

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL 4
LINKED LIST CIRCULAR DAN NON
CIRCULAR**



DISUSUN OLEH:

NAMA : Fahrur Rizqi

NIM : 2311102072

S1 IF-11-B

DOSEN:

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

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

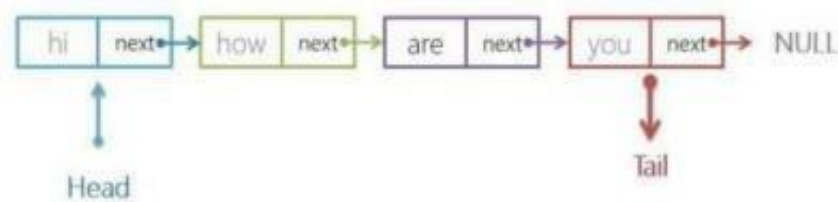
2024

A. DASAR TEORI

Linked list merupakan sejumlah objek yang di link atau dihubungkan satu dengan yang lainnya sehingga membentuk suatu list.

1) Linked List Non Circular

Linked list non circular merupakan *linked list* dengan node pertama (head) dan node terakhir (tail) yang tidak saling terhubung. Pointer terakhir (tail) pada *Linked List* ini selalu bernilai '*NULL*' sebagai pertanda data terakhir dalam *list*-nya. *Linked list non circular* dapat digambarkan sebagai berikut.



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

```

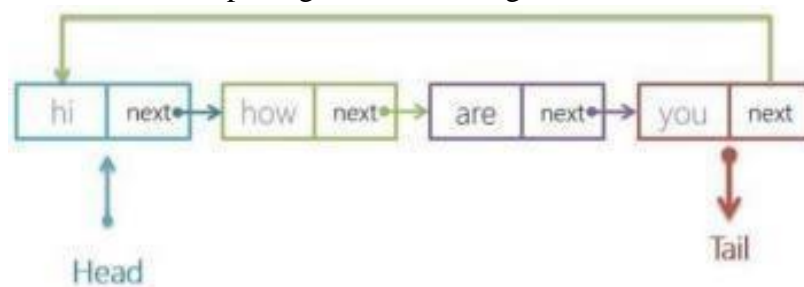
void tampil()
{
    if (isEmpty() == true)
    {
        cout << "List kosong!" << endl;
    }
    else
    {
        node *helper;
        helper = head;
        while (helper != NULL)
        {
            cout << helper->data << ends;
            helper = helper->next;
        }
    }
}

```

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;
        }
    }
}

```

7. Menampilkan Data Linked List

```

void tampil()
{
    if (isEmpty() == 0)
    {
        tail = head;
        do
        {
            cout << tail->data << ends;
            tail = tail->next;
        } while (tail != head);
        cout << endl;
    }
}

```

B. Guided

GUIDED 1 :

Linked List Non Circular

SOURCE CODE

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

struct Node {
    int data;
    Node *next;
};

Node *head;
Node *tail;

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

bool isEmpty() {
    return head == NULL;
}

void insertDepan(int nilai) {
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty()) {
        head = tail = baru;
    } else {
        baru->next = head;
        head = baru;
    }
}

void insertBelakang(int nilai) {
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty()) {
        head = tail = baru;
    } else {
        tail->next = baru;
        tail = baru;
    }
}
```

```

}

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

void insertTengah(int data, int posisi) {
    if (posisi < 1 || posisi > hitungList()) {
        cout << "Posisi diluar jangkauan" << endl;
    } else if (posisi == 1) {
        cout << "Posisi bukan posisi tengah" << endl;
    } else {
        Node *baru = new Node();
        baru->data = data;
        Node *bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1) {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

void hapusDepan() {
    if (!isEmpty()) {
        Node *hapus = head;
        if (head->next != NULL) {
            head = head->next;
        } else {
            head = tail = NULL;
        }
        delete hapus;
    } else {
        cout << "List kosong!" << endl;
    }
}

void hapusBelakang() {
    if (!isEmpty()) {

```



```

        Node *hapus = tail;
        if (head != tail) {
            Node *bantu = head;
            while (bantu->next != tail) {
                bantu = bantu->next;
            }
            tail = bantu;
            tail->next = NULL;
        } else {
            head = tail = NULL;
        }
        delete hapus;
    } else {
        cout << "List kosong!" << endl;
    }
}

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

void ubahDepan(int data) {
    if (!isEmpty()) {
        head->data = data;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

