

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

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



DISUSUN OLEH:

NAMA : SYARIEF RENDI ADITYA ANTONIUS

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**

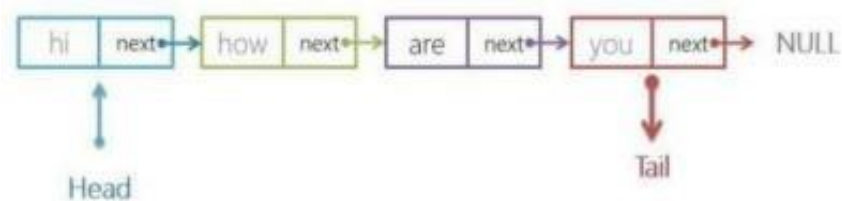
2024

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

```

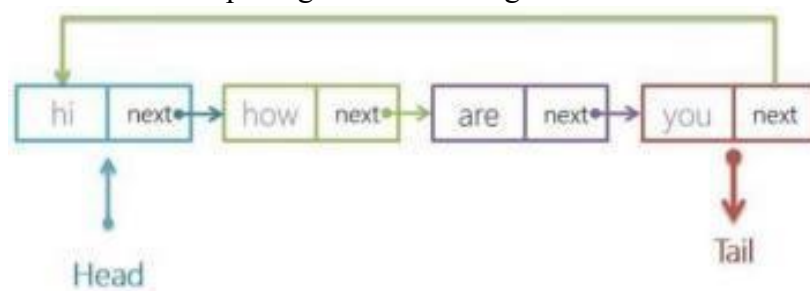
void tampil()
{
    if (isEmpty() == true)
    {
        cout << "List kosong!" << endl;
    }
    else
    {
        node *helper;
        helper = head;
        while (helper != NULL)
        {
            cout << helper->data << ends;
            helper = helper->next;
        }
    }
}

```

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) {

```

```

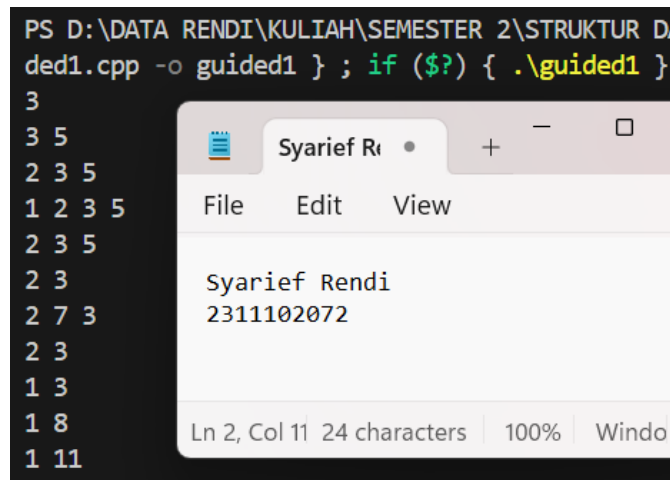
        cout << bantu->data << " ";
        bantu = bantu->next;
    }
    cout << endl;
} else {
    cout << "List masih kosong!" << endl;
}
}

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;
} //2311102072syarief

```

SCREENSHOT OUTPUT



The screenshot shows a terminal window on the left and a Notepad window on the right. The terminal window displays the output of a program, which consists of several lines of numbers: 3, 3 5, 2 3 5, 1 2 3 5, 2 3 5, 2 3, 2 7 3, 2 3, 1 3, 1 8, and 1 11. The Notepad window, titled 'Syarief Rendi', contains the text 'Syarief Rendi' and '2311102072'. The terminal window's command prompt shows the path 'PS D:\DATA RENDI\KULIAH\SEMESTER 2\STRUKTUR DA' and the command 'ded1.cpp -o guided1 } ; if (\$?) { .\guided1 }'.

