

**LAPORAN PRAKTIKUM STRUKTUR DATA DAN
PEMROGRAMAN**

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



Disusun oleh :

ZEFANYA.B.T.TARIGAN

2311102028

IF-11-A

Dosen Pengampu :

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

**PROGRAM STUDI TEKNIK INFORMATIKA FAKULTAS
INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO PURWOKERTO
2023**

BAB I

TUJUAN PRAKTIKUM

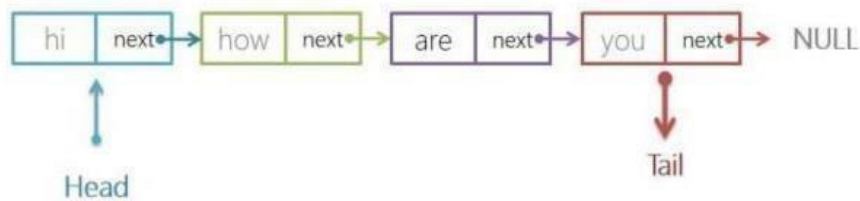
- Praktikan dapat mengetahui dan memahami linked list circular dan non circular.
- Praktikan dapat membuat linked list circular dan non circular.
- Praktikan dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat.

BAB II

DASAR TEORI

1. Linked List Non Circular

Linked list non circular adalah jenis linked list yang memiliki node pertama (head) dan node terakhir (tail) yang tidak saling terhubung. Pointer terakhir (tail) pada Linked List ini selalu memiliki nilai 'NULL' untuk menandakan bahwa data terakhir dalam list tersebut.



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;
        }
    }
}

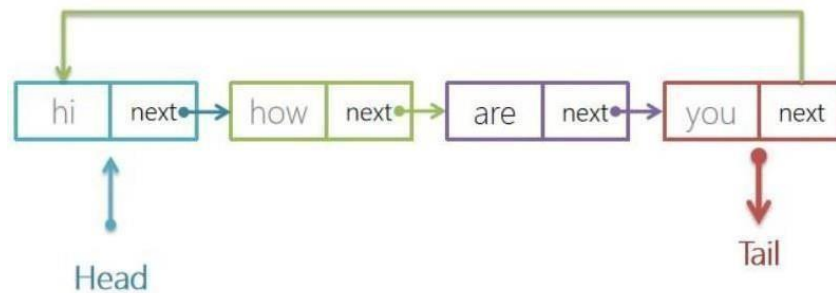
```

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;

    }

}z
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;
    }

}
```

BAB III

GUIDED

**GUIDED 1 SOURCE
CODE**

```

#include <iostream>

using namespace std;
// PROGRAM SINGLE LINKED LIST NON-CIRCULAR
// Deklarasi struct node struct
Node
{
    int data;
    Node *next;
};

Node *head; // Deklarasi head
Node *tail; // Deklarasi tail

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

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

// Tambah depan
void insertDepan(int nilai)
{
    // buat node baru
    Node *baru = new Node();
    baru->data = nilai;
    baru->next = NULL;    if
    (isEmpty() == true)
    {
        head = tail = baru;
    head->next = NULL;
    }
}

```

```

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

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

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

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

```

```

    {
        Node *baru,
        *bantu;
        baru = new Node();
        baru->data = data;

        //
        tranversing
        bantu = head;
        int nomor = 1;
        while (nomor < posisi
        - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next
        = bantu->next;
        bantu->next = baru;
    }
}

// Hapus depan
void hapusDepan()
{
    Node *hapus;
    if (isEmpty() ==
    false)
    {
        if (head->next
        != NULL)
        {
            hapus = head;
            head = head->next;
            delete hapus;
        }
        else
        {
            head =
            tail
            = NULL;
        }
    }
    else
    {

```

```

        cout <<
"Linked list masih
kosong" << endl;    }
}

// Hapus belakang void
hapusBelakang()
{
    Node *hapus;
Node *bantu;    if
(isEmpty() ==
false)    {
if (head != tail)
{ hapus = tail;
bantu = head;
while
(bantu->next != tail)
{
bantu = bantu->next;
}    tail =
bantu;
tail->next
= NULL;
delete hapus;
}    else
{    head =
tail
= NULL;
}    }
else    {
cout << "Linked list
masih kosong" << endl;
}
}

// Hapus tengah void
hapusTengah(int
posisi)
{
    Node *hapus,
*bantu, *sebelum;

```

```

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

```



```

sebelum->next = bantu;
{
    delete hapus;
}

// ubah depan void
ubahDepan(int data)
{
    if (isEmpty() == 0)
    {
        head->data = data;
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

// ubah tengah
void ubahTengah(int data, int posisi)
{
    Node *bantu;    if
(isEmpty() == 0)
    {
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "Posisi di luar jangkauan" << endl;
        }
        else if (posisi == 1)
        {
            cout << "Posisi bukan posisi tengah" << endl;
        }
    }
    else
    {
        int nomor = 1;
        bantu = head;
        while (nomor < posisi)
        {
            bantu = bantu->next;
            nomor++;
        }
        bantu->data = data;
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

// ubah belakang
void ubahBelakang(int data)
{
    if (isEmpty() == 0)

```



```

tail->data = data;
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

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

// Tampilkan list
void tampilList()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << " ";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

int main()
{
    init();
    insertDepan(3);
    tampilList();
    insertBelakang(5);
    tampilList();
    insertDepan(2);
    tampilList();
    insertDepan(1);
    tampilList();
    hapusDepan();
    tampilList();
    hapusBelakang();
}

```

```

tampilList();
insertTengah(7, 2);

    tampilList();
hapusTengah(2);

    tampilList();
ubahDepan(1);

    tampilList();
ubahBelakang(8);
tampilList();      ubahTengah(11,
2);      tampilList();

    return 0;
}

```

SCREENSHOOT PROGRAM

```

PS D:\KULIAH\SEMESTER2\struktur data praktikum> ./Guided1modul4
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:\KULIAH\SEMESTER2\struktur data praktikum>

```

DESKRIPSI PROGRAM

Program tersebut sebagai mendemonstrasikan operasi dasar pada single linked list non circular.

struct Node berfungsi sebagai mendeklarasikan struktur data Node yang memiliki dua anggota: data (bertipe integer) dan next (bertipe pointer ke Node).