```

```

    }
}

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

void ubahBelakang(int data) {
    if (!isEmpty()) {
        tail->data = data;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

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

void tampil() {
    Node *bantu = head;
    if (!isEmpty()) {
        while (bantu != NULL) {

```

```

        cout << bantu->data << " ";
        bantu = bantu->next;
    }
    cout << endl;
} else {
    cout << "List masih kosong!" << endl;
}
}

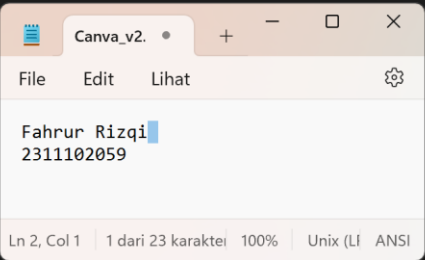
int main() {
    init();
    insertDepan(3);
    tampil();
    insertBelakang(5);
    tampil();
    insertDepan(2);
    tampil();
    insertDepan(1);
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, 2);
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1);
    tampil();
    ubahBelakang(8);
    tampil();
    ubahTengah(11, 2);
    tampil();

    return 0;
}

```

SCREENSHOT OUTPUT

```
PS C:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 4> c
d "c:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 4\"
; if ($?) { g++ Guided1.cpp -o Guided1 } ; if ($?) { .\Guided1 }
3
35
235
1235
235
23
273
23
13
18
Posisi bukan posisi tengah
111
PS C:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 4>
```



DESKRIPSI PROGRAM

Dalam program, program memakai struct untuk inisialisasi coding node dan juga linked list terdiri dari node yang berasal dari beberapa fungsi void untuk masing masing algoritma. Pada program int main yaitu pertama terdapat inisialisasi dengan void init. Program juga memanggil dari fungsi void yang telah dibuat seperti insertDepan, insertBelakang, hapusDepan, hapusBelakang, insertTengah, hapusTengah, ubahDepan, ubahBelakang, ubah Tengah. Pada fungsi tampil hanya digunakan untuk algoritma tampilan output. Nilai di hardcode kan dalam source code.

GUIDED 2 :

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() {
    return head == NULL;
}

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

int hitungList() {
    bantu = head;
    int jumlah = 0;
    while (bantu != NULL) {
        jumlah++;
        bantu = bantu->next;
    }
    return jumlah;
}

void insertDepan(string data) {
    buatNode(data);
    if (isEmpty()) {
        head = baru;
        tail = head;
        baru->next = head;
    } else {
```

```

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

void insertBelakang(string data) {
    buatNode(data);
    if (isEmpty()) {
        head = baru;
        tail = head;
        baru->next = head;
    } else {
        while (tail->next != head) {
            tail = tail->next;
        }
        tail->next = baru;
        baru->next = head;
    }
}

void insertTengah(string data, int posisi) {
    if (isEmpty()) {
        head = baru;
        tail = head;
        baru->next = head;
    } else {
        baru->data = data;
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1) {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

void hapusDepan() {
    if (!isEmpty()) {
        hapus = head;
        tail = head;
        if (hapus->next == head) {

```

```

        head = NULL;
        tail = NULL;
        delete hapus;
    } else {
        while (tail->next != hapus) {
            tail = tail->next;
        }
        head = head->next;
        tail->next = head;
        hapus->next = NULL;
        delete hapus;
    }
} else {
    cout << "List masih kosong!" << endl;
}
}

void hapusBelakang() {
    if (!isEmpty()) {
        hapus = head;
        tail = head;
        if (hapus->next == head) {
            head = NULL;
            tail = NULL;
            delete hapus;
        } else {
            while (hapus->next != head) {
                hapus = hapus->next;
            }
            while (tail->next != hapus) {
                tail = tail->next;
            }
            tail->next = head;
            hapus->next = NULL;
            delete hapus;
        }
    } else {
        cout << "List masih kosong!" << endl;
    }
}

void hapusTengah(int posisi) {
    if (!isEmpty()) {
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1) {
            bantu = bantu->next;

```

