

**LAPORAN PRAKTIKUM STRUKTUR  
DATA DAN ALGORITMA**

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



**DISUSUN OLEH :  
BINTANG YUDHISTIRA  
2311102052**

**DOSEN  
WAHYU ANDI SAPUTRA, S.PD., M.PD.**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

## **A. Dasar Teori**

Linked list adalah struktur data linier berbentuk rantai simpul di mana setiap simpul menyimpan 2 item, yaitu nilai data dan pointer ke simpul elemen berikutnya. Berbeda dengan array, elemen linked list tidak ditempatkan dalam alamat memori yang berdekatan melainkan elemen ditautkan menggunakan pointer.

Simpul pertama dari linked list disebut sebagai head atau simpul kepala. Apabila linked list berisi elemen kosong, maka nilai pointer dari head menunjuk ke NULL. Begitu juga untuk pointer berikutnya dari simpul terakhir atau simpul ekor akan menunjuk ke NULL.

Ukuran elemen dari linked list dapat bertambah secara dinamis dan mudah untuk menyisipkan dan menghapus elemen karena tidak seperti array, kita hanya perlu mengubah pointer elemen sebelumnya dan elemen berikutnya untuk menyisipkan atau menghapus elemen.

Linked list biasanya digunakan untuk membuat file system, adjacency list, dan hash table.

## **JENIS JENIS LINKEDLIST YANG AKAN DI BAHAS PADA MODUL INI :**

- **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.

## **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. Penambah 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;
        }
    }
}
```

- **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.

## 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

Source Code :

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

// Deklarasi Struct Node
struct Node {
    int data;
    Node* next;
};

Node* head;
```

```

Node* tail;

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

// Pengecekan apakah list kosong
bool isEmpty() {
    return head == NULL;
}

// Tambah Node di depan
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;
    }
}

// Tambah Node di belakang
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;
    }
}

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

```

```

}

// Tambah Node di posisi tengah
void insertTengah(int data, int posisi) {
    if (posisi < 1 || posisi > hitungList()) {
        cout << "Posisi diluar jangkauan" << endl;
    } else if (posisi == 1) {
        cout << "Posisi bukan posisi tengah" << endl;
    } else {
        Node* baru = 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;
    }
}

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

// Hapus Node di belakang
void hapusBelakang() {
    if (!isEmpty()) {
        if (head != tail) {
            Node* hapus = tail;
            Node* bantu = head;
            while (bantu->next != tail) {
                bantu = bantu->next;
            }
            tail = bantu;
        }
    }
}

```

```

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

// Hapus Node di posisi tengah
void hapusTengah(int posisi) {
    if (posisi < 1 || posisi > hitungList()) {
        cout << "Posisi diluar jangkauan" << endl;
    } else if (posisi == 1) {
        cout << "Posisi bukan posisi tengah" << endl;
    } else {
        Node* hapus;
        Node* bantu = head;
        for (int nomor = 1; nomor < posisi - 1; nomor++) {
            bantu = bantu->next;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    }
}

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

// Ubah data Node di posisi tengah
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;
            for (int nomor = 1; nomor < posisi; nomor++) {
                bantu = bantu->next;
            }
            bantu->data = data;
        }
    }
}

```

```

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

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

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

// Tampilkan semua data Node di list
void tampil() {
    if (!isEmpty()) {
        Node* bantu = head;
        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;
}

```

## Screenshots Output:

```

PS C:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 3 LinkedList> g++ Guided1.cpp
PS C:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 3 LinkedList> .\Guided1.exe
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
PS C:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 3 LinkedList>

```

## Deskripsi Program :

Fungsi `init()` bertanggung jawab untuk menginisialisasi pointer `head` dan `tail` menjadi `NULL`, menandakan bahwa linked list kosong. Fungsi `isEmpty()` digunakan untuk memeriksa apakah linked list kosong atau tidak. Kemudian, fungsi `insertDepan(int nilai)` dan `insertBelakang(int nilai)` menambahkan node baru di depan dan belakang linked list, berturut-turut. Fungsi `hitungList()` menghitung jumlah node dalam linked list. Fungsi `insertTengah(int data, int posisi)` menambahkan node baru di posisi tengah linked list.



Sementara itu, fungsi `hapusDepan()` dan `hapusBelakang()` menghapus node pertama dan terakhir dari linked list.

Untuk mengubah nilai data dari node, ada fungsi `ubahDepan(int data)`, `ubahTengah(int data, int posisi)`, dan `ubahBelakang(int data)`. Fungsi-fungsi tersebut mengubah nilai data dari node pertama, node di posisi tengah, dan node terakhir linked list. Terakhir, fungsi `clearList()` menghapus semua node dari linked list, sementara `tampil()` menampilkan semua nilai data dalam linked list. Fungsi `main()` digunakan untuk menguji implementasi linked list dengan memanggil fungsi-fungsi tersebut.

## Guided 2

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;
    }
}

```

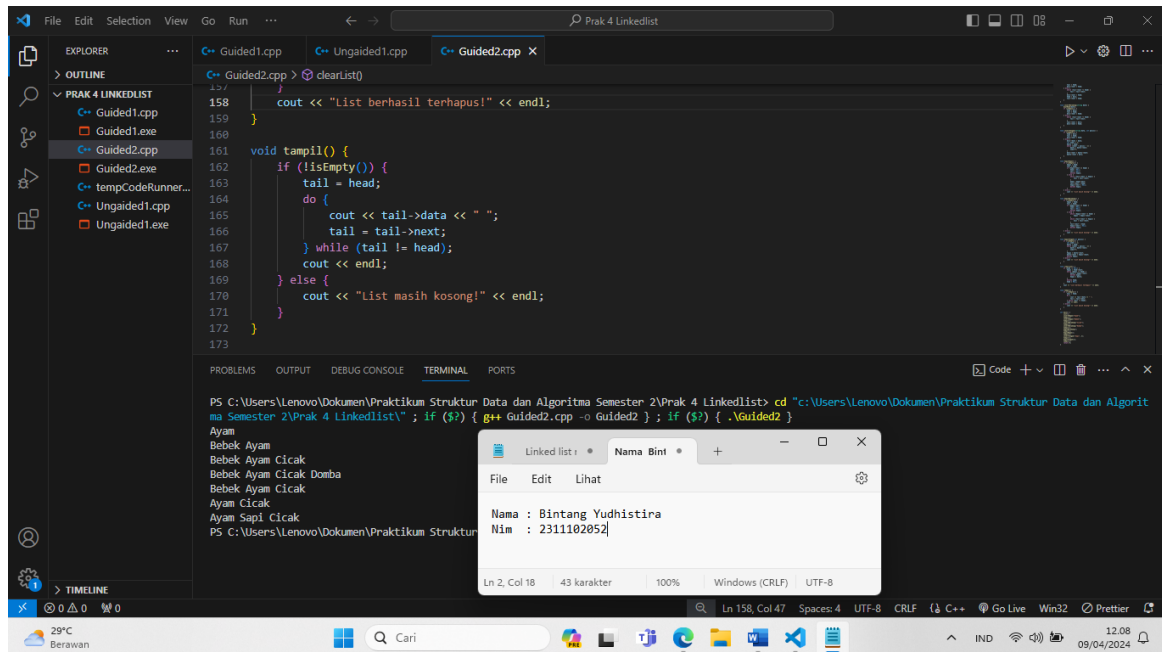
```

        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;
}

```

## Screenshots Output:



The screenshot shows a C++ IDE with a project named 'Prak 4 Linkedlist'. The Explorer panel on the left shows the project structure with files like Guided1.cpp, Ungaided1.cpp, Guided2.cpp, Guided2.exe, tempCodeRunner..., Ungaided1.cpp, and Ungaided1.exe. The main editor displays the code for Guided2.cpp, which includes functions like clearList(), isEmpty(), buatNode(), hitungList(), insertDepan(), insertBelakang(), insertTengah(), hapusDepan(), hapusBelakang(), hapusTengah(), and tampil(). The output window at the bottom shows the execution results, including the command prompt output and a small window titled 'Linked list' displaying the list contents: 'Nama : Bintang Yudhistira' and 'Nim : 2311102052'.

