

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

**MODUL 6
DOUBLY LINKED LIST (BAGIAN PERTAMA)**



Disusun Oleh :

NAMA : Muhammad Azzam Satria

NIM : 103112400112

Dosen

FAHRUDIN MUKTI WIBOWO S. Kom., M. Eng

**PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

A. Dasar Teori

Doubly linked list merupakan salah satu bentuk struktur data berantai yang setiap elemennya saling terhubung secara dua arah, yaitu ke elemen sebelumnya dan ke elemen setelahnya. Struktur ini memiliki dua penunjuk utama, yaitu HEAD yang mengarah ke node pertama dan TAIL yang mengarah ke node terakhir dalam daftar. Kondisi list dianggap kosong apabila pointer HEAD bernilai NULL. Pada node pertama, pointer prev selalu bernilai NULL karena tidak memiliki node sebelumnya, sedangkan pada node terakhir, pointer next bernilai NULL sebagai penanda bahwa tidak ada node setelahnya. Dengan menggunakan struktur data ini, data dalam list dapat diakses dan dimodifikasi dengan lebih mudah pada kedua ujungnya.

Berbagai operasi dapat dilakukan pada doubly linked list seperti menambah data, menghapus data dan menelusuri data. Setiap proses yang dilakukan pada doubly linked list akan selalu berhubungan dengan pengaturan ulang pointer next dan prev, agar hubungan antar node tetap tersambung dengan benar. Ketika sebuah data baru ditambahkan, pointer dari node sebelum dan sesudahnya akan disesuaikan agar elemen baru dapat terhubung di antara keduanya. Ketika data dihapus, node yang akan dihapus dilepaskan dengan cara memutus pointer penghubungnya, kemudian ruang memorinya dikembalikan agar tidak terjadi pemborosan. Cara kerja tersebut membuat doubly linked list lebih efisien dibanding array, karena setiap operasi penambahan atau penghapusan cukup dilakukan dengan menyesuaikan pointer antar elemen. Struktur data ini banyak digunakan pada aplikasi yang membutuhkan perubahan data secara cepat dan fleksibel, seperti sistem navigasi, aplikasi pemutar musik dan platform media sosial.

B. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1

```
#include <iostream>

using namespace std;
```

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

Node *ptr_first = NULL;
Node *ptr_last = NULL;

void add_first(int value)
{
    Node *newNode = new Node{ value, NULL, ptr_first};

    if (ptr_first == NULL)
    {
        ptr_last = newNode;
    }
    else
    {
        ptr_first->prev = newNode;
    }
    ptr_first = newNode;
}
```

```
void add_last(int value)
{
    Node *newNode = new Node{ value, ptr_last, NULL};

    if(ptr_last == NULL)
    {
        ptr_first = newNode;
    }
    else
    {
        ptr_last->next = newNode;
    }
    ptr_last = newNode;
}

void add_target(int targetValue, int newValue)
{
    Node *current = ptr_first;
    while (current != NULL && current->data != targetValue)
    {
        current = current->next;
    }

    if(current != NULL)
    {
```

```

        if (current == ptr_last)
        {
            add_last(newValue);
        }
        else
        {
            Node *newNode = new Node {newValue, current, current->next};
            current->next->prev = newNode;
            current->next = newNode;
        }
    }
}

void view()
{
    Node *current = ptr_first;
    if (current == NULL)
    {
        cout << "List Kosong\n";
        return;
    }
    while (current != NULL)
    {
        cout << current->data << (current->next != NULL ? "<->" : "");
        current = current->next;
    }
}

```

```
    }  
    cout << endl;  
}  
  
void delete_first()  
{  
    if (ptr_first == NULL)  
        return;  
  
    Node *temp = ptr_first;  
  
    if (ptr_first == ptr_last)  
    {  
        ptr_first = NULL;  
        ptr_last = NULL;  
    }  
    else  
    {  
        ptr_first = ptr_first->next;  
        ptr_first->prev = NULL;  
    }  
    delete temp;  
}  
  
void delete_last()
```

```
{  
    if(ptr_last == NULL)  
        return;  
  
