LAPORAN PRAKTIKUM

Modul 6

“DOUBLE LINKED LIST (BAGIAN PERTAMA)”

Disusun Oleh:

Rifqi Mohamad Ramdani 2311104044

Kelas

SE-07-02

Dosen :

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO 2024

1. Tujuan

1. Memahami konsep modul linked list.

2. Mengaplikasikan konsep double linked list dengan menggunakan pointer dan dengan bahasa C

2. Landasan Teori

Double Linked List adalah jenis linked list di mana setiap elemen memiliki dua pointer: satu ke elemen sebelumnya (prev) dan satu ke elemen berikutnya (next). Ini memungkinkan traversal dua arah, baik maju maupun mundur, sehingga lebih fleksibel dibandingkan dengan single linked list.

Komponen dalam Double Linked List:

First: Pointer ke elemen pertama dalam list.

Last: Pointer ke elemen terakhir dalam list.

Next: Pointer ke elemen setelahnya.

Prev: Pointer ke elemen sebelumnya.

Operasi Utama:

Insert First: Tambah elemen di awal.

Insert After: Tambah elemen setelah elemen tertentu.

Insert Before: Tambah elemen sebelum elemen tertentu.

Delete First: Hapus elemen pertama.

Delete Last: Hapus elemen terakhir.

Delete After: Hapus elemen setelah elemen tertentu.

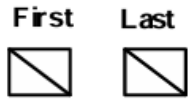
Delete Before: Hapus elemen sebelum elemen tertentu.

3. Guided

Double Linked List

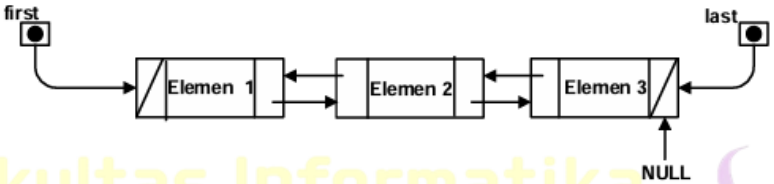
Double Linked list adalah linked list yang masing – masing elemen nya memiliki 2 successor, yaitu successor yang menunjuk pada elemen sebelumnya (prev) dan successor yang menunjuk pada elemen sesudahnya (next).

Gambar berikut menunjukan bentuk Double Linked list dengan elemen kosong:



Gambar 6-1 *Double Linked list* dengan Elemen Kosong

Gambar berikut menunjukan bentuk *Double Linked list* dengan 3 elemen:



Double linked list juga menggunakan dua buah successor utama yang terdapat pada list, yaitu first (successor yang menunjuk elemen pertama) dan last (susesor yang menunjuk elemen terakhir list).

Komponen-komponen dalam double linked list:

1. First : pointer pada list yang menunjuk pada elemen pertama list.

2. Last : pointer pada list yang menunjuk pada elemen terakhir list.

3. Next : pointer pada elemen sebagai successor yang menunjuk pada elemen didepannya.

4. Prev : pointer pada elemen sebagai successor yang menunjuk pada elemen dibelakangnya.

Contoh pendeklarasian struktur data untuk *double linked list*:

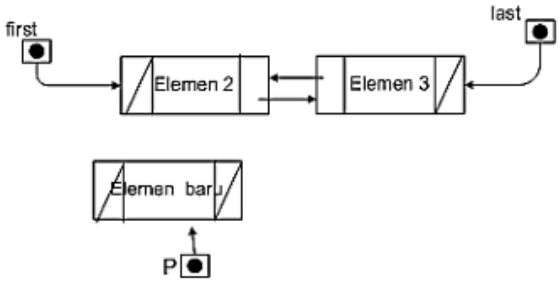
```
1   #ifndef doublelist_H
2   #define doublelist_H
3   #include "boolean.h"
4   #define Nil NULL
5   #define info(P) (P)->info
6   #define next(P) (P)->next
7   #define prev(P) (P)->prev
8   #define first(L) ((L).first)
9   #define last(L) ((L).last)
10
11  /*deklarasi record dan struktur data double linked list*/
12  typedef int infotype;
13  typedef struct elmlist *address;
14  struct elmlist {
15      infotype info;
16      address next;
17      address prev;
18  };
19
```

```
20  /* definisi list: */
21  /* list kosong jika First(L)=Nil */
22  struct list{
23       address first;
24       address last;
25  };
26  #endif
```

Insert A.

