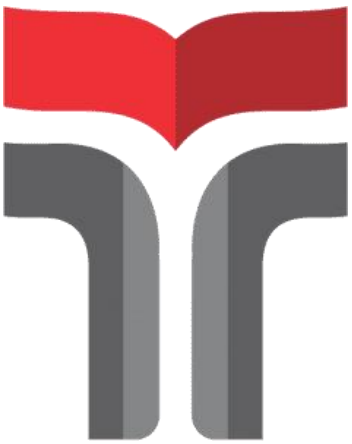
LAPORAN PRAKTIKUM STRUKTUR DATA DAN PEMROGRAGAMAN

MODUL 4

LINKED LIST CIRCULLAR DAN NON CIRCULLAR



Disusun oleh :

Rafa Aldhino Fatin

2311102023

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

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

BAB II
DASAR TEORI

LINKED LIST CIRCULLAR

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 circular adalah tipe dari struktur data linked list di mana setiap node memiliki referensi atau pointer ke node berikutnya dalam daftar, dan node terakhir menunjuk kembali ke node pertama. Ini menciptakan siklus atau lingkaran di antara node-node dalam struktur. Berbeda dengan linked list non-circular, di mana node terakhir menunjuk ke NULL, linked list circular dapat digambarkan sebagai suatu lingkaran, karena node terakhirnya kembali ke node pertama. Linked list circular memiliki keuntungan dan kelemahan tertentu tergantung pada konteks penggunaannya. Keuntungan utamanya adalah kemampuan untuk secara efisien memodelkan struktur data tertentu dan memprosesnya dalam beberapa algoritma.

LINKED LIST NON CIRCULLAR

linked list non-circular adalah struktur data linear di mana setiap elemen, disebut node, terdiri dari dua bagian: data itu sendiri dan sebuah pointer yang menunjuk ke node berikutnya dalam urutan. Di linked list non-circular, node terakhir menunjuk ke NULL, menandakan akhir dari daftar.linked list non-circular memiliki ujung yang jelas, di mana tidak ada siklus di dalam struktur. Ini berarti bahwa ketika mencapai node terakhir, tidak ada node berikutnya dalam urutan, dan iterasi melalui daftar dapat dilakukan dengan memeriksa apakah pointer menunjuk ke NULL. Linked list non-circular memiliki kelebihan dan kelemahan tertentu tergantung pada konteks penggunaannya. Keuntungan utamanya adalah fleksibilitas dalam penyisipan dan penghapusan elemen serta kemampuan untuk memanipulasi urutan data secara efisien. Namun, pengaksesan acak ke elemen dalam daftar memerlukan iterasi linier dari awal, yang bisa menjadi kurang efisien untuk daftar yang sangat besar.

BAB III
GUIDED

Guided 1

Source code

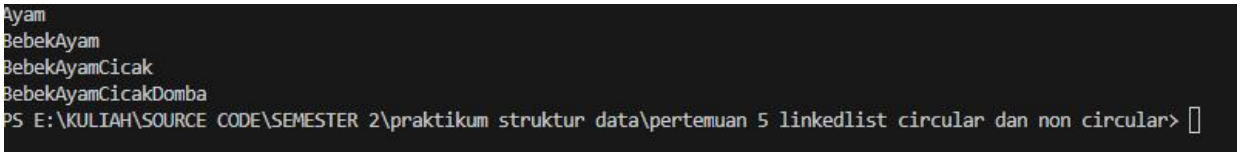
```
#include <iostream>
using namespace std;
/// PROGRAM SINGLE LINKED LIST CIRCULAR
// Deklarasi Struct Node
struct Node
{
    string data;
    Node *next;
};
Node *head, *tail, *baru, *bantu, *hapus;
void init()
{
    head = NULL;
    tail = head;
}
// Pengecekan
int isEmpty()
{
    if (head == NULL)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

```
// Tambah Belakang
void insertBelakang(string data)
{
   // Buat Node baru
   buatNode(data);
   if (isEmpty() == 1)
   {
      head = baru;
      tail = head;
      baru->next = head;
   }
   else
   {
      while (tail->next != head)
      {
         tail = tail->next;
      }
      tail->next = baru;
      baru->next = head;
   }
}
// Tambah Tengah
void insertTengah(string data, int posisi)
{
   if (isEmpty() == 1)
   {
      head = baru;
      tail = head;
      baru->next = head;
   }
   else
   {
      baru->data = data;
      // transversing
      int nomor = 1;
      bantu = head;
      while (nomor < posisi - 1)
      {
         bantu = bantu->next;
         nomor++;
      }
      baru->next = bantu->next;
      bantu->next = baru;
   }
}
// Hapus Depan
void hapusDepan()
{
   if (isEmpty() == 0)
   {
      hapus = head;
      tail = head;
      if (hapus->next == head)
          {
         head = NULL;
         tail = NULL;
      }
      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;
```