Node *head berfungsi sebagai deklarasi pointer head yang menunjuk ke node pertama dalam linked list.

Node *tail berfungsi sebagai deklarasi pointer tail yang menunjuk ke node terakhir dalam linked list.

GUIDED 2

SOURCE CODE

```
#include <iostream> using
namespace std;
/// PROGRAM SINGLE LINKED LIST CIRCULAR
// Deklarasi Struct Node
struct Node
{
    string
data;      Node
*next;
};
Node *head, *tail, *baru, *bantu, *hapus;
void init()
{
    head =
NULL;      tail =
head;
}
// Pengecekan
int isEmpty()
{
    if (head ==
NULL)

        return 1; // true
    else

        return 0; // false
}

// Buat Node Baru void
buatNode(string data){

    baru = new Node;
    baru->data = data;      baru->
next = NULL;
}
// Hitung List int
hitungList()
{
    bantu = head;      int
jumlah = 0;      while
(bantu != NULL)
    {
        jumlah++;
        bantu = bantu->next;
    }
    return jumlah;
}
// Tambah Depan
void insertDepan(string data)
{
    // Buat Node baru
    buatNode(data);      if
(isEmpty() == 1)
    {
```

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

```
    }    else    {    while

(tail->next != head)    {

tail = tail->next;

        }        baru-

>next = head;        head

= baru;        tail->next

= head;

        }

    }

// Tambah Belakang void

insertBelakang(string data)

{

    // Buat Node baru

    buatNode(data);    if

(isEmpty() == 1)

    {        head = baru;

tail = head;        baru-

>next = head;

        }

    else    {
```

```

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

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

    }

} // Tambah Tengah void
insertTengah(string data, int posisi)
{
    if (isEmpty() ==
1)

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

    }    else    {

baru->data = data;          //

transversing          int nomor =
1;          bantu = head;
while (nomor < posisi - 1)

    {

```

```

        bantu = bantu->next;

    nomor++;

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

    }

}

// Hapus Depan

void hapusDepan()

{    if (isEmpty() ==
0)

    {        hapus = head;

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

        {

    head = NULL;

    tail = NULL;

        }        else        {

    delete hapus;        while (tail-
>next != hapus)

        {

    tail = tail->next;

        }

```

```

        head = head->next;

tail->next = head;

hapus->next = NULL;

delete hapus;

        }    }    else    {    cout

<< "List masih kosong!" << endl;

    }

}

// Hapus Belakang

void hapusBelakang()

{    if (isEmpty()

== 0)

    {    hapus = head;

tail = head;    if

(hapus->next == head)

        {

head = NULL;

tail = NULL;    }

else    {

delete hapus;

```

```

        while (hapus->next != head)

        {

hapus = hapus->next;

        }           while

(tail->next != hapus)

        {

tail = tail->next;

        }

tail->next = head;

hapus->next = NULL;

delete hapus;

        }     } else     {           cout <<

"List masih kosong!" << endl;

        }

} // Hapus Tengah void

hapusTengah(int posisi)

{     if (isEmpty()

== 0)     {

        // transversing

int nomor = 1;

bantu = head;

while (nomor < posisi -

1)

        {           bantu = bantu->next;

nomor++;           }           hapus = bantu-

```



```

>next;          bantu->next = hapus->next;

delete hapus;    }      else      {

cout << "List masih kosong!" << endl;

    }

} // Hapus List void

clearList() {      if (head

!= NULL)        {

hapus = head->next;

while (hapus != head)

    {              bantu =

hapus->next;

delete hapus;

hapus = bantu;

    }

    delete head;          head = NULL;

}      cout << "List berhasil terhapus!" <<

endl;

}

// Tampilkan List

void tampil() {

if (isEmpty() == 0)

    {          tail = head;          do

    {          cout << tail->data << ends;

tail = tail->next;          } while (tail !=

head);          cout << endl;          }      else

```

```
{          cout << "List masih kosong!" <<

endl;

    } } int

main() {

init();
```

```

        insertDepan("Ayam");

tampil();

insertDepan("Bebek");

tampil();

insertBelakang("Cicak");

tampil();

insertBelakang("Domba");

tampil();

hapusBelakang();

tampil();      hapusDepan();

tampil();

insertTengah("Sapi", 2);

tampil();

hapusTengah(2);

tampil();      return 0;

}

```

SCREENSHOOT PROGRAM

```

glauncher.exe' '--stdin=Microsoft-MIEngine-In-tar51ixo.vxl' '--stdout=Micro
0b.jdl' '--pid=Microsoft-MIEngine-Pid-of1bu2l1.uig' '--dbgExe=D:\MinGW\bin\
PS D:\KULIAH\SEMESTER2\struktur data praktikum> ./Guided2modul4
Ayam
BebekAyam
BebekAyamCicak
BebekAyamCicakDomba
PS D:\KULIAH\SEMESTER2\struktur data praktikum>

```

DESKRIPSI PROGRAM

insertDepan(string data): fungsi untuk menambahkan node baru di depan list.

insertBelakang(string data): fungsi untuk menambahkan node baru di belakang list.

insertTengah(string data, int posisi): fungsi untuk menambahkan node baru di posisi tertentu dalam list.

isEmpty(): fungsi untuk mengecek apakah list kosong.

buatNode(string data): fungsi untuk membuat node baru dengan data yang diberikan.

hitungList(): fungsi untuk menghitung jumlah node dalam list.