```

        nomor++;
    }
    hapus = bantu->next;
    bantu->next = hapus->next;
    delete hapus;
} else {
    cout << "List masih kosong!" << endl;
}
}

void clearList() {
    if (head != NULL) {
        hapus = head->next;
        while (hapus != head) {
            bantu = hapus->next;
            delete hapus;
            hapus = bantu;
        }
        delete head;
        head = NULL;
    }
    cout << "List berhasil terhapus!" << endl;
}

void tampil() {
    if (!isEmpty()) {
        tail = head;
        do {
            cout << tail->data << " ";
            tail = tail->next;
        } while (tail != head);
        cout << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

int main() {
    init();
    insertDepan("Ayam");
    tampil();
    insertDepan("Bebek");
    tampil();
    insertBelakang("Cicak");
    tampil();
    insertBelakang("Domba");
    tampil();
}

```



```

hapusBelakang();
tampil();
hapusDepan();
tampil();
insertTengah("Sapi", 2);
tampil();
hapusTengah(2);
tampil();
return 0;
}

```

SCREENSHOT OUTPUT

```

PS C:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 4> cd "c:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 4\" ; if ($?) { g++ Guided2.cpp -o Guided2 } ; if ($?) { .\Guided2 }
Ayam
Bebek Ayam
Bebek Ayam Cicak
Bebek Ayam Cicak Domba
Bebek Ayam Cicak
Ayam Cicak
Ayam Sapi Cicak
Ayam Cicak
PS C:\Users\ASUS\Documents\File Fahrur\SEMESTER 2\Laprak Struktur Data dan Algoritma\MODUL 4>

```

DESKRIPSI PROGRAM

Dalam program ini, digunakan sebuah struct untuk merepresentasikan node dalam linked list, yang terdiri dari data dan pointer ke node berikutnya. Struktur data digunakan untuk menyimpan sejumlah data secara terurut dan dinamis.

Terdapat beberapa fungsi yang digunakan dalam program ini, antara lain untuk menginisialisasi linked list, memeriksa apakah linked list kosong, membuat simpul baru, menghitung jumlah simpul dalam linked list, serta melakukan operasi-insert dan operasi-hapus pada berbagai posisi dalam linked list. Fungsi tampil() digunakan untuk menampilkan isi linked list.

Di dalam fungsi main(), program melakukan inisialisasi linked list, kemudian melakukan serangkaian operasi pada linked list seperti insert dan delete pada berbagai posisi, serta menampilkan isi linked list setiap kali operasi dilakukan. Nilai-nilai dihardcode ke dalam program sebagai contoh nilai yang dimasukkan ke dalam linked list.

C. UNGUIDED

UNGUIDED 1 : Buatlah Program menu Linked List Non Circular untuk menyimpan **Nama** dan **NIM mahasiswa**, dengan menggunakan *input* dari user.

1. Buatlah menu untuk menambahkan, mengubah, menghapus, dan melihat Nama dan NIM mahasiswa, berikut **contoh** tampilan output dari nomor 1:

• Tampilan Menu:

```
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. TAMPILKAN
0. KELUAR

```
Pilih Operasi :
```

•Tampilan Operasi Tambah:

```
-Tambah Depan
```

```
Masukkan Nama :
```

```
Masukkan NIM :
```

```
Data telah ditambahkan
```

```
-Tambah Tengah
```

```
Masukkan Nama :
```

```
Masukkan NIM :
```

```
Masukkan Posisi :
```

```
Data telah ditambahkan
```

• Tampilan Operasi Hapus:

-Hapus Belakang
Data (nama mahasiswa yang dihapus) berhasil dihapus
-Hapus Tengah
Masukkan posisi :

• Tampilan Operasi Ubah:

-Ubah Belakang
Masukkan nama :
Masukkan NIM :
Data (nama lama) telah diganti dengan data (nama baru)
-Ubah Belakang
Masukkan nama :
Masukkan NIM :
Masukkan posisi :

• Tampilan Operasi Tampil Data:

DATA MAHASISWA
NAMA NIM
Nama1 NIM1
Nama2 NIM2

***Buat tampilan output sebgus dan secantik mungkin sesuai kreatifitas anda masing-masing, jangan terpaku pada contoh output yang diberikan**

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

SOURCE CODE

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

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

class LinkedList {
private:
    identitasMhs* head;
public:
    LinkedList() {
        head = NULL;
    }

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

    void addLast(string nama, string nim) {
        identitasMhs* newNode = new identitasMhs();
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = NULL;
        if (head == NULL) {
            head = newNode;
        }
    }
};
```

```

    } else {
        identitasMhs* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
    cout << endl << "Data telah ditambahkan"<<endl;
}

