LAPORAN PRAKTIKUM

MODUL IV
LINKED LIST CIRCULAR DAN NON CIRCULAR

Disusun oleh:
FATTAH RIZQY ADHIPRATAMA
NIM: 2311102019

Dosen Pengampu:
Wahyu Andi Saputra, S.Pd., M.Eng.

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

BAB I

TUJUAN PRAKTIKUM

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

BAB II
DASAR TEORI

Linked list adalah struktur data linier yang terdiri dari sejumlah simpul (node) yang saling terhubung melalui referensi atau pointer. Setiap simpul dalam linked list menyimpan data dan memiliki sebuah pointer yang menunjuk ke simpul berikutnya dalam urutan linear.

Konsep dasar dari linked list adalah bahwa setiap simpul mengandung dua bagian utama: data dan pointer. Data mewakili informasi yang ingin disimpan, misalnya bilangan, teks, atau objek lainnya. Pointer, juga disebut sebagai “next pointer,” menunjuk ke simpul berikutnya dalam urutan.

Perbedaan utama antara linked list dengan struktur data lainnya, seperti array, adalah kemampuannya untuk mengalokasikan ruang secara dinamis saat program berjalan. Hal ini memungkinkan penggunaan memori yang efisien, karena linked list dapat tumbuh atau menyusut sesuai kebutuhan.

Dalam linked list, simpul pertama disebut sebagai “head” atau “kepala,” sedangkan simpul terakhir dalam urutan disebut sebagai “tail” atau “ekor.” Ketika tail memiliki nilai pointer yang menunjuk ke null, itu menandakan akhir dari linked list. Salah satu kelebihan linked list adalah kemampuannya untuk menyisipkan dan menghapus elemen dengan cepat di tengah-tengah linked list, meskipun operasi ini memerlukan penyesuaian pointer. Namun, akses acak ke elemen dalam linked list lebih lambat dibandingkan dengan array, karena untuk mencapai elemen tertentu, perlu dilakukan perjalanan melalui simpul-simpul sebelumnya.

Circular linked list adalah linked list unidirectional. Kita hanya dapat melintasinya dalam satu arah. Tetapi jenis linked list ini memiliki simpul terakhir yang menunjuk ke simpul kepala. Jadi saat melintas, kita harus berhati-hati dan berhenti saat mengunjungi kembali simpul kepala.

BAB III

GUIDED

1. 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;
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)
        {
        }
        else
        {
            cout << "Posisi bukan posisi tengah" << endl;
            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;
    }
}
```

```cpp
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;
}
```

Screenshoot program

```
PS D:\Data Semester 2\Praktikum Struktur Data dan Algoritme\Modul 4>
$?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) {
3
35
235
1235
235
23
273
23
13
18
Posisi bukan posisi tengah
111
PS D:\Data Semester 2\Praktikum Struktur Data dan Algoritme\Modul 4>
```

Deskripsi program

Program ini menunjukkan operasi dasar pada Single Linked List Non-Circular C++. Fungsi-fungsi yang disediakan memungkinkan pengguna untuk menambahkan, menghapus, mengubah, dan menampilkan elemen dalam list dengan mudah.

2. Guided 2

Source Code

```
#include <iostream>
using namespace std;
/// PROGRAM SINGLE LINKED LIST CIRCULAR
// Deklarasi Struct Node
struct Node
{
    string data;
    Node *next;
};
Node *head, *tail, *baru, *bantu, *hapus;
void init()
{
    head = NULL;
    tail = head;
}
// Pengecekan
int isEmpty()
{
    if (head == NULL)
        return 1; // true
    else
        return 0; // false
}
// Buat Node Baru
void buatNode(string data)
{
    baru = new Node;
    baru->data = data;
    baru->next = NULL;
}
// Hitung List
int hitungList()
{
```

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

```
    }
    else
    {
        while (tail->next != head)
        {
            tail = tail->next;
        }
        tail->next = baru;
        baru->next = head;
    }
}
// Tambah Tengah
void insertTengah(string data, int posisi)
{
    if (isEmpty() == 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;
            delete hapus;
        }
        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;
            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;
    }
}
// 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

```
PS D:\Data Semester 2\Praktikum Struktur Data dan Algoritme\Modul 4>
$?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) {
Ayam
BebekAyam
BebekAyamCicak
BebekAyamCicakDomba
BebekAyamCicak
PS D:\Data Semester 2\Praktikum Struktur Data dan Algoritme\Modul 4>
```

Deskripsi Program

Program ini menunjukkan operasi dasar pada Single Linked List Circular C++. Fungsi-fungsi yang disediakan memungkinkan pengguna untuk menambahkan, menghapus, mengubah, dan menampilkan elemen dalam list dengan mudah. Perbedaan utama dengan Single Linked List Non-Circular adalah list ini melingkar, sehingga tail->next selalu menunjuk ke head.