```
PS D:\DATA RENDI\KULIAH\SEMESTER 2\STRUKTUR DA
ded1.cpp -o guided1 } ; if ($?) { .\guided1 }
3
3 5
2 3 5
1 2 3 5
2 3 5
2 3
2 7 3
2 3
1 3
1 8
1 11
```

Syarief Rendi
2311102072

Ln 2, Col 11 24 characters | 100% | Windo

DESKRIPSI PROGRAM

Dalam program, program memakai struct untuk inialisasi 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 inialisasi 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();
}

```

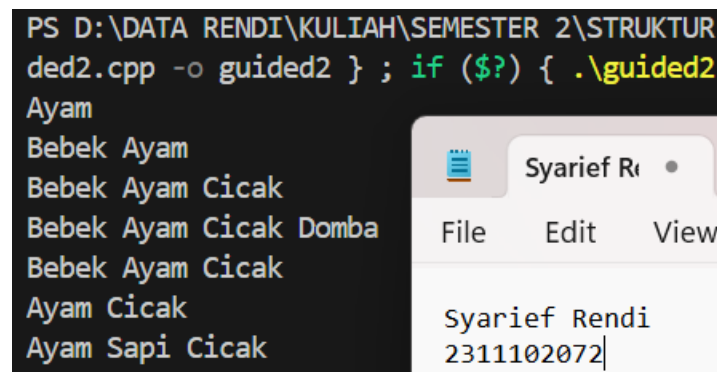


```

hapusBelakang();
tampil();
hapusDepan();
tampil();
insertTengah("Sapi", 2);
tampil();
hapusTengah(2);
tampil();
return 0;
} //2311102072syarief

```

SCREENSHOT OUTPUT



```

PS D:\DATA RENDI\KULIAH\SEMESTER 2\STRUKTUR
ded2.cpp -o guided2 } ; if ($?) { .\guided2
Ayam
Bebek Ayam
Bebek Ayam Cicak
Bebek Ayam Cicak Domba
Bebek Ayam Cicak
Ayam Cicak
Ayam Sapi Cicak

```

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 terkahir 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>
using namespace std;

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

Node* head;
Node* tail;

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

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

void insertDepan(string nama, int 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, int 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;
    }
}

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

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

void hapusTengah(int posisi) {

```

```

    if (!isEmpty()) {
        if (posisi < 1 || posisi > hitungList()) {
            cout << "Posisi diluar jangkauan" << endl;
        } else {
            Node* hapus;
            Node* bantu = head;
            Node* prev = nullptr;
            int nomor = 1;
            while (nomor < posisi) {
                prev = bantu;
                bantu = bantu->next;
                nomor++;
            }
            hapus = bantu;
            if (prev != nullptr) {
                prev->next = bantu->next;
            } else {
                head = bantu->next;
            }
            if (bantu == tail) {
                tail = prev;
            }
            delete hapus;
            cout << "Node pada posisi " << posisi << " berhasil
dihapus" << endl;
        }
    } else {
        cout << "List masih kosong!" << endl;
    }
}

void ubahTengah(string nama, int nim, string nama_lama) {
    if (!isEmpty()) {
        Node* bantu = head;
        while (bantu != nullptr && bantu->nama != nama_lama) {
            bantu = bantu->next;
        }
        if (bantu != nullptr) {
            bantu->nama = nama;
            bantu->nim = nim;
            cout << "Node dengan nama '" << nama_lama << "'
berhasil diubah menjadi '" << nama << "' dengan NIM baru " <<
nim << endl;
        } else {
            cout << "Node dengan nama '" << nama_lama << "'
tidak ditemukan" << endl;
        }
    }
}

```

```

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

void hapusDepan() {
    if (!isEmpty()) {
        Node* hapus = head;
        head = head->next;
        delete hapus;
        cout << "Node pertama berhasil dihapus" << endl;
    } else {
        cout << "List masih 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;
        }
        cout << "Node terakhir berhasil dihapus" << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

void hapusList() {
    while (!isEmpty()) {
        hapusDepan();
    }
}

void tampil() {
    cout << "DATA MAHASISWA\n\n";
    cout << "-----\n";
    cout << setw(15) << left << "NAMA" << setw(10) << "NIM" <<
endl;

```



