LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA

MODUL IV
LINKED LIST CIRCULAR DAN NON CIRCULAR

Disusun Oleh :

DIO GILBRAN PRAMANA

NIM : 2311102062

Dosen

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

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

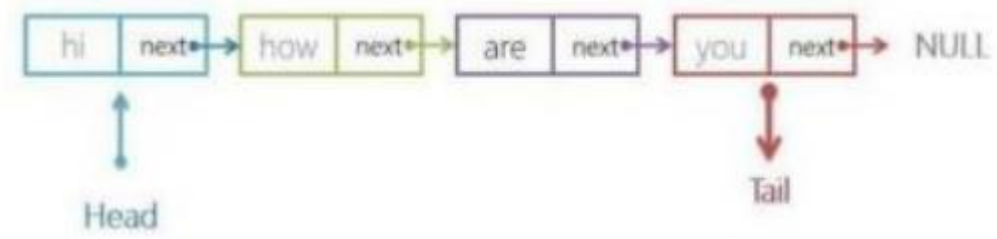
INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

A. DASAR TEORI

1. Linked List Non Circular

Linked list 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.



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)

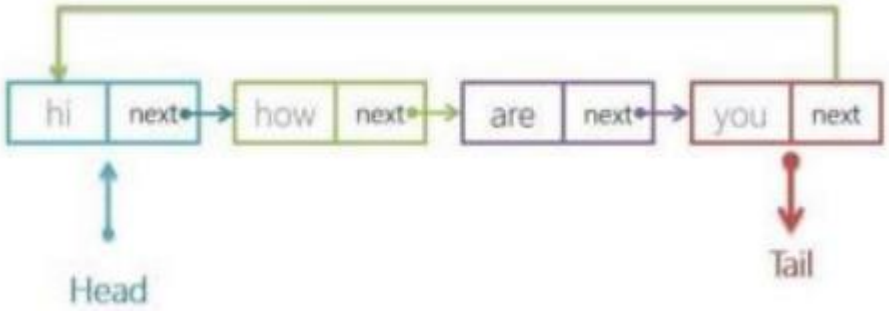
```
void hapusDepan()
{
    if (isEmpty() == true)
    {
        cout << "List kosong!" <<
    endl; }
    else
    {
        node *helper;
        helper = head;
        if (head == tail)
        {
            head = NULL;
            tail = NULL;
            delete helper;
        }
        else
 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



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;
      }
}
```

B. GUIDED

GUIDED 1

```cpp
#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() {
```

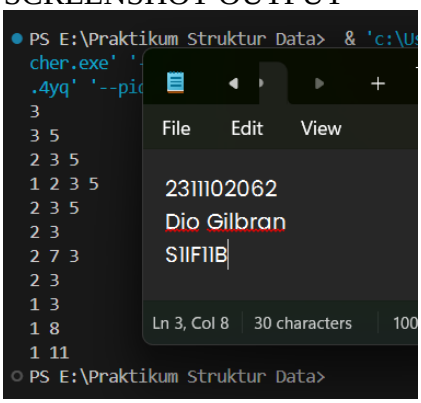
```
    Node *hitung = head;
    int jumlah = 0;
    while (hitung != NULL) {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

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 OUTPUT



DESKRIPSI

Program ini mengimplementasikan operasi dasar pada singly linked list menggunakan C++. Linked list diwakili oleh struktur Node yang memiliki anggota data dan next, sementara pointer global head dan tail digunakan untuk melacak Node pertama dan terakhir. Program ini menyediakan fungsi untuk inisialisasi (init), pengecekan apakah list kosong (isEmpty), penambahan Node di depan (insertDepan) dan di belakang (insertBelakang), serta di posisi tengah (insertTengah). Selain itu, terdapat fungsi untuk penghapusan Node dari depan (hapusDepan), belakang (hapusBelakang), dan posisi tengah (hapusTengah). Program juga memungkinkan pengubahan nilai Node di depan (ubahDepan), tengah (ubahTengah), dan belakang (ubahBelakang), serta menyediakan fungsi untuk menghapus seluruh Node dalam list (clearList) dan menampilkan semua elemen list (tampil). Fungsi main mengilustrasikan penggunaan semua operasi ini dengan serangkaian contoh sederhana.

GUIDED 2