BAB IV UNGUIDED

UNGUIDED 1

Source code

```
#include <iostream>
#include <iomanip> using
namespace std; //
Deklarasi Struct Node
struct Node
{
    string nama;
    string nim;      Node
    *next;
};
Node *head = NULL;
// Buat Node Baru
Node *buatNode(string nama, string nim)
{
    Node *baru = new Node;
    baru->nama = nama;      baru->
nim = nim;      baru->next =
NULL;      return baru;
}
// Pengecekan List Kosong bool
isEmpty()
{
    return head == NULL;
}
// Tambah Depan
void tambahDepan(string nama, string nim)
{
    Node *baru = buatNode(nama, nim);
    baru->next = head;      head = baru;
}
// Tambah Belakang
```

```

void tambahBelakang(string nama, string nim)
{
    Node *baru = buatNode(nama, nim);
    if (isEmpty())
    {
        head
= baru;
    }
    else
    {
        Node *bantu = head;
        while (bantu->next != NULL)
        {
            bantu =
bantu->next;
        }
        bantu->next = baru;
    }
}

// Tambah Tengah void tambahTengah(string nama,
string nim, int posisi)
{
    if (posisi <= 1 ||
isEmpty())
    {
        tambahDepan(nama, nim);
    }
    else
    {
        Node *baru = buatNode(nama, nim);
        Node *bantu =
head;
        for (int i = 1; i < posisi - 1 && bantu->next !=
NULL; i++)
        {
            bantu =
bantu->next;

```

```

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

// Ubah Depan void ubahDepan(string
nama, string nim)
{    if
(!isEmpty())
    {        head->nama
= nama;        head->nim
= nim;
    }
}

// Ubah Belakang void ubahBelakang(string
nama, string nim)
{    if
(!isEmpty())
    {
        Node *bantu = head;
while (bantu->next != NULL)
    {
        bantu = bantu->next;
    }
        bantu->nama = nama;
bantu->nim = nim;
    }
}

// Ubah Tengah void ubahTengah(string nama, string
nim, int posisi)
{    if
(!isEmpty())
    {
        Node *bantu = head;
        for (int i = 1; i < posisi && bantu != NULL; i++)
        {
            bantu =
bantu->next;
        }
        if (bantu != NULL)

```

```

        {
            bantu-
>nama = nama;
bantu->nim = nim;
        }
    }
} // Hapus Depan
void
hapusDepan()
{
    if
(!isEmpty())
    {
        Node *hapus = head;
head = head->next;
delete hapus;
    }
}
// Hapus Belakang
void hapusBelakang()
{
    if (!isEmpty()) {
if (head->next == NULL)
    {
delete head;
head = NULL;
    }
else
    {
        Node *bantu = head;
while (bantu->next->next != NULL)
    {
        bantu
= bantu->next;
    }
delete bantu->next;
bantu->next = NULL;
    }
}
} // Hapus Tengah void
hapusTengah(int posisi)
{
    if (!isEmpty())
    {
        if (posisi <= 1)
        {
            hapusDepan();
        }
    }
else
    {
        Node *bantu =
head;
for (int i =
1; i < posisi
- 1 && bantu-

```

```

>next !=
NULL; i++)
    {
        bantu
    = bantu->next;
    }
    if
    (bantu->next != NULL)
    {
        Node *hapus = bantu->next;
        bantu->next = bantu->next->next;
        delete hapus;
    }
}

// Hapus List
void hapusList()
{
    while
    (!isEmpty())
    {
        hapusDepan();
    }
}

// Tampilkan Data
void tampilkanData()
{
    if
    (isEmpty())
    {
        cout << "List masih kosong!"
    << endl;
    }
    else
    {
        Node *bantu = head;
        cout <<
        "===== " << endl;
        cout << "| DATA MAHASISWA |" << endl;
        cout << "===== " << endl;
        cout << "| " << setw(25) << left << "NAMA"
        << " | " << setw(20) << left
        << "NIM" << " |" << endl;
        cout <<
        "===== " << endl;
        while (bantu != NULL)
        {
            cout << "| " << setw(25) << left << bantu->nama
            << " | " << setw(20)
            << left << bantu->nim << " |" << endl;
            bantu = bantu->next;
        }
        cout <<
        "===== " << endl;
    }
}

// Menu Utama
void menu()
{
    system("cls");
    cout << "\033[1mMade by : Zefanya Brana Tertius
    Tarigan\033[0m" << endl;
    cout << "\033[1mNim : 2311102028\033[0m" <<

```



```

endl;      cout << endl;      cout <<
"=====
      << endl;      cout << "| PROGRAM SINGLE LINKED LIST NON-
CIRCULAR |" << endl;      cout <<
"=====
      << endl;      cout << "01. Tambah Depan" << endl;      cout <<
"02. Tambah Belakang" << endl;      cout << "03. Tambah Tengah" <<
endl;      cout << "04. Ubah Depan" << endl;      cout << "05. Ubah
Belakang" << endl;      cout << "06. Ubah Tengah" << endl;      cout <<
"07. Hapus Depan" << endl;      cout << "08. Hapus Belakang" << endl;
cout << "09. Hapus Tengah" << endl;      cout << "10. Hapus List" <<
endl;      cout << "11. TAMPILKAN" << endl;      cout << "-----
-----" << endl;      cout << "0. KELUAR"
<< endl;      cout <<
"=====
      << endl;
} int
main()
{      system("cls");      int
pilihan, posisi;      string nama,
nim;      do      {      menu();
cout << "Pilih Operasi : ";
cin >> pilihan;      switch
(pilihan)
{
case 1:
      system("cls");
cout << "Masukkan Nama : ";
cin >> nama;      cout <<
"Masukkan NIM : ";      cin >>
nim;      tambahDepan(nama,
nim);      system("pause");
break;      case 2:
      system("cls");
cout << "Masukkan Nama : ";
cin >> nama;      cout <<
"Masukkan NIM : ";      cin >>
nim;      tambahBelakang(nama,
nim);      system("pause");
break;      case 3:
      system("cls");      cout
<< "Masukkan Nama : ";      cin >>
nama;      cout << "Masukkan NIM :
";      cin >> nim;      cout
<< "Masukkan Posisi : ";      cin >>
posisi;      tambahTengah(nama, nim,
posisi);      system("pause");
break;      case 4:
      system("cls");
cout << "Masukkan Nama : ";
cin >> nama;      cout <<
"Masukkan NIM : ";      cin >>