    Node *temp = ptr_last;  
  
    if(ptr_first == ptr_last)  
    {  
        ptr_first = NULL;  
        ptr_last = NULL;  
    }  
    else  
    {  
        ptr_last = ptr_last->prev;  
        ptr_last->next = NULL;  
    }  
    delete temp;  
}  
  
void delete_target(int targetValue)  
{  
    Node *current = ptr_first;  
    while (current != NULL && current->data != targetValue)  
    {  
        current = current->next;  
    }
```

```

    }

    if (current != NULL)
    {
        if (current == ptr_first)
        {
            delete_first();
            return;
        }

        current->prev->next = current->next;
        current->next->prev = current->prev;
        delete current;
    }
}

void edit_node(int targetValue,int newValue)
{
    Node *current = ptr_first;
    while (current != NULL && current->data != targetValue)
    {
        current = current->next;
    }

    if (current != NULL)

```



```
{
    current->data = newValue;
}
}

int main()
{
    add_first(10);
    add_first(5);
    add_last(20);
    cout << "Awal\t\t\t: ";
    view();

    delete_first();
    cout << "Setelah delete_first\t: ";
    view();
    delete_last();
    cout << "Setelah delete_last\t: ";
    view();

    add_last(30);
    add_last(40);
    cout << "Setelah tambah\t\t: ";
    view();
```

```

delete_target(30);

cout << "Setelah delete_target\t: ";

view();