```cpp
#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) {
```

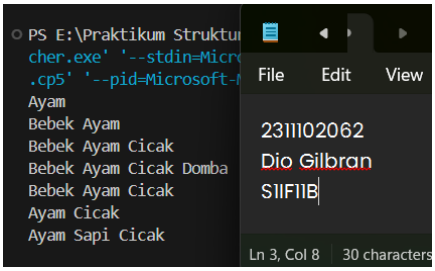
```
        jumlah++;
        bantu = bantu->next;
    }
    return jumlah;
}

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 OUTPUT



DESKRIPSI

Program ini mengimplementasikan operasi dasar pada circular singly linked list menggunakan C++. Struktur Node menyimpan data dan pointer ke Node berikutnya, dengan pointer global head, tail, baru, bantu, dan hapus untuk melacak Node pada linked list. Fungsi init menginisialisasi list, sementara isEmpty mengecek apakah list kosong. Fungsi buatNode membuat Node baru, dan hitungList menghitung jumlah Node dalam list. Operasi penambahan Node dilakukan oleh insertDepan, insertBelakang, dan insertTengah, sedangkan operasi penghapusan Node dilakukan oleh hapusDepan, hapusBelakang, dan hapusTengah. Fungsi clearList menghapus semua Node dalam list, dan tampil menampilkan semua data dalam list. Fungsi main mengilustrasikan penggunaan operasi-operasi ini dengan menambahkan, menghapus, dan menampilkan Node pada linked list.

C. UNGUIDED

UNGUIDED 1

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
```

```
-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
```

```cpp
#include <iostream>
#include <iomanip>
using namespace std;
struct Node
{
    string nama;
    int nim;
    Node *next;
};
Node *head;
Node *tail;
void init()
{
    head = NULL;
    tail = NULL;
}
bool isEmpty()
{
    return head == NULL;
}
void insertDepan(string nama, int nim)
{
    Node *baru = new Node;
    baru->nama = nama;
    baru->nim = nim;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}
void insertBelakang(string nama, int nim)
{
    Node *baru = new Node;
    baru->nama = nama;
    baru->nim = nim;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
```

```
    {
        tail->next = baru;
        tail = baru;
    }
}
int hitungList()
{
    Node *hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}
void insertTengah(string nama, int nim, int posisi)
{
    if (posisi < 1 || posisi > hitungList() + 1)
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        Node *baru = new Node();
        baru->nama = nama;
        baru->nim = nim;
        Node *bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}
void hapusTengah(int posisi)
{
    if (!isEmpty())
    {
        if (posisi < 1 || posisi > hitungList())
```

```
        {
            cout << "Posisi diluar jangkauan" << endl;
        }
        else
        {
            Node *hapus;
            Node *bantu = head;
            Node *prev = nullptr;
            int nomor = 1;
            while (nomor < posisi)
            {
                prev = bantu;
                bantu = bantu->next;
                nomor++;
            }
            hapus = bantu;
            if (prev != nullptr)
            {
                prev->next = bantu->next;
            }
            else
            {
                head = bantu->next;
            }
            if (bantu == tail)
            {
                tail = prev;
            }
            delete hapus;
            cout << "Node pada posisi " << posisi << " berhasil
dihapus " << endl;
        }
    }
    else
    {
        cout << "List masih kosong" << endl;
    }
}
void ubahTengah(string nama, int nim, string nama_lama)
{
    if (!isEmpty())
    {
        Node *bantu = head;
        while (bantu != nullptr && bantu->nama != nama_lama)
        {
            bantu = bantu->next;
        }
```

