# LAPORAN PRAKTIKUM

**MODUL IV**

**LINKED LIST CIRCULAR DAN NON CIRCULAR**


**Disusun Oleh:**

Aldi ransi

2211102328

**Dosen**

Muhammad Afrizal Amrustian, S. Kom., M. Kom.


## PROGRAM STUDI S1 TEKNIK INFORMATIKA

## FAKULTAS INFORMATIKA

## INSTITUT TEKNOLOGI TELKOM PURWOKERTO

## 2024

# BAB I

# TUJUAN PRAKTIKUM

- Mahasiswa memahami perbedaan konsep Linked List Circular dan Non Circular.
- Mahasiswa mampu menerapkan Linked List Circular dan Non Circular ke dalam pemrograman

# BAB II

# DASAR TEORI

1. Linked List Circular

   Linked list circular adalah sebuah struktur data yang digunakan yang terdiri dari kumpulan simpul (node) yang terhubung dengan pointer yang saling terkait membentuk sirkuit, memiliki simpul akhir yang kembali ke simpul pertama sehingga membuat suatu siklus(cycle) dan dapat digunakan dalam situasi di mana data perlu diolah secara berkelanjutan.

2. Linked List Non Circular

   Linked list non circular adalah struktur data yang terdiri dari node dengan pointer yang menunjuk ke node berikutnya, kecuali pada node terakhir yang menunjuk ke nilai null sebagai penanda akhir dari linked list. Digunakan pada situasi di mana data perlu diolah secara berurutan dan tidak membentuk lingkaran atau sirkuit. Lebih efisien dalam penggunaannya karena mudah menambah atau menghapus node pada awal atau akhir linked list dan tidak perlu memperhatikan hubungan antar node yang saling terkait seperti pada linked list circular.

# 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;
Node *tail;

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

// Pengecekan
bool isEmpty()
{
    if (head == NULL)
        return true;
    else
        return false;
}

// Tambah Depan
void insertDepan(int nilai)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        baru->next = head;
```

```
        head = baru;
    }
}

// Tambah Belakang
void insertBelakang(int nilai)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }

    else
    {
        tail->next = baru;
        tail = baru;
    }
}

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

// Tambah Tengah
void insertTengah(int data, int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else

    {
```

```
        Node *baru, *bantu;
        baru = new Node();
        baru
            ->data = data;
        // tranversing
        bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1)
        {
            bantu = bantu
                        ->next;

            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(int data)
{
    if (isEmpty() == 0)
    {
        head->data = data;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah Tengah
void ubahTengah(int data, int posisi)
{
    Node *bantu;
    if (isEmpty() == 0)
    {

        if (posisi < 1 || posisi > hitungList())
        {
            cout << "Posisi di luar jangkauan" << endl;
        }
        else if (posisi == 1)
        {
            cout << "Posisi bukan posisi tengah" << endl;
        }
        else
        {
            bantu = head;
            int nomor = 1;
            while (nomor < posisi)
            {
                bantu = bantu->next;
```

```
                nomor++;
            }
            bantu->data = data;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah Belakang
void ubahBelakang(int data)
{
    if (isEmpty() == 0)
    {
        tail->data = data;
    }

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

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

// Tampilkan List
void tampil()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << ends;
            bantu = bantu->next;
        }
```

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

int main()
{
    init();
    insertDepan(3);
    tampil();
    insertBelakang(5);
    tampil();
    insertDepan(2);
    tampil();
    insertDepan(1);
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, 2);
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1);
    tampil();
    ubahBelakang(8);
    tampil();
    ubahTengah(11, 2);
    tampil();

    return 0;
}
```

**Output**

```
PS C:\Users\ACER>  & 'c:\Users\ACER\.vscode\extensio
ns\ms-vscode.cpptools-1.14.5-win32-x64\debugAdapters
\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MI
Engine-In-zrqpgsqf.3hm' '--stdout=Microsoft-MIEngine
-Out-3urgvkoi.a3h' '--stderr=Microsoft-MIEngine-Erro
r-hwfd2kff.ivx' '--pid=Microsoft-MIEngine-Pid-bbobvn
ua.tz4' '--dbgExe=C:\Not System\College\MinGW\bin\gd
b.exe' '--interpreter=mi'
3
35
235
1235
235
23
273
23
13
18
111
PS C:\Users\ACER>
```

**Deskripsi program**

Program tersebut menggunakan linked list non-circular dan memiliki fitur untuk menambah, menghapus, mengubah. Program tersebut juga menggunakan beberapa fungsi if, else if, dan while do. Selain itu, program tersebut juga menggunakan Struct