return 0;
}

```

Screenshots Output

```

PS C:\Users\USER\OneDrive\Dokumen\Kuliah\Semester 3\Praktikum Struktur Data\modul 6> & 'c:\Users\USER\.vscode\extensions\ms-vscode.cpptools-1.28.3-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-evjlj3ob.y15' '--stdout=Microsoft-MIEngine-Out-1b0bk2b2.54q' '--stderr=Microsoft-MIEngine-Error-mhi4uybk.ktu' '--pid=Microsoft-MIEngine-Pid-czbxgy11.jre' '--dbgExe=C:\Users\USER\mingw32\bin\gdb.exe' '--interpreter=mi'
Awal : 5<->10<->20
Setelah delete_first : 10<->20
Setelah delete_last : 10
Setelah tambah : 10<->30<->40
Setelah delete_target : 10<->40
PS C:\Users\USER\OneDrive\Dokumen\Kuliah\Semester 3\Praktikum Struktur Data\modul 6>

```

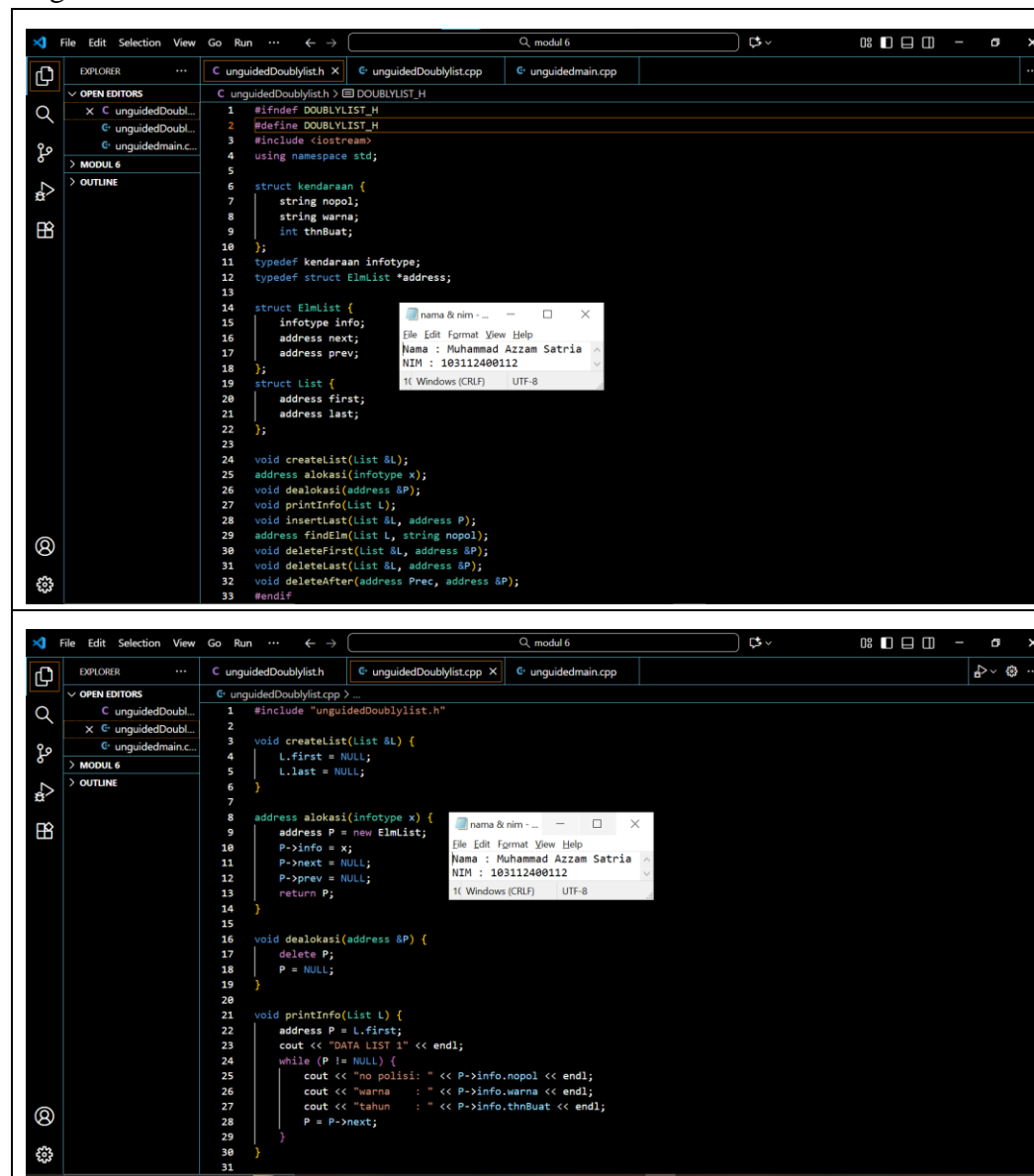
Deskripsi:

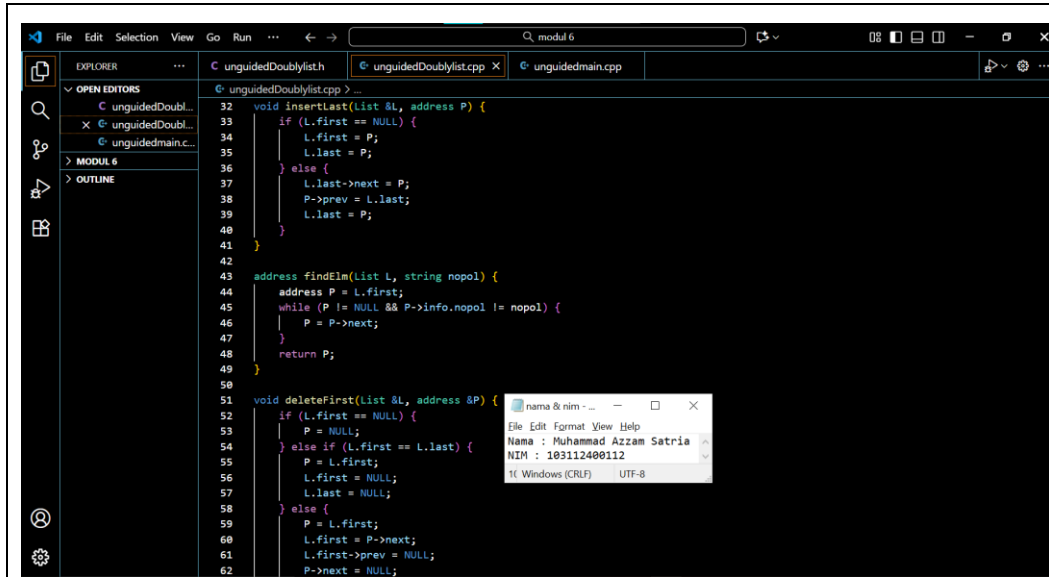
Program ini dibuat untuk mengelola data menggunakan doubly linked list, yaitu struktur data yang setiap elemennya saling terhubung dua arah ke depan dan ke belakang. Setiap elemen (node) memiliki tiga bagian, yaitu isi data, pointer ke node sebelumnya (prev), dan pointer ke node setelahnya (next). Program ini menyimpan alamat node pertama (ptr_first) dan terakhir (ptr_last), sehingga proses penambahan atau penghapusan data bisa dilakukan dari kedua arah.

Beberapa fungsi yang digunakan untuk menjalankan operasi dasar pada list yaitu add_first() dan add_last() untuk menambahkan data di bagian awal atau akhir, add_target() untuk menambahkan data setelah nilai tertentu. Kemudian terdapat fungsi untuk menghapus data seperti delete_first(), delete_last(), dan delete_target(). Selain itu, fungsi edit_node() dipakai untuk mengubah isi data yang sudah ada, dan view() digunakan untuk menampilkan isi list secara berurutan.

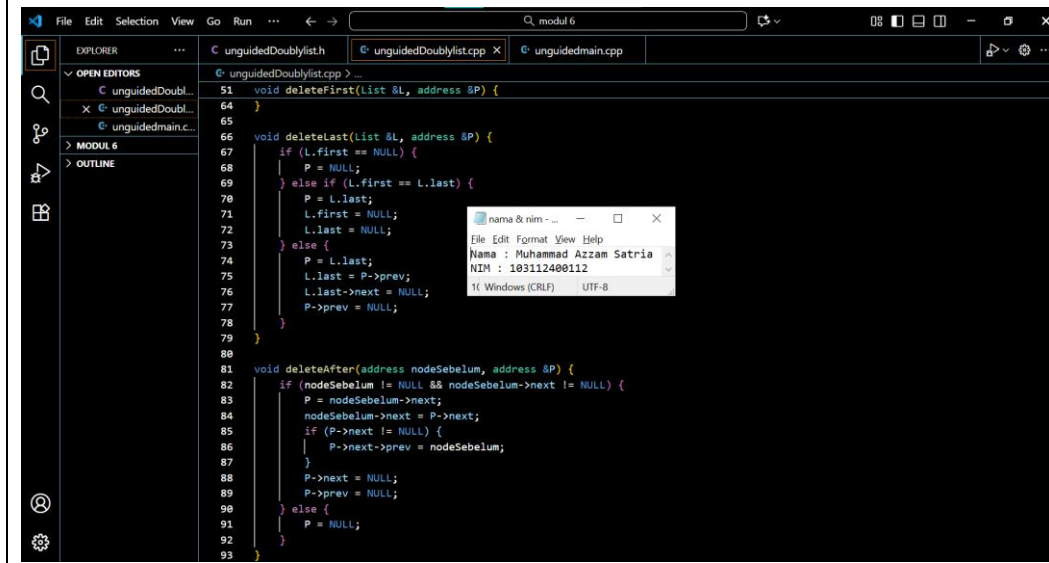