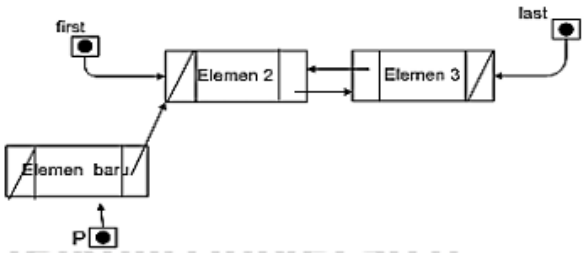
Insert First

Langkah-langkah dalam proses insert first:



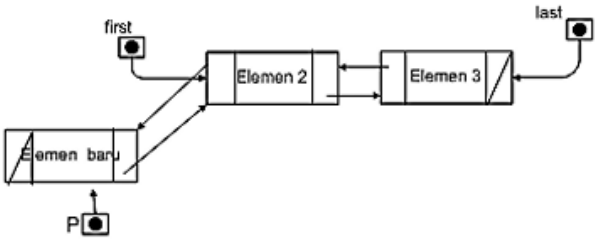

```
P = alokasi(X);
next(P) = Nil;
prev(P) = Nil;
```

Gambar 6-3 *Double Linked list Insert First* 1



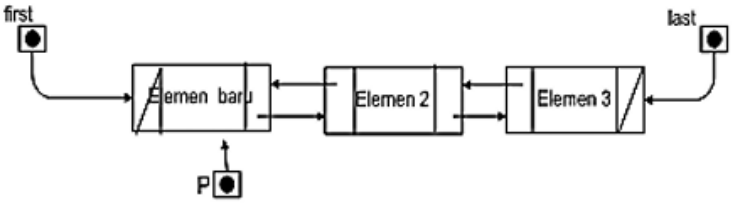

```
next(P) = first(L);
```

Gambar 6-4 *Double Linked list Insert First* 2




```
prev(first(L)) = P;
```

Gambar 6-5 *Double Linked list Insert First* 3



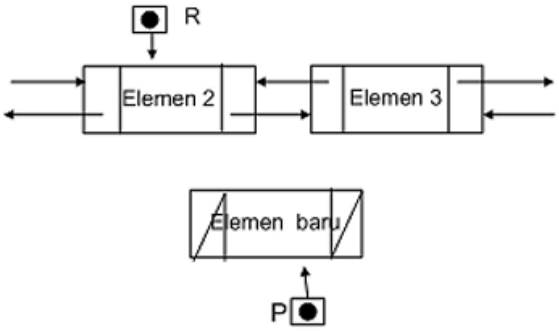

```
first(L) = P;
```

Gambar 6-6 *Double Linked list Insert First* 4

```
void insertFirst(list &L, address &P){
    next(P) = first(L);
    prev(first(L)) = P;
    first(L) = P;
}
```

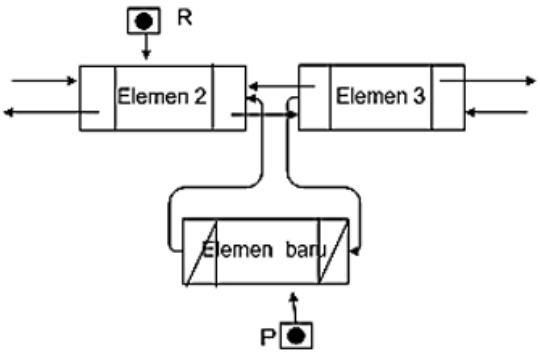
Insert After

Langkah-langkah dalam proses insert after:



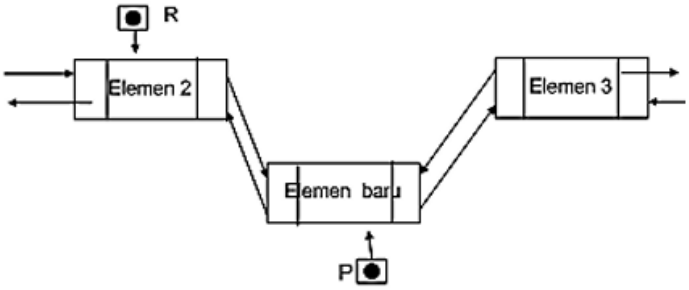

```
P = alokasi(X);
next(P) = Nil;
prev(P) = Nil;
```