## 2. Guided 2

**Source code**

```
#include <iostream>

using namespace std;

/// PROGRAM SINGLE LINKED LIST CIRCULAR

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

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

void init()
{
```

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

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

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

// Hitung List
int hitungList()
{
    bantu = head;
    int jumlah = 0;

    while (bantu != NULL)
    {
        jumlah++;
        bantu = bantu->next;
    }

    return jumlah;
}

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

    if (isEmpty() == 1)
    {

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

```
        {
            tail = tail->next;
        }
        baru->next = head;
        head = baru;
        tail->next = head;
    }
}

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

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

        tail->next = baru;
        baru->next = head;
    }
}

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

```
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

// Hapus Depan

void hapusDepan()
{
    if (isEmpty() == 0)
    {
        hapus = head;
        tail = head;

        if (hapus->next == head)
        {
            head = NULL;
            tail = NULL;

            delete hapus;
        }
        else
        {
            while (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;
}
```

**Output**

```
PS C:\Users\ACER>  & 'c:\Users\ACER\.vscode\extensio
ns\ms-vscode.cpptools-1.14.5-win32-x64\debugAdapters
\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MI
Engine-In-5rdrtguf.2u4' '--stdout=Microsoft-MIEngine
-Out-xqx1r4sy.nkr' '--stderr=Microsoft-MIEngine-Erro
r-rvzdapaq.q4p' '--pid=Microsoft-MIEngine-Pid-3xkk0j
fl.gpp' '--dbgExe=C:\Not System\College\MinGW\bin\gd
b.exe' '--interpreter=mi'
Ayam
BebekAyam
BebekAyamCicak
BebekAyamCicakDomba
BebekAyamCicak
AyamCicak
AyamSapiCicak
```

**Deskripsi program**

Program tersebut merupakan program Linked List Circular, program tersebut dapat menambah dan menghaous pada depan dan belakang. Struktur data tersebut terdiri dari Struct Node yang berisi data dan pointer ke Node selanjutnya.

## UNGUIDED

1. **Unguided 1**
   **Source code**

```
#include <iostream>
// Muhammad Rayhan Pratama/2211102179/IF-10-D
#include <iomanip>
using namespace std;
struct mahasiswa
{
    string nama;
    int nim;
};
struct node
{
    mahasiswa iden;
    node *next;
};
node *head, *tail, *bantu, *hapus, *before, *baru;
void init()
{
    head = NULL;
    tail = NULL;
    bantu = NULL;
    hapus = NULL;
    before = NULL;
}
bool isEmpty()
{
    if (head == NULL)
    {
        return true;
    }
    else
    {
        return false;
    }
}
mahasiswa mintaData()
{
    system("cls");
    mahasiswa iden;
    cout << "\nMasukkan Nama\t: ";
    cin.ignore();
    getline(cin, iden.nama);
    cout << "Masukkan NIM\t: ";
    cin >> iden.nim;
    return iden;
}
void insertDepan(mahasiswa iden)
{
```

```
    node *baru = new node;
    baru->iden.nama = iden.nama;
    baru->iden.nim = iden.nim;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
    cout << "Data " << iden.nama << " berhasil diinput!\n";
    system("pause");
}
void insertBelakang(mahasiswa iden)
{
    node *baru = new node;
    baru->iden.nama = iden.nama;
    baru->iden.nim = iden.nim;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}
int hitungList()
{
    int penghitung = 0;
    node *bantu;
    bantu = head;
    while (bantu != NULL)
    {
        penghitung++;
        bantu = bantu->next;
    }
    return penghitung;
}
void insertTengah(mahasiswa identitas, int posisi)
{
    node *baru = new node;
    baru->iden.nama = identitas.nama;
    baru->iden.nim = identitas.nim;
```

```
    node *bantu;
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "posisi diluar jangakauan";
    }
    else if (posisi == 1)
    {
        cout << "INi bukan posisi tengah\n";
    }
    else
    {
        bantu = head;
        int penghitung = 1;
        while (penghitung != posisi - 1)
        {
            penghitung++;
            bantu = bantu->next;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}
void ubahDepan(mahasiswa data)
{
    string namaLama = head->iden.nama;
    head->iden.nama = data.nama;
    head->iden.nim = data.nim;

    cout << "data " << namaLama << " telah diganti dengan
data "
         << data.nama << endl;
}
void ubahBelakang(mahasiswa data)
{
    string namaLama = tail->iden.nama;
    tail->iden.nama = data.nama;
    tail->iden.nim = data.nim;
    cout << "data " << namaLama << " telah diganti dengan
data "
         << data.nama << endl;
}
void ubahTengah(mahasiswa data)
{
    int posisi;
    cout << "\nMasukkan posisi data yang akan diubah : ";
    cin >> posisi;
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "\nPosisi diluar jangkauan\n";
    }
    else if (posisi == 1)
    {
```

