

**LAPORAN PRAKTIKUM STRUKTUR DATA  
DAN ALGORITMA**

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



**Disusun Oleh :**

NAMA : Pandia Arya Brata

NIM : 2311102076

**Dosen**

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

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

## MODUL 4

### LINKED LIST CIRCULAR DAN NON CIRCULAR

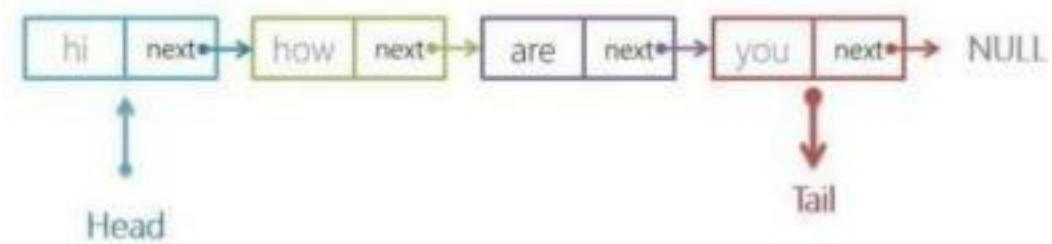
#### A. TUJUAN PRAKTIKUM

- 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

##### 1. Linked List Non Circular

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



**Gambar 1** *Single Linked List Non Circular*

#### OPERASI PADA LINKED LIST NON CIRCULAR

##### 1. Deklarasi Simpul (Node)

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

##### 2. Membuat dan Menginisialisasi Pointer Head dan Tail

```
node *head, *tail;

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

### 3. Pengecekan Kondisi Linked List

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

### 4. Penambahan Simpul (Node)

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

    else
    {
```

```

        node *baru = new node;
        baru->data = dataUser;
        baru->next = NULL;
        tail->next = baru;
        tail = baru;
    }
};

```

## 5. Penghapusan Simpul (Node)

```

void hapusDepan()
{
    if (isEmpty() == true)
    {
        cout << "List kosong!" <<
endl; }
    else
    {
        node *helper;
        helper = head;
        if (head == tail)
        {
            head = NULL;
            tail = NULL;
            delete helper;
        }
        else
            head = head->next;
        helper->next = NULL;
        delete helper;
    }
}
}

```

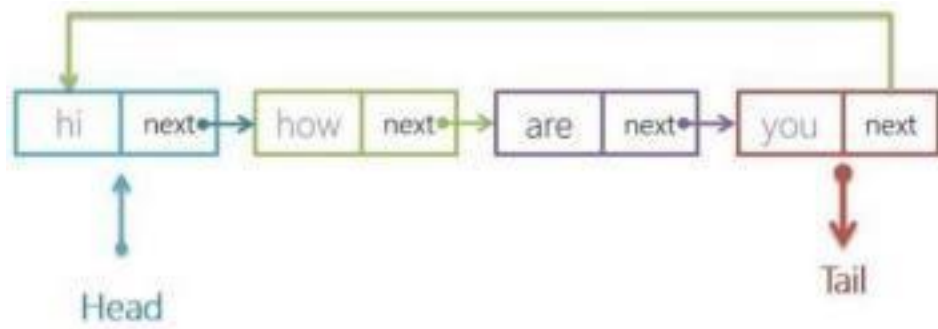
## 6. Tampil Data Linked List

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

## 2. Linked List Circular

*Linked list circular* merupakan *linked list* yang tidak memiliki akhir karena node terakhir (tail) tidak bernilai '*NULL*', tetapi terhubung dengan node pertama (head). Saat menggunakan *linked list circular* kita membutuhkan *dummy node* atau node pengecoh yang biasanya dinamakan dengan node *current* supaya program dapat berhenti menghitung data ketika node *current* mencapai node pertama (head).

*Linked list circular* dapat digunakan untuk menyimpan data yang perlu diakses secara berulang, seperti daftar putar lagu, daftar pesan dalam antrian, atau penggunaan memori berulang dalam suatu aplikasi. *Linked list circular* dapat digambarkan sebagai berikut.



**Gambar 2** Single Linked List Circular

## OPERASI PADA LINKED LIST CIRCULAR

### 1. Deklarasi Simpul (Node)

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

### 2. Membuat dan Menginisialisasi Pointer Head dan Tail

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

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

### 3. Pengecekan Kondisi Linked List

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

#### 4. Pembuatan Simpul (Node)

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

#### 5. Penambahan Simpul (Node)

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

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

#### 6. Penghapusan Simpul (Node)

```
void hapusBelakang()
```

```

{
    if (isEmpty() == 0)
    {
        hapus = head;
        tail = head;
        if (hapus->next == head)
        {
            head = NULL;
            tail = NULL;

            delete hapus;
        }
        else
        {
            while (hapus->next != head)
            {
                hapus = hapus->next;
            }
            while (tail->next != hapus)
            {
                tail = tail->next;
            }
            tail->next = head;
            hapus->next = NULL;

            delete hapus;
        }
    }
}

```

## 7. Menampilkan Data Linked List

```

void tampil()
{
    if (isEmpty() == 0)
    {
        tail = head;
        do
        {
            cout << tail->data << ends;

```



```

        tail = tail->next;
    } while (tail != head);
    cout << endl;
}
}

```

## C. GUIDED

### 1. Linked List Non Circular

#### PROGRAM SINGLE LINKED LIST NON-CIRCULAR

```

#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;
}

```

**Screenshot program :**

The screenshot shows a Windows command prompt window with the following output:

```

PS C:\Users\Administrator\Desktop\STRUKDAT (PRAKTIK)\Modul4> cd 'c:\...'
PS C:\Users\Administrator\Desktop\STRUKDAT (PRAKTIK)\Modul4\output>
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\Administrator\Desktop\STRUKDAT (PRAKTIK)\Modul4\output>

```

Overlaid on the command prompt is a text editor window titled "Pandia Ary". The editor contains the following text:

```

Pandia Arya Brata
2311102076

```

The status bar at the bottom of the text editor shows: "Ln 2, Col 11 | 28 characters | 100% | Window | UTF-8".

