

LAPORAN PRAKTIKUM
MODUL 4
LINKED LIST CIRCULAR DAN NON CIRCULAR



Disusun oleh:
Arvan Murbiyanto

NIM :
2311102074

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

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM 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 Circular

Circular Linked List merupakan suatu linked list dimana tail (node terakhir) menunjuk ke head (node pertama). Jadi tidak ada pointer yang menunjuk NULL. Ada 2 jenis Circular Linked List, yaitu: Circular Single Linked dan List Circular Double Linked List.[1]

Linked List Non Circular

Linked list non circular adalah struktur data di mana simpul terakhir tidak menunjuk kembali ke simpul pertama, sehingga tidak membentuk lingkaran atau sirkular. Ini berarti bahwa simpul terakhir memiliki nilai pointer NULL, menandakan akhir dari linked list.[2]

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; // Deklarasi head
Node *tail; // Deklarasi tail

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

// Pengecekan apakah linked list kosong
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;
        head->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;
            head->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 di luar 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 << "Linked list masih 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 << "Linked list masih kosong" << endl;
    }
}
// Hapus tengah
void hapusTengah(int posisi)
{
    Node *hapus, *bantu, *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 << "Linked list masih kosong" << endl;
    }
}

// ubah tengah
void ubahTengah(int data, int posisi)
{
    Node *bantu;
    if (isEmpty() == 0)
    {
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "Posisi di luar jangkauan" << endl;
        }
        else if (posisi == 1)
        {
            cout << "Posisi bukan posisi tengah" << endl;
        }
    }
}

```

```

        else
        {
            int nomor = 1;
            bantu = head;
            while (nomor < posisi)
            {
                bantu = bantu->next;
                nomor++;
            }
            bantu->data = data;
        }
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

// ubah belakang
void ubahBelakang(int data)
{
    if (isEmpty() == 0)
    {
        tail->data = data;
    }
    else
    {
        cout << "Linked 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;
}

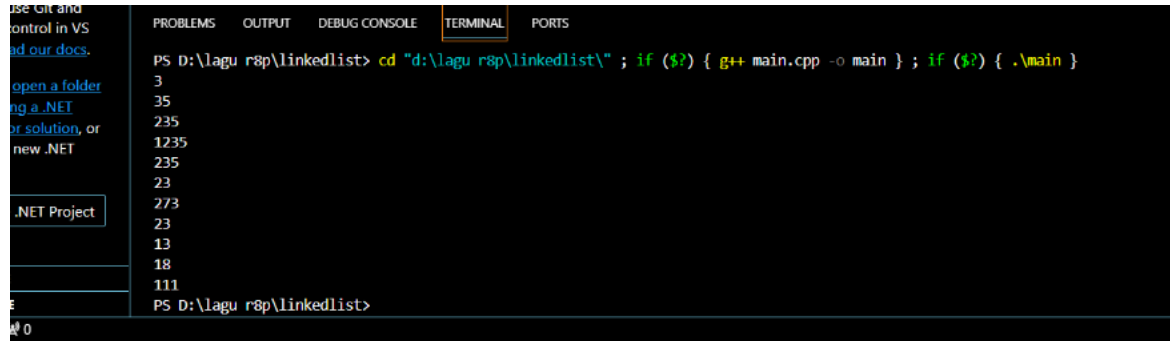
// Tampilkan list
void tampilList()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << " ";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

int main()
{

```

```
init();  
insertDepan(3);  
tampilList();  
insertBelakang(5);  
tampilList();  
insertDepan(2);  
tampilList();  
insertDepan(1);  
tampilList();  
hapusDepan();  
tampilList();  
hapusBelakang();  
tampilList();  
insertTengah(7, 2);  
tampilList();  
hapusTengah(2);  
tampilList();  
ubahDepan(1);  
tampilList();  
ubahBelakang(8);  
tampilList();  
ubahTengah(11, 2);  
tampilList();  
  
return 0;  
}
```

SCREENSHOOT PROGRAM



```
PS D:\lagu r8p\linkedList> cd "d:\lagu r8p\linkedList\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
3
35
235
1235
235
23
273
23
13
18
111
PS D:\lagu r8p\linkedList>
```

DESKRIPSI PROGRAM

Kode di atas merupakan implementasi dari sebuah program yang mengelola sebuah linked list non-circular dalam bahasa C++.

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
        {
            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)
    {