```
        cout << "\nBukan posisi tengah\n";
    }
    else
    {
        bantu = head;
        int penghitung = 1;
        while (penghitung != posisi)
        {
            penghitung++;
            bantu = bantu->next;
        }
        bantu->iden.nama = data.nama;
        bantu->iden.nim = data.nim;
    }
}
void tampil()
{
    system("cls");
    node *bantu = head;
    cout << "Nama "
         << " Nim\n";
    while (bantu != NULL)
    {
        cout << bantu->iden.nama << " " << bantu->iden.nim <<
endl;
        bantu = bantu->next;
    }
}
void hapusDepan()
{
    string dataLama = head->iden.nama;
    hapus = head;
    if (head != tail)
    {
        head = head->next;
        delete hapus;
    }
    else
    {
        head = tail = NULL;
    }
    cout << "Data " << dataLama << " berhasil dihapus\n";
}
void hapusBelakang()
{
    string dataLama = head->iden.nama;
    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;
    }
    cout << "Data " << dataLama << " berhasil dihapus\n";
}
void hapusTengah()
{
    tampil();
    cout << endl;
    if (isEmpty() == false)
    {
    back:
        int posisi;
        cout << "Masukkan Posisi yang dihapus : ";
        cin >> posisi;
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "\nPosisi di luar jangkauan!\n";
            system("pause");
            system("cls");
            cout << "Masukkan posisi baru\n";
            goto back;
        }
        else if (posisi == 1 || posisi == hitungList())
        {
            cout << "\nBukan Posisi tengah\n";
            system("pause");
            // system("cls");
            cout << "Masukkan posisi baru\n";
            goto back;
        }
        else
        {
            bantu = head;
            int penghitung = 1;

            while (penghitung <= posisi)
            {
                if (penghitung == posisi - 1)
                {
                    before = bantu;
                }
                if (penghitung == posisi)
                {
                    hapus = bantu;
```

```
                }
                bantu = bantu->next;
                penghitung++;
            }
            string dataLama = hapus->iden.nama;
            before->next = bantu;
            delete hapus;

            cout << "\nData " << dataLama << " berhasil
                dihapus !\n ";
                system("pause");
        }
    }
    else
    {
        cout << "\n!!! List Data Kosong !!!\n";
        system("pause");
    }
}
void hapusList()
{
    bantu = head;
    while (bantu != NULL)
    {
        hapus = bantu;
        delete hapus;
        bantu = bantu->next;
    }
    init();
    cout << "\nsemua data berhasil dihapus\n";
}
int main()
{
    init();
    mahasiswa iden;
back:
    int operasi, posisi;

    cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR\n\n";
    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.Exit" << endl;
    cout << "\nPilih Operasi :> ";
```

```
    cin >> operasi;
    switch (operasi)
    {
    case 1:
        cout << "tambah depan\n";
        insertDepan(mintaData());
        cout << endl;
        goto back;
        break;
    case 2:
        cout << "tambah belakang\n";
        insertBelakang(mintaData());
        cout << endl;
        goto back;
        break;
    case 3:
        cout << "tambah tengah\n";
        cout << "nama : ";
        cin >> iden.nama;
        cout << "NIM : ";
        cin >> iden.nim;
        cout << "Posisi: ";
        cin >> posisi;

        insertTengah(iden, posisi);
        cout << endl;
        goto back;
        break;
    case 4:
        cout << "ubah depan\n";
        ubahDepan(mintaData());
        cout << endl;
        goto back;
        break;
    case 5:
        cout << "ubah belakang\n";
        ubahBelakang(mintaData());
        cout << endl;
        goto back;
        break;
    case 6:
        cout << "ubah tengah\n";
        ubahTengah(mintaData());
        cout << endl;
        goto back;
        break;
    case 7:
        cout << "hapus depan\n";
        hapusDepan();
        cout << endl;
        goto back;
        break;
```