```

```

nim;                ubahDepan(nama, nim);
system("pause");    break;
case 5:
    system("cls");
cout << "Masukkan Nama : ";
cin >> nama;        cout <<
"Masukkan NIM : ";  cin >>
nim;                ubahBelakang(nama,
nim);                system("pause");
break;              case 6:
    system("cls");
cout << "Masukkan Nama : ";
cin >> nama;        cout <<
"Masukkan NIM : ";  cin >>
nim;                cout << "Masukkan Posisi
: ";                cin >> posisi;
ubahTengah(nama, nim, posisi);
system("pause");    break;
case 7:
    system("cls");
hapusDepan();
system("pause");
break;              case 8:
    system("cls");

hapusBelakang();
system("pause");
break;              case 9:
    system("cls");
cout << "Masukkan Posisi : ";
cin >> posisi;
hapusTengah(posisi);
system("pause");    break;
case 10:             hapusList();
break;              case 11:
    system("cls");
tampilkanData();
system("pause");    break;
case 0:              cout << "Terima kasih!"
<< endl;            break;
system("pause");    default:
    cout << "Pilihan tidak valid!" << endl;
system("pause");
}
} while (pilihan != 0);
return 0;
}

```

Screenshoot Program

```
Made by : Zefanya Brana Tertius Tarigan
Nim : 2311102028

=====
| PROGRAM SINGLE LINKED LIST NON-CIRCULAR |
=====
01. Tambah Depan
02. Tambah Belakang
03. Tambah Tengah
04. Ubah Depan
05. Ubah Belakang
06. Ubah Tengah
07. Hapus Depan
08. Hapus Belakang
09. Hapus Tengah
10. Hapus List
11. TAMPILKAN
-----
0. KELUAR
=====
Pilih Operasi : █
```

Deskripsi Program:

Program ini merupakan implementasi dari sebuah fungsi yang memiliki nama 'menu'. Fungsi tersebut akan menampilkan daftar menu yang terstruktur secara berurutan, meliputi:

- ☐ "Tambah Depan"
- ☐ "Tambah Belakang"
- ☐ "Tambah Tengah"
- ☐ "Ubah Depan"
- ☐ "Ubah Belakang"
- ☐ "Ubah Tengah"
- ☐ "Hapus Depan"
- ☐ "Hapus Belakang"
- ☐ "Hapus Tengah"
- ☐ "Hapus List"
- ☐ "TAMPILKAN"

Selain itu, fungsi menu juga akan menampilkan informasi tentang pembuat program, yaitu "Made by:Zefanya Brana Tertius Tarigan" dan "Nim: 2311102028". Untuk melihat konten dari setiap pilihan menu, program akan menjalankan fungsi yang sesuai dengan pilihan tersebut.

- ☐ Pengguna dapat keluar dari program dengan memilih opsi "0".

KELUAR".

B. Inputan Tambah Depan, Tengah dan Belakang

B.1 Input Tambah Depan

```
Masukkan Nama : Egi  
Masukkan NIM : 2344501928  
Press any key to continue . . . █
```

B.2 Input Tambah Belakang

```
Masukkan Nama : Egi  
Masukkan NIM : 2344501928  
Press any key to continue . . . █
```

B.3 Input Tambah Tengah

```
Masukkan Nama : raja  
Masukkan NIM : 23020208  
Masukkan Posisi : 5  
Press any key to continue . . . █
```

DESKRIPSI PROGRAM:

Ada tiga fungsi dalam program ini untuk menambahkan node baru ke linked list.

- Fungsi tambahDepan(string nama, string nim) digunakan untuk menambahkan node baru di bagian depan linked list.
- Fungsi tambahBelakang(string nama, string nim) digunakan untuk menambahkan node baru di bagian belakang linked list.
- Fungsi tambahTengah(string nama, string nim, int posisi) digunakan untuk menambahkan node baru pada posisi tertentu dalam linked list.

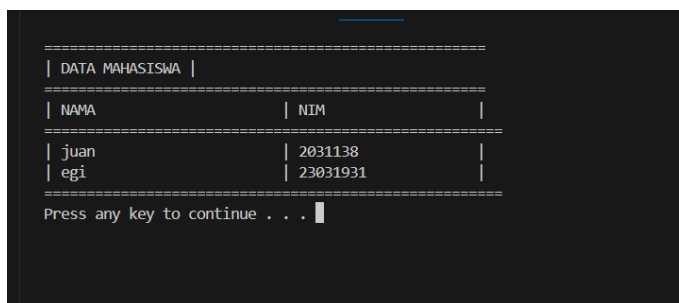
Sebelum menambahkan node baru, program akan membuat node baru terlebih dahulu dengan menggunakan fungsi buatNode(nama, nim). Fungsi ini akan membuat node baru dengan data berupa nama dan nim.