**Deskripsi program :**

Program di atas adalah implementasi single linked list non-circular menggunakan bahasa C++. Single linked list non-circular merupakan struktur data berupa kumpulan node yang terhubung satu sama lain melalui pointer next. Setiap node terdiri dari data dan pointer next yang menunjuk ke node berikutnya.

Program ini memiliki beberapa fungsi utama, antara lain:

1. Inisialisasi Node: menginisialisasi head dan tail menjadi NULL.
2. Pengecekan apakah list kosong atau tidak.
3. Tambah Node di Depan: menambahkan node baru di depan list.
4. Tambah Node di Belakang: menambahkan node baru di belakang list.
5. Hitung Jumlah Node: menghitung jumlah node dalam list.
6. Tambah Node di Tengah: menambahkan node baru di posisi tertentu dalam list.
7. Hapus Node di Depan: menghapus node pertama dalam list.
8. Hapus Node di Belakang: menghapus node terakhir dalam list.
9. Hapus Node di Tengah: menghapus node pada posisi tertentu dalam list.
10. Ubah Node di Depan: mengubah data pada node pertama dalam list.
11. Ubah Node di Tengah: mengubah data pada node pada posisi tertentu dalam list.
12. Ubah Node di Belakang: mengubah data pada node terakhir dalam list.
13. Hapus Seluruh Node: menghapus seluruh node dalam list.
14. Tampilkan List: menampilkan semua data dalam list.

## 2. Linked List Circular

### PROGRAM SINGLE LINKED LIST CIRCULAR

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

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

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

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

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

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

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

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

```



```

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

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

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

void hapusDepan()

```

```

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

void hapusBelakang()
{
    if (!isEmpty())
    {
        hapus = head;
        tail = head;
        if (hapus->next == head)
        {
            head = NULL;
            tail = NULL;
            delete hapus;
        }
        else
        {
            while (hapus->next != head)
            {
                hapus = hapus->next;
            }
            while (tail->next != hapus)
            {
                tail = tail->next;
            }
            tail->next = head;
            hapus->next = NULL;
            delete hapus;
        }
    }
}

```

```

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

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

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

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

```

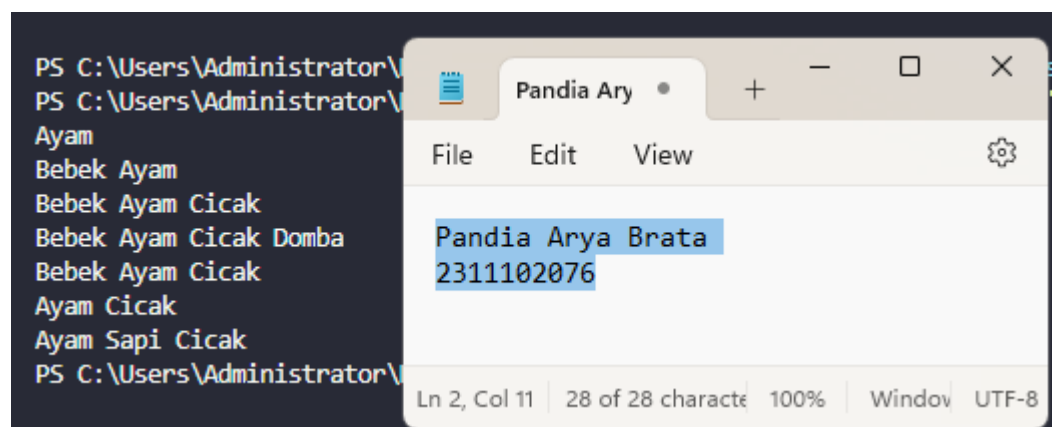
```

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

int main()
{
    init();
    insertDepan("Ayam");
    tampil();
    insertDepan("Bebek");
    tampil();
    insertBelakang("Cicak");
    tampil();
    insertBelakang("Domba");
    tampil();
    hapusBelakang();
    tampil();
    hapusDepan();
    tampil();
    insertTengah("Sapi", 2);
    tampil();
    hapusTengah(2);
    tampil();
    return 0;
}

```

### Screenshot Program :



## **Deskripsi program :**

Program ini merupakan implementasi dari struktur data linked list dengan menggunakan konsep circular linked list, yang dapat menambahkan, menghapus, dan menampilkan data dalam list. Program ini juga memiliki fungsi-fungsi dasar seperti `init()` untuk menginisialisasi list, `isEmpty()` untuk mengecek apakah list kosong atau tidak, `hitungList()` untuk menghitung jumlah elemen dalam list, serta `clearList()` untuk menghapus seluruh elemen dalam list.

Selain itu, program ini juga memiliki beberapa fungsi untuk menambahkan dan menghapus elemen dalam list, antara lain:

- `insertDepan()` untuk menambahkan elemen di awal list
- `insertBelakang()` untuk menambahkan elemen di akhir list
- `insertTengah()` untuk menambahkan elemen pada posisi tertentu dalam list •
- `hapusDepan()` untuk menghapus elemen di awal list
- `hapusBelakang()` untuk menghapus elemen di akhir list
- `hapusTengah()` untuk menghapus elemen pada posisi tertentu dalam list

Program ini juga memiliki fungsi `tampil()` untuk menampilkan seluruh elemen dalam list. Program ini juga sudah dilengkapi dengan komentar yang menjelaskan setiap bagian kode agar lebih mudah dipahami.

#### D. UNGUIDED

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

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

• Tampilan Menu:

```
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
0. KELUAR

Pilih Operasi :

• Tampilan Operasi Tambah:

-Tambah Depan

Masukkan Nama :

Masukkan NIM :

Data telah ditambahkan

-Tambah Tengah

Masukkan Nama :

Masukkan NIM :

Masukkan Posisi :

Data telah ditambahkan

- Tampilan Operasi Hapus :

-Hapus Belakang

Data (nama mahasiswa yang dihapus) berhasil dihapus

-Hapus Tengah

Masukkan posisi :

