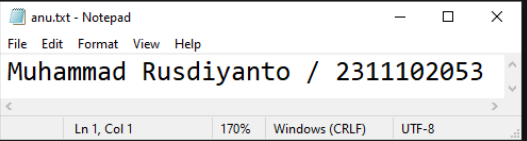
- Tambah David

```
Pilih Operasi : 2

- Tambah Belakang

Masukkan nama : David
Masukkan NIM : 23300100

Data telah ditambahkan
```

A screenshot of a Notepad window titled 'anu.txt - Notepad'. The text inside the Notepad is 'Muhammad Rusdiyanto / 2311102053'. The status bar at the bottom shows 'Ln 1, Col 1', '170%', 'Windows (CRLF)', and 'UTF-8'.

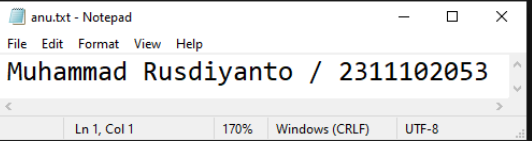
- Ubah Udin

```
Pilih Operasi : 6

- Ubah Tengah

Masukkan nama lama : Udin
Masukkan NIM lama : 23300048
Masukkan nama : Idin
Masukkan NIM : 23300045

Data Udin telah diganti dengan data Idin
```

A screenshot of a Notepad window titled 'anu.txt - Notepad'. The text inside the Notepad is 'Muhammad Rusdiyanto / 2311102053'. The status bar at the bottom shows 'Ln 1, Col 1', '170%', 'Windows (CRLF)', and 'UTF-8'.

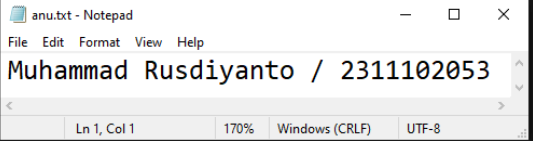
- Ubah Belakang

```
Pilih Operasi : 5

- Ubah Depan

Masukkan nama : Lucy
Masukkan NIM : 23300101

Data David telah diganti dengan
Data Lucy
```

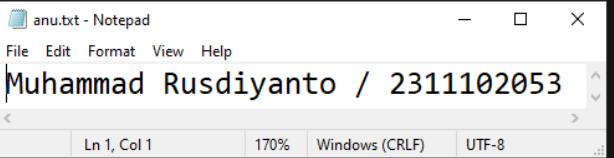
A screenshot of a Notepad window titled 'anu.txt - Notepad'. The text inside the Notepad is 'Muhammad Rusdiyanto / 2311102053'. The status bar at the bottom shows 'Ln 1, Col 1', '170%', 'Windows (CRLF)', and 'UTF-8'.

- Hapus Depan

```
Pilih Operasi : 7

- Hapus Depan

Data Owi berhasil dihapus
```

A screenshot of a Notepad window titled 'anu.txt - Notepad'. The text inside the Notepad is 'Muhammad Rusdiyanto / 2311102053'. The status bar at the bottom shows 'Ln 1, Col 1', '170%', 'Windows (CRLF)', and 'UTF-8'.

- Ubah Depan

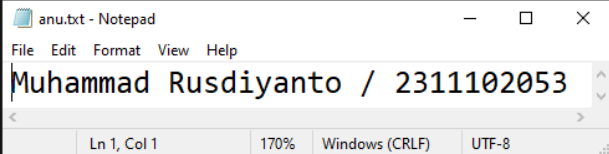


```
Pilih Operasi : 4

- Ubah Depan

Masukkan nama : Bagas
Masukkan NIM : 2330002

Data Jawab telah diganti dengan data Bagas
```

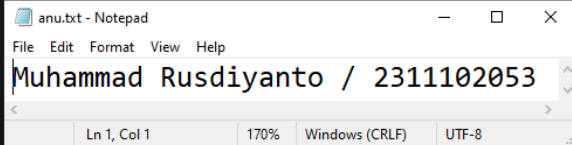


- Hapus Belakang

```
Pilih Operasi : 8

- Hapus Belakang

Data Lucy berhasil dihapus
```



- Tampil

```
Pilih Operasi : 11

Data MAHASISWA

NAMA          NIM
Bagas          23300002
Rusdi          2311102053
Farrel         23300003
Wati           23300004
Anis           23300008
Bowo           23300015
Gahar          23300040
Idin           23300045
Ucok           23300050
Budi           23300099
```



Deskripsi:

Program di atas adalah sebuah program hasil modifikasi dari program Guided 1. Layaknya program Guided 1, program tersebut menggunakan single linked list non circular, hanya saja program ini bisa menyimpan dua data yaitu nama serta NIM dari mahasiswa. Cara kerja kode kurang lebih sama untuk mayoritas dari program, kecuali untuk beberapa hal seperti parameter fungsi dan struktur data dari program. Perbedaan paling signifikan dalam kode terdapat dalam fungsi yang *ubahTengah* dan *hapusTengah*. Fungsi tersebut sebelumnya menerima posisi data sebagai parameter, tapi sekarang fungsi menerima 2 data yaitu nama dan NIM. Fungsi tersebut nantinya akan looping ke setiap node dalam list dan mencari mahasiswa dengan nama dan NIM yang sesuai. Jika telah ditemukan, maka data mahasiswa akan diubah atau dihapus, menyesuaikan dengan fungsi yang dipanggil program. Jika tidak, maka akan ditampilkan pesan bahwa mahasiswa tidak ditemukan. Ada pula modifikasi lain berupa menu dan input pengguna dalam program. Untuk input pengguna dibuat menggunakan fungsi supaya bisa digunakan berulang kali, sehingga menghemat kode

## **D. Kesimpulan**

Linked list non circular memiliki node terakhir yang menunjuk ke null, sehingga traversal berhenti saat mencapai node terakhir, sedangkan pada linked list circular, node terakhir menunjuk kembali ke node pertama, membentuk lingkaran tertutup, sehingga looping dapat berlanjut secara tak terbatas kecuali dihentikan secara eksplisit. Keuntungan linked list circular adalah kemampuannya untuk melakukan looping berulang kali tanpa pengecekan akhir, namun hal tersebut dapat meningkatkan kompleksitas implementasi dan berpotensi menyebabkan masalah seperti looping tak terbatas jika tidak dikelola dengan baik.

## **E. Referensi**

Asisten Praktikum, “Modul 4 Linked List Circular dan Non Circular”

Reema Thareja. (2014). Data Structures using C. OXFORD. New Delhi.