- Dalam fungsi tambahDepan(string nama, string nim), node baru akan ditambahkan di bagian depan linked list dengan cara membuat pointer baru yang menunjuk ke node baru, dan pointer baru->next akan menunjuk ke head yang sebelumnya menunjuk ke node pertama dalam linked list. Setelah itu, head akan diubah menjadi pointer baru.

□ Dalam fungsi tambahBelakang(string nama, string nim), node baru akan ditambahkan di bagian belakang linked list dengan cara mencari node terakhir dalam linked list menggunakan pointer bantu. Setelah menemukan node terakhir, pointer bantu->next akan diubah menjadi pointer baru.

□ Dalam fungsi tambahTengah(string nama, string nim, int posisi), node baru akan ditambahkan pada posisi tertentu dalam linked list dengan cara mencari node yang akan menjadi node sebelum node baru menggunakan pointer bantu. Setelah menemukan node tersebut, pointer baru->next akan diubah menjadi pointer bantu->next, dan pointer bantu->next akan diubah menjadi pointer baru. Posisi node baru ditentukan oleh parameter posisi. Jika posisi kurang dari atau sama dengan 1, maka node baru akan ditambahkan di bagian depan linked list. Jika posisi lebih dari jumlah node dalam linked list, maka node baru akan ditambahkan di bagian belakang linked list. Jika posisi berada antara 1 dan jumlah node, maka node baru akan ditambahkan di posisi yang sesuai.

B. 4 Output Tampilan Data Sementara



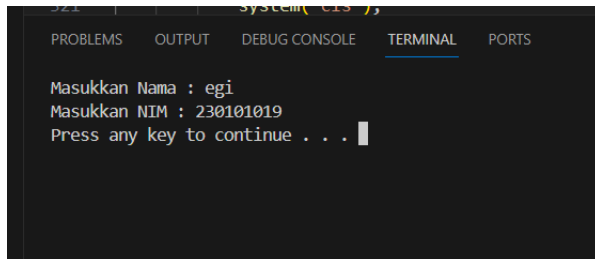
```
=====
| DATA MAHASISWA |
=====
| NAMA                | NIM                |
=====
| juan                | 2031138            |
| egi                 | 23031931           |
=====
Press any key to continue . . .
```

Deskripsi

Tampilan ini merupakan output akhir setelah user menginputkan fungsi tambahDepan(), tambahTengah, dan tambahBelakang() dan sekaligus output sementara, sebelum user menjalankan program lainnya.

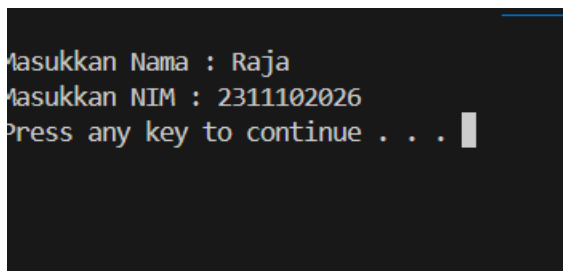
C. Ubah data

C.1 Ubah data depan



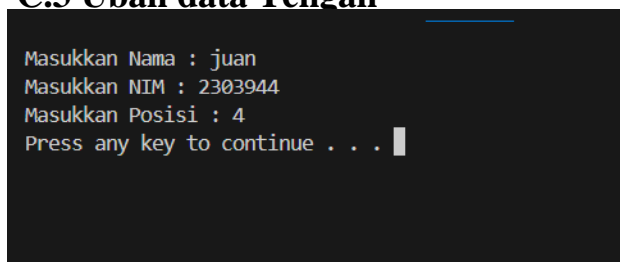
```
324 |         system("cls");  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
Masukkan Nama : egi  
Masukkan NIM : 230101019  
Press any key to continue . . .
```

C.2 Ubah data Belakang



```
Masukkan Nama : Raja  
Masukkan NIM : 2311102026  
Press any key to continue . . .
```

C.3 Ubah data Tengah



```
Masukkan Nama : juan  
Masukkan NIM : 2303944  
Masukkan Posisi : 4  
Press any key to continue . . .
```

Deskripsi

Ubah Data a) Fungsi `ubahDepan(string nama, string nim)` digunakan untuk mengubah data nama dan nim pada node pertama atau depan linked list. Fungsi ini akan memeriksa terlebih dahulu apakah linked list kosong atau tidak sebelum melakukan perubahan. b) Fungsi `ubahBelakang(string nama, string nim)` berperan dalam mengubah data nama dan nim pada node terakhir atau belakang linked list. Fungsi ini juga melakukan pengecekan terhadap kondisi linked list sebelum melakukan perubahan. c) Fungsi `ubahTengah(string nama, string nim, int posisi)` berfungsi untuk mengubah data nama dan nim pada node yang berada di posisi tertentu dalam linked list. Fungsi ini juga melakukan pengecekan terhadap keadaan linked list sebelum melakukan perubahan. Dalam semua fungsi di atas, apabila linked list kosong, maka tidak akan ada perubahan yang dilakukan pada node dan fungsi akan berhenti. Setiap fungsi di atas memiliki langkah-langkah yang berbeda dalam mengubah data mahasiswa pada linked list, tergantung dari posisi node yang ingin diubah.

C.4 Tampilan Data Sementara

```
=====
| DATA MAHASISWA |
=====
| NAMA                | NIM                |
=====
| Juan                | 2031138            |
| egi                 | 23031931           |
=====
Press any key to continue . . .
```

Deskripsi

Tampilan ini merupakan output akhir setelah user menginputkan fungsi tambahDepan(), tambahTengah(), dan tambahBelakang() kemudian dilanjutkan dengan user menjalankan program dengan fungsi ubahDepan(), ubahTengah(), dan ubahBelakang() dan sekaligus output sementara, sebelum user menjalankan program lainnya.