Pada bagian main(), program menjalankan beberapa contoh operasi mulai dari menambah data, menghapus dari depan atau belakang, menambah kembali data baru, hingga menghapus nilai tertentu. Setiap perubahan ditampilkan di layar supaya pengguna dapat melihat bagaimana struktur data berubah langkah demi langkah. Program ini merupakan contoh sederhana cara kerja doubly linked list yang digunakan untuk mengelola data dinamis dalam pemrograman.

Unguided 1

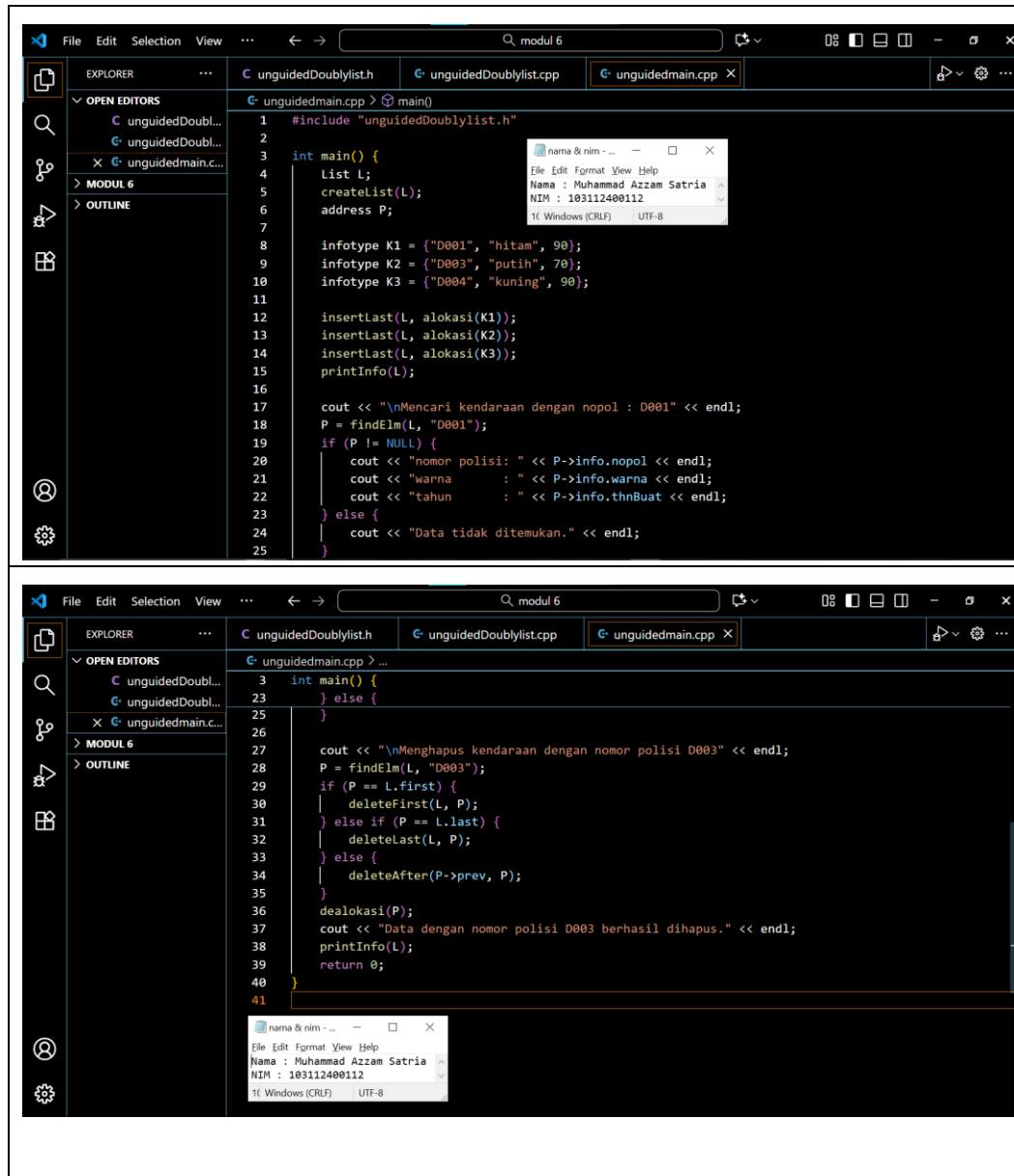




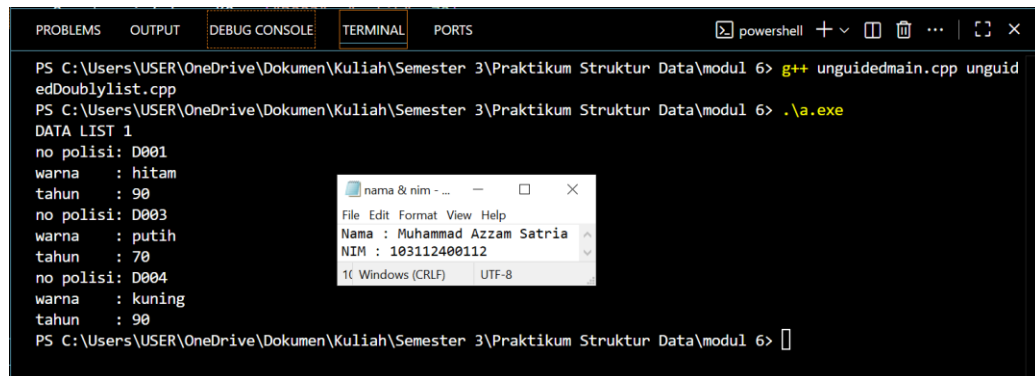
```
File Edit Selection View Go Run ...  
modul 6  
EXPLORER  
C unguidedDoublylist.h  
unguidedDoublylist.cpp  
unguidedmain.cpp  
OPEN EDITORS  
C unguidedDoublylist.h  
C unguidedDoublylist.cpp  
C unguidedmain.cpp  
MODUL 6  
OUTLINE  
32 void insertLast(List &L, address P) {  
33     if (L.first == NULL) {  
34         L.first = P;  
35         L.last = P;  
36     } else {  
37         L.last->next = P;  
38         P->prev = L.last;  
39         L.last = P;  
40     }  
41 }  
42  
43 address findElm(List L, string nopol) {  
44     address P = L.first;  
45     while (P != NULL && P->info.nopol != nopol) {  
46         P = P->next;  
47     }  
48     return P;  
49 }  
50  
51 void deleteFirst(List &L, address &P) {  
52     if (L.first == NULL) {  
53         P = NULL;  
54     } else if (L.first == L.last) {  
55         P = L.first;  
56         L.first = NULL;  
57         L.last = NULL;  
58     } else {  
59         P = L.first;  
60         L.first = P->next;  
61         L.first->prev = NULL;  
62         P->next = NULL;  
nama & nim - ...  
File Edit Format View Help  
Nama : Muhammad Azzam Satria  
NIM : 109112400112  
1( Windows (CRJF) UTF-8
```