```

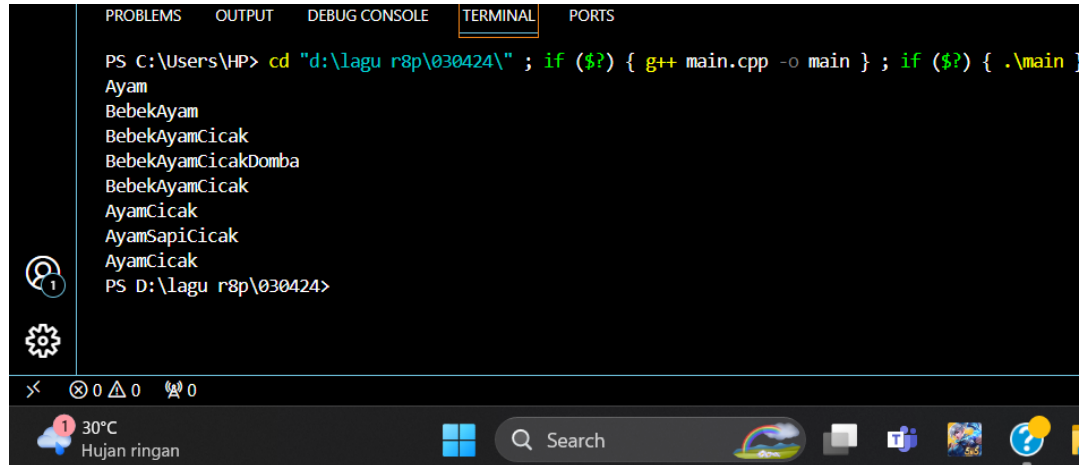
```

        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



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\HP> cd "d:\lagu r8p\030424\" ; if ($?) { g++ main.cpp -o main } ; if ($?) { .\main }
Ayam
BebekAyam
BebekAyamCicak
BebekAyamCicakDomba
BebekAyamCicak
AyamCicak
AyamSapiCicak
AyamCicak
PS D:\lagu r8p\030424>
```

DESKRIPSI PROGRAM

Kode di atas adalah implementasi dari program C++ yang menggunakan konsep linked list circular untuk menyimpan data. Menambahkan beberapa node ke linked list menggunakan berbagai fungsi penambahan node.

UNGUIDED

1. 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: Setelah membuat menu tersebut, masukkan data sesuai urutan berikut, lalu tampilkan data yang telah dimasukkan. (Gunakan insert depan, belakang atau tengah) Lakukan perintah berikut:

- a) Tambahkan data berikut diantara Farrel dan Denis: Wati 2330004
- b) Hapus data Denis
- c) Tambahkan data berikut di awal: Owi 2330000
- 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: Bagus 2330002
- i) Hapus data akhir
- j) Tampilkan seluruh data