LATIHAN KELAS - UNGUIDED

1. **Unguided 1**

   Source code

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

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

class LinkedList {
private:
    Node* head;

public:
    LinkedList() {
        head = nullptr;
    }

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

    void tambahBelakang(string nama, string nim) {
        Node* newNode = new Node;
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = nullptr;
        if (head == nullptr) {
            head = newNode;
            return;
```

```
        }
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
        cout << "Data telah ditambahkan" << endl;
    }

    void tambahTengah(string nama, string nim, int posisi) {
        if (posisi <= 0) {
            cout << "Posisi tidak valid" << endl;
            return;
        }
        Node* newNode = new Node;
        newNode->nama = nama;
        newNode->nim = nim;
        Node* temp = head;
        for (int i = 0; i < posisi - 1; i++) {
            if (temp == nullptr) {
                cout << "Posisi tidak valid" << endl;
                return;
            }
            temp = temp->next;
        }
        if (temp == nullptr) {
            cout << "Posisi tidak valid" << endl;
            return;
        }
        newNode->next = temp->next;
        temp->next = newNode;
        cout << "Data telah ditambahkan" << endl;
    }

    void hapusDepan() {
        if (head == nullptr) {
            cout << "Linked list kosong" << endl;
            return;
        }
        Node* temp = head;
        head = head->next;
```

```
        delete temp;
        cout << "Data berhasil dihapus" << endl;
    }

    void hapusBelakang() {
        if (head == nullptr) {
            cout << "Linked list kosong" << endl;
            return;
        }
        if (head->next == nullptr) {
            delete head;
            head = nullptr;
            cout << "Data berhasil dihapus" << endl;
            return;
        }
        Node* temp = head;
        while (temp->next->next != nullptr) {
            temp = temp->next;
        }
        delete temp->next;
        temp->next = nullptr;
        cout << "Data berhasil dihapus" << endl;
    }

    void hapusTengah(int posisi) {
        if (posisi <= 0 || head == nullptr) {
            cout << "Linked list kosong atau posisi tidak valid" <<
endl;
            return;
        }
        if (posisi == 1) {
            hapusDepan();
            return;
        }
        Node* temp = head;
        for (int i = 0; i < posisi - 2; i++) {
            if (temp->next == nullptr) {
                cout << "Posisi tidak valid" << endl;
                return;
            }
            temp = temp->next;
```

```
        }
        if (temp->next == nullptr) {
            cout << "Posisi tidak valid" << endl;
            return;
        }
        Node* nodeToDelete = temp->next;
        temp->next = temp->next->next;
        delete nodeToDelete;
        cout << "Data berhasil dihapus" << endl;
    }

     void ubahDepan(string namaBaru, string nimBaru) {
        if (head == nullptr) {
            cout << "Linked list kosong" << endl;
            return;
        }
        head->nama = namaBaru;
        head->nim = nimBaru;
        cout << "Data berhasil diubah" << endl;
    }

    void ubahBelakang(string namaBaru, string nimBaru) {
        if (head == nullptr) {
            cout << "Linked list kosong" << endl;
            return;
        }
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->nama = namaBaru;
        temp->nim = nimBaru;
        cout << "Data berhasil diubah" << endl;
    }

    void ubahTengah(string namaBaru, string nimBaru, int posisi) {
        if (posisi <= 0 || head == nullptr) {
            cout << "Linked list kosong atau posisi tidak valid" <<
endl;
            return;
        }
```