```
File Edit Selection View Go Run ...  
Prak 4 Linkedlist  
EXPLORER  
OUTLINE  
PRAK 4 LINKEDLIST  
Guided1.cpp  
Guided1.exe  
Guided2.cpp  
Guided2.exe  
tempCodeRunner...  
Ungaided1.cpp  
Ungaided1.exe  
Guided2.cpp  
158 cout << "List berhasil terhapus!" << endl;  
159 }  
160  
161 void tampil() {  
162     if (!isEmpty()) {  
163         tail = head;  
164         do {  
165             cout << tail->data << " ";  
166             tail = tail->next;  
167         } while (tail != head);  
168         cout << endl;  
169     } else {  
170         cout << "List masih kosong!" << endl;  
171     }  
172 }  
173  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 4 Linkedlist> cd "c:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 4 Linkedlist"; 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  
PS C:\Users\Lenovo\Dokumen\Praktikum Struktur Data dan Algoritma Semester 2\Prak 4 Linkedlist>  
Linked list * Nama Bint *  
File Edit Lihat  
Nama : Bintang Yudhistira  
Nim : 2311102052  
Ln 2, Col 18 | 43 karakter | 100% Windows (CRLF) UTF-8
```

## Deskripsi Program :

Kode yang diberikan adalah implementasi dari struktur data circular linked list dalam bahasa pemrograman C++. Circular linked list adalah kumpulan simpul data yang terhubung secara berurutan, di mana simpul terakhir terhubung kembali ke simpul pertama, membentuk suatu lingkaran. Pada awalnya, program menginisialisasi pointer head dan tail menjadi NULL melalui fungsi init(), menandakan bahwa linked list masih kosong. Terdapat juga beberapa variabel global seperti baru, bantu, dan hapus yang digunakan untuk operasi-operasi pada linked list. Fungsi isEmpty() digunakan untuk memeriksa apakah linked list kosong atau tidak, dan fungsi buatNode(string data) digunakan untuk membuat simpul baru dengan data yang diberikan. Fungsi hitungList() menghitung jumlah simpul dalam linked list. Kemudian, terdapat beberapa fungsi untuk menyisipkan simpul baru ke dalam linked list, baik di depan (insertDepan()), di belakang (insertBelakang()), maupun di tengah (insertTengah()). Operasi penghapusan juga diimplementasikan melalui fungsi-fungsi seperti hapusDepan(), hapusBelakang(), dan hapusTengah(). Fungsi clearList() digunakan untuk menghapus semua simpul dari linked list. Sedangkan tampil() adalah fungsi yang bertugas menampilkan isi dari linked list ke layar. Dalam fungsi main(), dilakukan serangkaian operasi untuk menguji implementasi linked list, seperti menyisipkan elemen-elemen ke dalam linked list, menghapus elemen-elemen tersebut, dan menampilkan isi linked list setiap kali operasi dilakukan. Melalui

kode ini, dapat dipahami bagaimana circular linked list diimplementasikan dalam bahasa pemrograman C++ dan berbagai operasi yang dapat dilakukan terhadapnya.

### C. Unguided

Source Code:

```
#include <iostream>
#include <iomanip>
using namespace std;
struct mahasiswa
{
    string nama;
    string nim;
};

struct node
{
    mahasiswa ITTP;
    node *next;
};
node *head, *tail, *bantu, *hapus, *before, *baru;
void init()
{
    head = NULL;
    tail = NULL;
}
bool isEmpty()
{
    if (head == NULL)
    {
        return true;
    }
    else
    {
        return false;
    }
}
mahasiswa Pendataan()
{
    mahasiswa ITTP;
    cout << "\nMasukkan Nama\t: ";
    cin.ignore();
```

```

getline(cin, ITTP.nama);
cout << "Masukkan NIM\t: ";
cin >> ITTP.nim;
return ITTP;
}

void insertDepan(mahasiswa ITTP)
{
    node *baru = new node;
    baru->ITTP.nama = ITTP.nama;
    baru->ITTP.nim = ITTP.nim;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }

    else
    {
        baru->next = head;
        head = baru;
    }
    cout << "Data " << ITTP.nama << " berhasil diinput!\n";

}

void insertBelakang(mahasiswa ITTP)
{
    node *baru = new node;
    baru->ITTP.nama = ITTP.nama;
    baru->ITTP.nim = ITTP.nim;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}

int hitungList()
{
    int penghitung = 0;

```



```

node *bantu;
bantu = head;
while (bantu != NULL)
{
    penghitung++;
    bantu = bantu->next;
}
return penghitung;
}

void insertTengah(mahasiswa iden0tas, int posisi)
{
    node *baru = new node;
    baru->ITTP.nama = iden0tas.nama;
    baru->ITTP.nim = iden0tas.nim;
    node *bantu;
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "posisi diluar jangkauan";
    }
    else if (posisi == 1)
    {
        cout << "Ini bukan posisi tengah\n";
    }
    else
    {
        bantu = head;
        int penghitung = 1;
        while (penghitung != posisi - 1)
        {
            penghitung++;
            bantu = bantu->next;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

void ubahDepan(mahasiswa data)
{
    string namaBefore = head->ITTP.nama;
    head->ITTP.nama = data.nama;
    head->ITTP.nim = data.nim;
    cout << "data " << namaBefore << " telah digan0 dengan data " <<
    data.nama << endl;
}

void ubahBelakang(mahasiswa data)

```

```

{
string namaBefore = tail->ITTP.nama;
tail->ITTP.nama = data.nama;
tail->ITTP.nim = data.nim;
cout << "data " << namaBefore << " telah digan0 dengan data " <<
data.nama << endl;
}

void ubahTengah(mahasiswa data)
{
int posisi;
cout << "\nMasukkan posisi data yang akan diubah : ";
cin >> posisi;

if (posisi < 1 || posisi > hitungList())
{
cout << "\nPosisi diluar jangkauan\n";
}
else if (posisi == 1)
{
cout << "\nBukan posisi tengah\n";
}
else
{
bantu = head;
int penghitung = 1;
while (penghitung != posisi)
{
penghitung++;
bantu = bantu->next;
}
bantu->ITTP.nama = data.nama;
bantu->ITTP.nim = data.nim;
}
}

void tampil()
{
node *bantu = head;
cout << "Nama "
<< " Nim\n";
while (bantu != NULL)
{

cout << bantu->ITTP.nama << " " << bantu->ITTP.nim << endl;
bantu = bantu->next;
}
}
}

```

```

void hapusDepan()
{
    string dataBefore = head->ITTP.nama;
    hapus = head;
    if (head != tail)
    {
        head = head->next;
        delete hapus;
    }
    else
    {
        head = tail = NULL;
    }
    cout << "Data " << dataBefore << " berhasil dihapus\n";
}

void hapusBelakang()
{
    string dataBefore = head->ITTP.nama;
    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;
    }
    cout << "Data " << dataBefore << " berhasil dihapus\n";
}

void hapusTengah()
{
    tampil();
    cout << endl;
    if (isEmpty() == false)
    {
        back:
        int posisi;
        cout << "Masukkan Posisi yang dihapus : ";
    }
}

```

```

cin >> posisi;
if (posisi < 1 || posisi > hitungList())
{
cout << "\nPosisi di luar jangkauan!\n";

cout << "Masukkan posisi baru\n";
goto back;

}
else if (posisi == 1 || posisi == hitungList())
{
cout << "\nBukan Posisi tengah\n";

cout << "Masukkan posisi baru\n";
goto back;
}
else
{
bantu = head;
int penghitung = 1;
while (penghitung <= posisi)
{
if (penghitung == posisi - 1)
{
before = bantu;
}
if (penghitung == posisi)
{
hapus = bantu;
}
bantu = bantu->next;
penghitung++;
}
string dataBefore = hapus->ITTP.nama;
before->next = bantu;
delete hapus;
cout << "\nData " << dataBefore << " berhasil dihapus!\n";

}
}
else
{
cout << "\n!!! List Data Kosong !!!\n";

}
}

```

```

void hapusList()
{
    bantu = head;
    while (bantu != NULL)
    {
        hapus = bantu;
        delete hapus;
        bantu = bantu->next;
    }
    init();
    cout << "\nsemua data berhasil dihapus\n";
}

int main()
{
    init();
    mahasiswa ITTP;
back:
    int operasi, posisi;
    cout << " PROGRAM SINGLE LINKED LIST NON-CIRCULAR" << endl;
    cout << " =====\n\n" << 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 Teangah" << endl;
    cout << "10.Hapus list" << endl;
    cout << "11.Tampilkan" << endl;
    cout << "0. Exit" << endl;

    cout << "\nPilih Operasi :> ";
    cin >> operasi;

    switch (operasi)
    {
        case 1:
            cout << "tambah depan\n";
            insertDepan(Pendataan());
            cout << endl;
            goto back;
            break;

        case 2:
            cout << "tambah belakang\n";

```

```
insertBelakang(Pendataan());
cout << endl;
goto back;

break;
case 3:
cout << "tambah tengah\n";
cout << "nama : ";
cin >> ITTP.nama;
cout << "NIM : ";
cin >> ITTP.nim;
cout << "Posisi: ";
cin >> posisi;
insertTengah(ITTP, posisi);
cout << endl;
goto back;
break;
case 4:
cout << "ubah depan\n";
ubahDepan(Pendataan());
cout << endl;
goto back;
break;
case 5:
cout << "ubah belakang\n";
ubahBelakang(Pendataan());
cout << endl;
goto back;
break;
case 6:
cout << "ubah tengah\n";
ubahTengah(Pendataan());
cout << endl;
goto back;

break;
case 7:
cout << "hapus depan\n";
hapusDepan();
cout << endl;
goto back;
break;
case 8:
cout << "hapus belakang\n";
hapusBelakang();
cout << endl;
goto back;
break;
case 9:
```