D. Hapus Depan

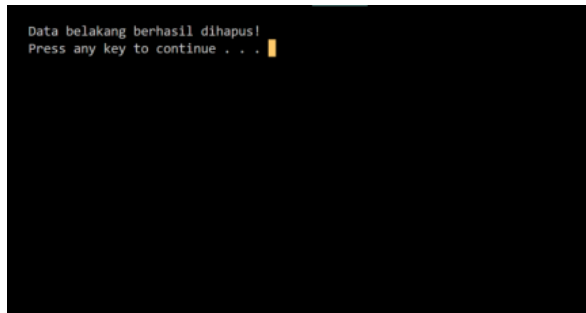
```
Data depan berhasil dihapus!
Press any key to continue . . .
```

Deskripsi

Kode di atas merupakan definisi fungsi hapusDepan yang digunakan untuk menghapus node pertama dari Single Linked List. Berikut penjelasannya:

1. Fungsi hapusDepan dideklarasikan sebagai void, yang artinya fungsi ini tidak mengembalikan nilai apapun.
2. Terdapat pengecekan apakah list kosong menggunakan fungsi isEmpty(). Fungsi ini digunakan untuk mengecek apakah list kosong atau tidak.
3. Jika list tidak kosong (isEmpty() bernilai false), maka dilakukan penghapusan node pertama.
4. Alamat node pertama disimpan dalam variabel hapus.
5. Pointer head diubah untuk menunjuk ke node kedua (setelah node pertama dihapus).
6. Node pertama dihapus dari memori menggunakan operator delete.

E. Hapus Belakang



Deskripsi

Kode di atas merupakan definisi fungsi `hapusBelakang` yang digunakan untuk menghapus node terakhir dari Single Linked List. Berikut penjelasannya:

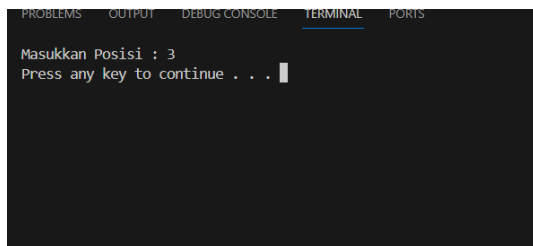
1. Deklarasi Fungsi `void hapusBelakang()`: Mendefinisikan fungsi `hapusBelakang` yang tidak mengembalikan nilai (`void`).
2. Pengecekan List Kosong `if (!isEmpty())`: Memeriksa apakah list kosong. `isEmpty()`: Fungsi untuk mengecek apakah list kosong. Jika list tidak kosong (`isEmpty()` bernilai `false`), maka lanjutkan ke langkah berikutnya.

3. Penghapusan Node Belakang

a) Kasus 1: List Hanya Memiliki Satu Node `if (head->next == NULL)`: Memeriksa apakah list hanya memiliki satu node. `delete head`: Menghapus node tersebut dari memori. `head = NULL`: Mengatur `head` menjadi `NULL` karena list menjadi kosong.

b) Kasus 2: List Memiliki Lebih Dari Satu Node `Node* bantu = head`: Menyimpan alamat node pertama ke dalam variabel `bantu`. `while (bantu->next->next != NULL)`: Melintasi list sampai menemukan node sebelum node terakhir. `bantu = bantu->next`: Memindahkan `bantu` ke node berikutnya. `delete bantu->next`: Menghapus node terakhir dari memori. `bantu->next = NULL`: Mengatur pointer `next` dari node sebelum node terakhir menjadi `NULL`.

F. Hapus Tengah



Deskripsi

Kode di atas merupakan definisi fungsi `hapusTengah` yang digunakan untuk menghapus node pada posisi tertentu dari Single Linked List. Berikut penjelasannya:

1. Fungsi `hapusTengah` Fungsi ini dideklarasikan dengan tipe data `void`, yang berarti tidak mengembalikan nilai. Fungsi ini menerima satu parameter yaitu `posisi`, yang menunjukkan

posisi node yang ingin dihapus. Pengecekan List Kosong Dilakukan pengecekan apakah list kosong atau tidak.

2. Pengecekan ini menggunakan fungsi isEmpty() yang mengembalikan nilai true jika list kosong, dan false jika tidak kosong. Jika list tidak kosong (isEmpty() bernilai false), maka langkah berikutnya akan dilanjutkan.

3. Penghapusan Node Tengah Terdapat dua kasus yang mungkin terjadi saat menghapus node tengah.

a) Kasus pertama adalah jika posisi node yang ingin dihapus kurang dari atau sama dengan 1. Jika kondisi ini terpenuhi, maka akan dipanggil fungsi hapusDepan() untuk menghapus node pertama.

b) Kasus kedua adalah jika posisi node yang ingin dihapus lebih dari 1.

c) Pada kasus ini, dilakukan iterasi melalui list sampai menemukan node sebelum node yang ingin dihapus.

4. Variabel bantu digunakan untuk menyimpan alamat node pertama.

5. Variabel i digunakan sebagai penghitung iterasi.

6. Variabel bantu akan dipindahkan ke node berikutnya menggunakan bantu->next.

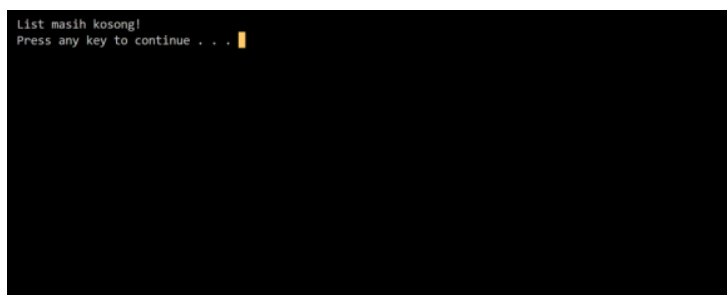
7. Jika bantu->next tidak NULL, artinya node yang ingin dihapus bukan node terakhir.