```
File Edit Selection View Go Run ...  
modul 6  
EXPLORER  
C unguidedDoublylist.h  
unguidedDoublylist.cpp  
unguidedmain.cpp  
OPEN EDITORS  
C unguidedDoublylist.h  
C unguidedDoublylist.cpp  
C unguidedmain.cpp  
MODUL 6  
OUTLINE  
51 void deleteFirst(List &L, address &P) {  
64 }  
65  
66 void deleteLast(List &L, address &P) {  
67     if (L.first == NULL) {  
68         P = NULL;  
69     } else if (L.first == L.last) {  
70         P = L.last;  
71         L.first = NULL;  
72         L.last = NULL;  
73     } else {  
74         P = L.last;  
75         L.last = P->prev;  
76         L.last->next = NULL;  
77         P->prev = NULL;  
78     }  
79 }  
80  
81 void deleteAfter(address nodeSebelum, address &P) {  
82     if (nodeSebelum != NULL && nodeSebelum->next != NULL) {  
83         P = nodeSebelum->next;  
84         nodeSebelum->next = P->next;  
85         if (P->next != NULL) {  
86             P->next->prev = nodeSebelum;  
87         }  
88         P->next = NULL;  
89         P->prev = NULL;  
90     } else {  
91         P = NULL;  
92     }  
93 }
```



Screenshots Output kasus kendaraan