```
        if (bantu != nullptr)
        {
            bantu->nama = nama;
            bantu->nim = nim;
            cout << "Node dengan nama '" << nama_lama << "'
berhasil diubah menjadi '" << nama << "' dengan NIM baru " << nim
<< endl;
        }
        else
        {
            cout << "Node dengan nama '" << nama_lama << "' tidak
ditemukan " << endl;
        }
    }
    else
    {
        cout << "List masih kosong" << endl;
    }
}
void hapusDepan()
{
    if (!isEmpty())
    {
        Node *hapus = head;
        head = head->next;
        delete hapus;
        cout << "Node pertama berhasil dihapus" << endl;
    }
    else
    {
        cout << "List masih kosong" << endl;
    }
}
void hapusBelakang()
{
    if (!isEmpty())
    {
        if (head == tail)
        {
            delete head;
            head = tail = NULL;
        }
        else
        {
            Node *bantu = head;
            while (bantu->next != tail)
            {
```

```
                bantu = bantu->next;
            }
            delete tail;
            tail = bantu;
            tail->next = NULL;
        }
        cout << "Node terakhir berhasil dihapus" << endl;
    }
    else
    {
        cout << "List masih kosong" << endl;
    }
}
void hapusList()
{
    while (!isEmpty())
    {
        hapusDepan();
    }
}
void tampil()
{
    cout << "DATA MAHASISWA\n\n";
    cout << "---------------------------------\n";
    cout << setw(15) << left << "NAMA" << setw(10) << "NIM" <<
endl;
    cout << "---------------------------------\n";
    if (!isEmpty())
    {
        Node *bantu = head;
        while (bantu != NULL)
        {
            cout << setw(15) << left << bantu->nama << setw(10)
                 << bantu->nim << endl;
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong" << endl;
    }
    cout << "---------------------------------\n";
}
int main()
{
    init();
```

```
    string nama;
    int nim;
    int choice;
    cout << "Masukkan nama anda: ";
    cin >> nama;
    cout << "Masukkan NIM anda: ";
    cin >> nim;
    insertBelakang(nama, nim);
    while (true)
    {
        cout << "\nPROGRAM SINGLE LINKED LIST NON-CIRCULAR\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. KELUAR" << endl;
        cout << "Pilih Operasi: ";
        cin >> choice;
        switch (choice)
        {
        case 1:
        {
            cout << "-Tambah Data Depan-\n"
                 << endl;
            cout << "Masukkan Nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            insertDepan(nama, nim);
            cout << "Data telah ditambahkan yaa";
            break;
        }
        case 2:
        {
            cout << "-Tambah Data Belakang-\n"
                 << endl;
            cout << "Masukkan Nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
```

```
            insertBelakang(nama, nim);
            cout << "Data telah ditambahkan yaa";
            break;
        }
        case 3:
        {
            cout << "-Tambah Data Tengah-\n"
                 << endl;
            int posisi;
            cout << "Masukkan Nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            cout << "Masukkan posisi: ";
            cin >> posisi;
            insertTengah(nama, nim, posisi);
            cout << "Data telah ditambahkan yaa";
            break;
        }
        case 4:
        {
            cout << "-Ubah Data Depan-\n"
                 << endl;
            string nama_baru;
            int nim_baru;
            cout << "Masukkan nama baru: ";
            cin >> nama_baru;
            cout << "Masukkan NIM baru: ";
            cin >> nim_baru;
            string nama_lama = head->nama;
            ubahTengah(nama_baru, nim_baru, head->nama);
            cout << "Data (" << nama_lama << ") telah terganti
dengan data '" << nama_baru << "' " << endl;
            break;
        }
        case 5:
        {
            cout << "-Ubah Data Belakang-\n"
                 << endl;
            string nama_baru;
            int nim_baru;
            cout << "Masukkan nama baru: ";
            cin >> nama_baru;
            cout << "Masukkan NIM baru: ";
            cin >> nim_baru;
            string nama_lama = tail->nama;
            ubahTengah(nama_baru, nim_baru, tail->nama);
```

