

**LAPORAN PRAKTIKUM  
STRUKTUR DATA DAN ALGORITMA**

**MODUL 4  
“LINKED LIST CIRCULAR DAN NON CIRCULAR”**



**Disusun Oleh :**

Nama : Fitri Kusumaningtyas  
NIM : 2311102068  
Kelas : IF 11 B

**DOSEN:**

WAHYU ANDI SAPUTRA, S.Pd. , M.Eng

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

## A. Dasar Teori

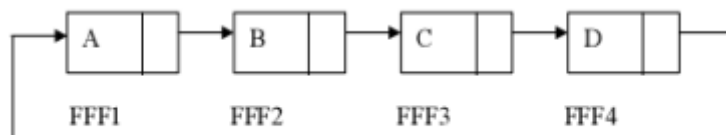
### 1. NON-CIRCULAR

Lingkaran linked list tunggal adalah linked list tunggal yang pointer berikutnya menunjuk ke dirinya sendiri.

- Jika satu linked list terdiri dari beberapa node, pointer berikutnya dari node terakhir menunjuk ke node pertama.

- Definisi : Tunggal : artinya hanya ada satu pointer bidang dan satu arah.

Melingkar: berarti pointer berikutnya menunjuk ke dirinya sendiri dan karenanya berputar.



Ilustrasi SLLC

Deklarasi node baru

```
#include "stdio.h"
typedef struct Tnode
{ int data;
  Tnode *next;
};
node *bantu; //mendeklarasikan pointer bantu
node *baru; //mendeklarasikan pointer baru
node *head; //mendeklarasikan pointer head
node *tail; //mendeklarasikan pointer tail
void main()
{
  head=new node;
  head ->next=head;
}
```

### 2. CIRCULAR

Linked list melingkar adalah jenis linked list yang node pertama dan terakhirnya juga dihubungkan membentuk lingkaran. Pada dasarnya ada dua jenis linked list melingkar.

- a. Circular singly linked list Alamat node terakhir terdiri dari alamat node pertama.



- b. Circular Double Linked List

Di sini, node pertama tidak hanya menyimpan alamat node pertama tetapi juga alamat node terakhir.



### Perbedaan Linked list Non-Circular dan Circular

#### Linked List Non-Circular

- Struktur: Node (elemen) terhubung dalam rantai linear. Setiap node memiliki data dan pointer (next) yang menunjuk ke node berikutnya dalam urutan tersebut.
- Node Terakhir: Pointer next dari node terakhir menunjuk ke NULL, menandakan akhir dari list.
- Penelusuran: Anda hanya dapat melintasi list dalam satu arah, mulai dari head (node pertama) dan mengikuti pointer next sampai Anda mencapai NULL.
- Operasi: Operasi umum termasuk penyisipan (di awal atau akhir), penghapusan (dari awal atau akhir), dan pencarian data tertentu.

#### Linked List Circular

- Struktur: Mirip dengan list non-circular, tetapi pointer next dari node terakhir kembali ke head, membentuk loop tertutup.

- Node Terakhir: Pointer next dari node terakhir menunjuk ke node head, menciptakan siklus berkelanjutan.
- Penelusuran: Anda dapat melintasi list tanpa henti, mulai dari node mana pun dan mengikuti pointer next sampai Anda kembali ke node awal.
- Operasi: Beberapa operasi, seperti menemukan akhir list, memerlukan penanganan khusus karena sifatnya yang circular. Penyisipan dan penghapusan bisa mirip dengan list non-circular, tetapi perlu mempertimbangkan untuk mempertahankan struktur circular.

## B. Guided

### 1. Guided 1

Source code :

```
#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;
    }

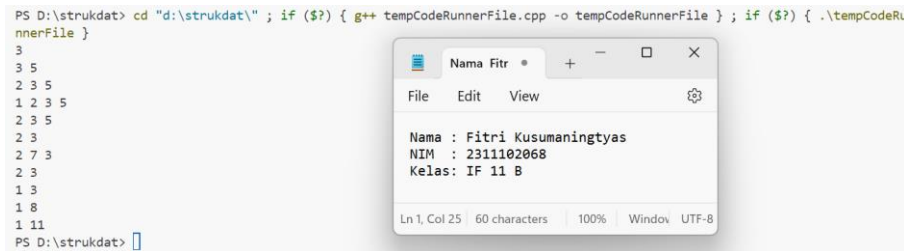
    void tampil() {
        Node *bantu = head;
        if (!isEmpty()) {
            while (bantu != NULL) {
                cout << bantu->data << " ";
                bantu = bantu->next;
            }
            cout << endl;
        } else {
            cout << "List masih kosong!" << endl;
        }
    }

    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;
    }

```

Output :



```
PS D:\strukdat> cd "d:\strukdat\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRu
nnerFile }
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:\strukdat>
```

```
Nama Fitr
File Edit View
Nama : Fitri Kusumaningtyas
NIM : 2311102068
Kelas: IF 11 B
Ln 1, Col 25 | 60 characters | 100% | Window UTF-8
```

Deskripsi program :

Program ini memiliki struktur node yang terdiri dari dua bagian: buah data dan sebuah pointer ke node berikutnya. Program ini memiliki beberapa fungsi:

- `init()`: Fungsi inialisasi linked list akan kosong.
- `isEmpty()`: Berfungsi untuk memeriksa apakah linked list kosong.
- `insertDepan(int nilai)`: Fungsi yang menambahkan node baru dengan data nilai ke awal linked list.
- `insertBelakang(int nilai)`: Fungsi yang menambahkan node baru dengan data nilai ke belakang linked list.
- `hitungList()`: Berfungsi untuk menghitung jumlah node dalam linked list.
- `insertTengah(int data, int Posisi)`: Fungsi yang menambahkan node baru dengan data antar posisi dalam linked list.
- `hapusDepan()`: Berfungsi untuk menghapus node pertama dari linked list.
- `hapusBelakang()`: Berfungsi untuk menghapus node terakhir dari linked list.
- `hapusTengah(int posisi)`: Berfungsi untuk menghapus node dari posisi ke posisi dalam linked list.
- `ubahDepan(int data)`: Fungsi yang mengubah data node pertama menjadi data.
- `ubahTengah (int data, int Posisi)`: Fungsi yang mengubah data node dari posisi ke posisi dalam data.
- `ubahBelakang(int data)`: Fungsi yang mengubah data dari node terakhir menjadi data.
- `clearList()`: Berfungsi untuk menghapus semua node dalam linked list.
- `tampil()`: Berfungsi untuk menampilkan seluruh data dalam linked list.