Data (nama mahasiswa yang dihapus) berhasil dihapus

- Tampilan Operasi Ubah:

-Ubah Belakang

Masukkan nama :

Masukkan NIM :

Data (nama lama) telah diganti dengan data (nama baru)

-Ubah Belakang

Masukkan nama :

Masukkan NIM :

Masukkan posisi :

Data (nama lama) telah diganti dengan data (nama baru)

- Tampilan Operasi Tampil Data:

DATA MAHASISWA

NAMA NIM

Nama1 NIM1

Nama2 NIM2

**\*Buat tampilan output sebagus dan secantik mungkin sesuai kreatifitas anda masing-masing, jangan terpaku pada contoh output yang diberikan**

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

Nama	NIM
Jawad	23300001
[Nama Anda]	[NIM Anda]
Farrel	23300003
Denis	23300005
Anis	23300008
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099

Lakukan perintah berikut:

- a) Tambahkan data berikut diantara Farrel dan Denis:

**Wati 23300004**

- b) Hapus data Denis

- c) Tambahkan data berikut di awal:

**Owi 23300000**

- d) Tambahkan data berikut di akhir:

**David 23300100**

- e) Ubah data Udin menjadi data berikut:

**Idin 23300045**

- f) Ubah data terkahir menjadi berikut:

**Lucy 23300101**

- g) Hapus data awal

- h) Ubah data awal menjadi berikut:

**Bagas 23300002**

- i) Hapus data akhir

- j) Tampilkan seluruh data



### Source Code :

```
#include <iostream>
#include <stdlib.h>
#include <conio.h>

using namespace std;

struct Mahasiswa
{
    string nama;
    string nim;
    Mahasiswa *next;
} *head, *tail, *curr;

void tambahDepan()
{
    curr = new Mahasiswa;

    cout << "\nMasukkan Nama : ";
    cin >> curr->nama;
    cout << "Masukkan NIM : ";
    cin >> curr->nim;

    if (head == NULL)
    {
        head = curr;
        tail = curr;
    }
    else
    {
        curr->next = head;
        head = curr;
    }
    cout << "\nData " << curr->nama << " berhasil diinput!\n";
}

void tambahBelakang()
{
    Mahasiswa *baru = new Mahasiswa;
    baru->next = NULL;

    cout << "\nMasukkan Nama : ";
    cin >> baru->nama;
    cout << "Masukkan NIM : ";
    cin >> baru->nim;

    if (head == NULL)
    {
        head = baru;
        tail = baru;
    }
    else
```

```

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

    cout << "\nData " << baru->nama << " berhasil diinput!\n";
}

void tambahTengah()
{
    curr = new Mahasiswa;

    cout << "\nMasukkan Nama : ";
    cin >> curr->nama;
    cout << "Masukkan NIM : ";
    cin >> curr->nim;

    int pos;
    cout << "Masukkan posisi : ";
    cin >> pos;

    if (pos == 1)
    {
        curr->next = head;
        head = curr;

        cout << "\nData " << curr->nama << " berhasil diinput!\n";
        return;
    }

    Mahasiswa *temp = head;

    for (int i = 1; i < pos - 1; i++)
    {
        if (temp == NULL)
        {
            cout << "\nPosisi terlalu besar!\n";
            return;
        }

        temp = temp->next;
    }

    curr->next = temp->next;
    temp->next = curr;

    cout << "\nData " << curr->nama << " berhasil diinput !\n ";
}

void ubahDepan()
{
    if (head == NULL)
    {
        cout << "\nData kosong!\n";
    }
}

```

```

        return;
    }

    string namaBaru, nimBaru;
    cout << "\nMasukkan Nama Baru : ";
    cin >> namaBaru;
    cout << "Masukkan NIM Baru : ";
    cin >> nimBaru;

    string namaLama = head->nama;
    head->nama = namaBaru;
    head->nim = nimBaru;

    cout << "\nData " << namaLama << " berhasil diganti dengan data " <<
namaBaru << "!\n";
}

void ubahBelakang()
{
    if (head == NULL)
    {
        cout << "\nData kosong!\n";
        return;
    }

    cout << "\nMasukkan Nama baru : ";
    string namaBaru;
    cin >> namaBaru;
    cout << "Masukkan NIM baru : ";
    string nimBaru;
    cin >> nimBaru;

    string namaLama = tail->nama;
    cout << namaLama << endl;
    tail->nama = namaBaru;
    tail->nim = nimBaru;

    cout << "\nData berhasil diubah!\n";
    cout << "Data " << namaLama << " telah diganti dengan data " << tail->nama
<< "!\n";
}

void ubahTengah()
{
    if (head == NULL)
    {
        cout << "\nData kosong!\n";
        return;
    }

    int pos;
    cout << "Masukkan posisi : ";
    cin >> pos;

```

```

Mahasiswa *temp = head;

for (int i = 1; i < pos; i++)
{
    if (temp == NULL)
    {
        cout << "\nPosisi terlalu besar!\n";
        return;
    }

    temp = temp->next;
}

string old_name = temp->nama; // simpan nama lama untuk ditampilkan
cout << "\nMasukkan Nama Baru : ";
cin >> temp->nama;
cout << "Masukkan NIM Baru : ";
cin >> temp->nim;

cout << "\nData " << old_name << " telah diganti dengan data " << temp-
>nama << "!\n";
}

void hapusDepan()
{
    if (head == NULL)
    {
        cout << "\nData kosong!\n";
        return;
    }

    curr = head;
    head = head->next;

    cout << "\nData dengan nama " << curr->nama << "berhasil dihapus !\n ";
    delete curr;
}

void hapusBelakang()
{
    if (head == NULL)
    {
        cout << "\nData kosong!\n";
        return;
    }

    if (head == tail)
    {
        cout << "\nData dengan nama " << tail->nama << " berhasil dihapus!\n";
        delete head;
        head = NULL;
        tail = NULL;
    }
}