```
            cout << "Data (" << nama_lama << ") telah terganti
dengan data '" << nama_baru << "' " << endl;
            break;
        }
        case 6:
        {
            cout << "-Ubah Data Tengah-\n"
                 << endl;
            if (!isEmpty())
            {
                int posisi;
                cout << "Masukkan posisi yang ingin diubah: ";
                cin >> posisi;
                string nama_baru;
                int nim_baru;
                cout << "Masukkan nama baru: ";
                cin >> nama_baru;
                cout << "Masukkan NIM baru: ";
                cin >> nim_baru;
                Node *bantu = head;
                int nomor = 1;
                while (nomor < posisi)
                {
                    bantu = bantu->next;
                    nomor++;
                }
                string nama_lama = bantu->nama;
                ubahTengah(nama_baru, nim_baru, bantu->nama);
                cout << "Data pada posisi " << posisi <<
"  berhasil diubah." << endl;
                cout << "Data (" << nama_lama << ") telah terganti
dengan data(" << nama_baru << ") " << endl;
            }
            else
            {
                cout << "List masih kosong" << endl;
            }
            break;
        }
        case 7:
        {
            cout << "-Hapus Data Depan-\n"
                 << endl;
            if (!isEmpty())
            {
                string nama_hapus = head->nama;
                hapusDepan();
```