```
cout << "hapus tengah\n";
hapusTengah();
cout << endl;
goto back;
break;
case 10:
cout << "hapus list\n";
hapusList();
cout << endl;
goto back;
break;
case 11:
tampil();
cout << endl;
goto back;
break;

case 0:
cout << "\nEXIT PROGRAM\n";
break;

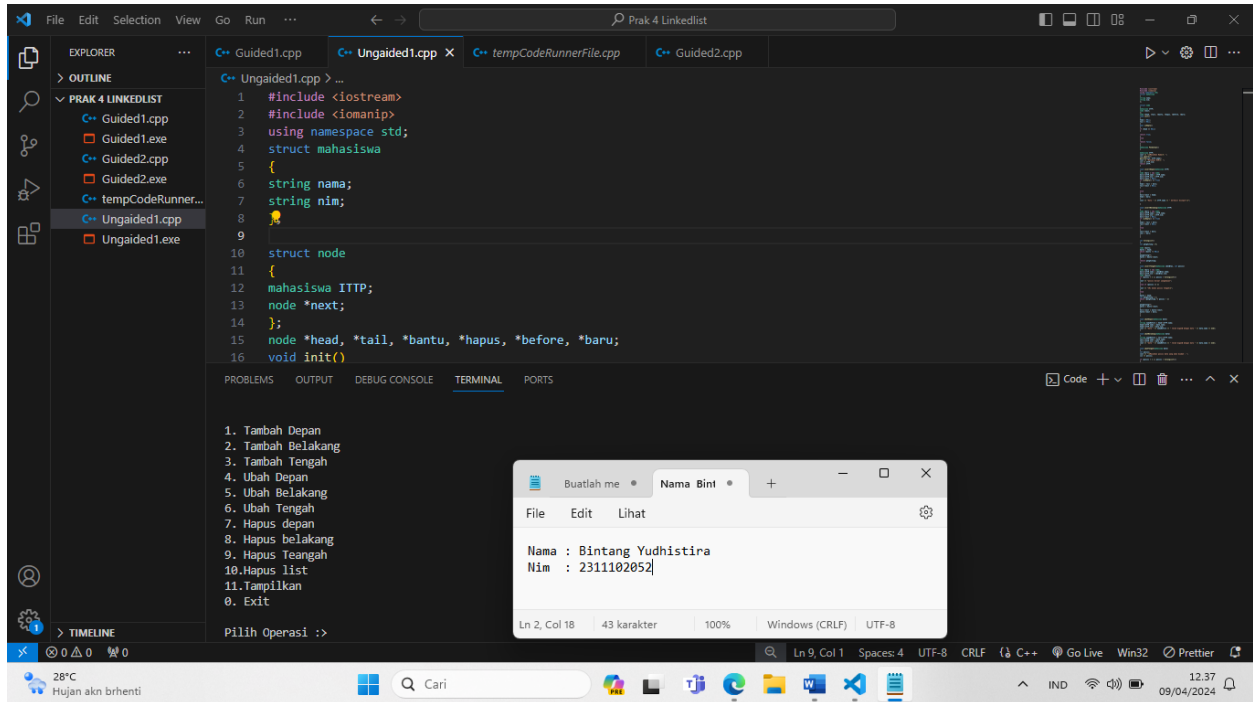
default:
cout << "\nSalah input operasi\n";
cout << endl;
goto back;
break;
}