```
PS C:\Users\USER\OneDrive\Dokumen\Kuliah\Semester 3\Praktikum Struktur Data\modul 6> g++ unguidedmain.cpp unguidedDoublylist.cpp
PS C:\Users\USER\OneDrive\Dokumen\Kuliah\Semester 3\Praktikum Struktur Data\modul 6> .\a.exe
DATA LIST 1
no polisi: D001
warna : hitam
tahun : 90
no polisi: D003
warna : putih
tahun : 70
no polisi: D004
warna : kuning
tahun : 90
PS C:\Users\USER\OneDrive\Dokumen\Kuliah\Semester 3\Praktikum Struktur Data\modul 6>
```

nama & nim - ...

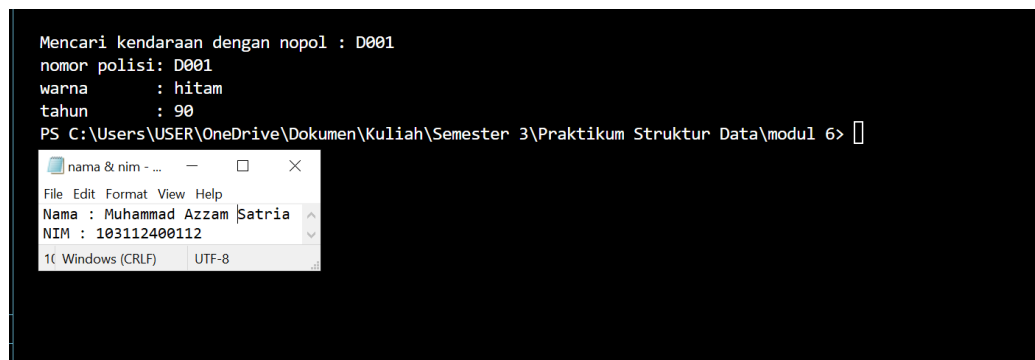
File Edit Format View Help

Nama : Muhammad Azzam Satria

NIM : 103112400112

1(Windows (CRLF) UTF-8

Screenshots Output mencari nomor polisi