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

void changeFirst(string nama, string nim) {
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    head->nama = nama;
    head->nim = nim;
    cout << endl << "Data di depan telah diubah"<<endl;
}

void changeLast(string nama, string nim) {
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    identitasMhs* temp = head;

```

```

        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->nama = nama;
        temp->nim = nim;
        cout << endl << "Data di belakang telah diubah"<<endl;
    }

    void changeMiddle(string nama, string nim, int posisi) {
        if (posisi < 1) {
            cout << endl << "Posisi tidak valid"<<endl;
            return;
        }
        if (head == NULL) {
            cout << endl << "Linked List kosong"<<endl;
            return;
        }
        identitasMhs* temp = head;
        for (int i = 1; i < posisi; i++) {
            if (temp == NULL) {
                cout << endl << "Posisi tidak valid"<<endl;
                return;
            }
            temp = temp->next;
        }
        temp->nama = nama;
        temp->nim = nim;
        cout << endl << "Data di posisi " << posisi << " telah
diubah"<<endl;
    }

    void deleteFirst() {
        if (head == NULL) {
            cout << endl << "Linked List kosong"<<endl;
            return;
        }
        identitasMhs* temp = head;
        head = head->next;
        delete temp;
        cout << endl << "Data di depan telah dihapus"<<endl;
    }

    void deleteLast() {
        if (head == NULL) {
            cout << endl << "Linked List kosong"<<endl;
            return;
        }
        if (head->next == NULL) {
            delete head;
            head = NULL;
            cout << endl << "Data di belakang telah dihapus"<<endl;
            return;
        }
    }

```

```

    }
    identitasMhs* temp = head;
    while (temp->next->next != NULL) {
        temp = temp->next;
    }
    delete temp->next;
    temp->next = NULL;
    cout << endl << "Data di belakang telah dihapus"<<endl;
}

void deleteMiddle(int posisi) {
    if (posisi < 1) {
        cout << endl << "Posisi tidak valid"<<endl;
        return;
    }
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    identitasMhs* temp = head;
    if (posisi == 1) {
        head = head->next;
        delete temp;
        cout << endl << "Data di posisi " << posisi << " telah
dihapus"<<endl;
        return;
    }
    for (int i = 1; i < posisi - 1; i++) {
        if (temp == NULL || temp->next == NULL) {
            cout << endl << "Posisi tidak valid"<<endl;
            return;
        }
        temp = temp->next;
    }
    identitasMhs* nextNode = temp->next->next;
    delete temp->next;
    temp->next = nextNode;
    cout << endl << "Data di posisi " << posisi << " telah
dihapus"<<endl;
}

void print() {
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    cout << "=====DATA MAHASISWA===== "<< endl;
    cout << "-----" << endl;
    cout << setw(20) << left << "NAMA" << setw(20) << left << "NIM"
<< endl;
    identitasMhs* temp = head;
    while (temp != NULL) {

```



```

        cout << setw(20) << left << temp->nama << setw(20) << left
<< temp->nim << endl;
        temp = temp->next;
    }
}

void deleteAll() {
    while (head != NULL) {
        identitasMhs* temp = head;
        head = head->next;
        delete temp;
    }
    cout << endl << "Semua data mahasiswa telah dihapus"<<endl;
}

};

int main() {
    LinkedList linkedList;
    int pilihan;
    string nama, nim;
    int posisi;
    while (true) {
        cout << endl;
        linkedList.print();
        cout << endl;
        cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR"<<endl;
        cout << "===== " << endl;
        cout << 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;

        cout << endl;

        cout << "Pilih Operasi: ";
        cin >> pilihan;

        cout << endl;

        switch (pilihan) {
            case 1:
                cout << "-Tambah Depan"<<endl;
                cout << endl;

```

```

        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        linkedList.addFirst(nama, nim);
        break;
    case 2:
        cout << "-Tambah Belakang"<<endl;
        cout << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        linkedList.addLast(nama, nim);
        break;
    case 3:
        cout << "-Tambah Tengah"<<endl;
        cout << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.addMiddle(nama, nim, posisi);
        break;
    case 4:
        cout << "-Ubah Depan"<<endl;
        cout << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        linkedList.changeFirst(nama, nim);
        break;
    case 5:
        cout << "-Ubah Belakang"<<endl;
        cout << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        linkedList.changeLast(nama, nim);
        break;
    case 6:
        cout << "-Ubah Tengah"<<endl;
        cout << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";

```

