LAPORAN PRAKTIKUM

MODUL 4
LINKED LIST CIRCULAR DAN NON CIRCULAR

Disusun Oleh:
Riyon Aryono **: 2211102241**

Dosen

Muhammad Afrizal Amrustian

PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM
PURWOKERTO

2023

BAB I
TUJUAN PRAKTIKUM

A. Tujuan Praktikum

1. Mahasiswa memahami perbedaan konsep Linked List Circular dan Non Circular
2. Mahasiswa mampu menerapkan Linked List Circular dan Non Circular

BAB II
DASAR TEORI

1. Linked List Non Circular

Linked list non circular merupakan linked list dimana antara kepala (head) dan node terakhir (tail) tidak memiliki hubungan. Pada Linked List ini maka pointer terakhir selalu menunjuk ‘NULL’ sebagai pertanda data terakhir dalam list-nya.

2. Linked List Circular

Linked list circular adalah struktur data linked list yang memiliki sifat berputar atau mengulang dari node head ke node tail. Dalam linked list circular, node tail tidak menunjuk ke ‘NULL’, melainkan mengarah kembali ke node head, sehingga membentuk lingkaran. Pada node tail dalam linked list circular, pointer menunjuk kembali ke node head.

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

BAB III
LATIHAN & TUGAS

1. Guided

- Demo Linked List Non Circular

Source Code

```
#include <iostream>
using namespace std;
/// PROGRAM SINGLE LINKED LIST NON-CIRCULAR
// Deklarasi Struct Node
struct Node
{
   int data;
   Node *next;
};
Node *head;
Node *tail;

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

// Pengecekan
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;
      tail->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;
        tail->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 diluar 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 << "List 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 << "List kosong!" << endl;
    }
}

// Hapus Tengah
void hapusTengah(int posisi)
{
    Node *bantu, *hapus, *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 << "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{
            bantu = head;
            int nomor = 1;
            while (nomor < posisi){
                bantu = bantu->next;
                nomor++;
            }
            bantu->data = data;
        }
    }else{
        cout << "List masih kosong!" << endl;
    }
}

// Ubah Belakang
void ubahBelakang(int data){
    if (isEmpty() == 0){
        tail->data = data;
    } else {
        cout << "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 tampil(){
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)  {
        while (bantu != NULL)   {
            cout << bantu->data << ends;
            bantu = bantu->next;
        }

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

int main(){
    init();
    insertDepan(3);
    tampil();
    insertBelakang(5);
    tampil();
    insertDepan(2);
    tampil();
    insertDepan(1);
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, 2);
    tampil();
```

```
    hapusTengah(2);
    tampil();
    ubahDepan(1);
    tampil();
    ubahBelakang(8);
    tampil();
    ubahTengah(11, 2);
    tampil();
    return 0;
}
```

Output

Penjelesan

Pada Program diatas kita mengimplementasi program single linked list non-circular. Kita juga mengimplementasi beberapa hal seperti menambahkan data didepan tengah dan belakang lalu juga ada ubah depan tengah dan belakang dan yang terakhir kita juga mengimplementasi kan hapus depan tengah dan belakang.

- Demo Linked List Circular

  Source Code

```
#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(){
    if(head == NULL){
        return 1;
    }else{
        return 0;
    }
}

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

int hitungList(){
    bantu = head;
    int jumlah = 0;

    while(bantu != 0){
        jumlah++;
        bantu = bantu->next;
    }

    return jumlah;
}

void insertDepan(string data){
    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;
    }
}

void insertBelakang(string data){
    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;
    }
}

void insertTengah(string data, int posisi){
    if(isEmpty() == 1){
        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() == 0){
```

```
        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() == 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;
```

```
        }
    }else{
        cout << "List masih Kosong!" <<endl;
    }
}

void hapusTengah(int posisi){
    if(isEmpty() == 0){
        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 dihapus!" <<endl;
}

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;
}
```

Output

```
Ayam
Bebek Ayam
Bebek Ayam Cicak
Bebek Ayam Cicak Domba
Bebek Ayam Cicak
Ayam Cicak
Ayam Sapi Cicak
Ayam Cicak
```