8. Alamat node yang ingin dihapus disimpan dalam variabel hapus.

9. Node sebelum node yang ingin dihapus dihubungkan dengan node setelahnya menggunakan bantu->next = bantu->next->next.

10. Node yang ingin dihapus dihapus dari memori menggunakan delete hapus.

G. Hasil Akhir



Deskripsi:

1. Fungsi tampilkanData() digunakan untuk menampilkan data mahasiswa yang tersimpan dalam Single Linked List.

2. Tahap pertama adalah melakukan pengecekan apakah list kosong dengan menggunakan fungsi isEmpty().

3. Jika list kosong, program akan mencetak pesan "List masih kosong!".

4. Selanjutnya, jika list tidak kosong, program akan melakukan penelusuran terhadap list menggunakan perulangan while.

5. Dalam penelusuran tersebut, data mahasiswa (nama dan NIM) akan dicetak dengan format yang sudah diatur agar rapi. Program ini menggunakan **system("cls")** perintah untuk menghapus layar konsol, dan **system("pause")** untuk menghentikan sementara program dan menunggu masukan pengguna.

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

3. 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 terakhir menjadi berikut:

Lucy 23300101

g) Hapus data awal

h) Ubah data awal menjadi berikut:

Bagas 23300002

i) Hapus data akhir

j) Tampilkan seluruh data

Screenshoot Output

```
=====
| DATA MAHASISWA |
=====
| NAMA                | NIM                |
=====
| Zefanya              | 2311102028         |
| Farrel               | 23300003           |
| Denis               | 23300005           |
| Anis                 | 23300008           |
| Bowo                 | 23300015           |
| Gahar                | 23300040           |
| Udin                 | 23300048           |
| Ucok                 | 23300050           |
| Budi                 | 23300099           |
=====
Press any key to continue . . .
```

Deskripsi

Data diatas merupakan salah satu bentuk (output) dari program yang sudah dibuat pada bagian A dengan menggunakan fungsi yang sudah dibuat yaitu

- a) tambahDepan();
- b) tambahTengah();
- c) tambahBelakang();

CASE

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

```
=====
| DATA MAHASISWA |
=====
| NAMA                | NIM                |
=====
| Zefanya              | 2311102028         |
| Farrel               | 23300003           |
| Wati                 | 23300004           |
| Denis               | 23300005           |
| Anis                 | 23300008           |
| Bowo                 | 23300015           |
| Gahar                | 23300040           |
| Udin                 | 23300048           |
| Ucok                 | 23300050           |
| Budi                 | 23300099           |
=====
Press any key to continue . . .
```

Deskripsi

Case pertama, user diminta untuk menginputkan data Wati yang terletak diantara Farrel dan Denis. Berdasarkan program yang sudah dibuat sebelumnya kita menggunakan fungsi tambahTengah() dengan memasukkan posisi Wati pada posisi yang ke 3 maka Farrel terletak di urutan ke 2 dan Denis akan berada di urutan ke 4. Sehingga Wati berada diantara tengah .

b) Hapus data Denis

```
=====
| DATA MAHASISWA |
=====
| NAMA          | NIM          |
=====
| Zefanya       | 2311102028   |
| Farrel        | 23300003     |
| Wati          | 23300004     |
| Anis          | 23300008     |
| Bowo          | 23300015     |
| Gahar         | 23300040     |
| Udin          | 23300048     |
| Ucok          | 23300050     |
| Budi          | 23300099     |
=====
Press any key to continue . . .
```

Deskripsi

Berdasarkan case diatas , user diminta untuk menghapus data Denis. Maka pada kondisi ini, user akan menggunakan fungsi hapusTengah() dengan memasukkan posisi atau urutan Denis pada data. Pada kondisi ini, Denis terletak pada urutan ke 4, maka pada inputan user memasukkan nilai yaitu 4.

c) Tambahkan data berikut di awal: Owi 2330000

```
=====
| DATA MAHASISWA |
=====
| NAMA          | NIM          |
=====
| Owi           | 2330000      |
| Zefanya       | 2311102028   |
| Farrel        | 23300003     |
| Wati          | 23300004     |
| Anis          | 23300008     |
| Bowo          | 23300015     |
| Gahar         | 23300040     |
| Udin          | 23300048     |
| Ucok          | 23300050     |
| Budi          | 23300099     |
=====
Press any key to continue . . .
```

Deskripsi

Pada case ini, user diminta untuk menginputkan data Owi di awal, maka user menggunakan fungsi tambahDepan() dengan memasukkan inputan nama dan nim. Dengan otomatis, data akan tersimpan pada posisi pertama.

d) Tambahkan data berikut di akhir: David 23300100

```
=====
| DATA MAHASISWA |
=====
| NAMA                | NIM                |
=====
| Owi                 | 2330000            |
| Zefanya             | 2311102028         |
| Farrel              | 23300003           |
| Wati                | 23300004           |
| Anis                | 23300008           |
| Bowo                | 23300015           |
| Gahar               | 23300040           |
| Udin                | 23300048           |
| Ucok                | 23300050           |
| Budi                | 23300099           |
| David               | 23300100           |
=====
Press any key to continue . . .
```

Deskripsi

Pada kondisi ini, user diminta untuk menginputkan data David pada urutan terakhir. Dengan mengetahui persoalan yang ada, user akan menggunakan fungsi `tambahBelakang()`, dengan mengisi inputan nama dan nim. Maka, data akan tersimpan pada urutan data terakhir.