Di main(), program pertama-tama menggunakan init() untuk membuat linked list kosong. Data kemudian dimasukkan ke dalam linked list menggunakan insertDepan() dan insertBelakang(). Data tersebut kemudian ditampilkan dalam linked list menggunakan tampil(). Untuk menghapus data, gunakan hapusDepan(), hapusBelakang(), dan hapusTengah(). Selain itu, ubahDepan(), ubahTengah(), dan ubahBelakang() digunakan untuk mengubah data dalam linked list. Selain itu, clearList() digunakan untuk menghapus semua data dalam linked list.

## 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()) {
        insertDepan(data);
    } else {
        buatNode(data);
        Node* bantu = head;
        for (int i = 1; i < posisi - 1; i++) {
            bantu = bantu->next;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

void hapusDepan() {
    if (!isEmpty()) {

```

```

        Node* hapus = head;
        if (head == tail) {
            head = nullptr;
            tail = nullptr;
        } else {
            tail->next = head->next;
            head = head->next;
        }
        delete hapus;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

void hapusBelakang() {
    if (!isEmpty()) {
        Node* hapus = tail;
        if (head == tail) {
            head = nullptr;
            tail = nullptr;
        } else {
            Node* bantu = head;
            while (bantu->next != tail) {
                bantu = bantu->next;
            }
            bantu->next = head;
            tail = bantu;
        }
        delete hapus;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

void hapusTengah(int posisi) {
    if (!isEmpty()) {
        if (posisi == 1) {
            hapusDepan();
        } else {
            Node* bantu = head;
            for (int i = 1; i < posisi - 1; i++) {
                bantu = bantu->next;
            }
            Node* hapus = bantu->next;
            bantu->next = hapus->next;

```

```

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

void clearList() {
    while (!isEmpty()) {
        hapusDepan();
    }
    cout << "List berhasil terhapus!" << endl;
}

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

// Main Function
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 :

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

```
PS D:\strukdat> cd "d:\strukdat\" ; if ($?) { g++ prak4g2.cpp -o prak4g2 } ; if ($?) { .\prak4g2 }
Ayam
Bebek Ayam
Bebek Ayam Cicak
Bebek Ayam Cicak Domba
Bebek Ayam
Ayam
Ayam Sapi
Ayam
PS D:\strukdat>
```

Overlaid on the terminal is a text editor window titled 'Nama Fitr'. It contains the following text:

```
Nama : Fitri Kusumaningtyas
NIM : 2311102068
Kelas: IF 11 B
```

The text editor's status bar at the bottom indicates 'Ln 1, Col 25 | 60 characters | 100% | Window | UTF-8'.

Deskripsi program :

Program ini merupakan implementasi dari single linked list melingkar dengan menggunakan bahasa C++. Struktur data berikut digunakan:

- Node: Struktur data yang berisi dua elemen: data dan next.
- data adalah string yang disimpan dalam node dan next adalah pointer ke node berikutnya.
- head, tail, new, help, delete: Variabel global digunakan sebagai pointer ke node dalam linked list.

Fungsi berikut tersedia dalam program ini.

- init(): Fungsi inisialisasi linked list. Inisialisasi dilakukan dengan menyetel head dan tail ke NULL.
- isEmpty(): Berfungsi untuk memeriksa apakah linked list kosong.
- Fungsi ini mengembalikan 1 jika linked list kosong dan 0 jika linked list tidak kosong.
- buatNode (string data): Fungsi yang membuat node baru dengan data yang ditentukan.
- hitungList(): Berfungsi untuk menghitung jumlah node dalam linked list.



- insertDepan (string data): Fungsi yang menambahkan node baru dengan data tertentu ke awal linked list.
- insertBelakang (string data): Fungsi yang menambahkan node baru dengan data tertentu ke bagian belakang linked list.
- insertTengah(string data, int Position): Fungsi yang menambahkan node baru dengan data tertentu pada posisi tertentu dalam linked list.
- hapusDepan(): Berfungsi untuk menghapus node pertama dari linked list.
- hapusBelakang(): Berfungsi untuk menghapus node terakhir dari linked list.
- hapusTengah(intposition): fungsi yang menghapus node pada posisi tertentu dalam linked list.
- clearList(): Berfungsi untuk menghapus semua node dalam linked list.
- tampil(): Fungsi yang menampilkan data untuk setiap node dalam linked list.

Program ini menyatakan bahwa linked list bersifat melingkar. Dalam fungsi main(), linked list diinisialisasi dengan memanggil fungsi init(). Data kemudian dimasukkan ke dalam linked list menggunakan fungsi insertDepan(), insertBelakang(), dan insertTengah(). Data dalam linked list kemudian ditampilkan menggunakan fungsi tampil(). Fungsi hapusDepan(), hapusBelakang(), dan hapusTengah() digunakan untuk menghapus data. Selain itu, fungsi clearList() digunakan untuk menghapus semua data dalam linked list.