```
        Node* temp = head;
        for (int i = 0; i < posisi - 1; i++) {
            if (temp == nullptr) {
                cout << "Posisi tidak valid" << endl;
                return;
            }
            temp = temp->next;
        }
        if (temp == nullptr) {
            cout << "Posisi tidak valid" << endl;
            return;
        }
        temp->nama = namaBaru;
        temp->nim = nimBaru;
        cout << "Data berhasil diubah" << endl;
    }

    void hapusList() {
        Node* current = head;
        Node* next;
        while (current != nullptr) {
            next = current->next;
            delete current;
            current = next;
        }
        head = nullptr;
        cout << "Linked list berhasil dihapus" << endl;
    }

    void tampilkanData() {
        Node* temp = head;
        cout << "DATA MAHASISWA" << endl;
        cout << "NAMA\tNIM" << endl;
        while (temp != nullptr) {
            cout << temp->nama << "\t" << temp->nim << endl;
            temp = temp->next;
        }
    }
};

int main() {
```

```
    LinkedList linkedList;
    int choice;
    string nama, nim;
    int posisi;

    do {
        cout << "PROGRAM 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. Hapus List" << endl; // Menu untuk hapus list
        cout << "11. Tampilkan Data" << endl;
        cout << "12. Keluar" << endl;
        cout << "Pilih Operasi : ";
        cin >> choice;

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

```
            cout << "Masukkan Nama : ";
            cin >> nama;
            cout << "Masukkan NIM : ";
            cin >> nim;
            cout << "Masukkan Posisi : ";
            cin >> posisi;
            linkedList.tambahTengah(nama, nim, posisi);
            break;
        case 4:
            cout << "-Ubah Depan-" << endl;
            cout << "Masukkan Nama Baru : ";
            cin >> nama;
            cout << "Masukkan NIM Baru : ";
            cin >> nim;
            linkedList.ubahDepan(nama, nim);
            break;
        case 5:
            cout << "-Ubah Belakang-" << endl;
            cout << "Masukkan Nama Baru : ";
            cin >> nama;
            cout << "Masukkan NIM Baru : ";
            cin >> nim;
            linkedList.ubahBelakang(nama, nim);
            break;
        case 6:
            cout << "-Ubah Tengah-" << endl;
            cout << "Masukkan Nama Baru : ";
            cin >> nama;
            cout << "Masukkan NIM Baru : ";
            cin >> nim;
            cout << "Masukkan Posisi : ";
            cin >> posisi;
            linkedList.ubahTengah(nama, nim, posisi);
            break;
        case 7:
            linkedList.hapusDepan();
            break;
        case 8:
            linkedList.hapusBelakang();
            break;
        case 9:
```

```
                cout << "-Hapus Tengah-" << endl;
                cout << "Masukkan Posisi : ";
                cin >> posisi;
                linkedList.hapusTengah(posisi);
                break;
            case 10:
                linkedList.hapusList(); // Hapus List
                break;
            case 11:
                linkedList.tampilkanData();
                break;
            case 12:
                cout << "Program selesai." << endl;
                break;
            default:
                cout << "Pilihan tidak valid." << endl;
        }
    } while (choice != 12);

    return 0;
}
```

Screenshoot program

Tampilan data awal

```
11. Tampilkan Data
0. Keluar
Pilih Operasi : 11
DATA MAHASISWA
NAMA    NIM
Jawad   23300001
Fattah  2311102019
Farrel  23300003
Denis   23300005
Anis    23300008
Bowo    23300015
Gahar   23300040
Udin    23300048
Ucok    23300050
Budi    23300099
```

Tambahkan data Wati diantara Farrel dan Denis

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA    NIM
Jawad   23300001
Fattah  2311102019
Farrel  23300003
Wati    2330004
Denis   23300005
Anis    23300008
Bowo    23300015
Gahar   23300040
Udin    23300048
Ucok    23300050
Budi    23300099
```

Hapus data Denis

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA    NIM
Jawad   23300001
Fattah  2311102019
Farrel  23300003
Wati    2330004
Anis    23300008
Bowo    23300015
Gahar   23300040
Udin    23300048
Ucok    23300050
Budi    23300099
```

Tambahkan data Owi di awal

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA    NIM
Owi     2330000
Jawad   23300001
Fattah  2311102019
Farrel  23300003
Wati    2330004
Anis    23300008
Bowo    23300015
Gahar   23300040
Udin    23300048
Ucok    23300050
Budi    23300099
```

Tambahkan data David di akhir

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA    NIM
Owi     2330000
Jawad   23300001
Fattah  2311102019
Farrel  23300003
Wati    2330004
Anis    23300008
Bowo    23300015
Gahar   23300040
Udin    23300048
Ucok    23300050
Budi    23300099
David   23300100
```