```
Mencari kendaraan dengan nopol : D001
nomor polisi: D001
warna : hitam
tahun : 90
PS C:\Users\USER\OneDrive\Dokumen\Kuliah\Semester 3\Praktikum Struktur Data\modul 6>
```

nama & nim - ...

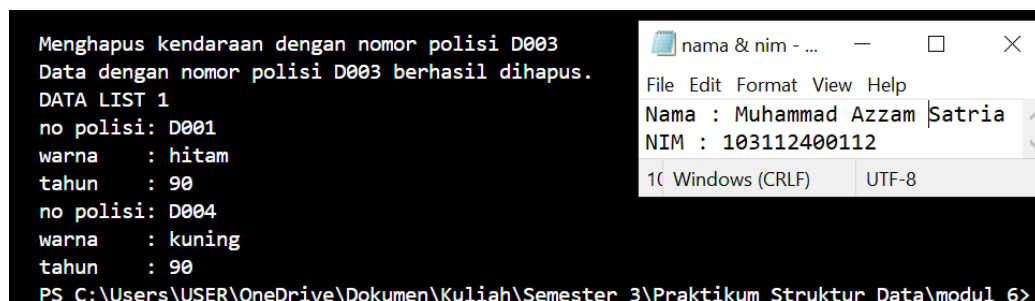
File Edit Format View Help

Nama : Muhammad Azzam Satria

NIM : 103112400112

1(Windows (CRLF) UTF-8

Screenshots Output menghapus data nomor polisi