```
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();
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Program ini menyediakan fungsi-fungsi dasar untuk menambah, menghapus, dan menampilkan elemen-elemen dalam linked list tersebut. Struktur data Node digunakan untuk menyimpan data string dan pointer ke node berikutnya dalam linked list. Fungsi-fungsi seperti insertDepan(), insertBelakang(), insertTengah(), hapusDepan(), hapusBelakang(), hapusTengah(), serta tampil() digunakan untuk operasi-operasi dasar pada linked list seperti menambah, menghapus, dan menampilkan elemen-elemen. Selain itu, terdapat fungsi-fungsi pendukung seperti init() untuk menginisialisasi linked list, isEmpty() untuk memeriksa apakah linked list kosong, dan hitungList() untuk menghitung jumlah elemen dalam linked list.

BAB IV
UNGUIDED

UNGUIDED 1

SOURCE CODE

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

// Struct Node Declaration
struct Node {
    string nama;
    string nim;
    Node *next;
};

Node *head = NULL;
Node *tail = NULL;

// Function to Initialize List
void init() {
    head = NULL;
    tail = NULL;
}

// Function to Check if List is Empty
bool isEmpty() {
    return head == NULL;
}

// Function to Calculate the Number of Nodes in the List
int hitungList() {
    Node *hitung = head;
    int jumlah = 0;

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

    return jumlah;
}
void insertDepan(string nama, string 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, string 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;
   }
}
void insertTengah(string nama, string nim, int posisi) {
   if (posisi < 1 || posisi > hitungList() + 1) {
      cout << "Posisi di luar jangkauan." << endl;
   } else if (posisi == 1) {
      insertDepan(nama, nim);
   } else if (posisi == hitungList() + 1) {
      insertBelakang(nama, nim);
   } else {
      Node *baru = new Node;
      baru->nama = nama;
      baru->nim = nim;
      baru->next = NULL;

      Node *bantu = head;
      int nomor = 1;

      while (nomor < posisi - 1) {
         bantu = bantu->next;
         nomor++;
      }

      baru->next = bantu->next;
      bantu->next = baru;
   }
}
void ubahDepan(string nama, string nim) {
   if (!isEmpty()) {
      head->nama = nama;
      head->nim = nim;
   } else {
      cout << "List masih kosong!" << endl;
   }
}
void ubahBelakang(string nama, string nim) {
   if (!isEmpty()) {
      tail->nama = nama;
      tail->nim = nim;
   } else {
      cout << "List masih kosong!" << endl;
   }
}
```

```
void ubahTengah(string nama, string nim, int posisi) {
   if (!isEmpty()) {
      if (posisi < 1 || posisi > hitungList()) {
         cout << "Posisi di luar jangkauan" << endl;
      } else {
         Node *bantu = head;
         int nomor = 1;

         while (nomor < posisi) {
            bantu = bantu->next;
            nomor++;
         }

         bantu->nama = nama;
         bantu->nim = nim;
      }
   } else {
      cout << "List masih kosong!" << endl;
   }
}
void hapusDepan() {
   if (!isEmpty()) {
      Node *hapus = head;
      head = head->next;
      delete hapus;
   } else {
      cout << "List 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;
      }
   } else {
      cout << "List kosong!" << endl;
   }
}
```

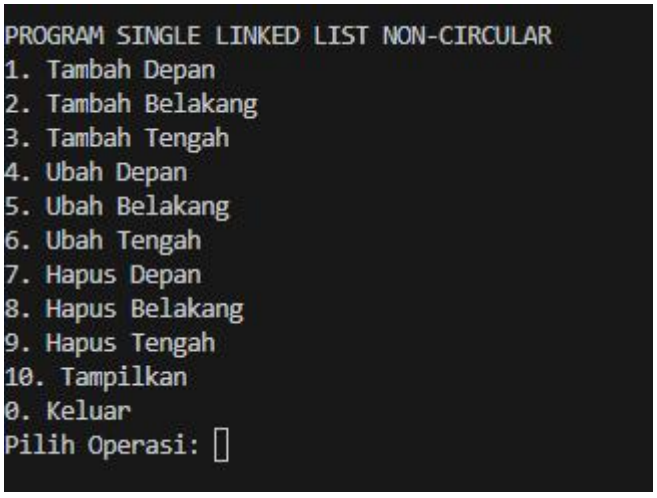
```
void hapusTengah(int posisi) {
   if (!isEmpty()) {
      if (posisi < 1 || posisi > hitungList()) {
         cout << "Posisi di luar jangkauan" << endl;
      } else if (posisi == 1) {
         hapusDepan();
      } else if (posisi == hitungList()) {
         hapusBelakang();
      } else {
         Node *hapus;
         Node *bantu = head;
         int nomor = 1;
         while (nomor < posisi - 1) {
            bantu = bantu->next;
            nomor++;
         }
         hapus = bantu->next;
         bantu->next = hapus->next;
         delete hapus;
      }
   } else {
      cout << "List kosong!" << endl;
   }
}
void tampil() {
   if (!isEmpty()) {
      Node *bantu = head;
      while (bantu != NULL) {
         cout << "Nama: " << bantu->nama << ", NIM: " << bantu->nim << endl;
         bantu = bantu->next;
      }
   } else {
      cout << "List kosong!" << endl;
   }
}
int main() {
   init(); // Initialize linked list