```

        cout << "-----\n";
        if (!isEmpty()) {
            Node* bantu = head;
            while (bantu != NULL) {
                cout << setw(15) << left << bantu->nama << setw(10)
<< bantu->nim << endl;
                bantu = bantu->next;
            }
            cout << endl;
        } else {
            cout << "List masih kosong!" << endl;
        }
        cout << "-----\n";
    }
}

```

```

int main() {
    init();
    string nama;
    int nim;
    int choice;
    cout << "Masukkan nama anda: ";
    cin >> nama;
    cout << "Masukkan NIM anda: ";
    cin >> nim;
    insertBelakang(nama, nim);

    while (true) {
        cout << "\nPROGRAM SINGLE LINKED LIST NON-CIRCULAR\n";
        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 << "Pilih Operasi: ";
        cin >> choice;

        switch (choice) {
            case 1: {

```

```

        cout << "-Tambah Depan\n" << endl;
        cout << "Masukkan Nama: ";
        cin >> nama;
        cout << "Masukkan NIM: ";
        cin >> nim;
        insertDepan(nama, nim);
        cout << "Data telah ditambahkan";
        break;
    }
    case 2: {
        cout << "-Tambah Belakang\n" << endl;
        cout << "Masukkan Nama: ";
        cin >> nama;
        cout << "Masukkan NIM: ";
        cin >> nim;
        insertBelakang(nama, nim);
        cout << "Data telah ditambahkan";
        break;
    }
    case 3: {
        cout << "-Tambah Tengah\n" << endl;
        int posisi;
        cout << "Masukkan Nama: ";
        cin >> nama;
        cout << "Masukkan NIM: ";
        cin >> nim;
        cout << "Masukkan posisi: ";
        cin >> posisi;
        insertTengah(nama, nim, posisi);
        cout << "Data telah ditambahkan";
        break;
    }
    case 4: {
        cout << "-Ubah Depan\n" << endl;
        string nama_baru;
        int nim_baru;
        cout << "Masukkan nama baru: ";
        cin >> nama_baru;
        cout << "Masukkan NIM baru: ";
        cin >> nim_baru;
        string nama_lama = head->nama;
        ubahTengah(nama_baru, nim_baru, head->nama);
        cout << "Data (" << nama_lama << ") telah
terganti dengan data '" << nama_baru << "'" << endl;
        break;
    }
    case 5: {

```

```

        cout << "-Ubah Belakang\n" << endl;
        string nama_baru;
        int nim_baru;
        cout << "Masukkan nama baru: ";
        cin >> nama_baru;
        cout << "Masukkan NIM baru: ";
        cin >> nim_baru;
        string nama_lama = tail->nama;
        ubahTengah(nama_baru, nim_baru, tail->nama);
        cout << "Data (" << nama_lama << ") telah
terganti dengan data '" << nama_baru << "'" << endl;
        break;
    }
    case 6: {
        cout << "-Ubah Tengah\n" << endl;
        if (!isEmpty()) {
            int posisi;
            cout << "Masukkan posisi yang ingin diubah:
";

            cin >> posisi;
            string nama_baru;
            int nim_baru;
            cout << "Masukkan nama baru: ";
            cin >> nama_baru;
            cout << "Masukkan NIM baru: ";
            cin >> nim_baru;
            Node* bantu = head;
            int nomor = 1;
            while (nomor < posisi) {
                bantu = bantu->next;
                nomor++;
            }
            string nama_lama = bantu->nama;
            ubahTengah(nama_baru, nim_baru, bantu-
>nama);

            cout << "Data pada posisi " << posisi << "
berhasil diubah." << endl;
            cout << "Data (" << nama_lama << ") telah
terganti dengan data (" << nama_baru << ")" << endl;
        } else {
            cout << "List masih kosong!" << endl;
        }
        break;
    }
    case 7: {
        cout << "-Hapus Depan\n" << endl;
        if (!isEmpty()) {

```