SOURCE CODE

```
#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;
cout << "11. TAMPILKAN" << endl;
cout << "0. 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;
```

```

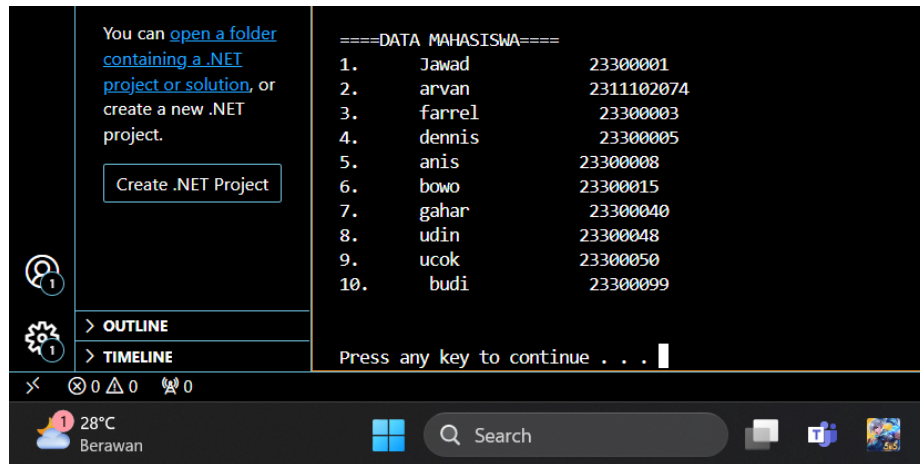
        linkedList.ubahTengah(nama, nim, posisi);
        break;
    case 7:
        linkedList.hapusDepan();
        break;
    case 8:
        linkedList.hapusBelakang();
        break;
    case 9:
        cout << "-Hapus Tengah-" << endl;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.hapusTengah(posisi);
        break;
    case 10:
        linkedList.hapusList();
        break;
    case 11:
        linkedList.tampilkanData();
        break;
    default:
        cout << "Pilihan tidak valid." << endl;
    }
} while (choice != 12);

return 0;
}

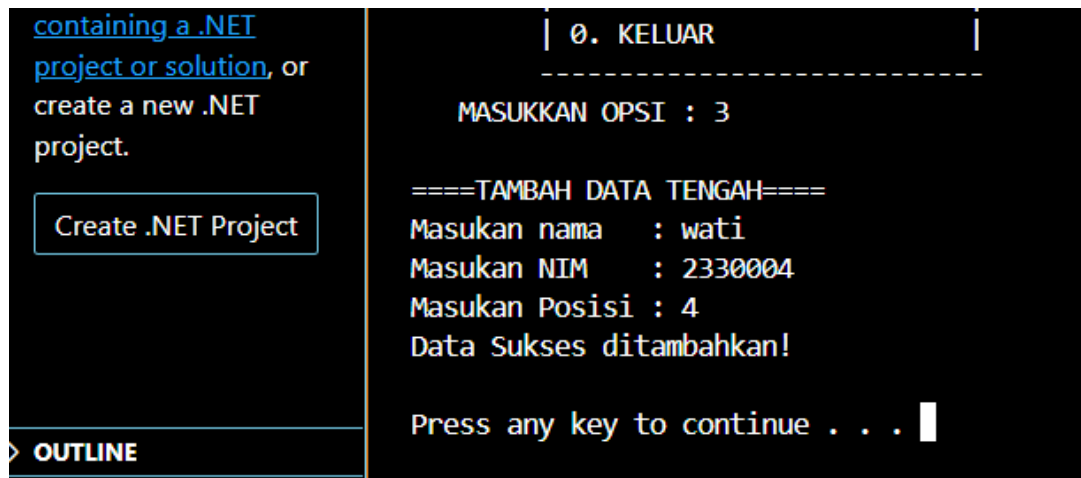
```