Penjelesan

Kode program diatas adalah contoh Impelementasi Linked List Circular. Jika kita memperhatikan kode nya tidak ada perbedan yang signifikan antara Linkedl List Circular dan Non Circular yang membedakan saat inisialisasi tail akan berisi head beda dengan non circular tail berisi NULL

2. Unguided

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

  Source Code

```
#include <iostream>
#include <iomanip>

using namespace std;

struct Node{
    string name;
    long long nim;

    Node *next;
};

Node *head, *tail;

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


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

// - insertfirst
void insertFirst(string name, long long nim){
    Node *baru = new Node;
    baru->name = name;
    baru->nim  = nim;
    baru->next = NULL;
    if(isEmpty() == true){
        head = tail = baru;
        tail->next  = NULL;
    }else{
        baru->next = head;
        head = baru;
```

```
    }
}

// insertBack
void insertBack(string name, long long nim){
    Node *newNode = new Node;
    newNode->name = name;
    newNode->nim = nim;
    newNode->next = NULL;

    if(isEmpty() == true){
        head = tail = newNode;
        tail->next  = NULL;
    }else{
        tail->next =newNode;
        tail = newNode;
    }
}

int countList(){
    Node *count;
    count = head;
    int num = 0;

    while(count != NULL){
        num++;
        count = count->next;
    }

    return num;
}

    // - insertMiddle
void insertMiddle(string name, long long nim, int pos ){
    if(pos < 1 || pos > countList()){
        cout << "Posisi diluar jangkauan cuy" <<endl;
    }else if(pos == 1){
        cout << "Posisi bukan ditengah" <<endl;
    }else{
        Node *newNode, *temp;
        newNode = new Node();
        newNode->name = name;
        newNode->nim  = nim;
        temp = head;

        int num = 1;
```

```
        while(num < pos - 1){
            temp = temp->next;
            num++;
        }
        newNode->next = temp->next;
        temp->next = newNode;
    }
}

void changeFirst(string name, long long nim){
        if(isEmpty() == false){
            head->name = name;
            head->nim = nim;
        }else{
            cout << "List masih kosong!" << endl;
        }
}


    // - changeMiddle
void changeMiddle(string name, long long nim, int pos){
    Node *temp;
    if(isEmpty() == 0){
        if(pos < 1 || pos > countList()){
            cout << "Posisi diluar jangakauan cuy" << endl;
        }else if(pos == 1){
            cout << "Posisi bukan ditengah" <<endl;
        }else{
            temp = head;
            int num = 1;
            while(num < pos){
                temp = temp->next;
                num++;
            }
            string n = temp->name;
            temp->name = name;
            temp->nim  = nim;

            cout << "Data " << n << " telah diganti dengan data " << name <<
   endl<<endl;
        }
    }else{
        cout << "List masih kosong" << endl;
    }

}
```

```
   // - changeBack
void changeBack(string name, long long nim){
   if(isEmpty() == 0){
      string n = tail->name;
      tail->name = name;
      tail->nim  = nim;
      cout << "Data " << n << " telah diganti dengan data " << name
  <<endl<<endl;
   }else{
      cout << "List masih kosong cuy" << endl;
   }
}
   // - delete back
void deleteBack(){
   Node *del, *temp;

   if(isEmpty() == false){
      if(head != tail){
         string n = tail->name;
         del = tail;
         temp = head;
         while(temp->next != tail){
            temp = temp->next;
         }
         tail = temp;
         tail->next = NULL;
         delete del;

         cout << "Data " << n << " berhasil dihapus"<<endl<<endl;
      }else{
         head = tail = NULL;
      }
   }else{
      cout << "List kamu masih kosong nih" << endl;
   }
}
// Delete First
void deleteFirst(){
   Node *del;
   if(isEmpty() == false){
      if(head->next != NULL){
         del = head;
         head  = head->next;
         delete del;
      }else{
```

```
            head = tail = NULL;
        }
    }else{
        cout << "List Kosong" << endl;
    }
}

// - deleteMiddle
void deleteMiddle(int pos){
    Node *temp, *del, *prev;
    if(pos < 1 || pos > countList()){
        cout << "Posisi diluar jangkauan cuy" <<endl;
    }else if(pos == 1){
        cout << "Posisi bukan ditengah" <<endl;
    }else{
        int num = 1;
        temp = head;

        while(num  <= pos){
            if(num == pos - 1){
                prev = temp;
            }if(num == pos){
                del = temp;