```

        string nama_hapus = head->nama;
        hapusDepan();
        cout << "Data (" << nama_hapus << ")
berhasil dihapus." << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
    break;
}
case 8: {
    cout << "-Hapus Belakang\n" << endl;
    if (!isEmpty()) {
        string nama_hapus = tail->nama;
        hapusBelakang();
        cout << "Data (" << nama_hapus << ")
berhasil dihapus." << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
    break;
}
case 9: {
    cout << "-Hapus Tengah\n" << endl;
    if (!isEmpty()) {
        int posisi;
        cout << "Masukkan posisi: ";
        cin >> posisi;
        hapusTengah(posisi);
    } else {
        cout << "List masih kosong!" << endl;
    }
    break;
}
case 10: {
    cout << "-Hapus List\n" << endl;
    hapusList();
    break;
}
case 11: {
    cout << "-Tampilkan\n" << endl;
    tampil();
    break;
}
case 0: {
    return 0;
}
default: {

```

```

        cout << "Pilihan tidak valid" << endl;
        break;
    }
}

return 0;
}
//2311102072syarief

```

SCREENSHOT OUTPUT

PS D:\DATA RENDI\KULIAH\SEMESTER 2\STRUKTUR DATA> g++ 10. Hapus List
; if (\$?) { .\unguided } 11. TAMPILKAN
Masukkan nama anda: Syarief 0. KELUAR
Masukkan NIM anda: 23302072 Pilih Operasi: 2
-Tambah Belakang

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: Denis
Masukkan NIM: 23300005
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: Denis
Masukkan NIM: 23300005
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: Gahar
Masukkan NIM: 23300040
Data telah ditambahkan
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
1. Tambah Depan
2. Tambah Belakang

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: Udin
Masukkan NIM: 23300048
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: Budi
Masukkan NIM: 23300099
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: 3

Masukkan Nama: Farrel
Masukkan NIM: 23300003
Masukkan posisi: 3
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: 11
-Tampilkan

DATA MAHASISWA

NAMA	NIM
Jawad	23300001
Syarief	23302072
Farrel	23300003
Denis	23300005
Anis	23300008
Bowo	23300015
Gahar	23300040


```
-Ubah Belakang
Masukkan nama baru: Lucy
Masukkan NIM baru: 23300101
Node dengan nama 'David' berhasil diubah menjadi 'Lucy' dengan NIM baru 23300101
Data (David) telah terganti dengan data 'Lucy'

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
-----
Owi        2330000
Jawad      23300001
Syarief    23302072
Farrel     23300003
Wati       23300004
Anis       23300008
Bowo       23300015

Bowo        23300015
Gahar       23300040
Idin        23300045
Ucok        23300050
Budi        23300099
Lucy        23300101

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

Node pertama berhasil dihapus
Data (Owi) berhasil 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
```

```
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
Node dengan nama 'Jawad' berhasil diubah menjadi 'Bagas' dengan NIM baru 2330002
Data (Jawad) telah terganti dengan data 'Bagas'

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
Syarief    23302072
Farrel     23300003
Wati       23300004
Anis       23300008
Bowo       23300015
Gahar      23300040
Idin       23300045
Ucok       23300050
Budi       23300099
Lucy       23300101

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

Node terakhir berhasil dihapus
Data (Lucy) berhasil 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
Syarief    23302072
Farrel     23300003
Wati       23300004
Anis       23300008
Bowo       23300015
Gahar      23300040
Idin       23300045
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: 0
PS D:\DATA RENDI\KULIAH\SEMESTER 2\STRUKTUR
```

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.