```

        cin >> posisi;
        linkedList.changeMiddle(nama, nim, posisi);
        break;
    case 7:
        cout << "-Hapus Depan"<<endl;
        cout << endl;
        linkedList.deleteFirst();
        break;
    case 8:
        cout << "-Hapus Belakang"<<endl;
        cout << endl;
        linkedList.deleteLast();
        break;
    case 9:
        cout << "-Hapus Tengah"<<endl;
        cout << endl;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.deleteMiddle(posisi);
        break;
    case 10:
        cout << "-Hapus List"<<endl;
        cout << endl;
        linkedList.deleteAll();
        break;
    case 11:
        cout << "-Tampilkan"<<endl;
        cout << endl;
        linkedList.print();
        break;
    case 0:
        exit(0);
    default:
        cout << "Pilihan tidak valid"<<endl;
    }
}

return 0;
}

```

SCREENSHOT OUTPUT

PROGRAM SINGLE LINKED LIST NON-CIRCULAR

=====

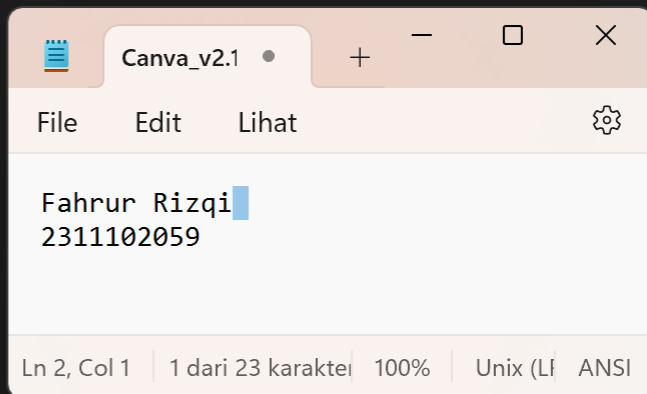
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar

Pilih Operasi: 11

-Tampilkan

=====DATA MAHASISWA=====

NAMA	NIM
Jawad	2300001
Fahrur	2311102059
Farrel	23300003
Denis	23300005
Anis	23300008
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099



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: 3