```
                cout << "Data (" << nama_hapus << ") berhasil
dihapus." << endl;
            }
            else
            {
                cout << "List masih kosong" << endl;
            }
            break;
        }
        case 8:
        {
            cout << "-Hapus Data Belakang-\n"
                 << endl;
            if (!isEmpty())
            {
                string nama_hapus = tail->nama;
                hapusBelakang();
                cout << "Data (" << nama_hapus << ") berhasil
dihapus." << endl;
            }
            else
            {
                cout << "List masih kosong" << endl;
            }
            break;
        }
        case 9:
        {
            cout << "-Hapus Data Tengah-\n"
                 << endl;
            if (!isEmpty())
            {
                int posisi;
                cout << "Masukkan posisi: ";
                cin >> posisi;
                hapusTengah(posisi);
            }
            else
            {
                cout << "List masih kosong" << endl;
            }
            break;
        }
        case 10:
        {
            cout << "-Hapus List Data-\n"
                 << endl;
```

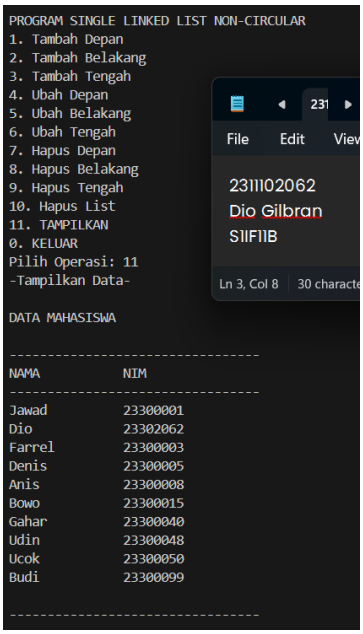
```
            hapusList();
            break;
        }
        case 11:
        {
            cout << "-Tampilkan Data-\n"
                 << endl;
            tampil();
            break;
        }