## C. Unguided

### 1. Unguided 1

Source code :

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

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

class LinkedList {
private:
    mahasiswa* head;

public:
    LinkedList() {
        head = NULL;
    }

    void insertDepan(string nama, string nim) {
        mahasiswa* newNode = new mahasiswa();
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = head;
        head = newNode;
        cout << endl << "Data telah ditambahkan"<<endl;
    }

    void insertBelakang(string nama, string nim) {
        mahasiswa* newNode = new mahasiswa();
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = NULL;
        if (head == NULL) {
            head = newNode;
        } else {
            mahasiswa* temp = head;
            while (temp->next != NULL) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
    }
};
```

```

        }
        temp->next = newNode;
    }
    cout << endl << "Data telah ditambahkan"<<endl;
}

void insertTengah(string nama, string nim, int posisi) {
    if (posisi < 1) {
        cout << endl << "Posisi tidak valid"<<endl;
        return;
    }
    mahasiswa* newNode = new mahasiswa();
    newNode->nama = nama;
    newNode->nim = nim;
    if (posisi == 1) {
        newNode->next = head;
        head = newNode;
    } else {
        mahasiswa* temp = head;
        for (int i = 1; i < posisi - 1; i++) {
            if (temp == NULL) {
                cout << endl << "Posisi tidak valid"<<endl;
                return;
            }
            temp = temp->next;
        }
        newNode->next = temp->next;
        temp->next = newNode;
    }
    cout << endl << "Data telah ditambahkan"<<endl;
}

void ubahDepan(string nama, string nim) {
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    head->nama = nama;
    head->nim = nim;
    cout << endl << "Data di depan telah diubah"<<endl;
}

void ubahBelakang(string nama, string nim) {
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
    }
}

```

```

        return;
    }
    mahasiswa* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->nama = nama;
    temp->nim = nim;
    cout << endl << "Data di belakang telah diubah"<<endl;
}

void ubahTengah(string nama, string nim, int posisi) {
    if (posisi < 1) {
        cout << endl << "Posisi tidak valid"<<endl;
        return;
    }
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    mahasiswa* temp = head;
    for (int i = 1; i < posisi; i++) {
        if (temp == NULL) {
            cout << endl << "Posisi tidak valid"<<endl;
            return;
        }
        temp = temp->next;
    }
    temp->nama = nama;
    temp->nim = nim;
    cout << endl << "Data di posisi " << posisi << " telah
diubah"<<endl;
}

void hapusDepan() {
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    mahasiswa* temp = head;
    head = head->next;
    delete temp;
    cout << endl << "Data di depan telah dihapus"<<endl;
}

```

```

void hapusBelakang() {
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    if (head->next == NULL) {
        delete head;
        head = NULL;
        cout << endl << "Data di belakang telah
dihapus"<<endl;
        return;
    }
    mahasiswa* temp = head;
    while (temp->next->next != NULL) {
        temp = temp->next;
    }
    delete temp->next;
    temp->next = NULL;
    cout << endl << "Data di belakang telah dihapus"<<endl;
}

void hapusTengah(int posisi) {
    if (posisi < 1) {
        cout << endl << "Posisi tidak valid"<<endl;
        return;
    }
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    mahasiswa* temp = head;
    if (posisi == 1) {
        head = head->next;
        delete temp;
        cout << endl << "Data di posisi " << posisi << "
telah dihapus"<<endl;
        return;
    }
    for (int i = 1; i < posisi - 1; i++) {
        if (temp == NULL || temp->next == NULL) {
            cout << endl << "Posisi tidak valid"<<endl;
            return;
        }
        temp = temp->next;
    }
}

```

```

        mahasiswa* nextNode = temp->next->next;
        delete temp->next;
        temp->next = nextNode;
        cout << endl << "Data di posisi " << posisi << " telah
dihapus"<<endl;
    }

    void print() {
        if (head == NULL) {
            cout << endl << "Linked List kosong"<<endl;
            return;
        }
        cout << "----- DATA MAHASISWA -----"<< endl;
        cout << setw(20) << left << "NAMA" << setw(20) << left
<< "NIM" << endl;
        mahasiswa* temp = head;
        while (temp != NULL) {
            cout << setw(20) << left << temp->nama << setw(20)
<< left << temp->nim << endl;
            temp = temp->next;
        }
    }

    void deleteAll() {
        while (head != NULL) {
            mahasiswa* temp = head;
            head = head->next;
            delete temp;
        }
        cout << endl << "Semua data mahasiswa telah
dihapus"<<endl;
    }
};

int main() {
    LinkedList linkedList;
    int pilihan;
    string nama, nim;
    int posisi;
    while (true) {
        cout << endl;
        linkedList.print();
        cout << endl;
        cout << "=====MENU
PILIHAN===== " << endl;

```

```

cout << endl ;
cout << "1. Tambah Depan"<<endl;
cout << "2. Tambah Belakang"<<endl;
cout << "3. Tambah Tengah"<<endl;
cout << "4. Ubah Depan"<<endl;
cout << "5. Ubah Belakang"<<endl;
cout << "6. Ubah Tengah"<<endl;
cout << "7. Hapus Depan"<<endl;
cout << "8. Hapus Belakang"<<endl;
cout << "9. Hapus Tengah"<<endl;
cout << "10. Hapus List"<<endl;
cout << "11. Tampilkan"<<endl;
cout << "12. Keluar"<<endl;

cout << endl;

cout << "Pilihan : ";
cin >> pilihan;

cout << endl;

switch (pilihan) {
    case 1:
        cout << "----Tambah Depan----"<<endl;
        cout << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        linkedList.insertDepan(nama, nim);
        break;
    case 2:
        cout << "----Tambah Belakang----"<<endl;
        cout << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        linkedList.insertBelakang(nama, nim);
        break;
    case 3:
        cout << "----Tambah Tengah----"<<endl;
        cout << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;

```

```

        cout << "Masukkan NIM : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.insertTengah(nama, nim, posisi);
        break;
    case 4:
        cout << "----Ubah Depan----"<<endl;
        cout << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        linkedList.ubahDepan(nama, nim);
        break;
    case 5:
        cout << "----Ubah Belakang----"<<endl;
        cout << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        linkedList.ubahBelakang(nama, nim);
        break;
    case 6:
        cout << "----Ubah Tengah----"<<endl;
        cout << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.ubahTengah(nama, nim, posisi);
        break;
    case 7:
        cout << "----Hapus Depan----"<<endl;
        cout << endl;
        linkedList.hapusDepan();
        break;
    case 8:
        cout << "----Hapus Belakang----"<<endl;
        cout << endl;
        linkedList.hapusBelakang();
        break;

```



```

        case 9:
            cout << "----Hapus Tengah----"<<endl;
            cout << endl;
            cout << "Masukkan Posisi : ";
            cin >> posisi;
            linkedList.hapusTengah(posisi);
            break;
        case 10:
            cout << "----Hapus List----"<<endl;
            cout << endl;
            linkedList.deleteAll();
            break;
        case 11:
            cout << "----Tampilkan----"<<endl;
            cout << endl;
            linkedList.print();
            break;
        case 12:
            exit(12);
        default:
            cout << "Pilihan tidak valid!"<<endl;
    }
}