   int opsi;
   string nama, nim;
   int posisi;

   do {
      cout << "\nPROGRAM SINGLE LINKED LIST NON-CIRCULAR" << 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. Tampilkan" << endl;
      cout << "0. Keluar" << endl;
      cout << "Pilih Operasi: ";
      cin >> opsi;
```

```
switch (opsi) {
        case 1:
           cout << "Masukkan Nama: ";
           cin >> nama;
           cout << "Masukkan NIM: ";
           cin >> nim;
           insertDepan(nama, nim);
           break;
        case 2:
           cout << "Masukkan Nama: ";
           cin >> nama;
           cout << "Masukkan NIM: ";
           cin >> nim;
           insertBelakang(nama, nim);
           break;
        case 3:
           cout << "Masukkan Nama: ";
           cin >> nama;
           cout << "Masukkan NIM: ";
           cin >> nim;
           cout << "Masukkan Posisi: ";
           cin >> posisi;
           insertTengah(nama, nim, posisi);
           break;
        case 4:
           cout << "Masukkan Nama Baru untuk Depan: ";
           cin >> nama;
           cout << "Masukkan NIM Baru untuk Depan: ";
           cin >> nim;
           ubahDepan(nama, nim);
           break;
        case 5:
           cout << "Masukkan Nama Baru untuk Belakang: ";
           cin >> nama;
           cout << "Masukkan NIM Baru untuk Belakang: ";
           cin >> nim;
           ubahBelakang(nama, nim);
           break;
        case 6:
           cout << "Masukkan Nama Baru untuk Tengah: ";
           cin >> nama;
           cout << "Masukkan NIM Baru untuk Tengah: ";
           cin >> nim;
           cout << "Masukkan Posisi: ";
           cin >> posisi;
           ubahTengah(nama, nim, posisi);
           break;
        case 7:
           hapusDepan();
           break;
        case 8:
           hapusBelakang();
           break;
```

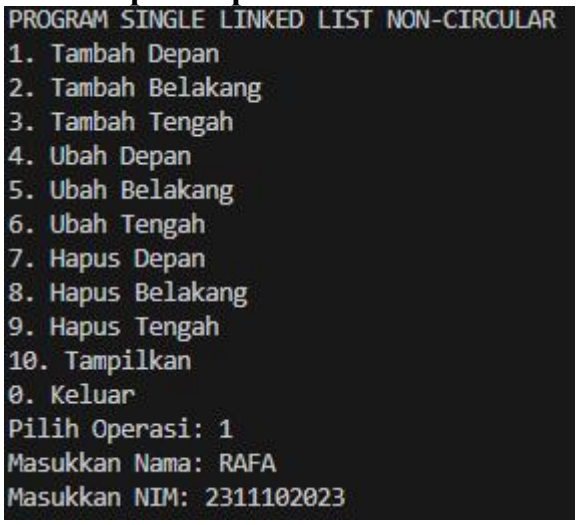
```
case 9:
            cout << "Masukkan Posisi untuk Menghapus: ";
            cin >> posisi;
            hapusTengah(posisi);
            break;
        case 10:
            tampil();
            break;
        case 0:
            cout << "Terima kasih!" << endl;
            break;
        default:
            cout << "Pilihan tidak valid!" << endl;
    }
  } while (opsi != 0);

  return 0;
}
```

SCREENSHOOT PROGRAM

- **Tampilan Menu**



- **Tampilan operasi tambah**