return 0;
}
```

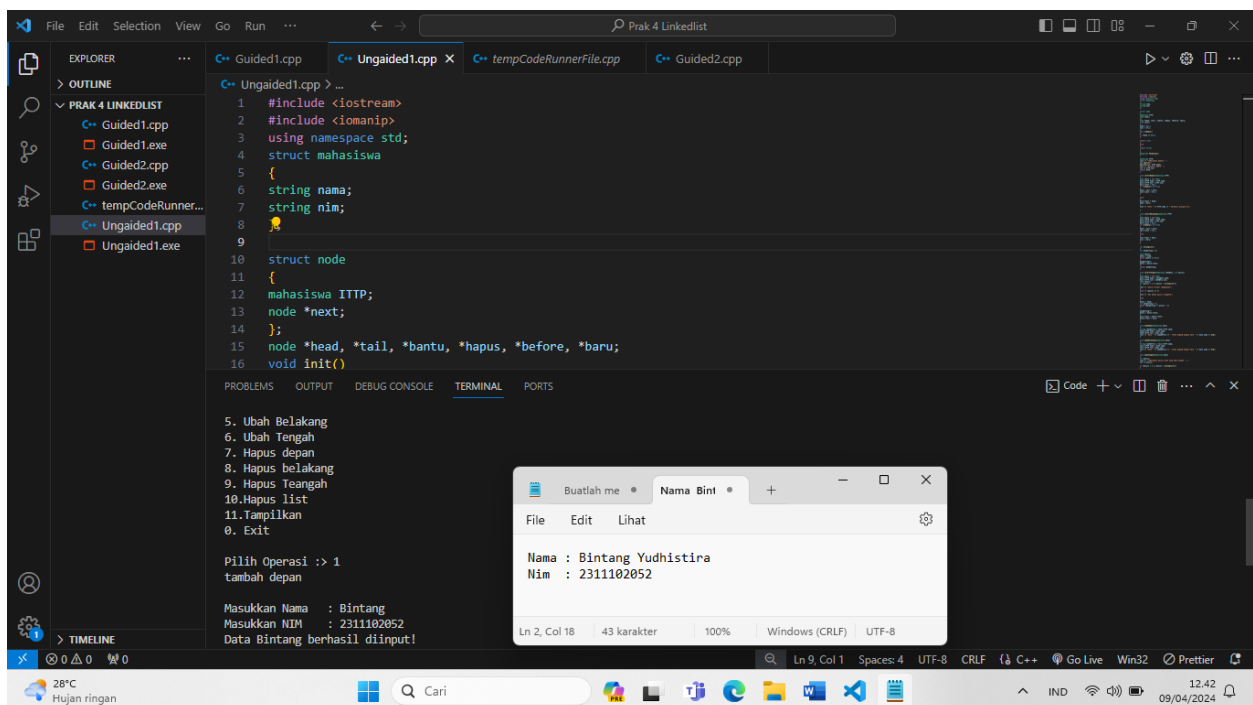
Screenshot output:

1. Buatlah menu untuk menambahkan, mengubah, menghapus, dan melihat Nama dan NIM Mahasiswa.

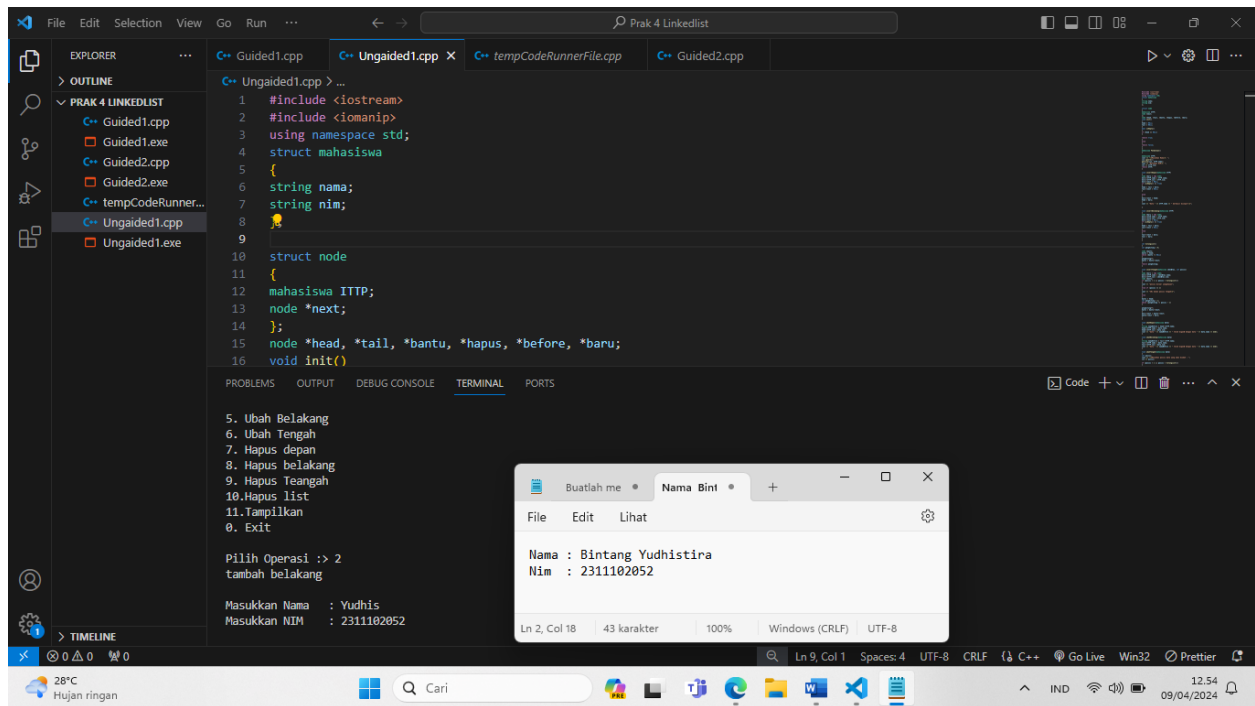
- Tampilan Menu



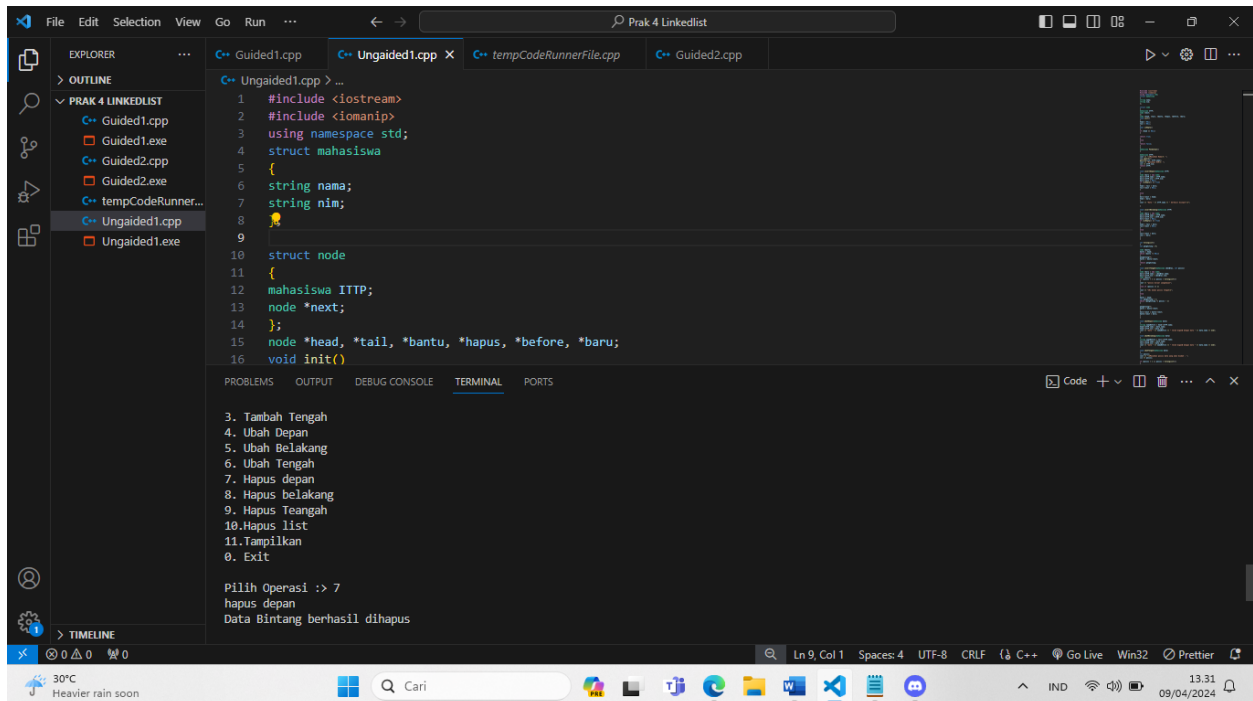
- Tampilan Operasi Tambah



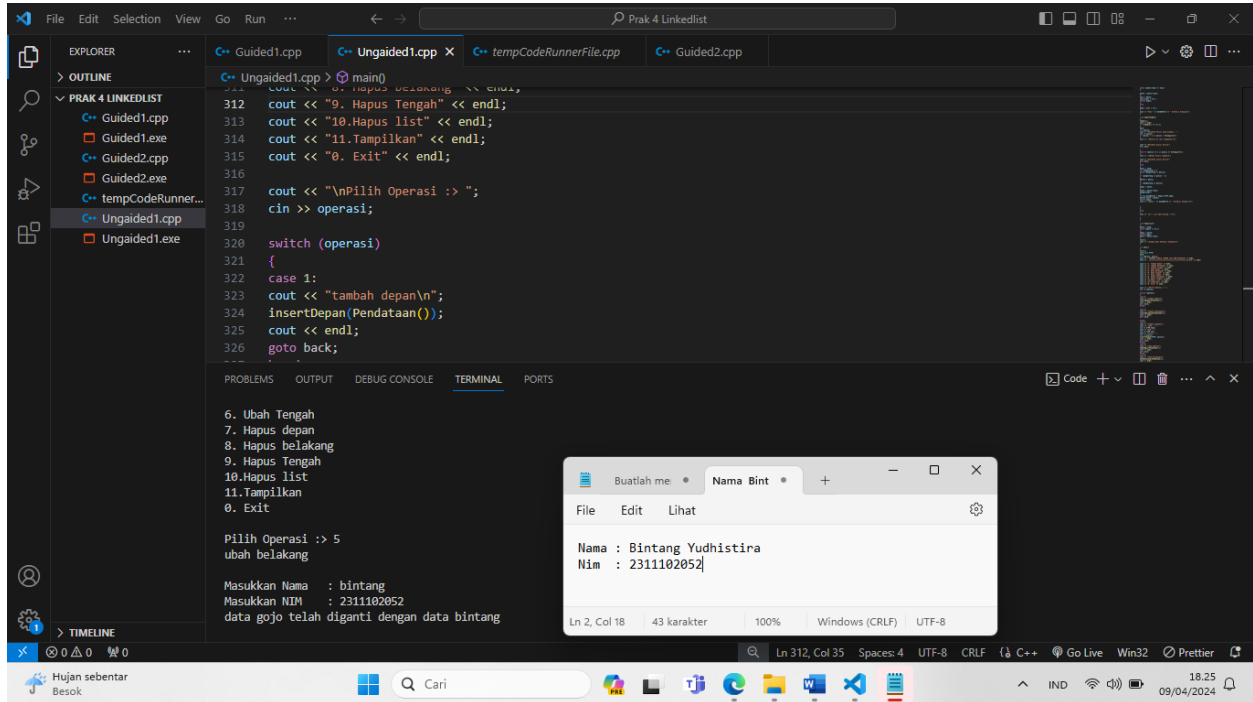




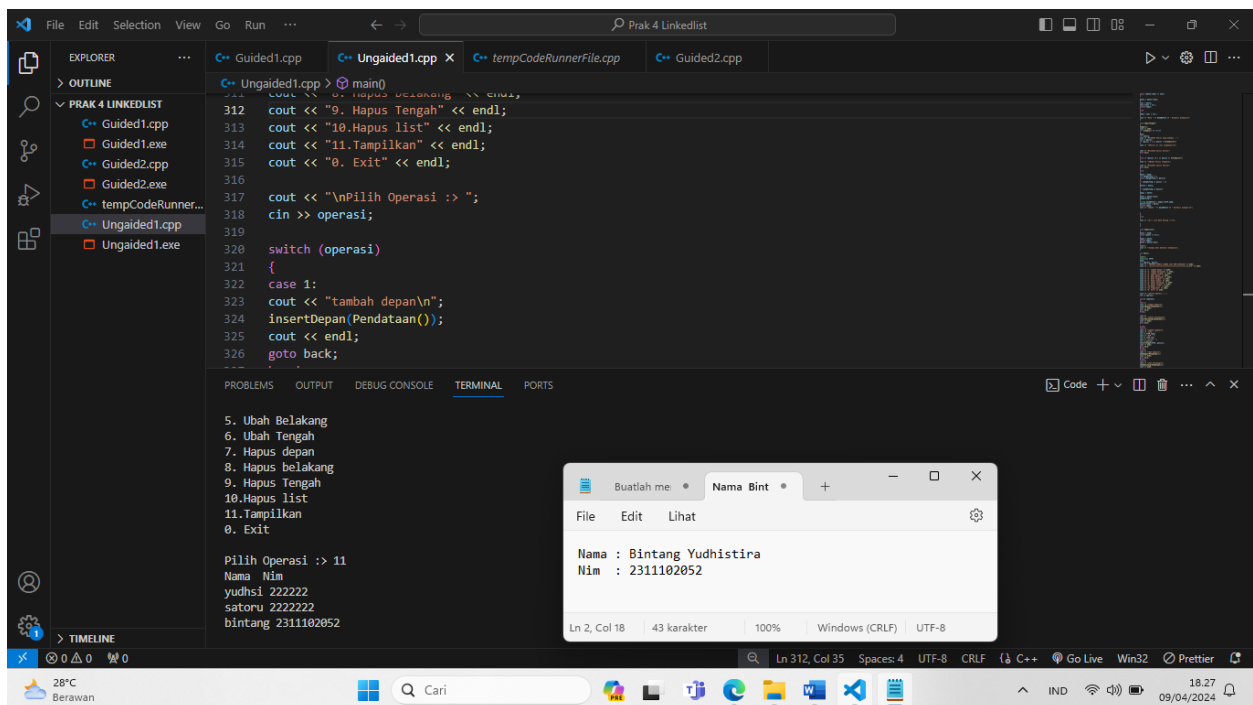
- Tampilan Operasi Hapus



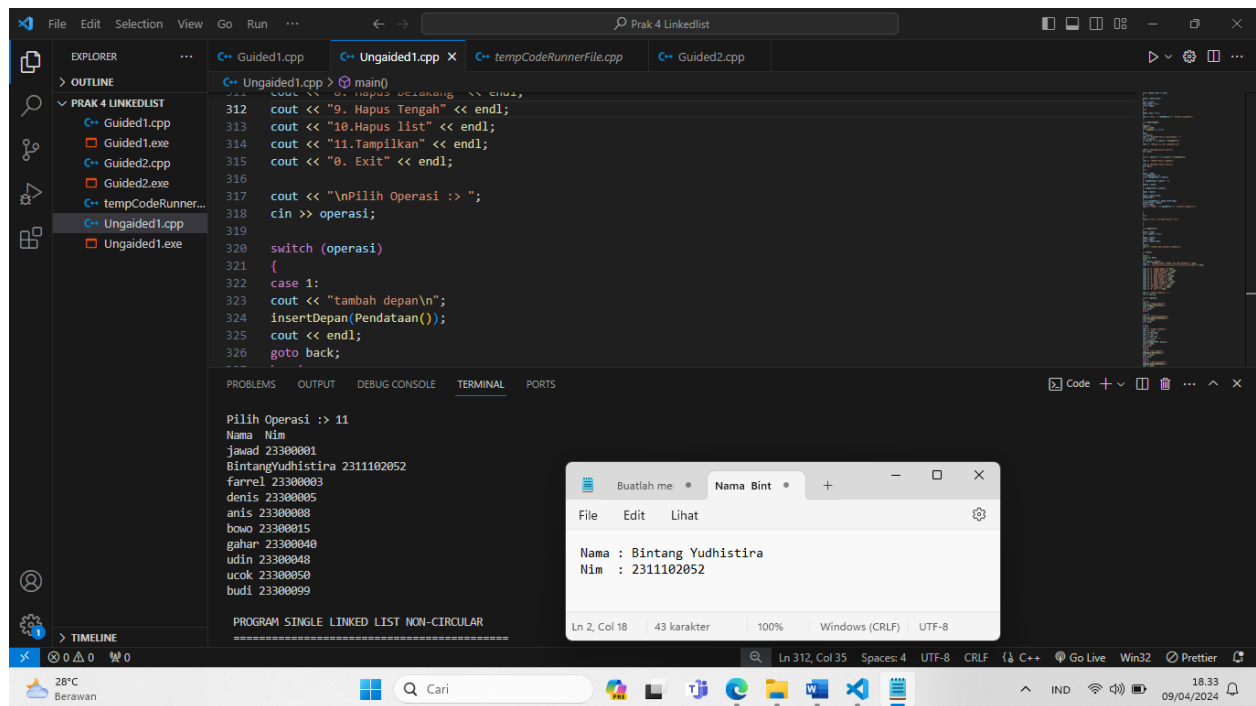
- Tampilan Operasi Ubah



- Tampilan Operasi Tampil Data



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



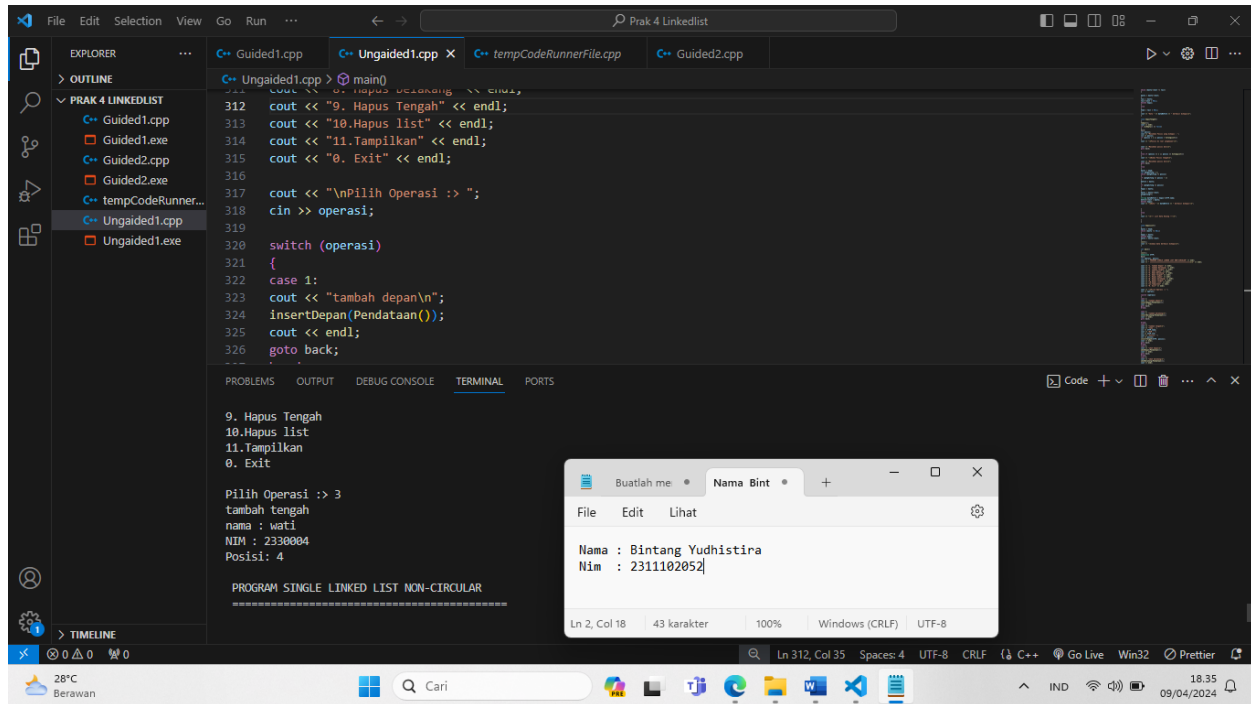
```
File Edit Selection View Go Run ...
EXPLORER
PRAK 4 LINKEDLIST
  Guided1.cpp
  Guided1.exe
  Guided2.cpp
  Guided2.exe
  tempCodeRunner...
  Ungaided1.cpp
  Ungaided1.exe
OUTLINE
  main()
  312 cout << "9. Hapus Tengah" << endl;
  313 cout << "10. Hapus list" << endl;
  314 cout << "11. Tampilkan" << endl;
  315 cout << "0. Exit" << endl;
  316
  317 cout << "\nPilih Operasi :> ";
  318 cin >> operasi;
  319
  320 switch (operasi)
  321 {
  322 case 1:
  323   cout << "tambah depan\n";
  324   insertDepan(Pendataan());
  325   cout << endl;
  326   goto back;
  327
  328 case 2:
  329   cout << "tambah belakang\n";
  330   insertBelakang(Pendataan());
  331   cout << endl;
  332   goto back;
  333
  334 case 3:
  335   cout << "hapus tengah\n";
  336   deleteTengah();
  337   cout << endl;
  338   goto back;
  339
  340 case 4:
  341   cout << "hapus list\n";
  342   deleteList();
  343   cout << endl;
  344   goto back;
  345
  346 case 5:
  347   cout << "tampilkan\n";
  348   display();
  349   cout << endl;
  350   goto back;
  351