```

```

else
{
    curr = head;

    while (curr->next != tail)
    {
        curr = curr->next;
    }

    cout << "\nData dengan nama " << tail->nama << " berhasil dihapus!\n";
    tail = curr;
    curr = curr->next;

    delete curr;
    tail->next = NULL;
}
}

void hapusTengah()
{
    if (head == NULL)
    {
        cout << "\nData kosong!\n";
        return;
    }

    int pos;
    cout << "Masukkan posisi : ";
    cin >> pos;
    if (pos == 1)
    {
        curr = head;
        head = head->next;

        cout << "\nData dengan nama " << curr->nama << " berhasil dihapus!\n";
        delete curr;

        cout << "\nData berhasil dihapus!\n";
        return;
    }

    Mahasiswa *temp = head;

    for (int i = 1; i < pos - 1; i++)
    {
        if (temp == NULL)
        {
            cout << "\nPosisi terlalu besar!\n";
            return;
        }

        temp = temp->next;
    }
}

```

```

curr = temp->next;
temp->next = curr->next;

cout << "\nData dengan nama " << curr->nama << " berhasil dihapus!\n";
delete curr;

cout << "\nData berhasil dihapus!\n";
}

void tampilkan()
{
    if (head == NULL)
    {
        cout << "\nData kosong!\n";
        return;
    }

    curr = head;

    cout << "\nData Mahasiswa:\n";
    while (curr != NULL)
    {
        cout << "Nama : " << curr->nama << endl;
        cout << "NIM : " << curr->nim << endl
            << endl;
        curr = curr->next;
    }
}

int main()
{
    int pilihan;

    do
    {
        cout << "\n---PROGRAM SINGLE LINKED LIST NON-CIRCULAR--- \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. Tampilkan\n";
        cout << "11. Keluar\n";
        cout << "Pilih: ";
        cin >> pilihan;

        switch (pilihan)
        {

```

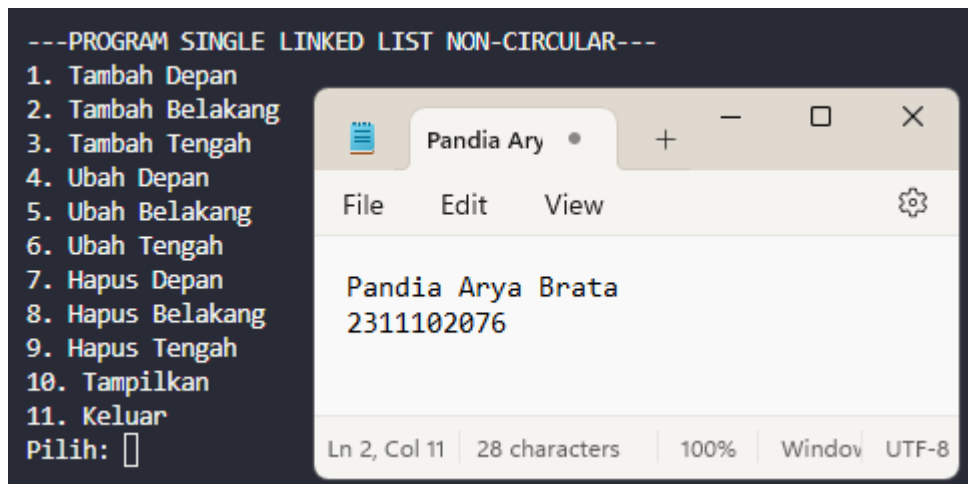
```
        case 1:
            tambahDepan();
            break;
        case 2:
            tambahBelakang();
            break;
        case 3:
            tambahTengah();
            break;
        case 4:
            ubahDepan();
            break;
        case 5:
            ubahBelakang();
            break;
        case 6:
            ubahTengah();
            break;
        case 7:
            hapusDepan();
            break;
        case 8:
            hapusBelakang();
            break;
        case 9:
            hapusTengah();
            break;
        case 10:
            tampilkan();
            break;
        case 11:
            cout << "\nProgram Selesai.\n";
            break;
        default:
            cout << "\nPilihan tidak valid!\n";
            break;
    }

    cout << "\n";

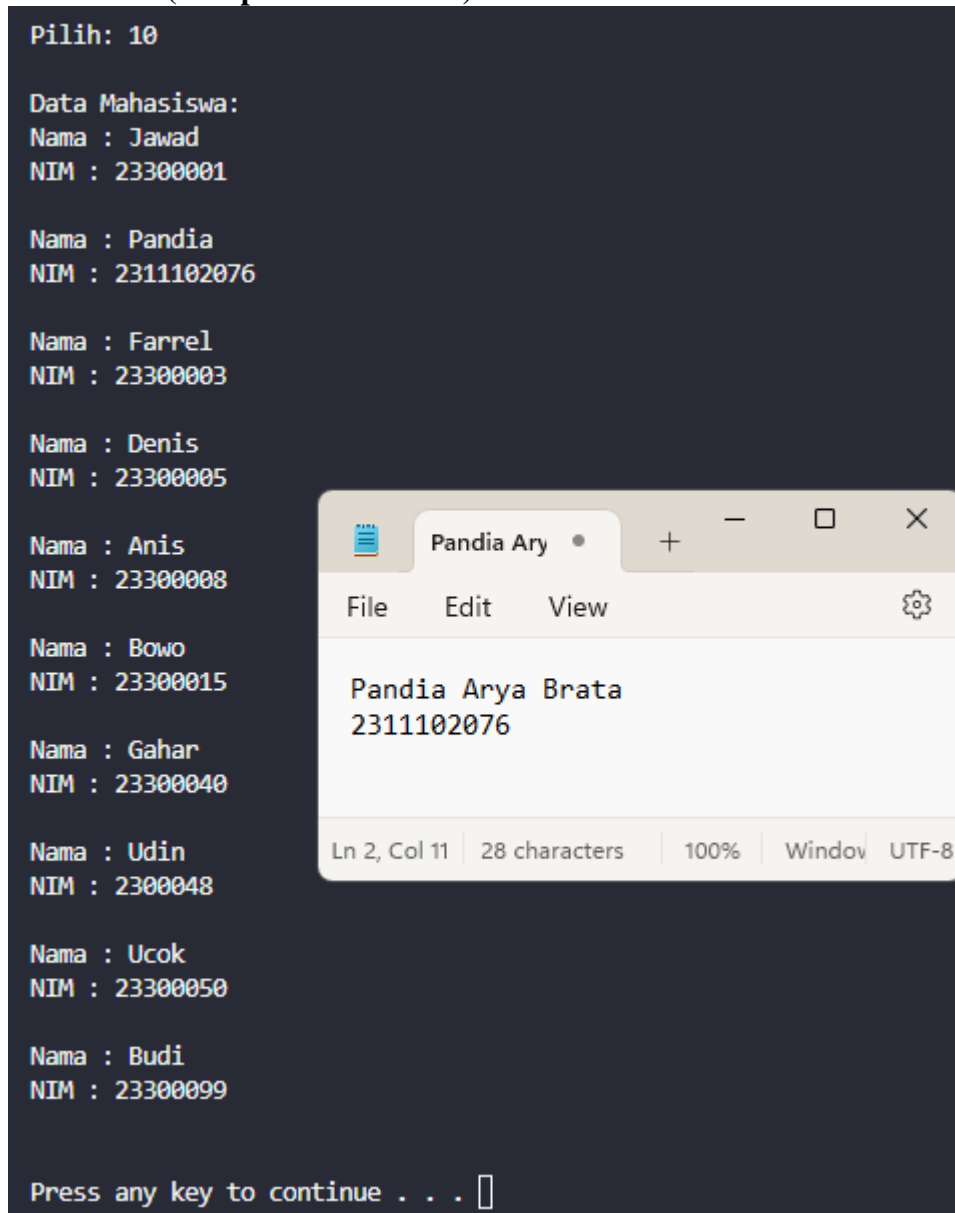
} while (pilihan != 11);

return 0;
}
```