```
    case 8:
        cout << "hapus belakang\n";
        hapusBelakang();
        cout << endl;
        goto back;
        break;
    case 9:
        cout << "hapus tengah\n";
        hapusTengah();
        cout << endl;
        goto back;
        break;
    case 10:
        cout << "hapus list\n";
        hapusList();
        cout << endl;
        goto back;
        break;
    case 11:
        tampil();
        cout << endl;
        goto back;
        break;
    case 0:
        cout << "\nEXIT PROGRAM\n";
        break;
    default:
        cout << "\nSalah input operasi\n";
        cout << endl;
        goto back;
        break;
    }
    return 0;
}
```

**Output**

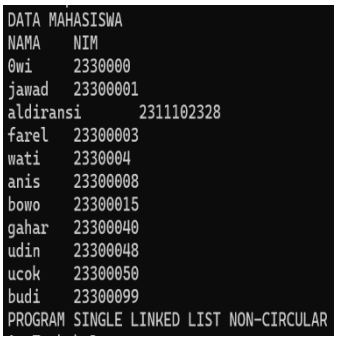
a. Tambahkan data berikut diantara Farrel dan Denis: Wati 2330004

```
DATA MAHASISWA
NAMA    NIM
jawad   23300001
aldiransi       2311102328
farel   23300003
wati    2330004
denis   23300005
anis    23300008
bowo    23300015
gahar   23300040
udin    23300048
ucok    23300050
budi    23300099
```

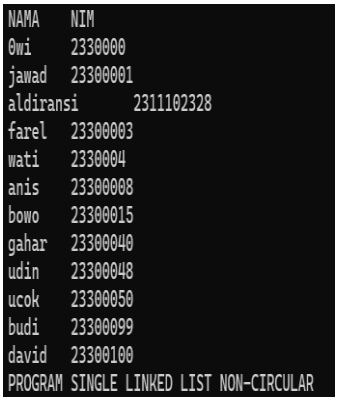
b. Hapus data Denis

```
DATA MAHASISWA
NAMA    NIM
jawad   23300001
aldiransi       2311102328
farel   23300003
wati    2330004
anis    23300008
bowo    23300015
gahar   23300040
udin    23300048
ucok    23300050
budi    23300099
```

c. Tambahkan data berikut di awal: Owi 2330000




d. tambahkan data berikut di akhir: David 23300100



e. Ubah data Udin menjadi data berikut: Idin 23300045

```
NAMA    NIM
0wi     2330000
jawad   23300001
aldiransi       2311102328
farel   23300003
wati    2330004
anis    23300008
bowo    23300015
gahar   23300040
idin    23300045
ucok    23300050
budi    23300099
david   23300100
```

f. Ubah data terkahir menjadi berikut: Lucy 23300101

```
DATA MAHASISWA
NAMA    NIM
0wi     2330000
jawad   23300001
aldiransi       2311102328
farel   23300003
wati    2330004
anis    23300008
bowo    23300015
gahar   23300040
idin    23300045
ucok    23300050
budi    23300099
lucy    23300101
```

g. Hapus data awal

```
DATA MAHASISWA
NAMA    NIM
jawad   23300001
aldiransi       2311102328
farel   23300003
wati    2330004
anis    23300008
bowo    23300015
gahar   23300040
idin    23300045
ucok    23300050
budi    23300099
lucy    23300101
```

h. Ubah data awal menjadi berikut: Bagas 2330002

```
DATA MAHASISWA
NAMA    NIM
bagas   2300045
aldiransi       2311102328
farel   23300003
wati    2330004
anis    23300008
bowo    23300015
gahar   23300040
idin    23300045
ucok    23300050
budi    23300099
lucy    23300101
```

i. Hapus data akhir

```
DATA MAHASISWA
NAMA    NIM
bagas   2300045
aldiransi       2311102328
farel   23300003
wati    2330004
anis    23300008
bowo    23300015
gahar   23300040
idin    23300045
ucok    23300050
budi    23300099
```

j. Tampilkan seluruh data

```
DATA MAHASISWA
NAMA    NIM
bagas   2300045
aldiransi       2311102328
farel   23300003
wati    2330004
anis    23300008
bowo    23300015
gahar   23300040
idin    23300045
ucok    23300050
budi    23300099
```

**Deskripsi program**

Program tersebut terdapat menu Linked List Non-Circullar untuk menyimpan nama dan NIM mahasiswayang  dimana data tersebut dari inputan user. Sturct untukmendeklarasikan node dan inisialisasi node dan pengecekan node dengan menggunakan bool.

# BAB IV
# KESIMPULAN

Linked List non circular adalah struktur data yang terdiri dari node dengan pointer yang menunjuk ke node berikutnya, kecuali pada node terakhir yang menunjuk ke nilai null sebagai penanda akhir dari linked list. Sedangkan Linked list circular adalah struktur data yang digunakan yang terdiri dari kumpulan simpul (node) yang terhubung dengan pointer yang saling terkait membentuk sirkuit