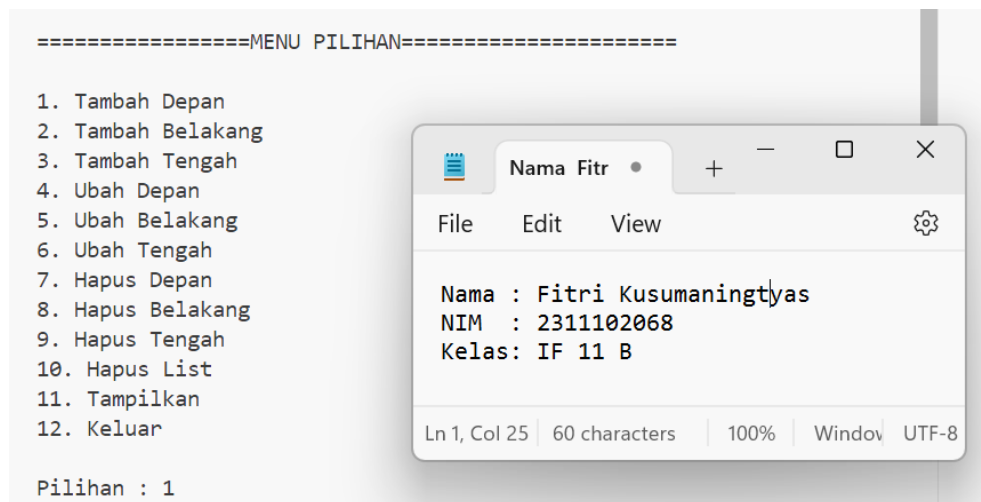
return 0;
}

```

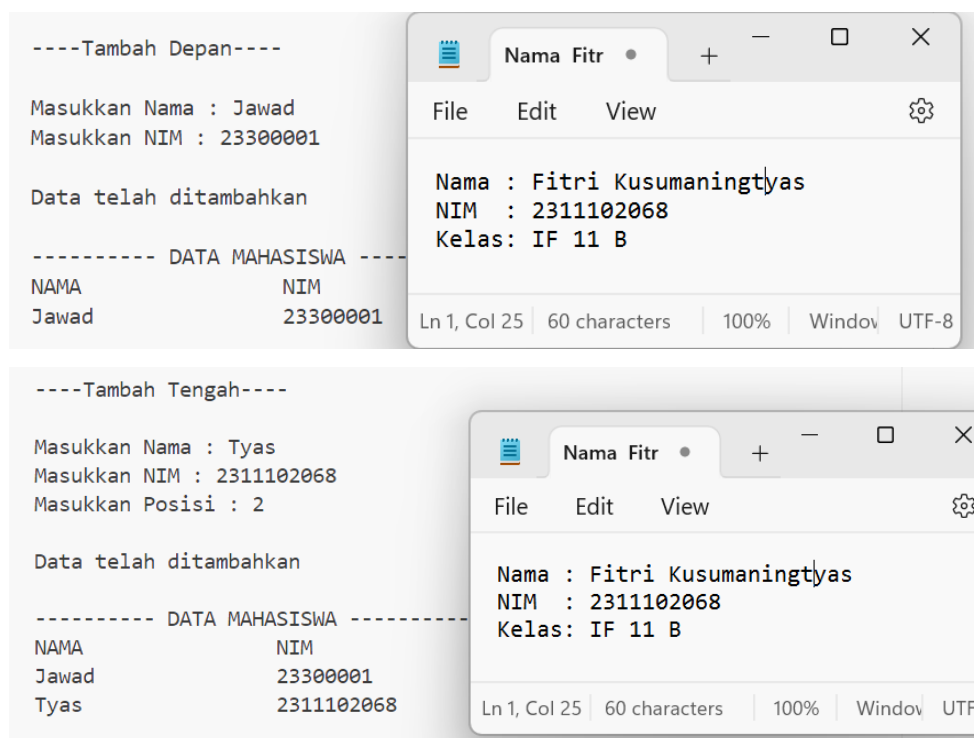
Output :

1).

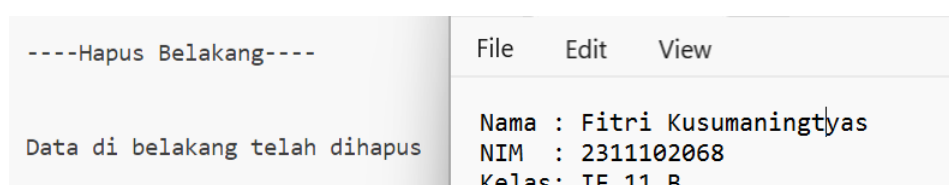
a) Tampilan menu

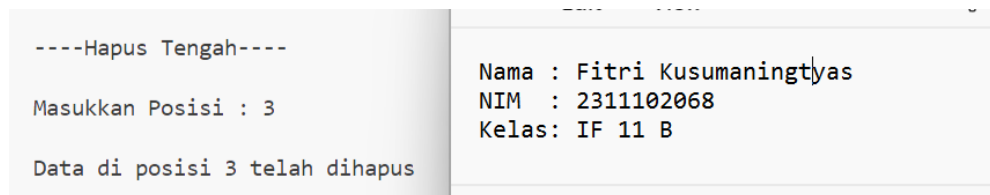


#### b) Tampilan operasi Tambah

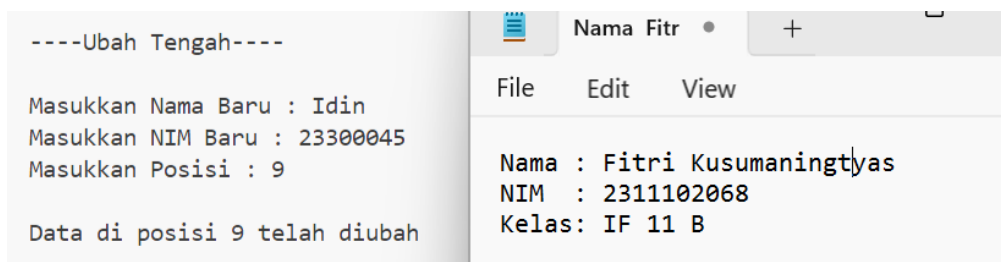
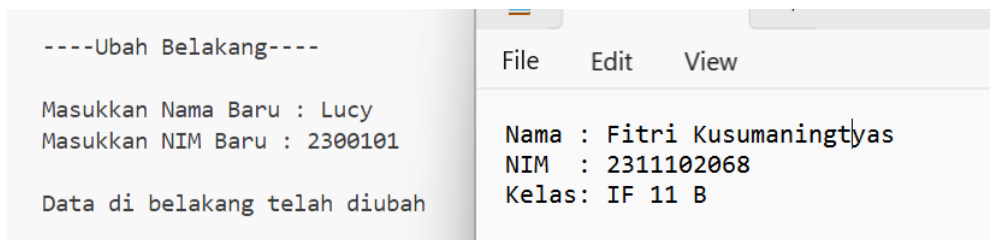