**Screenshot program (Tampilan awal) :**

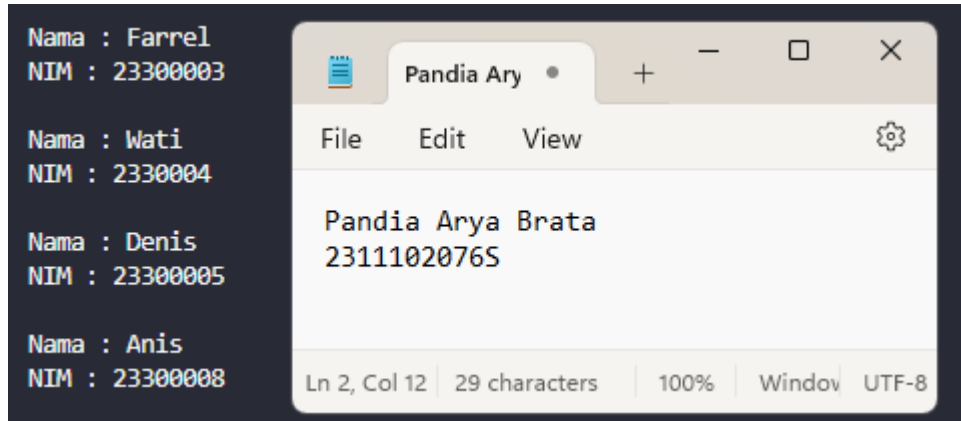


**Screenshot (Tampilan berisi data) :**

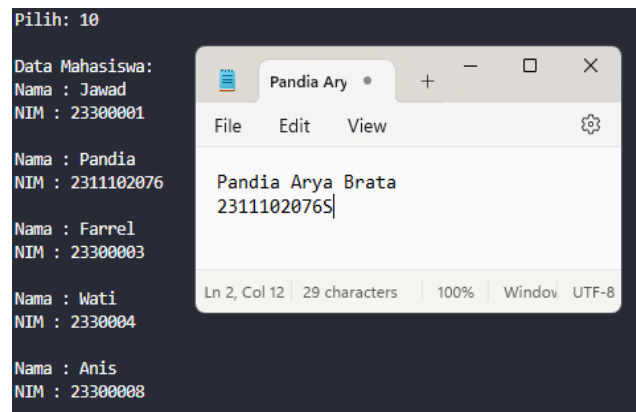
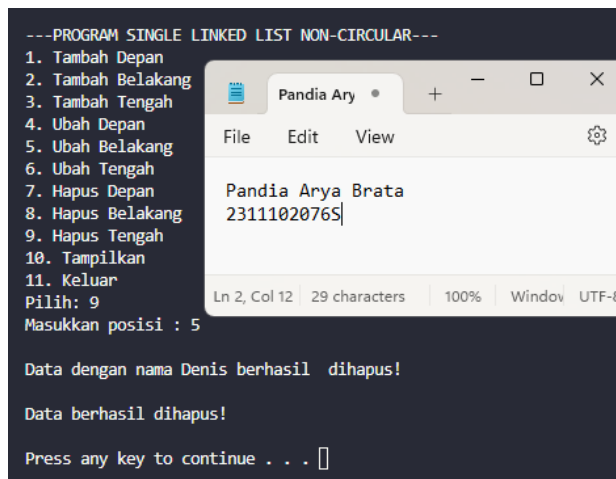