            }

            temp = temp->next;
            num++;
        }
        string n = del->name;
        prev->next = temp;
        delete del;

        cout << "Data " << n << " berhasil dihapus!"<<endl<<endl;
    }
}

// clear List
void clearList(){
     Node *temp, *del;
    temp = head;
    while(temp != NULL){
        del = temp;
        temp = temp->next;
        delete del;
    }
    head = tail = NULL;
```

```
    cout << "List berhasil dihapus!"<< endl;
}

   // - Display
void display(){
   Node *temp;
   temp = head;
   if(isEmpty() == false){
      cout << "DATA MAHASISWA" <<endl<<endl;
      cout << left << setw(15) << "Nama" << left << setw(20) << "Nim"
  <<endl;
      while(temp != NULL){
         cout << left << setw(15) << temp->name
             << left << setw(30) << temp->nim  << endl;
         temp = temp->next;
      }
      cout <<endl;
   }else{
      cout << "List kamu masih kosong nih, isi dulu ya";
   }
}


int main(){
   init();

   while(true){
      cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR"
  <<endl<<endl;
      cout << "1. Tambah Depan"<<endl;
      cout << "2. Tambah Belakang"<<endl;
      cout << "3. Tambah Tengah"<<endl;
      cout << "4. Ubah Depan"<<endl;
      cout << "5. Ubah Belakang"<<endl;
      cout << "6. Ubah Tengah"<<endl;
      cout << "7. Hapus Depan"<<endl;
      cout << "8. Hapus Belakang"<<endl;
      cout << "9. Hapus Tengah"<<endl;
      cout << "10. Hapus List"<<endl;
      cout << "11. Tampilkan"<<endl;
      cout << "0. Keluar"<<endl<<endl;

      int choice;
      cout << "Pilih Operasi : ";
      cin  >>choice;
```

```
        switch (choice){
            case 1:{
                string name;long long nim;
                cout << "\n-Tambah Depan-"<<endl<<endl;
                cout << "Masukan Nama : ";
                cin  >> name;
                cout << "Masukkan Nim : ";
                cin  >> nim;
                cout << endl;

                insertFirst(name, nim);
                cout << "Data " << name << " berhasil diinput!" <<endl<<endl;
                break;
            }
            case 2:{
                string name; long long nim;
                cout << "\n-Tambah Belakang-" <<endl<<endl;
                cout << "Masukkan Nama : ";
                cin  >> name;
                cout << "Masukkan Nim : ";
                cin  >> nim;
                cout << endl;
                insertBack(name,nim);
                cout << "Data " << name << " berhasil diinput!" <<endl<<endl;
                break;
            }
            case 3:{
                string name; long long nim; int pos;
                cout << "\n-Tambah Tengah-" <<endl<<endl;
                cout << "Masukkan Nama : ";
                cin  >> name;
                cout << "Masukkan Nim : ";
                cin  >> nim;
                cout << "Masukkan Posisi : ";
                cin  >> pos;
               cout << endl;

                insertMiddle(name, nim, pos);
                cout << "Data " << name << " berhasil diinput!" <<endl<<endl;
                break;
            }
            case 4:{
                changeFirst("Sample",2121);
                break;
            }
            case 5:{
```

```
            string name; long long nim;
            cout << "\n-Ubah Belakang-" <<endl<<endl;
            cout << "Masukkan Nama : ";
            cin  >> name;
            cout << "Masukkan Nim : ";
            cin  >> nim;
            cout << endl;

            changeBack(name, nim);
        break;
    }
    case 6:{
            string name; long long nim; int pos;
            cout << "\n-Ubah Tengah-" <<endl<<endl;
            cout << "Masukkan Nama : ";
            cin  >> name;
            cout << "Masukkan Nim : ";
            cin  >> nim;
            cout << "Masukkan Posisi : ";
            cin  >> pos;
            cout << endl;

            changeMiddle(name, nim, pos);
        break;
    }
    case 7:{
        deleteFirst();
        break;
    }
    case 8:{
        cout << "-Hapus Belakang-" <<endl<<endl;
        deleteBack();
        break;
    }
    case 9:{
        int num;
        cout << "Masukkan posisi : ";
        cin  >> num;
        cout <<endl;
        deleteMiddle(num);
        break;
    }
    case 10:{
        clearList();
        break;
    }
```

```
        case 11:{
            display();
            break;
        }
        case 0:{
            return 0;
            break;
        }
        default:{
            return 0;
            break;
        }
    }
  }

  return 0;
}
```