  352 case 0:
  353   cout << "Program Selesai\n";
  354   return 0;
  355
  356 default:
  357   cout << "Invalid Input\n";
  358   goto back;
  359
  360 }
  361
  362 back:
  363
  364 }
  365
  366 return 0;
  367
  368 }
```

Pilih Operasi :> 11  
Nama Nim  
Jawad 23300001  
BintangYudhistira 2311102052  
farrel 23300003  
denis 23300005  
anis 23300008  
bowo 23300015  
gahar 23300048  
udin 23300048  
ucok 23300050  
budi 23300099

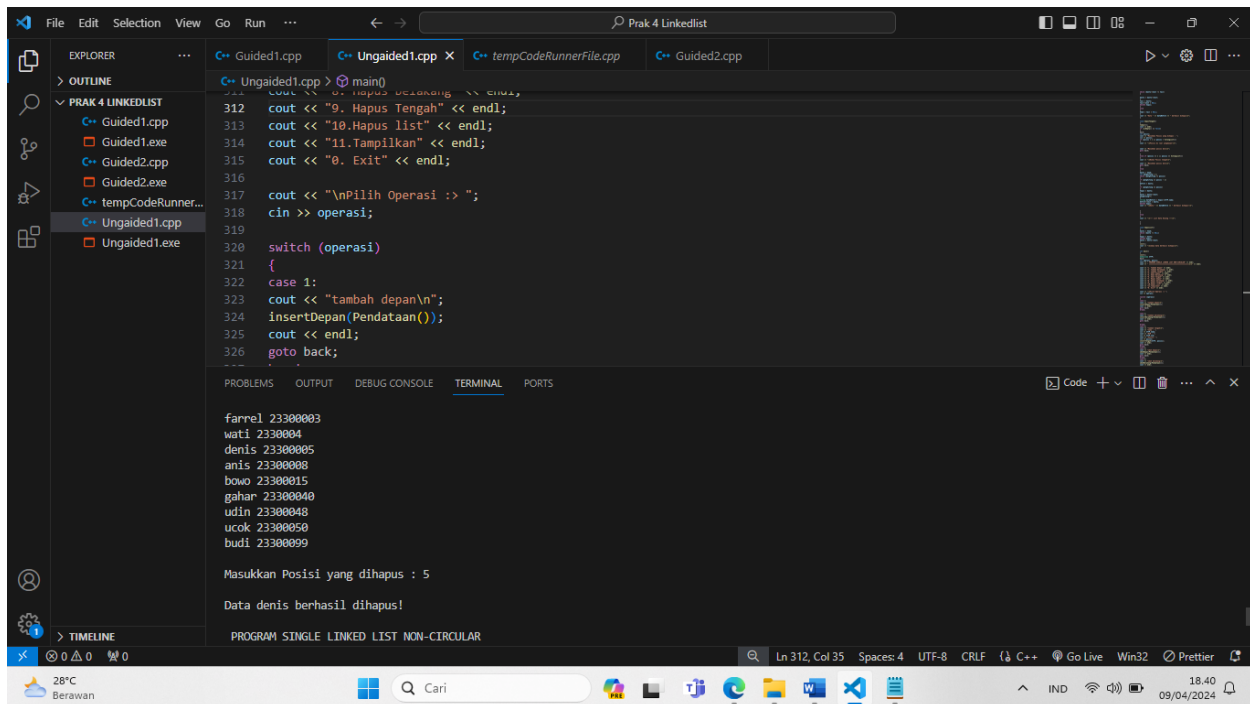
PROGRAM SINGLE LINKED LIST NON-CIRCULAR

Buatlah me Nama Bint  
File Edit Lihat  
Nama : Bintang Yudhistira  
Nim : 2311102052  
Ln 2, Col 18 43 karakter 100% Windows (CRLF) UTF-8

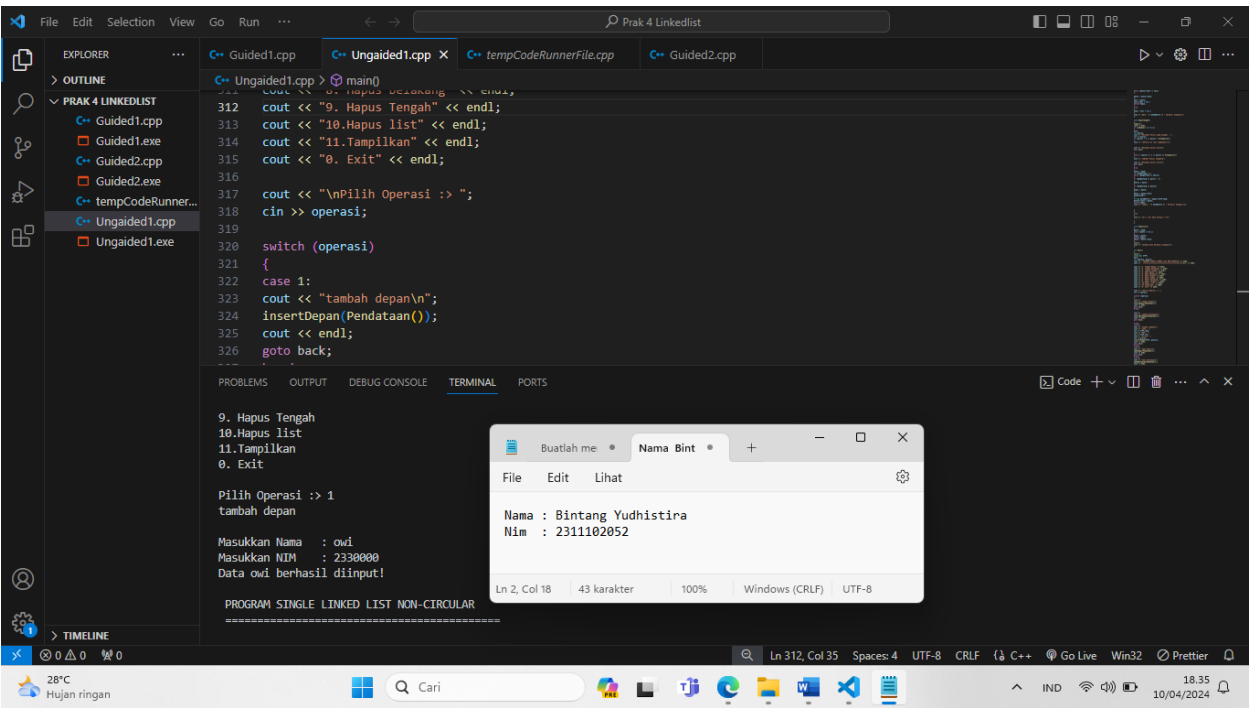
3. a. Tambahkan data berikut diantara Farrel dan Denis: Wati 23300004



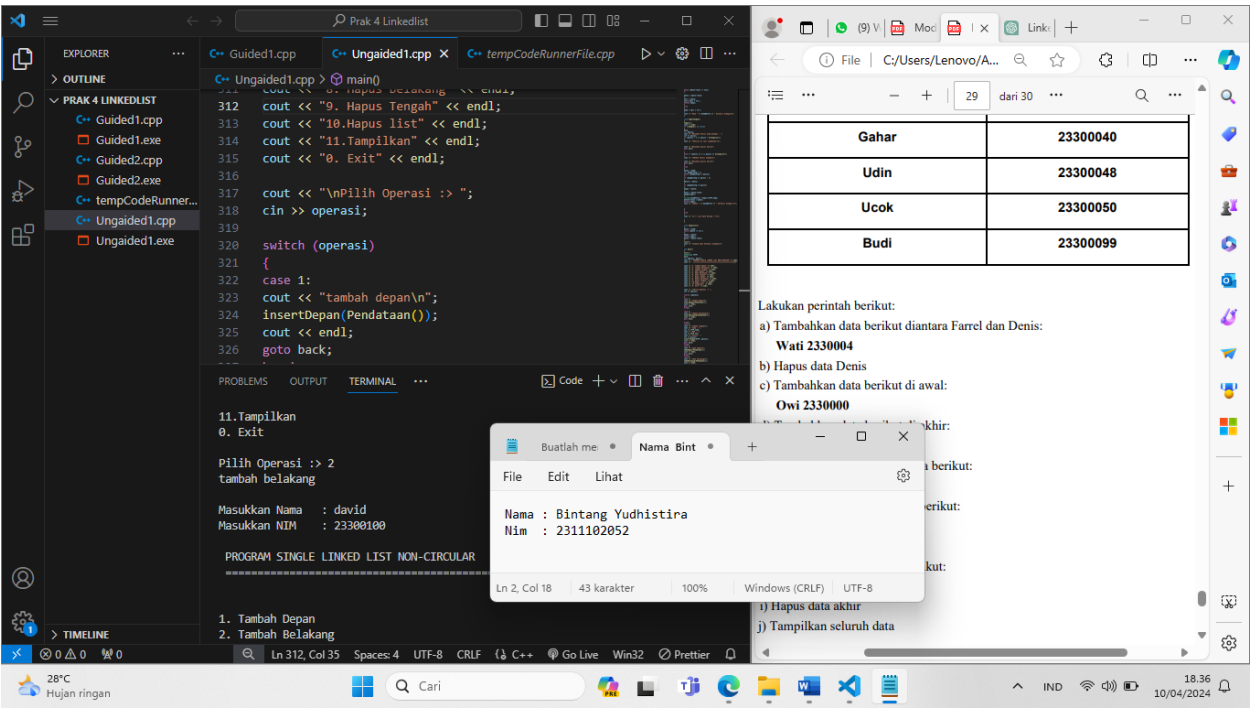
## b. Hapus data Denis



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

The screenshot shows a C++ IDE with the following components:

- EXPLORER:** Shows the project structure with files like Guided1.cpp, Guided1.exe, Guided2.cpp, Guided2.exe, tempCodeRunner..., Ungaided1.cpp, and Ungaided1.exe.
- CODE EDITOR:** Displays the code in Ungaided1.cpp. The code is a linked list program with operations like insert, delete, and display.
- TERMINAL:** Shows the program output and user input. The output includes "10.Hapus list", "11.Tampilkan", and "0. Exit". The user input shows "Pilih Operasi :> 6", "ubah tengah", "Masukkan Nama : idin", "Masukkan NIM : 23300045", and "Masukkan posisi data yang akan diubah : 9".
- Dialog Box:** A small dialog box titled "Nama Bint" shows the user's name "Nama : Bintang Yudhistira" and ID "Nim : 2311102052".