```
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. Tampilkan
0. Keluar
Pilih Operasi: 3
Masukkan Nama: udin
Masukkan NIM: 12342365
Masukkan Posisi: 2
```

- **Tampilan operasi hapus**

```
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. Tampilkan
0. Keluar
Pilih Operasi: 7
```

```
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. Tampilkan
0. Keluar
Pilih Operasi: 8
```

```
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. Tampilkan
0. Keluar
Pilih Operasi: 9
Masukkan Posisi untuk Menghapus: 1
```

- **Tampilan operasi ubah**

```
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. Tampilkan
0. Keluar
Pilih Operasi: 4
Masukkan Nama Baru untuk Depan: dino
Masukkan NIM Baru untuk Depan: 2311102023
```

```
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. Tampilkan
0. Keluar
Pilih Operasi: 5
Masukkan Nama Baru untuk Belakang: dina
Masukkan NIM Baru untuk Belakang: 8361552
```

```
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. Tampilkan
0. Keluar
Pilih Operasi: 6
Masukkan Nama Baru untuk Tengah: rafaal
Masukkan NIM Baru untuk Tengah: 72145753
Masukkan Posisi: 1
```

- **Tampilan operasi tampil data**

```
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. Tampilkan
0. Keluar
Pilih Operasi: 10
Nama: rafaal, NIM: 72145753
Nama: rafa, NIM: 2311102023
Nama: dina, NIM: 8361552
```

DESKRIPSI PROGRAM

Kode tersebut adalah implementasi dari linked list non-circular dalam bahasa C++. Program ini memungkinkan pengguna untuk melakukan berbagai operasi pada linked list, termasuk penambahan elemen di depan, di belakang, atau di tengah daftar, penghapusan elemen, dan pembaruan nilai elemen. Setiap elemen dalam linked list direpresentasikan oleh sebuah struct Node yang memiliki dua string untuk nama dan NIM mahasiswa, serta pointer next yang menunjuk ke node berikutnya dalam daftar. Implementasi ini menyediakan fungsi-fungsi seperti insertDepan, insertBelakang, insertTengah, ubahDepan, ubahBelakang, ubahTengah, hapusDepan, hapusBelakang, hapusTengah, dan tampil untuk mengelola linked list. Program utama menggunakan sebuah loop untuk menampilkan menu dan mengarahkan pengguna untuk memilih operasi yang diinginkan, serta melakukan operasi tersebut pada linked list yang sesuai dengan input pengguna. Jika operasi keluar dipilih, program akan berhenti.

.

UNGUIDED 2

```
Nama: jawad, NIM: 23300001
Nama: Rafa, NIM: 2311102023
Nama: farrel, NIM: 23300003
Nama: denis, NIM: 2330005
Nama: anis, NIM: 23300008
Nama: bowo, NIM: 23300040
Nama: gahar, NIM: 23300040
Nama: udin, NIM: 23300048
Nama: ucok, NIM: 23300050
Nama: budi, NIM: 23300099
```

UNGUIDED 3

Tambah data wati

```
Nama: jawad, NIM: 23300001
Nama: Rafa, NIM: 2311102023
Nama: farrel, NIM: 23300003
Nama: wati, NIM: 2330004
Nama: denis, NIM: 2330005
Nama: anis, NIM: 23300008
Nama: bowo, NIM: 23300040
Nama: gahar, NIM: 23300040
Nama: udin, NIM: 23300048
Nama: ucok, NIM: 23300050
Nama: budi, NIM: 23300099
```

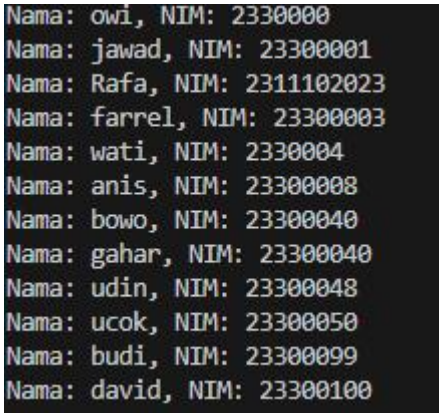
Hapus data denis