```
Menghapus kendaraan dengan nomor polisi D003
Data dengan nomor polisi D003 berhasil dihapus.
DATA LIST 1
no polisi: D001
warna : hitam
tahun : 90
no polisi: D004
warna : kuning
tahun : 90
PS C:\Users\USER\OneDrive\Dokumen\Kuliah\Semester 3\Praktikum Struktur Data\modul 6>
```

nama & nim - ...

File Edit Format View Help

Nama : Muhammad Azzam Satria

NIM : 103112400112

1(Windows (CRLF) UTF-8

Deskripsi:

Struktur program ini terbagi ke dalam tiga file berbeda yaitu unguideDoublylist.h, unguideDoublylist.cpp, dan unguidedmain.cpp. File pertama berfungsi untuk mendefinisikan struktur data yang digunakan dalam program, yaitu node atau simpul yang menyimpan data kendaraan berupa nomor polisi, warna dan tahun pembuatan. Setiap node pada program memiliki dua

penunjuk yaitu next dan prev untuk menghubungkan data sebelumnya dan setelahnya. Selain itu, pada file ini dideklarasikan berbagai fungsi yang digunakan untuk mengelola list, seperti membuat list baru, menambah data di akhir list, mencari data tertentu dan menghapus data tertentu.

File kedua berisi implementasi dari fungsi-fungsi yang telah dideklarasikan pada file pertama. Contohnya yaitu fungsi `insertLast()` digunakan untuk menambahkan data baru di bagian akhir list dengan menyesuaikan posisi pointer, `findElm()` berfungsi mencari node berdasarkan nomor polisi yang diinput. Prosedur `deleteAfter()` untuk menghapus node yang terletak setelah node tertentu dengan memperbarui pointer agar hubungan antar elemen tetap konsisten. Selain itu, setiap node yang sudah dihapus akan didealokasi dari memori untuk mencegah terjadinya kebocoran memori dan menjaga efisiensi program.

File ketiga merupakan bagian utama yang menjalankan seluruh program. Pada bagian ini, program terlebih dahulu membuat list kosong, kemudian menambahkan beberapa data kendaraan menggunakan fungsi `insertLast()`. Setelah semua data ditambahkan, program menampilkan isi list melalui fungsi `printInfo()`, lalu melakukan pencarian data dengan `findElm()` dan penghapusan data tertentu dengan `deleteAfter()`. Setiap perubahan yang terjadi pada list akan langsung ditampilkan di output mulai dari penambahan, pencarian hingga penghapusan data, sehingga hasil akhir program berupa kondisi list yang telah diperbarui sesuai urutan eksekusi kode program.

C. Kesimpulan

Dari hasil latihan pada program Doubly Linked List ini, dapat disimpulkan bahwa setiap data pada list saling terhubung dua arah melalui pointer next dan prev. Hal ini membuat setiap elemen di dalam list bisa diakses dari depan maupun dari belakang dengan mudah. Program juga memperlihatkan bagaimana proses penambahan, pencarian dan penghapusan data dilakukan dengan tetap terjaganya hubungan antar elemen. Setiap operasi tersebut bekerja secara teratur dikarenakan penggunaan pointer yang tepat, sehingga perubahan pada satu

elemen tidak memengaruhi elemen lainnya secara langsung.

Selain itu, program ini menunjukkan bahwa pengelolaan memori sangat penting dalam struktur data dinamis. Setiap kali data dihapus, node yang dihapus harus didealokasi agar memori yang digunakan bisa dikembalikan. Hal ini menjaga program tetap efisien dan bebas dari kebocoran memori. Kode dalam program ini bekerja dengan mengatur urutan proses mulai dari membuat list kosong, menambah data kendaraan, mencari data tertentu, menghapus data berdasarkan posisi hingga menampilkan hasil akhir list yang telah diperbarui sesuai urutan yang ada pada program.

D. Referensi

Khoerul, A. (2023). *Pengantar struktur data full*.

Lohmann, S., & Tutsch, D. (2023). *The doubly linked tree of singly linked rings: Providing hard real-time database operations on an FPGA*. *Computers*, 13(1), 8.