- Table:** A table on the right lists names and IDs: 

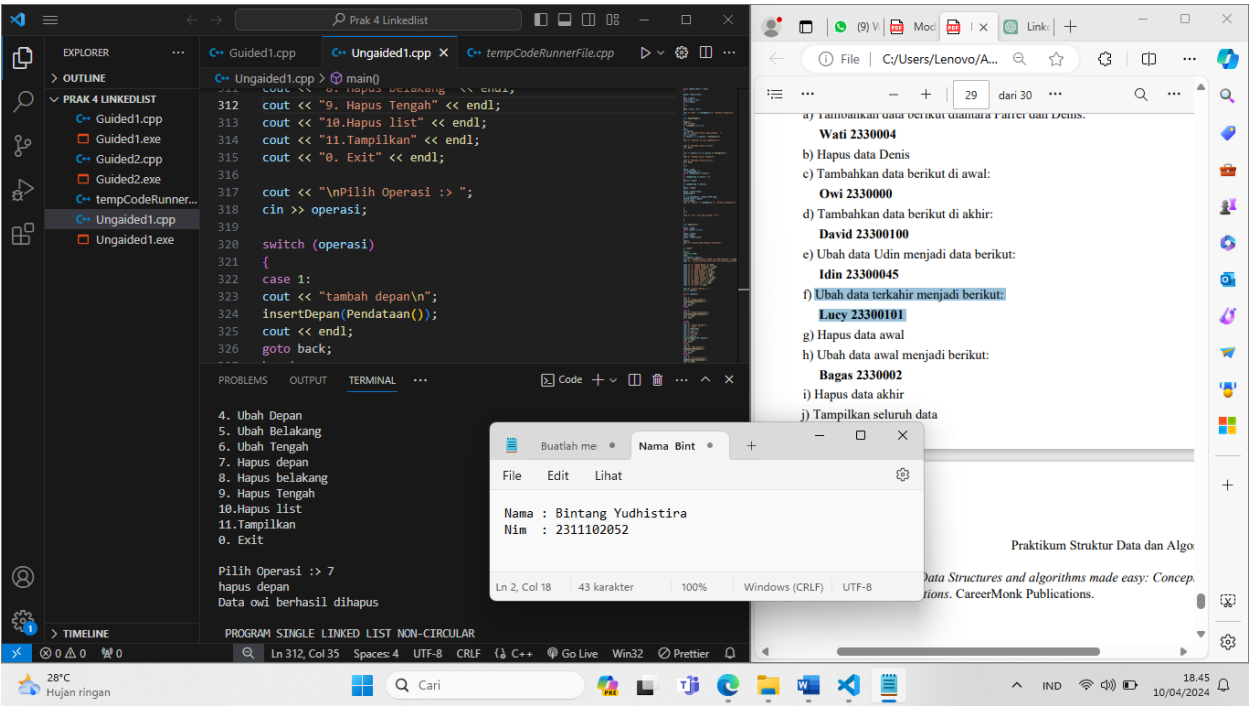
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099

f. Ubah data terakhir menjadi berikut: Lucy 23300101

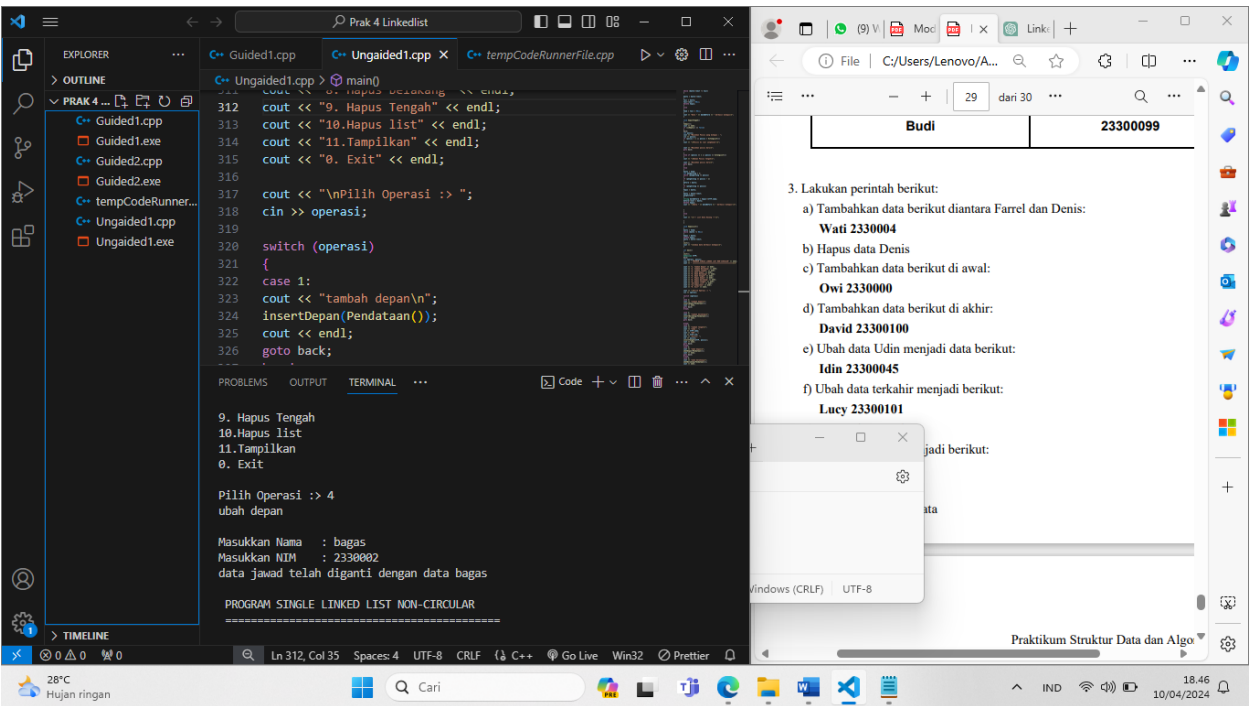
The screenshot shows a C++ IDE with the following components:

- EXPLORER:** Shows the project structure with files like Guided1.cpp, Guided1.exe, Guided2.cpp, Guided2.exe, tempCodeRunner..., Ungaided1.cpp, and Ungaided1.exe.
- CODE EDITOR:** Displays the code in Ungaided1.cpp. The code is a linked list program with operations like insert, delete, and display.
- TERMINAL:** Shows the program output and user input. The output includes "8. Hapus belakang", "9. Hapus Tengah", "10.Hapus list", "11.Tampilkan", and "0. Exit". The user input shows "Pilih Operasi :> 5", "ubah belakang", "Masukkan Nama : lucy", "Masukkan NIM : 23300101", and "data david telah diganti dengan data lucy".
- Dialog Box:** A small dialog box titled "Nama Bint" shows the user's name "Nama : Bintang Yudhistira" and ID "Nim : 2311102052".
- List of Instructions:** A list of instructions on the right includes the task to change the last data to Lucy 23300101:
  - a) Tambahkan data berikut diantara Farrel dan Denis:
  - b) Hapus data Denis
  - c) Tambahkan data berikut di awal:
  - d) Tambahkan data berikut di akhir:
  - e) Ubah data Udin menjadi data berikut:
  - f) Ubah data terakhir menjadi berikut:
  - g) Hapus data awal
  - h) Ubah data awal menjadi berikut:
  - i) Hapus data akhir
  - j) Tampilkan seluruh data

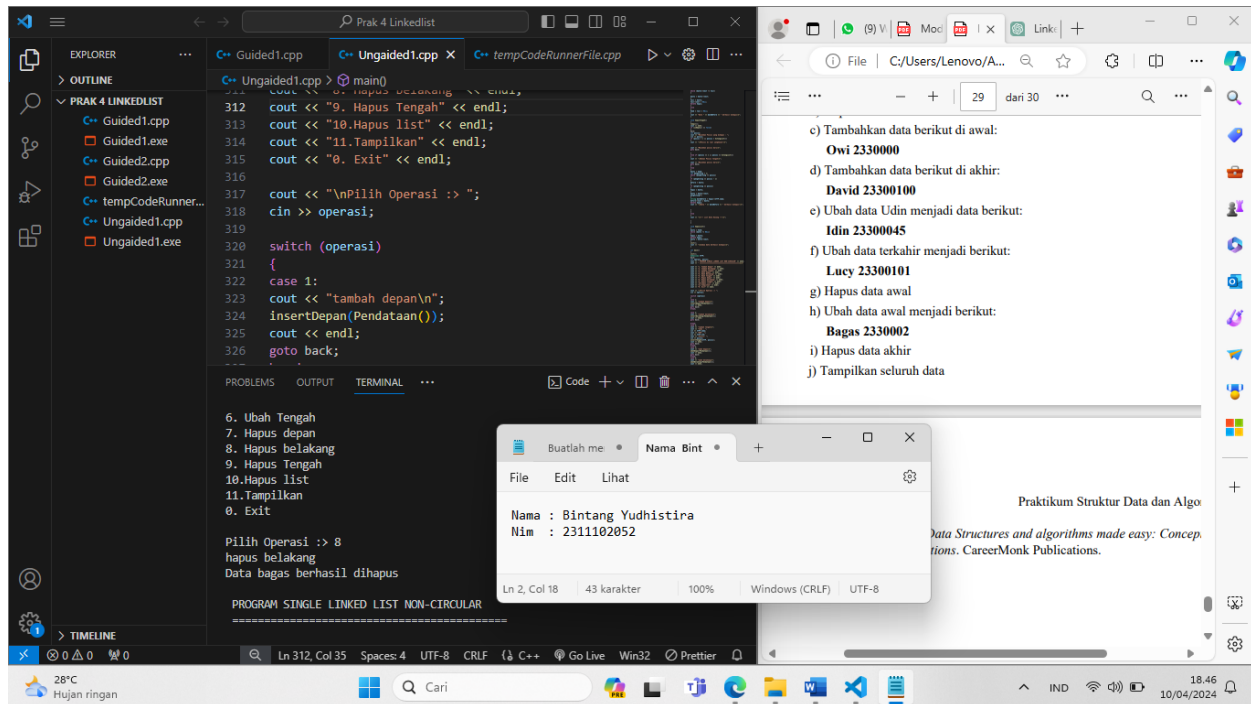
g. Hapus Data Awal



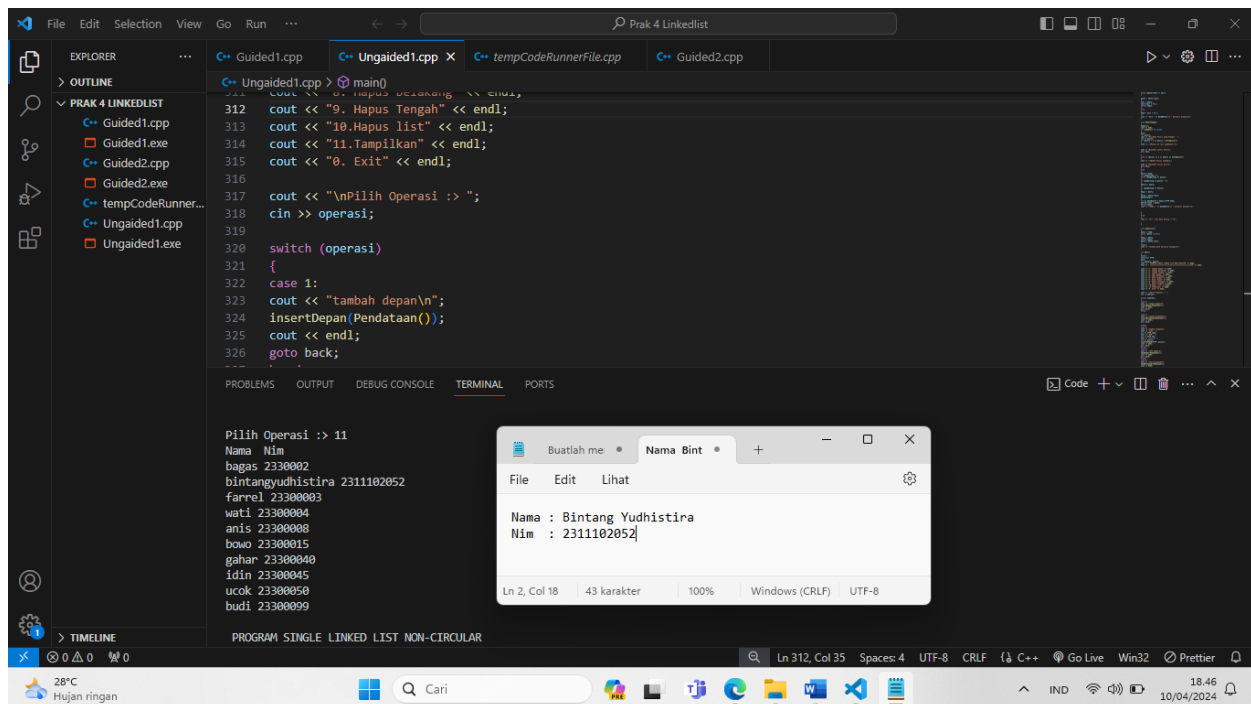
h. Ubah data awal menjadi berikut: Bagas 2330002