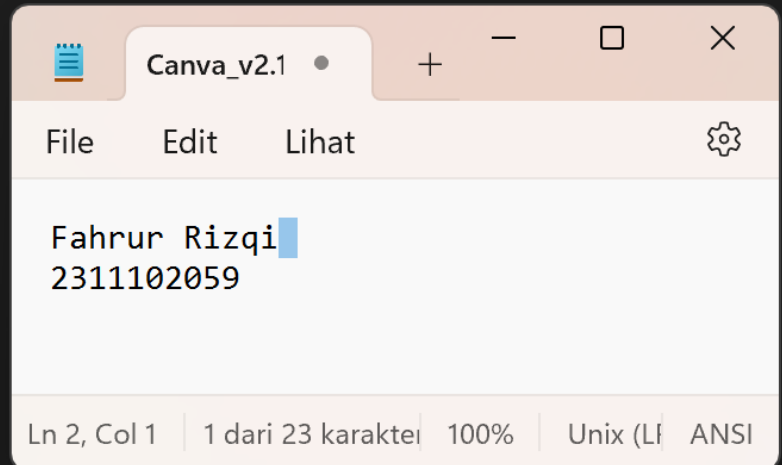
-Tambah Tengah

Masukkan Nama : Wati

Masukkan NIM : 2330004

Masukkan Posisi : 4

Data telah ditambahkan



PROGRAM SINGLE LINKED LIST NON-CIRCULAR

=====

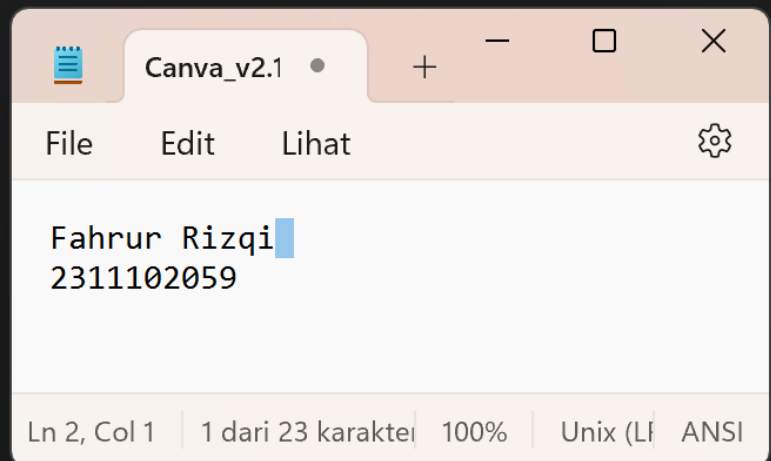
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar

Pilih Operasi: 9

-Hapus Tengah

Masukkan Posisi : 5

Data di posisi 5 telah dihapus



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

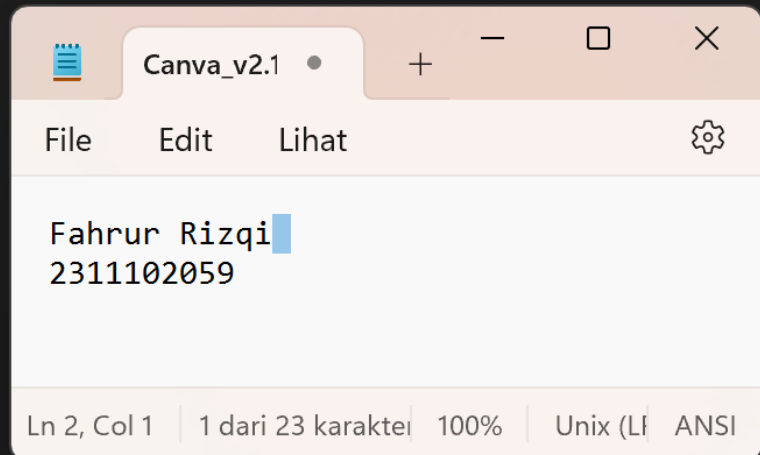
Pilih Operasi: 1

-Tambah Depan

Masukkan Nama : Owi

Masukkan NIM : 2330000

Data telah ditambahkan



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

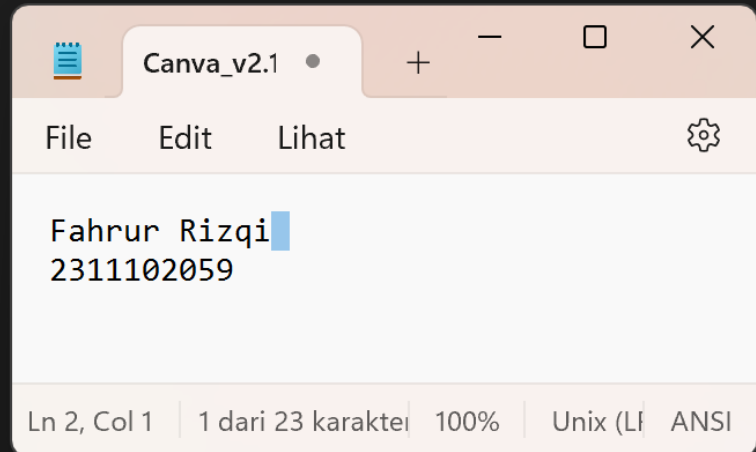
Pilih Operasi: 2

-Tambah Belakang

Masukkan Nama : David

Masukkan NIM : 23300100

Data telah ditambahkan



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: 6

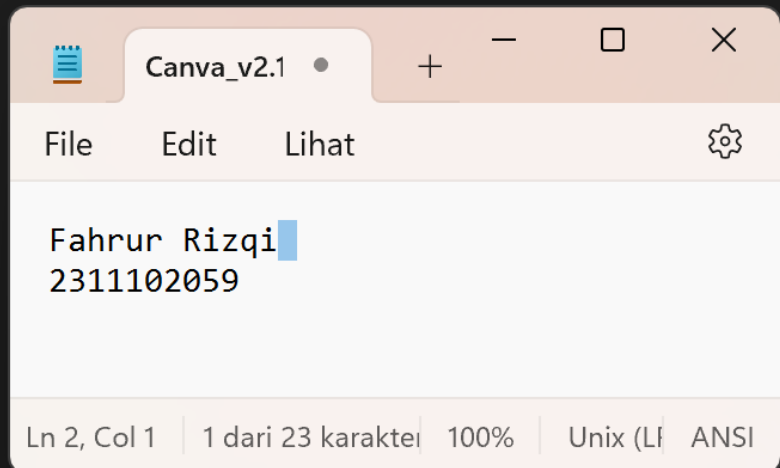
-Ubah Tengah

Masukkan Nama Baru : Idin

Masukkan NIM Baru : 23300045

Masukkan Posisi : 9

Data di posisi 9 telah diubah



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

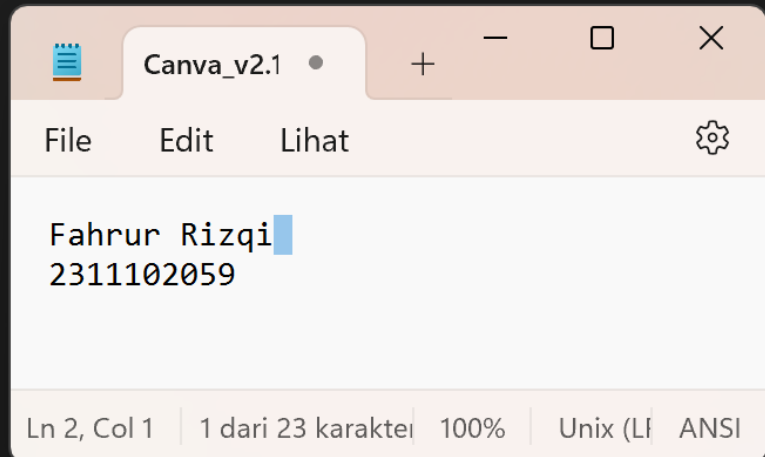
Pilih Operasi: 5

-Ubah Belakang

Masukkan Nama Baru : Lucy

Masukkan NIM Baru : 23300101

Data di belakang telah diubah



PROGRAM SINGLE LINKED LIST NON-CIRCULAR