#### c) Tampilan Operasi Hapus

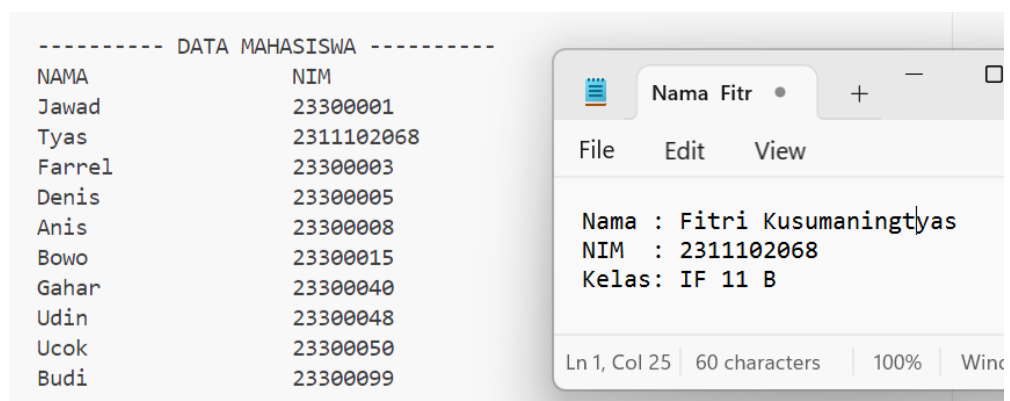




d) Tampilan Operasi Ubah

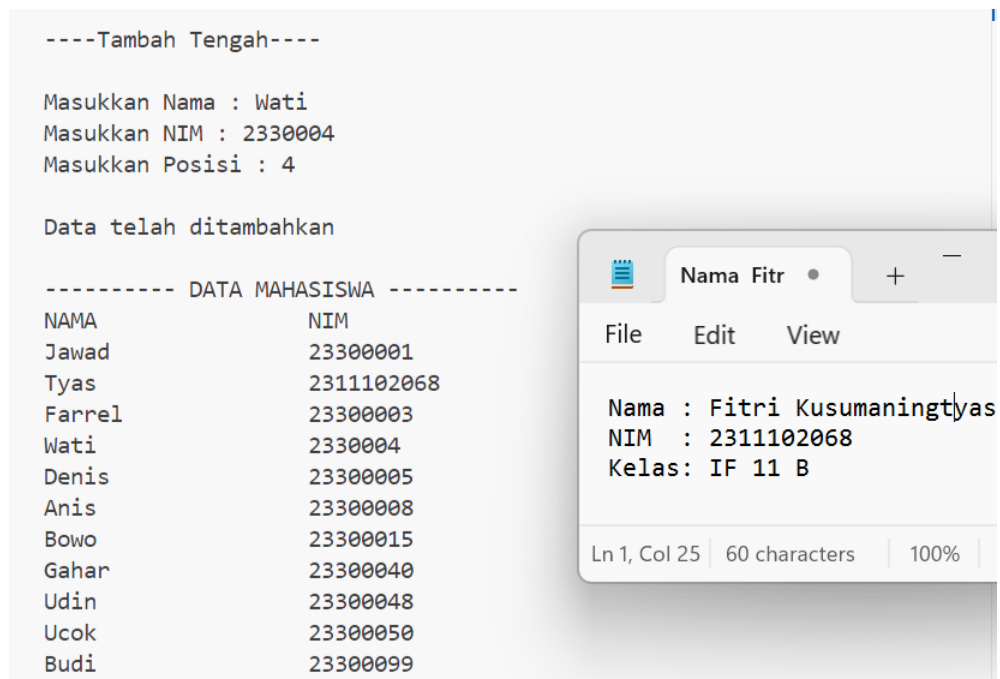


e) Tampilan Operasi Tampil data

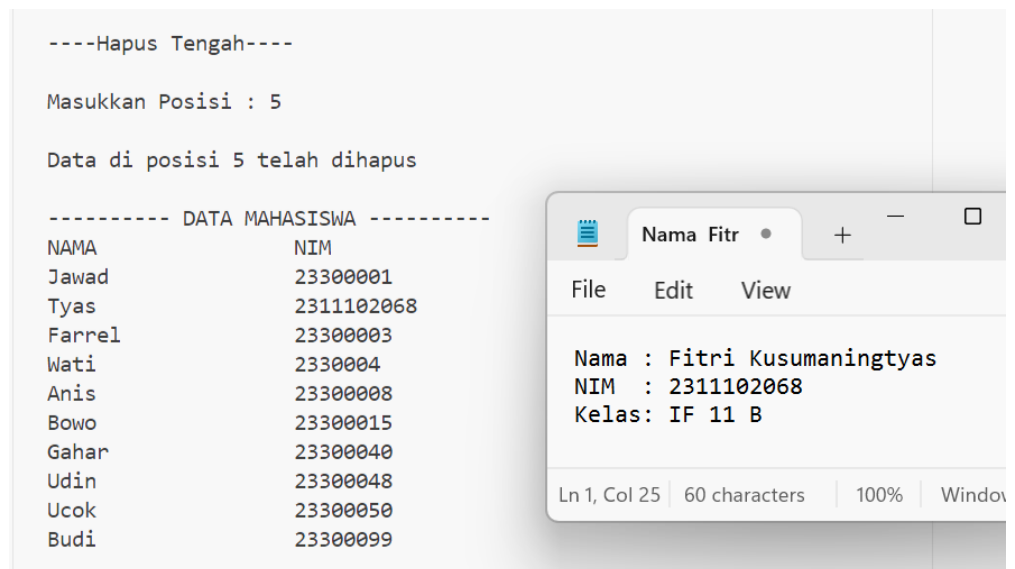


2)

a). Tambahkan data diantara Farrel dan Denis



b). Hapus data Denis



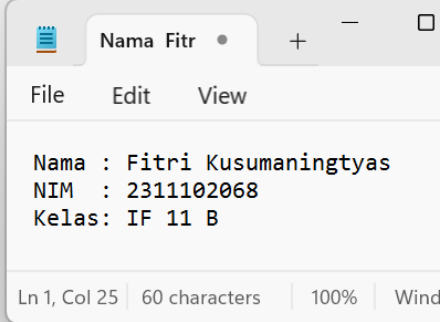
c). Tambah data di awal