SCREENSHOOT PROGRAM

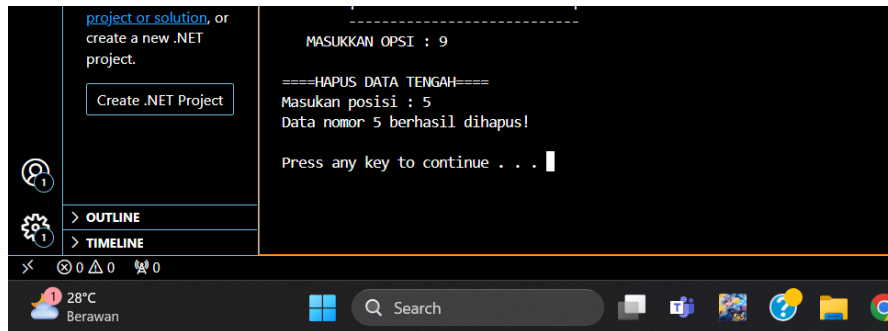
Screenshot tampilan data



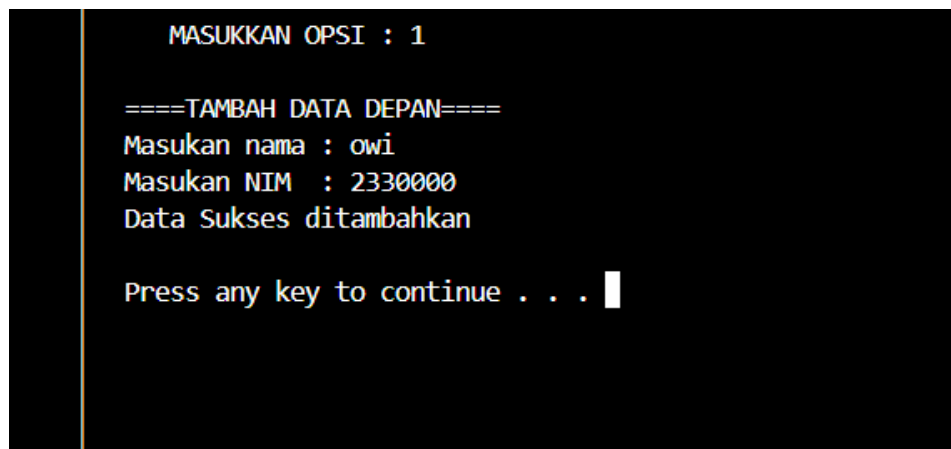
Screenshot perintah a



Screenshot perintah b



Screenshot perintah c



Screenshot perintah d

```
-----  
MASUKKAN OPSI : 2  
  
====TAMBAH DATA BELAKANG====  
Masukan nama : david  
Masukan NIM : 23300100  
Data Sukses ditambahkan  
  
Press any key to continue . . .
```

Screenshot perintah e

```
-----  
MASUKKAN OPSI : 6  
  
====UBAH DATA TENGAH====  
Masukan nama : idin  
Masukan NIM : 23300045  
Masukan Posisi : 9  
Data nomor 9 berhasil diganti  
  
Press any key to continue . . .
```

Screenshot perintah f

```
MASUKKAN OPSI : 5

====UBAH DATA BELAKANG====
Masukan nama : Lucy
Masukan NIM : 23300101
Data berhasil diganti

Press any key to continue . . .
```

Screenshot perintah g

```
MASUKKAN OPSI : 7

====HAPUS DATA DEPAN====
Data Berhasil Dihapus!

Press any key to continue . . .
```

Screenshot perintah h

```
-----  
MASUKKAN OPSI : 1  
  
====TAMBAH DATA DEPAN====  
Masukan nama : bagas  
Masukan NIM   : 2330002  
Data Sukses ditambahkan  
  
Press any key to continue . . .
```

Screenshot perintah i

```
-----  
MASUKKAN OPSI : 8  
  
====HAPUS DATA BELAKANG====  
Data Berhasil Dihapus!  
  
Press any key to continue . . .
```

Tampilkan seluruh data

```
-----
MASUKKAN OPSI : 11

====DATA MAHASISWA====
1.   bagas      2330002
2.   arvan      2311102074
3.   farrel     23300003
4.   wati       2330004
5.   anis       23300008
6.   bowo       23300015
7.   gahar      23300040
8.   idin       23300045
9.   ucok       23300050
10.  budi       23300099

Press any key to continue . . .
```

DESKRIPSI PROGRAM

Kode di atas merupakan implementasi dari sebuah program C++ yang menggunakan konsep linked list non-circular untuk mengelola data mahasiswa. Berdasarkan inputan para mahasiswa, untuk menu menggunakan perulangan while.

BAB IV

KESIMPULAN

Setelah melakukan pembelajaran mengenai tipe data di Bahasa Pemrograman C++ berikut poin utama yang telah dipelajari :

1. Linked list non circular adalah linked list dengan node pertama (head) dan node terakhir (tail) yang tidak saling terhubung.
2. Linked list circular adalah linked list yang tidak memiliki akhir karena node terakhir (tail) tidak bernilai 'NULL', tetapi terhubung dengan node pertama (head).
3. Praktikum Tujuan Praktikum menyediakan praktikum tentang linked list circular dan non circular.

DAFTAR PUSTAKA

[1] M. Bahrul Ulum, S.kom, M.Kom. 2019. Modul Kuliah Struktur Data Linked List. available: https://lms-paralel.esaunggul.ac.id/pluginfile.php?file=%2F86227%2Fmod_resource%2Fcontent%2F1%2FModul%20Struktur%20Data-Linked%20List.pdf

[2] S. Aggarwal, M. Gupta, "Performance Comparison of Circular and Non-Circular Linked Lists," 2020 5th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2020.