Ubah data Udin menjadi Idin

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA    NIM
Owi     2330000
Jawad   23300001
Fattah  2311102019
Farrel  23300003
Wati    2330004
Anis    23300008
Bowo    23300015
Gahar   23300040
Idin    23300045
Ucok    23300050
Budi    23300099
David   23300100
```

Ubah data terakhir menjadi Lucy

```
Pilih Operasi : 1
DATA MAHASISWA
NAMA    NIM
Owi     2330000
Jawad   23300001
Fattah  231110201
Farrel  23300003
Wati    2330004
Anis    23300008
Bowo    23300015
Gahar   23300040
Idin    23300045
Ucok    23300050
Budi    23300099
Lucy    23300101
```

Hapus data awal

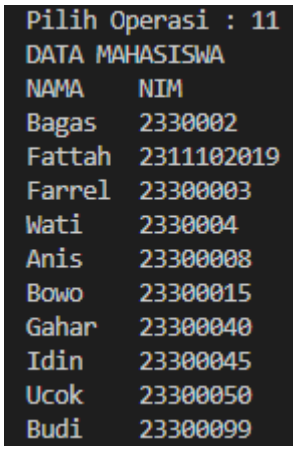
```
Pilih Operasi : 11
DATA MAHASISWA
NAMA    NIM
Jawad   23300001
Fattah  2311102019
Farrel  23300003
Wati    2330004
Anis    23300008
Bowo    23300015
Gahar   23300040
Idin    23300045
Ucok    23300050
Budi    23300099
Lucy    23300101
```

Ubah data awal menjadi Bagas

```
DATA MAHASISWA
NAMA    NIM
Bagas   2330002
Fattah  2311102019
Farrel  23300003
Wati    2330004
Anis    23300008
Bowo    23300015
Gahar   23300040
Idin    23300045
Ucok    23300050
Budi    23300099
Lucy    23300101
```

Hapus data akhir

Tampilkan seluruh data



Deskripsi program

Program ini menyediakan implementasi dasar Single Linked List Non-Circular dalam C++. Fungsi-fungsi yang disediakan memungkinkan pengguna untuk mengelola data dengan mudah dan efisien. Fungsi utama menyediakan menu interaktif untuk melakukan operasi pada list. Pengguna dapat memilih operasi yang diinginkan, seperti menambahkan, menghapus, mengubah, atau menampilkan data.

BAB IV

KESIMPULAN

Hasil praktikum Linked List Circular dan Non-Circular menunjukkan bahwa terdapat perbedaan penting antara kedua jenis ini. Berikut adalah kesimpulan dari hasil praktikum tersebut:

- Double Linked List Non-Circular (DLLNC) dan Double Linked List Circular (DLLC) merupakan dua jenis Linked List yang berbeda.
- Double Linked List Non-Circular memiliki pointer next dan prev yang menunjuk ke node berikutnya dan sebelumnya, sementara Double Linked List Circular memiliki pointer next yang menunjuk ke node berikutnya dan pointer prev yang menunjuk ke node sebelumnya, serta pointer tail yang selalu terhubung dengan pointer head.
- Double Linked List Non-Circular memiliki satu pointer next dan satu pointer prev, sedangkan Double Linked List Circular memiliki satu pointer next dan satu pointer prev, serta pointer tail yang selalu terhubung dengan pointer head.
- Double Linked List Non-Circular memiliki beberapa fungsi tambahan, seperti menambah data di tengah, menghapus data tertentu, mengurutkan data acak secara ascending dan descending.
- Double Linked List Non-Circular memiliki beberapa operasi, seperti menambah data depan, menghapus data depan, menghapus data belakang, dan menghapus data tengah.

DAFTAR PUSTAKA

Pratama. 2023. Panduan Lengkap Mengenai Linked List: Definisi, Implementasi, dan Penggunaan dalam Pemrograman. Diakses pada tanggal 15 April 2024, dari https://medium.com/@furatamarizuki/panduan-lengkap-mengenai-linked-list-definisi-implementasi-dan-penggunaan-dalam-pemrograman-a6159d548941

Trivusi. 2022. Struktur Data Linked List: Pengertian, Karakteristik, dan Jenis-jenisnya. Diakses pada tanggal 15 April 2024, dari https://www.trivusi.web.id/2022/07/struktur-data-linked-list.html