```
-----Tambah Depan-----

Masukkan Nama : Owi
Masukkan NIM : 2330000

Data telah ditambahkan

----- DATA MAHASISWA -----
NAMA          NIM
Owi            2330000
Jawad          23300001
Tyas           2311102068
Farrel         23300003
Wati           23300004
Anis           23300008
Bowo           23300015
Gahar          23300040
Udin           23300048
Ucok           23300050
Budi           23300099
```



The screenshot shows a text editor window titled "Nama Fitr" with a menu bar (File, Edit, View). The content of the window matches the terminal output, showing the student data table with "Owi" added at the top. The status bar at the bottom indicates "Ln 1, Col 25 | 60 characters | 100% | Wind".

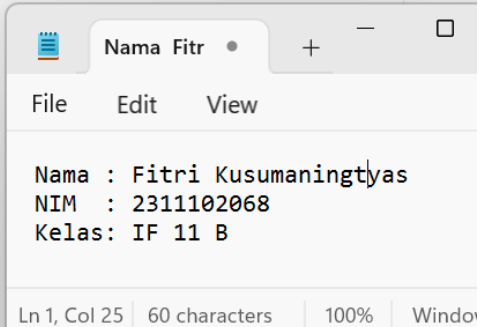
d). Tambah di Akhir

```
-----Tambah Belakang-----

Masukkan Nama : David
Masukkan NIM : 2300100

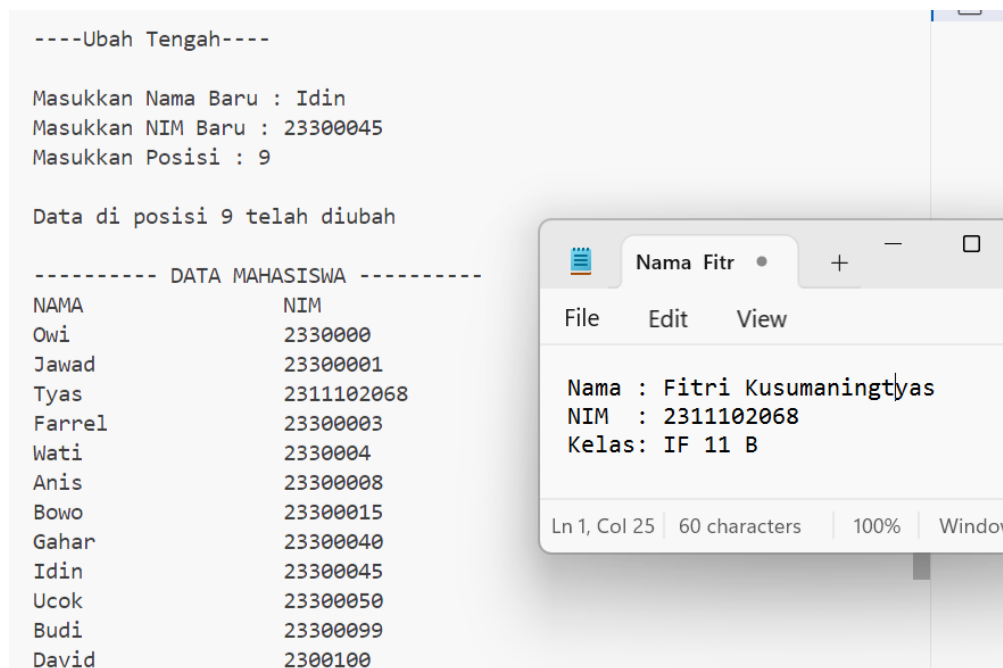
Data telah ditambahkan

----- DATA MAHASISWA -----
NAMA          NIM
Owi            2330000
Jawad          23300001
Tyas           2311102068
Farrel         23300003
Wati           23300004
Anis           23300008
Bowo           23300015
Gahar          23300040
Udin           23300048
Ucok           23300050
Budi           23300099
David          2300100
```

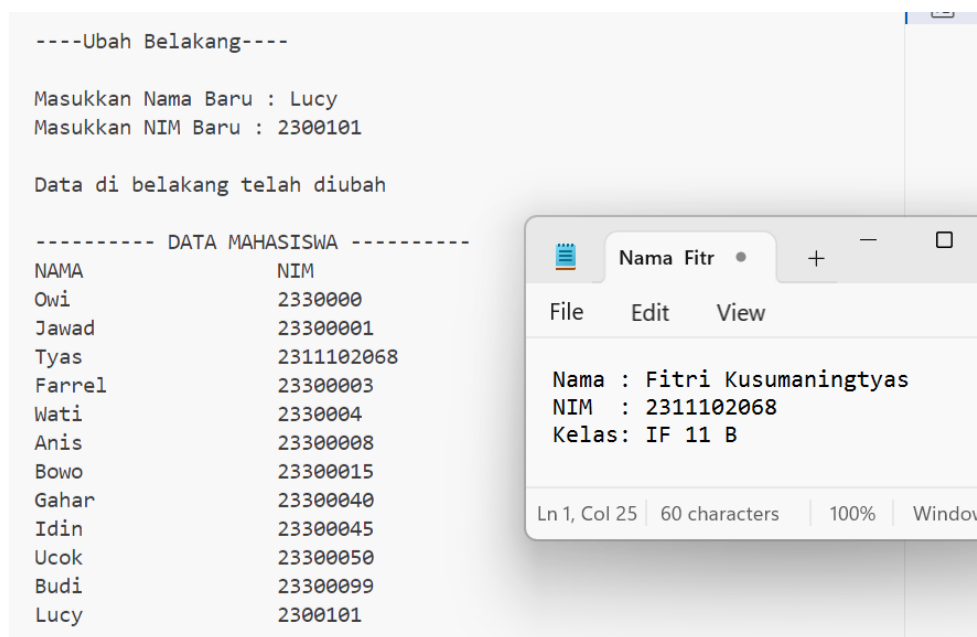