        case 0:
        {
            return 0;
        }
        default:
        {
            cout << "Pilihan tidak valid" << endl;
            break;
        }
        }
    }
    return 0;
}
```

SCREENSHOT OUTPUT

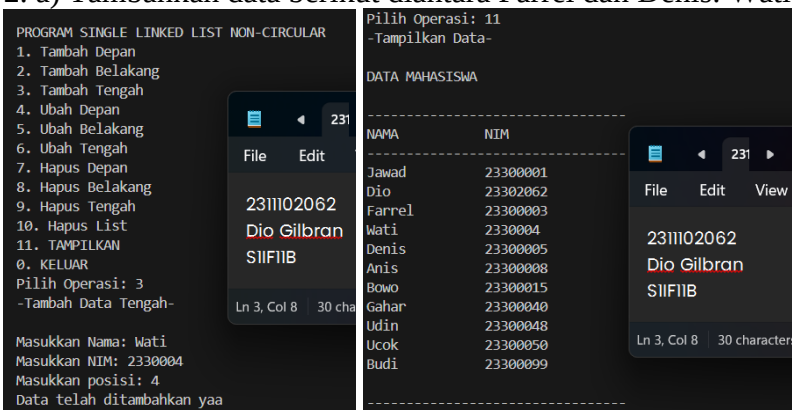
1.

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

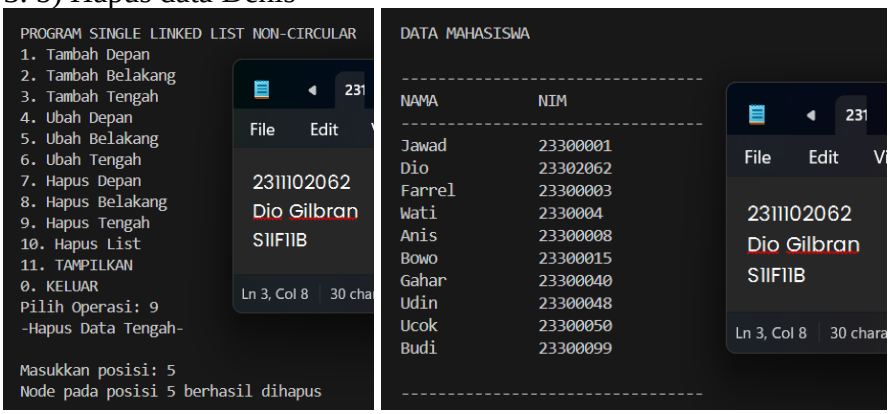
Nama	NIM
Jawad	23300001
[Nama Anda]	[NIM Anda]
Farrel	23300003
Denis	23300005
Anis	23300008
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099



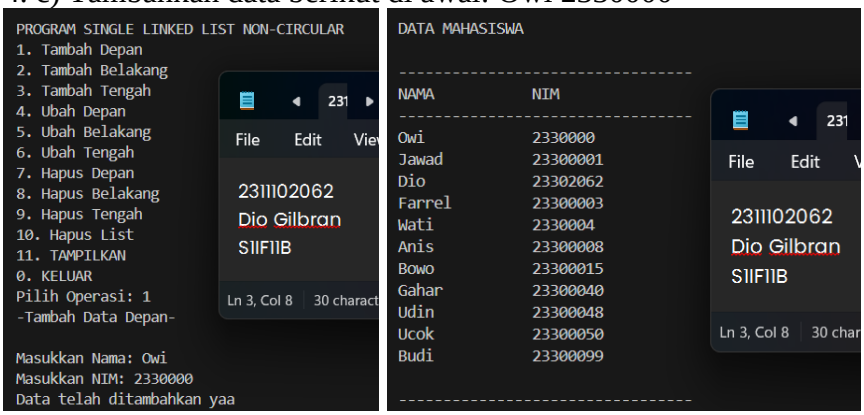
2. a) Tambahkan data berikut diantara Farrel dan Denis: Wati 2330004



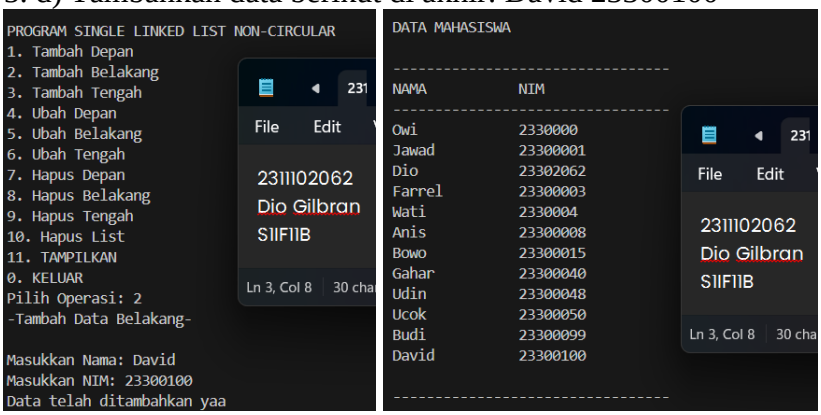
3. b) Hapus data Denis



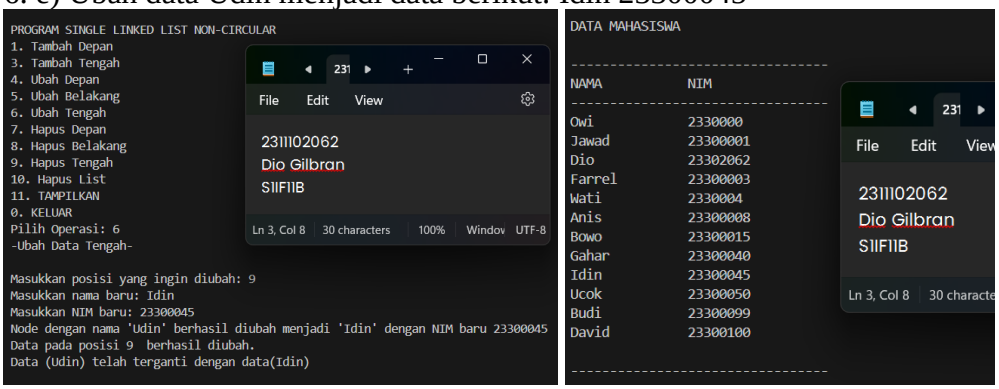
4. c) Tambahkan data berikut di awal: Owi 2330000



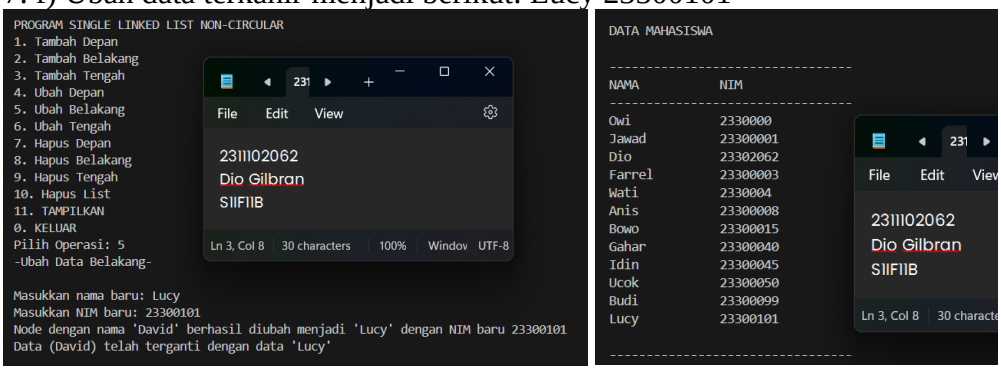
5. d) Tambahkan data berikut di akhir: David 23300100



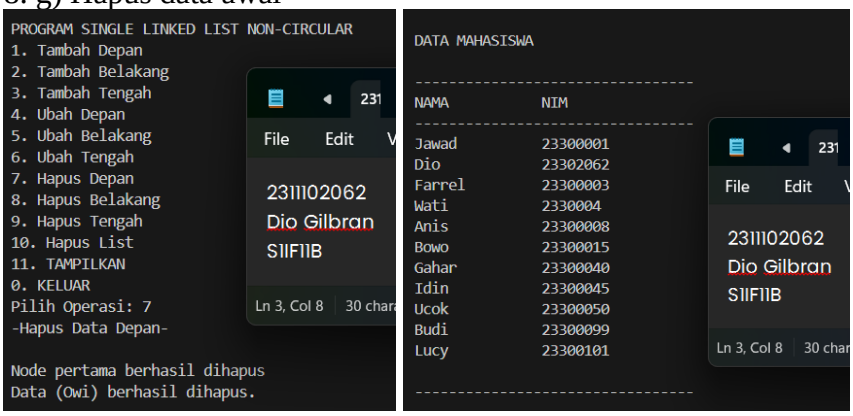
6. e) Ubah data Udin menjadi data berikut: Idin 23300045



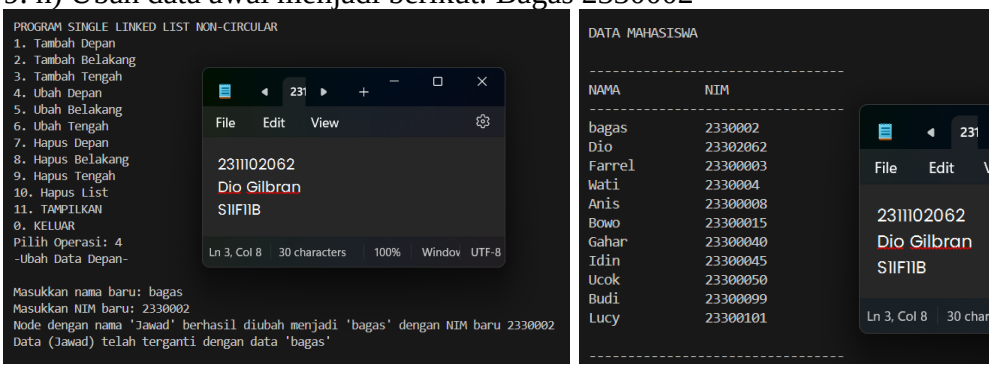
7. f) Ubah data terkahir menjadi berikut: Lucy 23300101



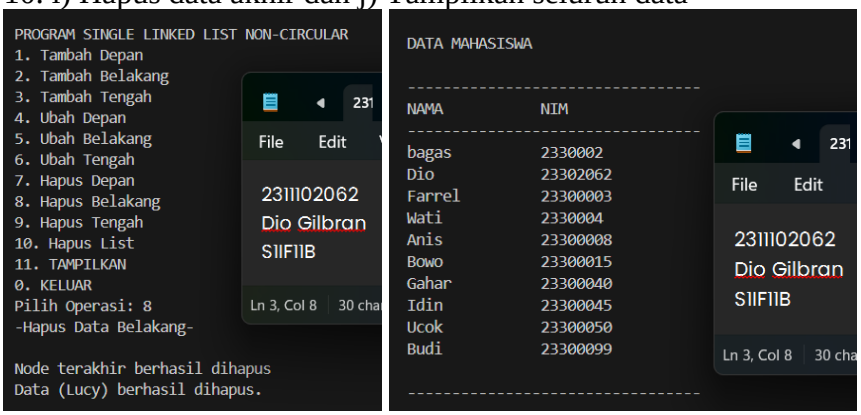
8. g) Hapus data awal



9. h) Ubah data awal menjadi berikut: Bagas 2330002



10. i) Hapus data akhir dan j) Tampilkan seluruh data



DESKRIPSI