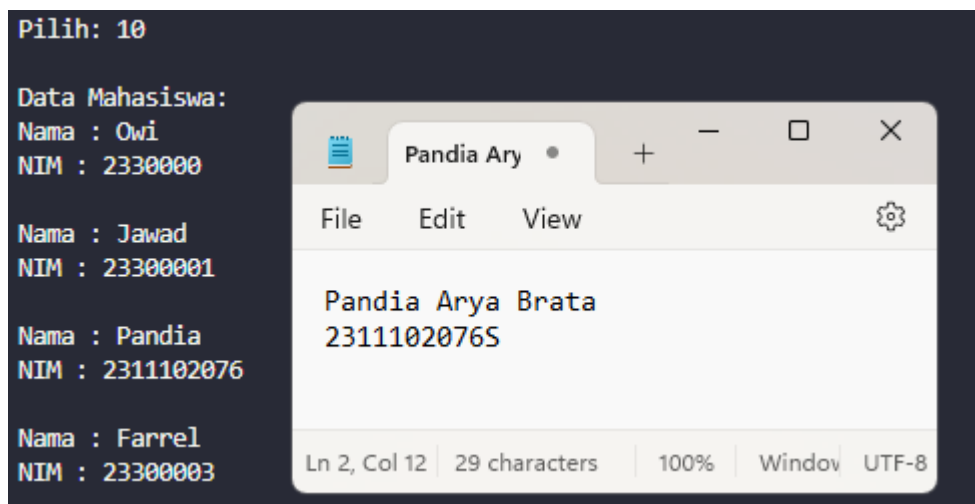
### Screenshot (Tambahkan data Wati diantara Farrel dan Denis) :



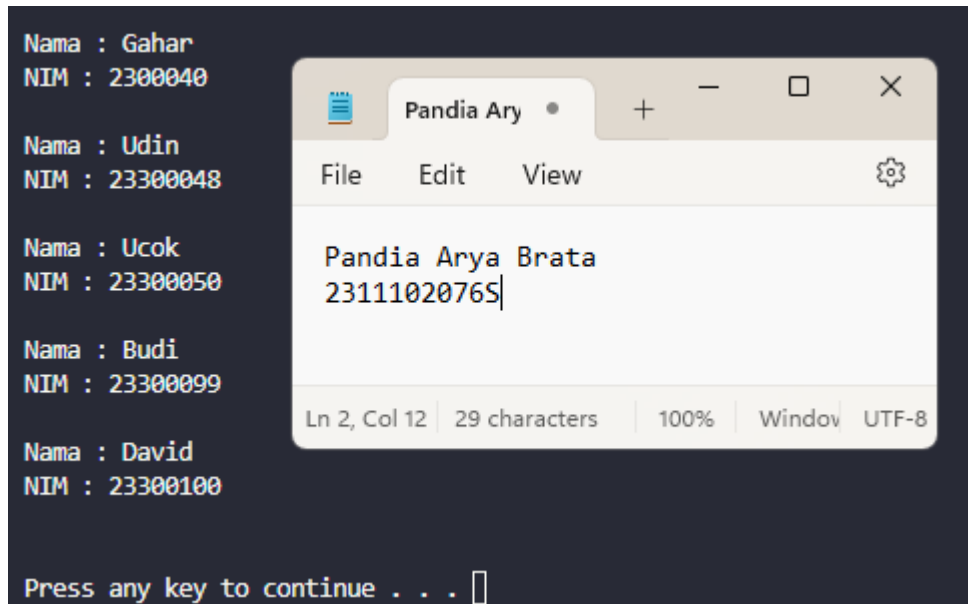
### Screenshot (Hapus data Denis) :



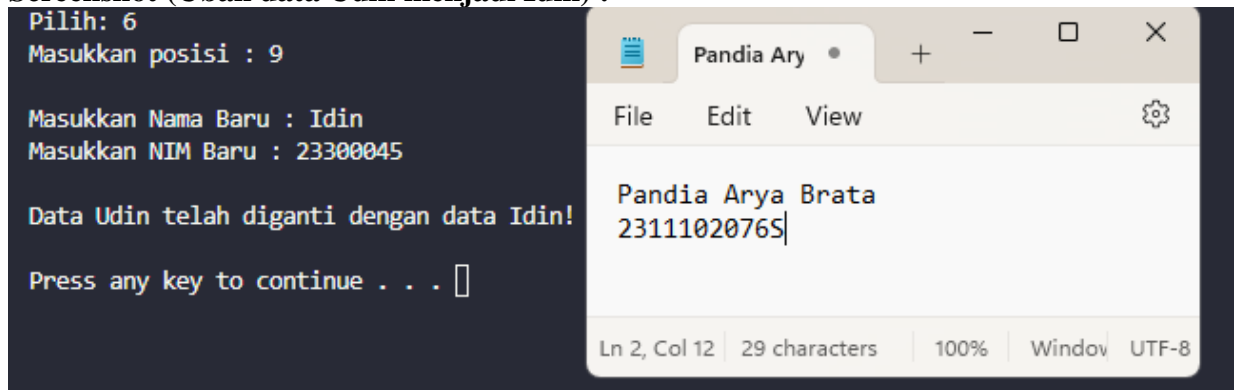
### Screenshot (Tambah data Owi di awal) :



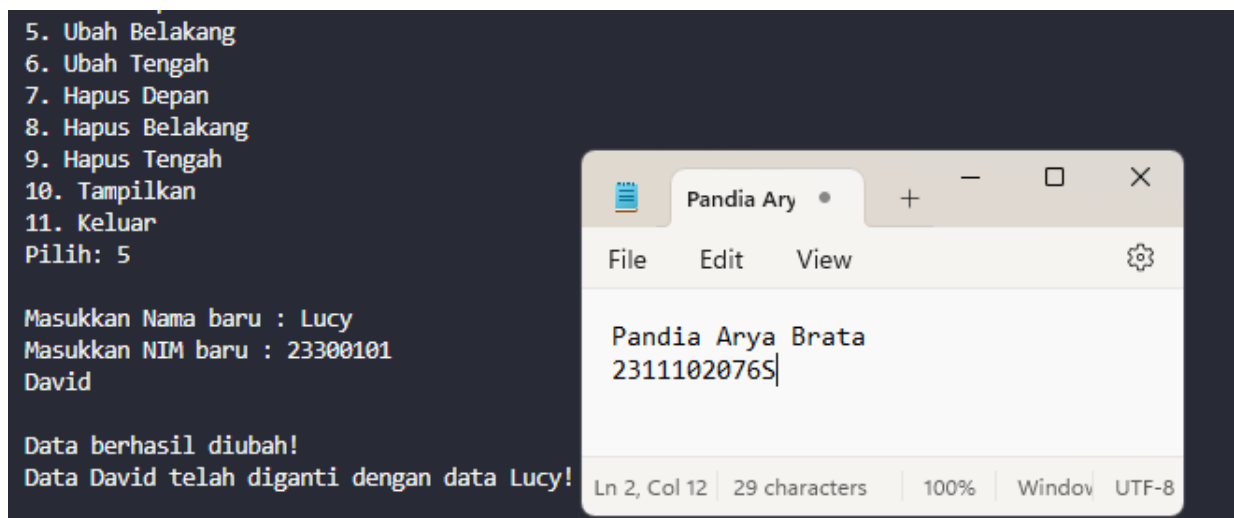
**Screenshot (Tambah data David di akhir) :**