The screenshot shows a text editor window titled "Nama Fitr" with a menu bar (File, Edit, View). The content of the window matches the terminal output, showing the student data table with "David" added at the bottom. The status bar at the bottom indicates "Ln 1, Col 25 | 60 characters | 100% | Window".

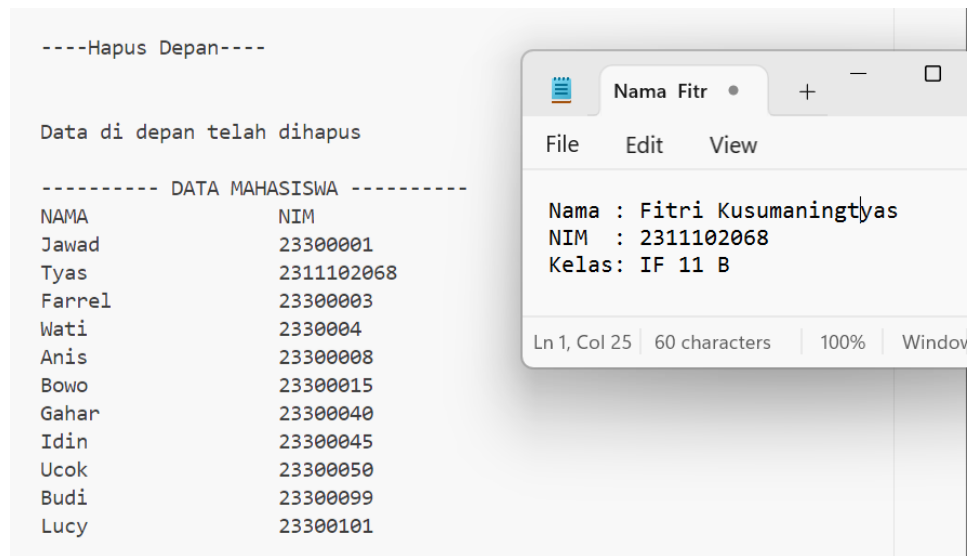
e). Ubah data Udin



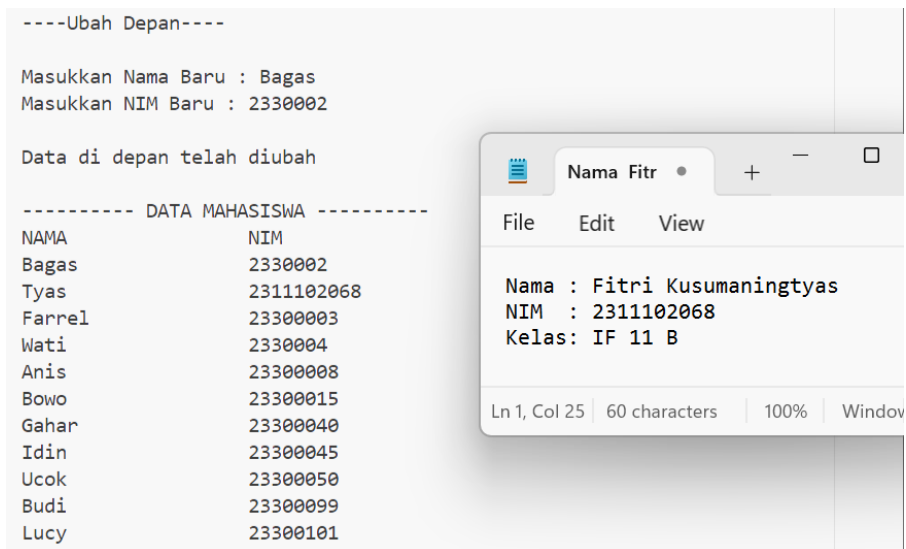
f). Ubah data David



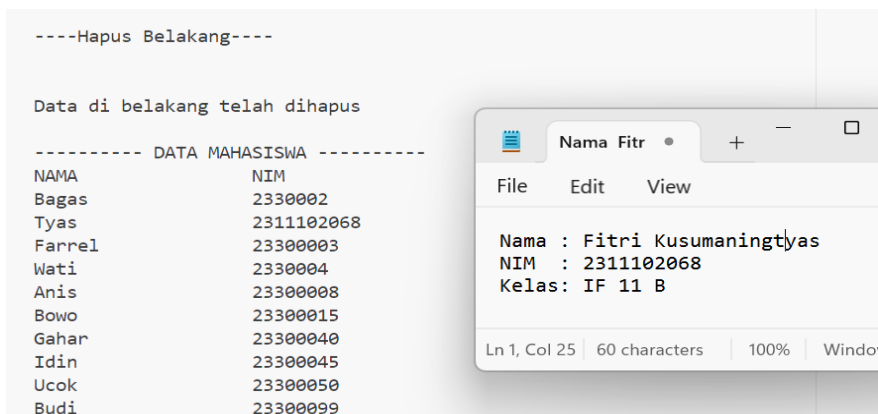
g). Hapus data awal



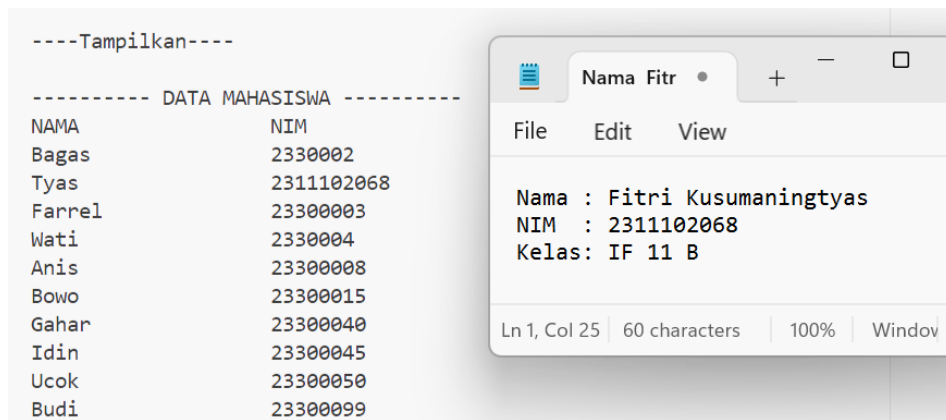
h). Ubah data awal



i)> Hapus data Akhir



j). Tampilkan seluruh Data



The screenshot shows a C++ program's output in a terminal window. The output displays a table of student data under the heading "DATA MAHASISWA". The table has two columns: "NAMA" and "NIM". The data rows are: Bagas (2330002), Tyas (2311102068), Farrel (23300003), Wati (23300004), Anis (23300008), Bowo (23300015), Gahar (23300040), Idin (23300045), Ucok (23300050), and Budi (23300099). To the right of the terminal, there is a window titled "Nama Fitr" showing the details of a selected student: Nama : Fitri Kusumaningtyas, NIM : 2311102068, and Kelas: IF 11 B. The window also shows a status bar with "Ln 1, Col 25", "60 characters", "100%", and "Window".