e) Ubah data Udin menjadi data berikut: Idin 23300045

```
=====
| DATA MAHASISWA |
=====
| NAMA                | NIM                |
=====
| Owi                 | 2330000            |
| Zefanya             | 2311102028         |
| Farrel              | 23300003           |
| Wati                | 23300004           |
| Anis                | 23300008           |
| Bowo                | 23300015           |
| Gahar               | 23300040           |
| Idin                | 2300045            |
| Ucok                | 23300050           |
| Budi                | 23300099           |
| David               | 23300100           |
=====
Press any key to continue . . .
```

Deskripsi

Pada kondisi tersebut, user diminta untuk mengubah data Udin menjadi Idin. Maka pada kondisi ini, user akan menggunakan fungsi `ubahTengah()`. Pada fungsi ini, user akan diminta untuk memasukkan nama, nim, dan posisi. Pada kondisi data saat ini, Udin terletak di urutan data ke 8 maka inputan user memasukkan posisi ke-8. Maka, dengan logika program yang sudah dibuat data akan berubah dari data Udin menjadi data Idin.

f) Ubah data terakhir menjadi berikut: Lucy 23300101

```
=====
| DATA MAHASISWA |
=====
| NAMA                | NIM                |
=====
| Owi                 | 2330000            |
| Zefanya             | 2311102028         |
| Farrel              | 23300003           |
| Wati                | 23300004           |
| Anis                | 23300008           |
| Bowo                | 23300015           |
| Gahar               | 23300040           |
| Idin                | 2300045            |
| Ucok                | 23300050           |
| Budi                | 23300099           |
| Lucy                | 2300101            |
=====
Press any key to continue . . .
```

Deskripsi

Pada case ini, user diminta untuk mengubah data terakhir menjadi data Lucy. Maka, pada kondisi ini. User menggunakan fungsi ubahBelakang(), dengan memasukkan inputan nama dan nim, secara otomatis data akan berubah dengan data yang baru.

g) Hapus data awal

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS
Data depan berhasil dihapus!
Press any key to continue . . .
```

Deskripsi

Pada case ini, user diminta untuk menghapus data awal. Melihat kondisi ini, fungsi yang digunakan adalah hapusDepan(). Maka, data awal kan terhapus

h) Ubah data awal menjadi berikut: Bagus 2330002

```
=====
| DATA MAHASISWA |
=====
| NAMA                | NIM                |
=====
| Bagus               | 230002             |
| Owi                 | 2330000            |
| Zefanya             | 2311102028         |
| Farrel              | 23300003           |
| Wati                | 23300004           |
| Anis                | 23300008           |
| Bowo                | 23300015           |
| Gahar               | 23300040           |
| Idin                | 2300045            |
| Ucok                | 23300050           |
| Budi                | 23300099           |
| Lucy               | 2300101            |
=====
Press any key to continue . . .
```

Deskripsi

Pada case ini, user diminta untuk mengubah data awal menjadi data yang baru yaitu data Bagus. Pada kondisi berikut, user menggunakan fungsi ubahDepan(), kemudian memasukkan inputan berupa nama dan nim. Maka, data pun akan berubah dengan bentuk data yang baru.

i) Hapus data akhir

```
=====
| DATA MAHASISWA |
=====
| NAMA                | NIM                |
=====
| Bagus               | 230002             |
| Owi                 | 2330000            |
| Zefanya             | 2311102028         |
| Farrel              | 23300003           |
| Wati                | 23300004           |
| Anis                | 23300008           |
| Bowo                | 23300015           |
| Gahar               | 23300040           |
| Idin                | 2300045            |
| Ucok                | 23300050           |
| Budi                | 23300099           |
=====
Press any key to continue . . .
```

Deskripsi

Pada case ini, user diminta untuk menghapus data akhir. Berdasarkan kondisi ini, user akan menggunakan fungsi hapusBelakang(), maka data akhir pun akan terhapus seperti pada gambar di atas, data yang bernama "Lucy" sudah tidak ada.

j) Tampilkan seluruh data

```
=====
| DATA MAHASISWA |
=====
| NAMA                | NIM                |
=====
| Bagas               | 230002             |
| Owi                 | 2330000            |
| Zefanya             | 2311102028         |
| Farrel              | 23300003           |
| Wati                | 2330004            |
| Anis                | 23300008           |
| Bowo                | 23300015           |
| Gahar               | 23300040           |
| Idin                | 2300045            |
| Ucok                | 23300050           |
| Budi                | 23300099           |
=====
Press any key to continue . . .
```

Pada case ini, user diminta untuk menampilkan seluruh data yang sudah di inputkan berdasarkan case-case sebelumnya. Dengan menggunakan fungsi tampilkanData(), maka bentuk data awal yang sebelumnya sudah user inputkan akan berubah dengan bentuk data yang baru seperti gambar diatas.

BAB V

Kesimpulan

1. Linked List Non Circular adalah struktur data dengan simpul-simpul yang terhubung sekuensial. Node terakhir memiliki pointer yang menunjuk ke NULL. Operasi yang dapat dilakukan termasuk penambahan simpul, penghapusan simpul, dan pencarian data.

2. Linked List Circular adalah struktur data di mana node terakhir terhubung kembali ke node pertama, membentuk lingkaran tertutup. Node terakhir dalam Linked List Circular tidak bernilai NULL, melainkan menunjuk kembali ke node pertama. Linked List Circular digunakan dalam situasi di mana data perlu diakses berulang kali, seperti daftar putar lagu atau antrian pesan.

-Keuntungan Linked List Circular adalah tidak perlu mencari kembali ke head saat mencapai akhir linked list.

- Operasi umum pada linked list meliputi penambahan simpul, penghapusan simpul, pencarian data, dan menampilkan seluruh data dalam linked list.

BAB VI

DAFTAR PUSTAKA

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