Output

Case 1

```
-Tambah Depan-

Masukan Nama : Riyon
Masukkan Nim : 2211102241

Data Riyon berhasil diinput!
```

```
Nama            Nim
Riyon           2211102241
Alvin           22200001
Candra          22200002
Niken           22200005
Joko            22200008
Friska          22200015
Gabriel         22200040
Karin           22200020
```

Case 2

```
-Hapus Belakang-

Data Karin berhasil dihapus
```

Case 3

```
-Tambah Tengah-

Masukkan Nama : Cika
Masukkan Nim : 22200003
Masukkan Posisi : 6

Data Cika berhasil diinput!
```

Case 4

```
Masukkan posisi : 5

Data Joko berhasil dihapus!
```

Case 5

```
-Tambah Depan-

Masukan Nama : Dimas
Masukkan Nim : 22200010

Data Dimas berhasil diinput!
```

Case 6

```
-Tambah Tengah-

Masukkan Nama : Vina
Masukkan Nim : 22200022
Masukkan Posisi : 2

Data Vina berhasil diinput!
```

Case7

```
-Ubah Belakang-

Masukkan Nama : Jamal
Masukkan Nim : 22200033

Data Gabriel telah diganti dengan data Jamal
```

```
-Ubah Tengah-

Masukkan Nama : April
Masukkan Nim : 22200017
Masukkan Posisi : 6

Data Niken telah diganti dengan data April
```

```
-Tambah Belakang-

Masukkan Nama : Budi
Masukkan Nim : 22200000

Data Budi berhasil diinput!
```

```
-Ubah Tengah-

Masukkan Nama : Candra
Masukkan Nim : 21200055
Masukkan Posisi : 5

Data Candra telah diganti dengan data Candra
```

```
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 : 11
DATA MAHASISWA

Nama            Nim
Dimas           22200010
Vina            22200022
Riyon           2211102241
Alvin           22200001
Candra          21200055
April           22200017
Cika            22200003
Friska          22200015
Jamal           22200033
Budi            22200000
```

Penjelesan

Kode Program diatas menggunakan Single List Non Circular menggunakan *Struct*. Langkah pertama kita inisalisasi *Strut* terlebih dahulu, pada kode diatas *Struct* memiliki dua data yaitu name dan nim dan memiliki satu pointer yaitu pointer next. Untuk operasi nya terdapat beberapa seperti tambah depan, tambah tengah, tambah belakang, delete depan, delete tengah, delete belakang, ubah depan, ubah tengah dan ubah belakang

BAB IV
KESIMPULAN

Linked List Non-Circular adalah jenis Linked List di mana simpul terakhir tidak menunjuk pada simpul pertama, sehingga tidak membentuk suatu lingkaran. Jadi, simpul terakhir memiliki nilai null sebagai alamat pointer, yang menandakan akhir dari Linked List.

Sedangkan Single Linked List Circular adalah jenis Linked List di mana simpul terakhir menunjuk pada simpul pertama sehingga membentuk lingkaran. Ini memungkinkan untuk melakukan iterasi ke depan secara tak terbatas, karena tidak ada akhir dari Linked List.