Program ini merupakan implementasi dari Single Linked List Non-Circular dalam bahasa C++. Daftar tertaut ini terdiri dari beberapa fungsi untuk menginisialisasi, menambah, mengubah, menghapus, dan menampilkan data. Fitur-fitur ini memungkinkan pengguna untuk menambahkan data ke depan, belakang, atau tengah daftar, serta mengubah dan menghapus data di posisi tersebut. Program ini juga mencakup fungsi untuk menghitung item dalam daftar dan menghapus seluruh daftar. Antarmuka pengguna merupakan menu interaktif yang memungkinkan pengguna memilih tindakan yang diinginkan melalui input keyboard, sehingga memudahkan untuk mengedit data siswa (termasuk nama dan atribut NIM) dalam daftar tertaut.

D. KESIMPULAN

Modul kali ini membahas implementasi Linked List, baik yang berbentuk Circular maupun Non-Circular, yang tentunya menggunakan bahasa pemrograman C++. Pada praktikum ini, saya mempelajari cara untuk mendeklarasikan dan menginisialisasi simpul (node), serta melakukan operasi dasar seperti penambahan, penghapusan, dan penampilan data pada Linked List. Saya diajarkan untuk menghitung jumlah simpul dalam list dan mengubah nilai data pada simpul tertentu. Implementasi ini bertujuan untuk memberikan pemahaman yang mendalam tentang bagaimana Linked List bekerja dan bagaimana operasi-operasi tersebut dapat diimplementasikan dalam program-program komputer.

E. REFERENSI

AsistenPraktikum,“Modul Linked List Circular dan Non Circular", LearningManagementSystem,2024.

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

Yesi Gusla, dkk. 2022. Konsep Algoritma dan Pemrograman. Bandung : Indie Press.