```
----Tampilkan----  
  
----- DATA MAHASISWA -----  
NAMA      NIM  
Bagas     2330002  
Tyas      2311102068  
Farrel     23300003  
Wati       23300004  
Anis       23300008  
Bowo       23300015  
Gahar      23300040  
Idin       23300045  
Ucok       23300050  
Budi       23300099
```

Nama Fitr

Nama : Fitri Kusumaningtyas  
NIM : 2311102068  
Kelas: IF 11 B

Ln 1, Col 25 | 60 characters | 100% | Window

Deskripsi Program :

Program ini menerapkan konsep linked list untuk menyimpan data siswa. Data mahasiswa terdiri dari nama dan NIM (Nomor Induk Mahasiswa). Program ini menyediakan berbagai operasi seperti menambah, mengubah, menghapus, dan melihat data siswa dalam linked list. Berikut penjelasan program:

- `#include` dan `#include`: Berisi file header yang diperlukan untuk penggunaan input/output dan manipulasi output.
- `#include`: Menyertakan file header yang diperlukan untuk menggunakan tipe data string.
- `using namespace std;`: using namespace standar.
- Struct mahasiswa: Membuat struct dengan nama Student dengan tiga variabel. Nama dan nim bertipe string, dan next adalah pointer ke struktur Student yang digunakan untuk menyimpan alamat node berikutnya.
- Class LinkedList: Membuat kelas bernama LinkedList yang mengimplementasikan konsep linked list.
- `LinkedList()`: Konstruktor untuk kelas LinkedList yang menginisialisasi variabel head ke NULL.
- `void insertDepan(nama string, string nim)`: Fungsi yang menambahkan data siswa ke awal linked list.
- `void insertBelakang(nama string, string nim)`: Fungsi yang menambahkan data siswa ke akhir linked list.



- void insertTengah (nama string, string nim, int Posisi): Fungsi yang menambahkan data siswa ke posisi tertentu di tengah linked list.
- void ubahDepan(nama string, string nim): Fungsi yang mengubah data siswa di awal linked list.
- void ubahBelakang(nama string, string nim): Fungsi yang mengubah data siswa di akhir linked list.
- void ubahTengah(string nama, string nim, int Posisi): Fungsi yang mengubah data siswa pada posisi tertentu di tengah linked list.
- void hapusDepan(): Berfungsi untuk menghapus data siswa di awal linked list.
- void hapusBelakang(): Berfungsi untuk menghapus data siswa di akhir linked list.
- void hapusTengah(int posisi): Berfungsi untuk menghapus data siswa pada posisi yang ditentukan di tengah-tengah linked list.
- void print(): Berfungsi untuk menampilkan seluruh data siswa pada linked list.
- void deleteAll(): Berfungsi untuk menghapus seluruh data siswa pada linked list.
- int main(): fungsi utama program. Menampilkan menu pilihan untuk melakukan operasi pada linked list.
- LinkedList LinkedInList;: Membuat objek kelas LinkedList bernama LinkedInList.
- while (true): Perulangan yang berjalan terus menerus hingga pengguna menghentikan program.
- Switch (pilihan) Gunakan pernyataan sakelar untuk memilih operasi yang akan dilakukan bergantung pada opsi yang dipilih oleh pengguna.
- Case 1 – 12 : Menjelaskan tindakan yang akan dilakukan untuk setiap opsi yang dipilih oleh pengguna.

#### D. Kesimpulan

Linked list non-circular lebih fleksibel untuk operasi list secara umum. Sedangkan Linked list circular berguna ketika diperlukan untuk terus menelusuri list atau saat perlu untuk merepresentasikan struktur data circular seperti antrean.

#### E. Referensi jurnal

Utami, E., Kom, M., Duhita, W. M. P., Kom, S., & Kom, M. (2017). *Langkah Mudah Belajar Struktur Data Menggunakan C/C++*. Elex Media Komputindo.

AKSU, M., & KARCI, A. (2015). Skip ring/circular skip list: circular linked list based new data structure. *structure*, 6(5).

Agrawal, A., O'Connor, M., Bolotin, E., Chatterjee, N., Emer, J., & Keckler, S. (2016, October). CLARA: Circular linked-list auto and self refresh architecture. In *Proceedings of the Second International Symposium on Memory Systems* (pp. 338-349).

Nugraha, A. S. (2019). Artikel Double Linked List Circulate.

kurniman Putra, A. List linier (linked list) dan variasinya\_struktur data.