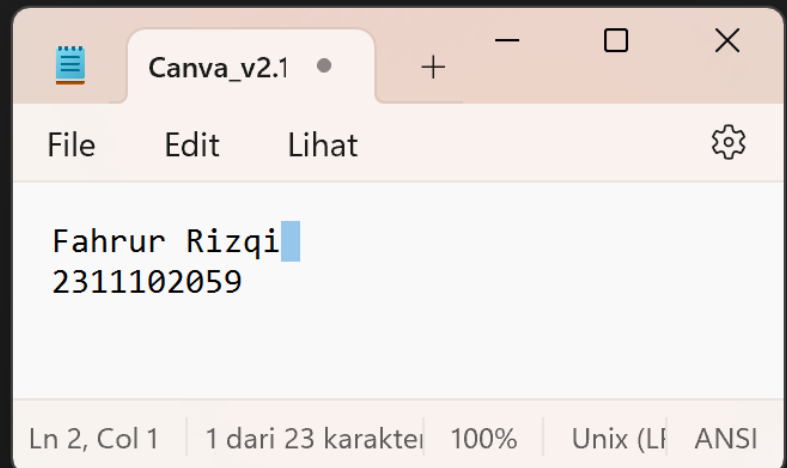
=====

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: 7

-Hapus Depan

Data di depan telah dihapus



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

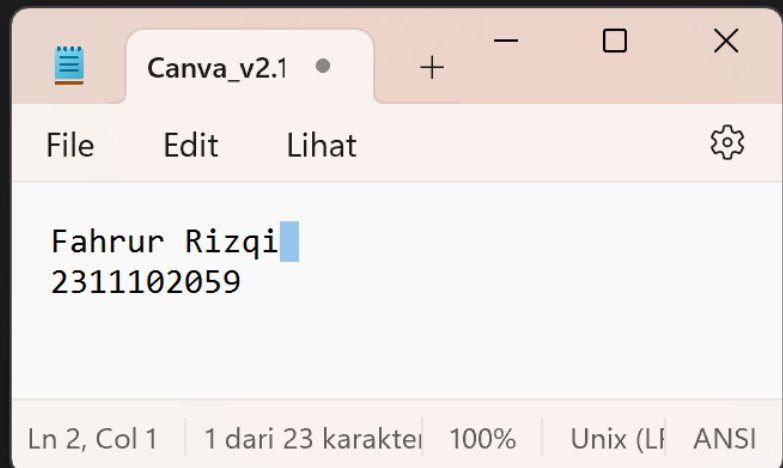
Pilih Operasi: 4

-Ubah Depan

Masukkan Nama Baru : Bagas

Masukkan NIM Baru : 2330002

Data di depan telah diubah



PROGRAM SINGLE LINKED LIST NON-CIRCULAR

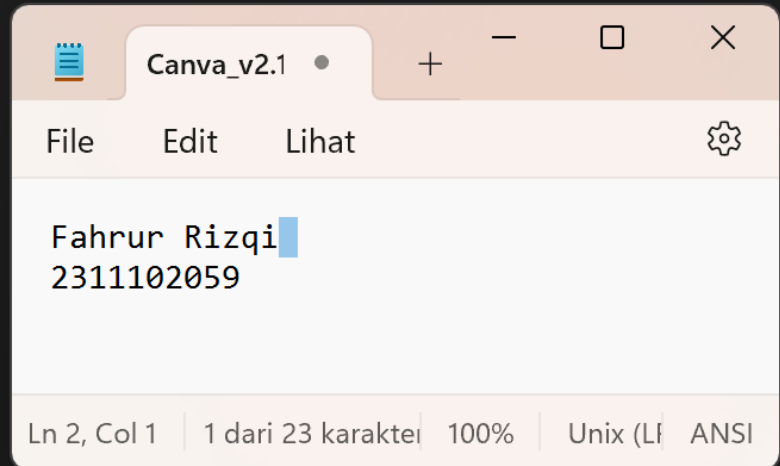
=====

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: 8

-Hapus Belakang

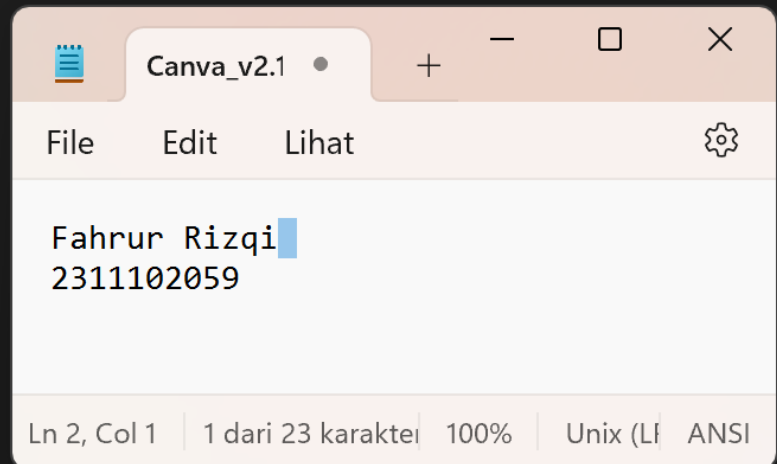
Data di belakang telah dihapus



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

-Tampilkan

=====DATA MAHASISWA=====

NAMA	NIM
Bagas	2330002
Fahrur	2311102059
Farrel	23300003
Wati	2330004
Anis	23300008
Bowo	23300015
Gahar	23300040
Idin	23300045
Ucok	23300050
Budi	23300099

DESKRIPSI PROGRAM

Program ini menyediakan fungsi-fungsi utama seperti inisialisasi, penambahan dan penghapusan node pada berbagai posisi, serta pengubahan data di dalam node. Program memuat beberapa opsi pilihan menu. Dalam program memuat beberapa fungsi yang memungkinkan pengguna untuk melakukan berbagai operasi pada linked list, seperti menambah, mengubah, atau menghapus node. Program juga memberikan pengkondisian untuk