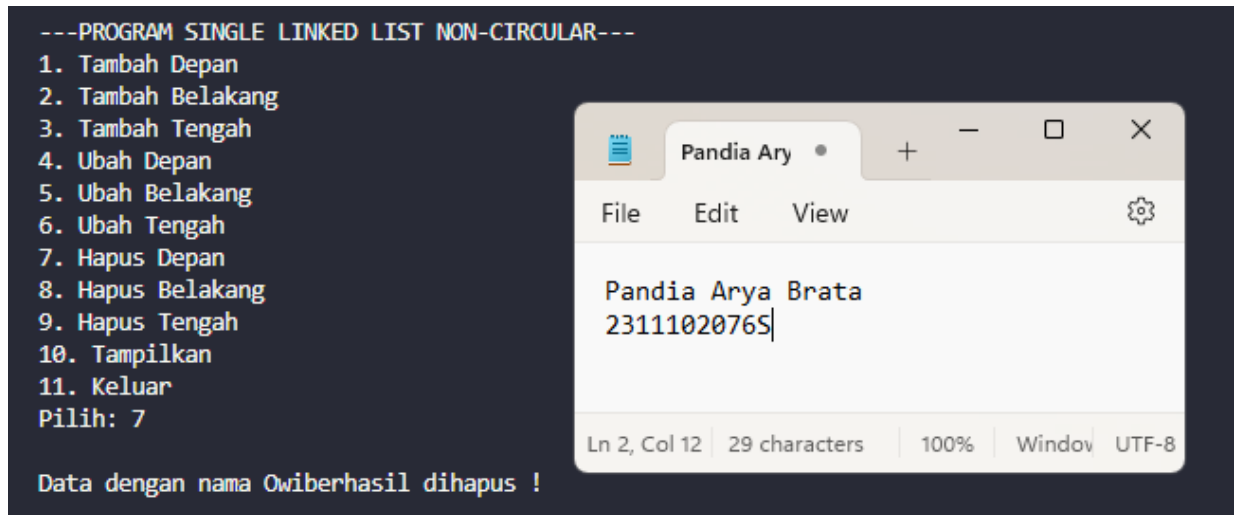
**Screenshot (Ubah data Udin menjadi Idin) :**



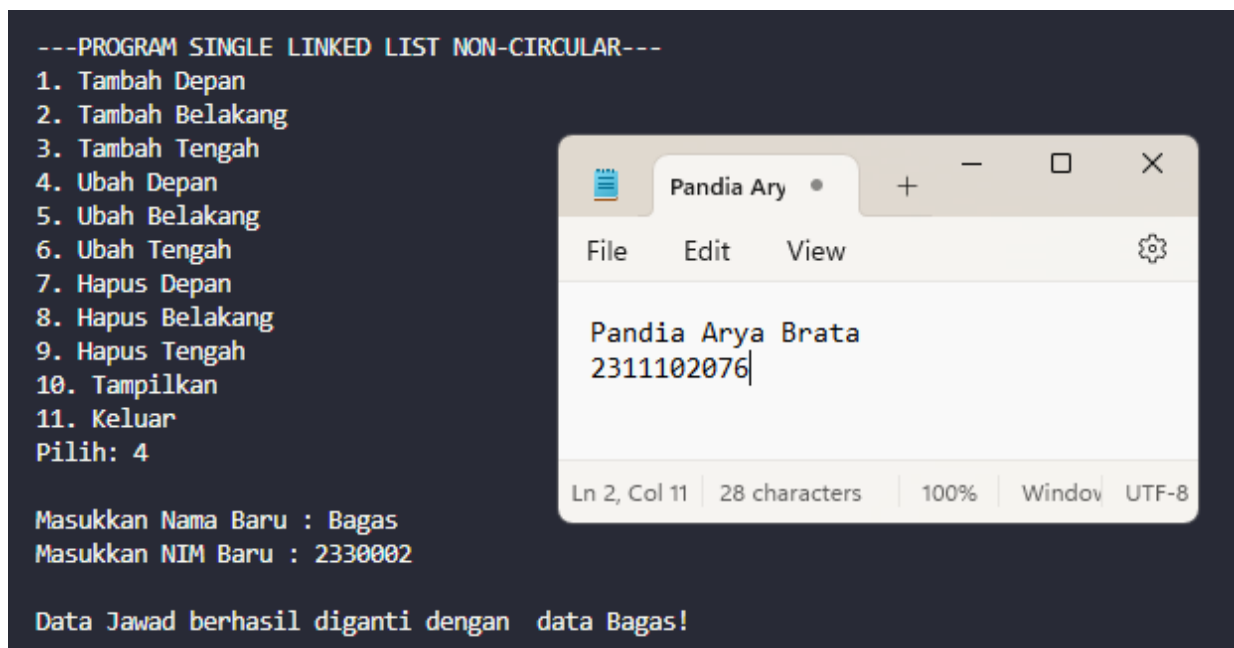
**Screenshot (Ubah data terakhir menjadi Lucy) :**