```
Nama: jawad, NIM: 23300001
Nama: Rafa, NIM: 2311102023
Nama: farrel, NIM: 23300003
Nama: wati, NIM: 2330004
Nama: anis, NIM: 23300008
Nama: bowo, NIM: 23300040
Nama: gahar, NIM: 23300040
Nama: udin, NIM: 23300048
Nama: ucok, NIM: 23300050
Nama: budi, NIM: 23300099
```

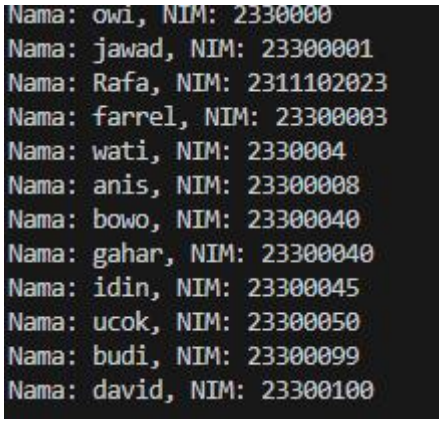
Tambah data Owi

```
Nama: owi, NIM: 2330000
Nama: jawad, NIM: 23300001
Nama: Rafa, NIM: 2311102023
Nama: farrel, NIM: 23300003
Nama: wati, NIM: 2330004
Nama: anis, NIM: 23300008
Nama: bowo, NIM: 23300040
Nama: gahar, NIM: 23300040
Nama: udin, NIM: 23300048
Nama: ucok, NIM: 23300050
Nama: budi, NIM: 23300099
```

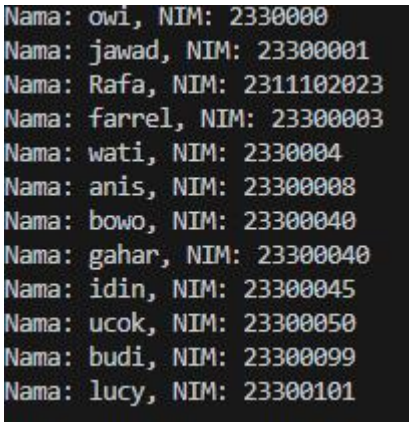
Tambah data david di akhir



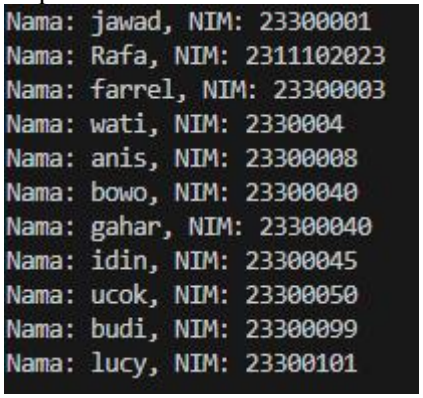
Ubah data udin



Ubah data terakhir



Hapus data awal



Ubah data awal

```
Nama: bagas, NIM: 2330002
Nama: Rafa, NIM: 2311102023
Nama: farrel, NIM: 23300003
Nama: wati, NIM: 2330004
Nama: anis, NIM: 23300008
Nama: bowo, NIM: 23300040
Nama: gahar, NIM: 23300040
Nama: idin, NIM: 23300045
Nama: ucok, NIM: 23300050
Nama: budi, NIM: 23300099
Nama: lucy, NIM: 23300101
```

Hapus data akhir

```
Nama: bagas, NIM: 2330002
Nama: Rafa, NIM: 2311102023
Nama: farrel, NIM: 23300003
Nama: wati, NIM: 2330004
Nama: anis, NIM: 23300008
Nama: bowo, NIM: 23300040
Nama: gahar, NIM: 23300040
Nama: idin, NIM: 23300045
Nama: ucok, NIM: 23300050
Nama: budi, NIM: 23300099
```

Tampilkan seluruh data

```
Nama: bagas, NIM: 2330002
Nama: Rafa, NIM: 2311102023
Nama: farrel, NIM: 23300003
Nama: wati, NIM: 2330004
Nama: anis, NIM: 23300008
Nama: bowo, NIM: 23300040
Nama: gahar, NIM: 23300040
Nama: idin, NIM: 23300045
Nama: ucok, NIM: 23300050
Nama: budi, NIM: 23300099
```

BAB V

KESIMPULAN

Linked list non-circular adalah struktur data linear di mana setiap elemen memiliki pointer yang menunjuk ke elemen berikutnya, dan elemen terakhir menunjuk ke null, sehingga tidak membentuk siklus tertutup. Operasi pada linked list non-circular memerlukan penanganan khusus saat menambah dan menghapus elemen di ujung list. linked list circular adalah versi modifikasi dari linked list non-circular di mana elemen terakhir menunjuk kembali ke elemen pertama, membentuk siklus tertutup. Ini memungkinkan akses langsung ke elemen-elemen di seluruh list tanpa perlu mengubah pointer tail, tetapi memerlukan penanganan tambahan saat menyisipkan atau menghapus elemen di berbagai posisi dalam list untuk memastikan integritas siklus.

DAFTAR PUSTAKA

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