memastikan operasi-operasi dilakukan sesuai dengan kondisi linked list, seperti memeriksa apakah linked list kosong sebelum melakukan operasi tertentu. Melalui fungsi tampil(), program mengoutputkan pengguna untuk melihat isi linked list setiap kali operasi dilakukan. Nilai-nilai yang dimasukkan ke dalam linked list dapat dimodifikasi langsung dalam kode program sebagai contoh nilai yang dimasukkan ke dalam linked list.

D. Kesimpulan

Linked list adalah sejumlah objek yang dihubungkan satu dengan yang lainnya sehingga membentuk suatu list. Variabel pointer tersebut merupakan salah satu variabel dalam struktur objek. Terdapat beberapa macam Linked List seperti : Single Linked List dan Circular Linked List. Linked List biasanya head pada posisi node pertama dan NULL berada di node terakhir. Sedangkan Circular Linked List node terakhir menunjuk node pertama, sehingga tidak terdapat NULL.

E. Referensi

Asisten Praktikum, "Modul Linked List Circular dan Non Circular", LearningManagementSystem, 2024.

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

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

Budi Raharjo. 2015. Pemrograman C++. Bandung: Informatika.

Rinaldi Munir, Leony Lidya. 2016. Algoritma dan Pemrograman. Bandung: Informatika Bandung.