Gambar 6-11 *Double Linked list Insert After* 1




```
next(P) = next(R);
prev(P) = R;
```

Gambar 6-12 *Double Linked list Insert After* 2




```
prev(next(R)) = P;
prev(R) = P;
```

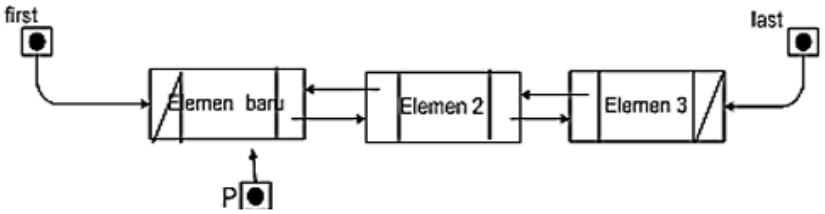
Gambar 6-13 *Double Linked list Insert After* 3

Insert Before

Diatas hanya dijelaskan tentang insert after. Insert before hanya kebalikan dari insert after. Perbedaan Insert After dan Insert Before terletak pada pencarian elemennya.

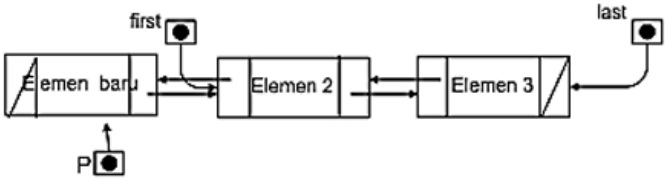
Delete A.

Delete First Langkah-langkah dalam proses delete first:



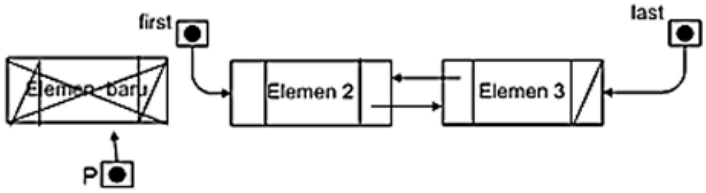

```
P = first(L);
```

Gambar 6-14 *Double Linked list Delete First* 1




```
first(L) = next(first(L));
next(P) = Nil;
```

Gambar 6-15 *Double Linked list Delete First* 2



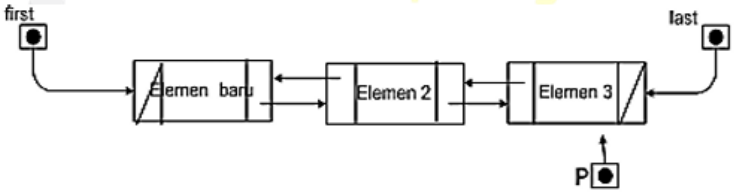

```
prev(first(L)) = Nil;
next(P) = Nil;
return P;
          atau
dealokasi(P);
```

Gambar 6-16 *Double Linked list Delete First* 3

```
/* contoh sintak delet first */
void deleteFirst(list &L, address &P){
    P = first(L);
    first(L) = next(first(L));
    prev (P) = null;
    prev(first(L)) = null;
    next(P) = null;
}
```

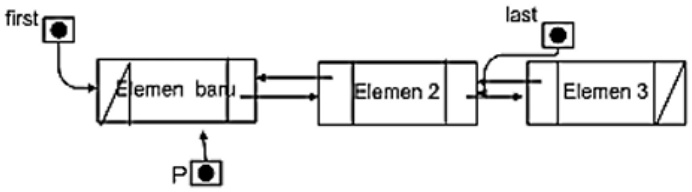
B. *Delete Last*

Langkah-langkah dalam proses *delete last*:



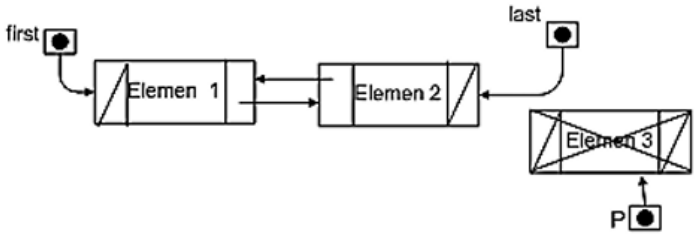

```
P = last(L);
```

Gambar 6-17 *Double Linked list Delete Last* 1




```
last(L) = prev(last(L));
```