## i. Hapus Data Akhir



## j. Tampilkan seluruh data





#### Deskripsi Program :

Linked list tersebut menyimpan data mahasiswa, yang terdiri dari nama dan NIM. Struktur data linked list diimplementasikan dengan menggunakan dua struktur, yaitu mahasiswa yang merepresentasikan data mahasiswa, dan node yang merepresentasikan simpul (node) dalam linked list. Pada awalnya, terdapat inisialisasi dari pointer head dan tail dalam fungsi `init()`, yang diatur menjadi NULL untuk menandakan bahwa linked list masih kosong. Kemudian, terdapat fungsi `isEmpty()` yang digunakan untuk memeriksa apakah linked list kosong atau tidak. Operasi-operasi dasar pada linked list seperti `insertDepan()`, `insertBelakang()`, dan `insertTengah()` digunakan untuk menyisipkan data mahasiswa ke dalam linked list, baik di depan, di belakang, maupun di posisi tertentu. Selain itu, terdapat fungsi-fungsi lain seperti `hapusDepan()`, `hapusBelakang()`, dan `hapusTengah()` untuk menghapus data dari linked list. Terdapat juga fungsi-fungsi `tampil()` untuk menampilkan seluruh data mahasiswa yang ada dalam linked list, serta fungsi-fungsi `ubahDepan()`, `ubahBelakang()`, dan `ubahTengah()` untuk mengubah data mahasiswa yang ada dalam linked list. Di dalam fungsi `main()`, terdapat implementasi dari menu operasi-operasi yang dapat dilakukan pada linked list, seperti penambahan data, penghapusan data, perubahan data, dan penampilan data. Program akan berjalan secara iteratif sehingga pengguna dapat melakukan operasi-operasi tersebut secara berulang hingga memilih untuk keluar dari program.