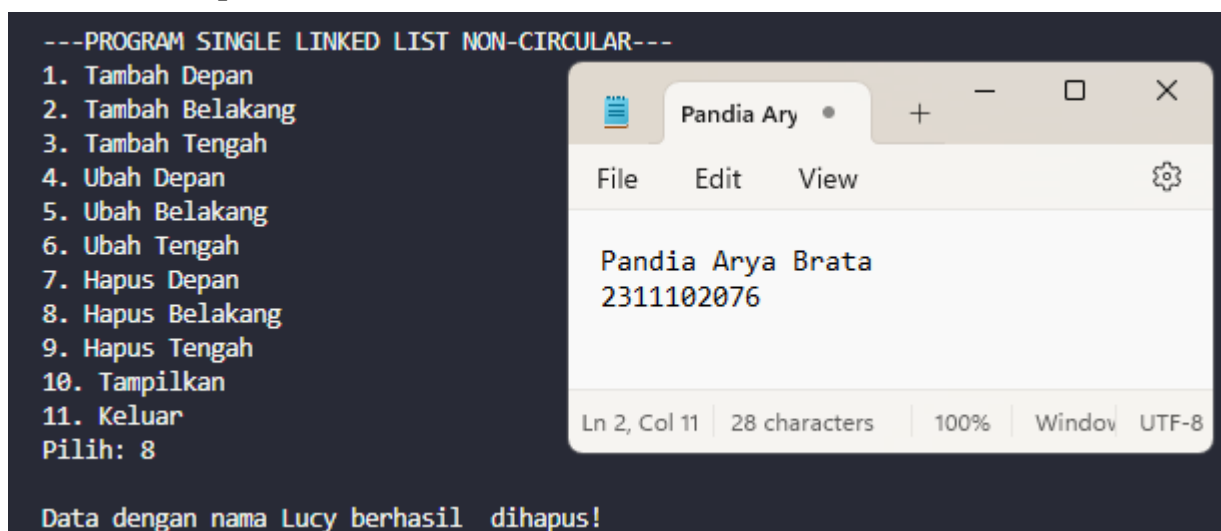
### Screenshot (Hapus data awal) :



### Screenshot (Ubah data awal menjadi Bagus) :



### Screenshot (Hapus data akhir) :



### Screenshot (Tampilan seluruh data) :

```
10. Tampilkan
11. Keluar
Pilih: 10

Data Mahasiswa:
Nama : Bagus
NIM : 23300002

Nama : Pandia
NIM : 2311102076

Nama : Farrel
NIM : 23300003

Nama : Wati
NIM : 23300004

Nama : Anis
NIM : 23300008

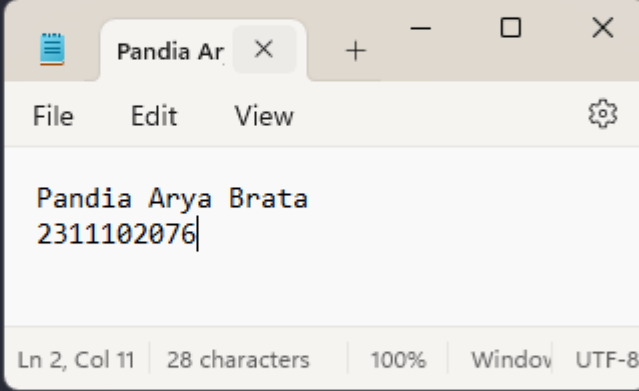
Nama : Bowo
NIM : 23300015

Nama : Gahar
NIM : 23300040

Nama : Idin
NIM : 23300045

Nama : Ucok
NIM : 23300050

Nama : Budi
NIM : 23300099
```



### Deskripsi Program :

Program di atas merupakan program untuk memanipulasi linked list yang berisi data mahasiswa. Program ini menggunakan struktur data linked list untuk menyimpan data mahasiswa dengan nama dan NIM.

Dalam program ini, terdapat beberapa fungsi untuk memanipulasi linked list, seperti tambahDepan(), tambahBelakang(), dan tambahTengah() untuk menambahkan data mahasiswa, ubahDepan(), ubahBelakang(), dan ubahTengah() untuk mengubah data mahasiswa, serta hapusDepan() dan hapusBelakang() untuk menghapus data mahasiswa.

Setiap fungsi ini memanipulasi linked list dengan cara yang berbeda-beda. Misalnya, tambahDepan() menambahkan data di awal linked list, tambahBelakang() menambahkan data di akhir linked list, dan tambahTengah() menambahkan data di tengah linked list sesuai dengan posisi yang ditentukan.

Program ini menggunakan variabel global head, tail, dan curr untuk mengakses elemen linked list, serta menggunakan struct Mahasiswa untuk menyimpan data mahasiswa beserta pointer next yang menghubungkan setiap elemen dalam linked list.

## **KESIMPULAN**

Linked list adalah suatu struktur data yang terdiri dari sekumpulan node yang terhubung satu sama lain melalui pointer. Terdapat dua jenis linked list, yaitu linked list circular dan non-circular.

Linked list circular adalah linked list di mana node terakhir menunjuk kembali ke node pertama, sehingga linked list tersebut membentuk sebuah lingkaran. Sedangkan linked list non-circular adalah linked list di mana node terakhir menunjuk ke NULL atau kosong.

Kedua jenis linked list ini dapat digunakan untuk menyimpan dan mengelola data secara dinamis, seperti menambah, menghapus, dan mengubah data dalam linked list. Kita dapat menggunakan linked list untuk berbagai macam kebutuhan seperti implementasi queue, stack, dan lainnya.

Pada linked list circular, penambahan dan penghapusan data lebih cepat dibandingkan dengan linked list non-circular. Namun, pada linked list non-circular, manipulasi data lebih mudah karena tidak perlu memeriksa apakah pointer node terakhir mengarah ke NULL atau tidak.

Keduanya memiliki kelebihan dan kekurangan masing-masing, sehingga pemilihan jenis linked list tergantung pada kebutuhan dan karakteristik dari aplikasi yang akan dikembangkan.

## **E. DAFTAR PUSTAKA**

Karumanchi, N. (2016). *Data Structures and algorithms made easy: Concepts, problems, Interview Questions*. CareerMonk Publications.