#### **D. Kesimpulan**

Linked list circular dan non-circular merupakan dua variasi struktur data sering digunakan dalam pengembangan perangkat lunak untuk mengatur dan mengelola kumpulan data. Perbedaan signifikan antara keduanya terletak pada cara simpul terakhir dalam linked list dihubungkan kembali ke simpul pertama dalam linked list circular, sementara dalam linked list non-circular, simpul terakhir hanya menunjuk ke NULL, menandakan akhir dari linked list. Linked list circular memiliki kelebihan dalam manajemen memori karena tidak ada simpul yang "mati", yang dapat mengoptimalkan penggunaan memori dalam beberapa kasus, terutama ketika operasi traverse (penelusuran) linked list secara terus-menerus diperlukan. Namun, implementasi linked list circular memerlukan operasi tambahan seperti menentukan apakah linked list kosong atau menghitung jumlah elemen, karena harus mempertimbangkan sirkularitas struktur. Pemilihan antara kedua jenis linked list harus didasarkan pada kebutuhan spesifik dari

aplikasi atau masalah yang dihadapi. Jika navigasi berulang dari awal hingga akhir linked list sering terjadi, linked list circular mungkin lebih sesuai, namun jika tidak ada kebutuhan khusus seperti itu, linked list non-circular mungkin lebih mudah diimplementasikan dan dikelola..

## **E. Referensi**

[1] Asisten Pratikum “Modul 4 Linkedlist Circular dan non Circular”, Learning Management System, 2024.

[2] Educative. Singly linked list in C++. Diakses pada 31 Maret 2024.

Diakses Pada 7 April 2024, dari

<https://www.educative.io/answers/singly-linked-list-in-cpp>

[3] Taufikkipo (2012, juli). “Single Linked List Non Circular”.

Diakses pada 7 April 2024, dari

<https://www.trivusi.web.id/2022/07/struktur-data-linked-list.html>