Gambar 6-18 *Double Linked list Delete Last* 2

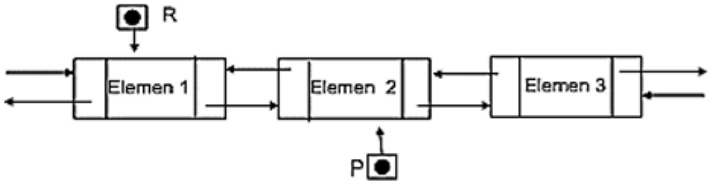



```
prev(P) = Nil;
next(last(L)) = Nil;
return P;
        atau
dealokasi(P);
```

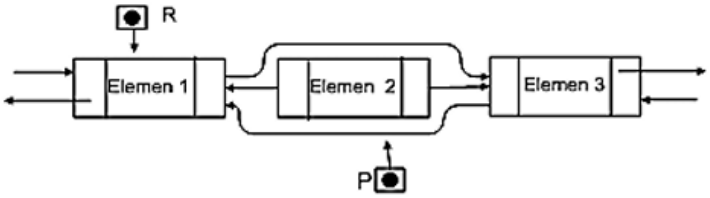
Gambar 6-19 *Double Linked list Delete Last* 3

C. *Delete After*

Langkah-langkah dalam proses *delete after*:

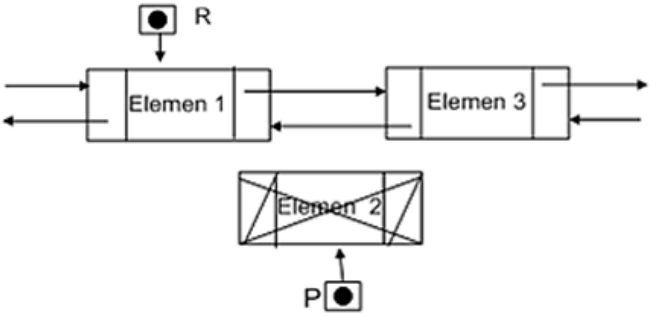


Gambar 6-20 *Double Linked list Delete After* 1




```
next(R) = next(P);
prev(next(P)) = R;
```

Gambar 6-21 *Double Linked list Delete After* 2




```
prev(P) = Nil;
next(P) = Nil;
return P;
        atau
dealokasi(P);
```

Gambar 6-22 *Double Linked list Delete After* 3

Delete Before

Diatas hanya dijelaskan tentang delete after. Delete before hanya kebalikan dari delete after. Perbedaan Delete After dan Delete Before terletak pada pencarian elemennya.

Update, View, dan Searching Proses pencarian, update data dan view data pada dasarnya sama dengan proses pada single linked list. Hanya saja pada double linked list lebih mudah dalam melakukan proses akses elemen, karena bisa melakukan iterasi maju dan mundur. Seperti halnya single linked list, double linked list juga mempunyai ADT yang pada dasarnya sama dengan ADT yang ada pada single linked list.

```c
/*file : doublelist .h*/
/* contoh ADT list berkait dengan representasi fisik pointer*/
/* representasi address    dengan pointer*/
/* info tipe adalah integer */
#ifndef doublelist_H
#define doublelist_H

#include <stdio.h>
#define Nil NULL
#define info(P) (P)->info
#define next(P) (P)->next
#define prev(P) (P)->prev
#define first(L) ((L).first)
#define last(L) ((L).last)

typedef int infotype;
typedef struct elmlist *address;
/* pendefinisian tipe data bentukan elemen list
   dengan dua successor, yaitu next dan prev */
struct elmlist{
    infotype info;
    address prev;
    address next;
};

/* definisi double linked list : list kosong jika first(L)=Nil
   setiap elemen address P dapat diacu info(P) atau next(P)
   elemen terakhir adalah last
   nama tipe list yang dipakai adalah 'list', sama dengan pada singe list*/
struct list {
    address first,last;
};

/** Deklarasi fungsi primitif lain **/
/** Sama dengan Single Linked list **/
```


LATIHAN GUIDED

main.cpp

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

class Node {
public:
    int data;
    Node* prev;
    Node* next;
};

class DoublyLinkedList {
public:
    Node* head;
    Node* tail;

    // Constructor untuk inisialisasi head dan tail
    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    // Fungsi untuk menambahkan elemen di depan list
    void insert(int data) {
        Node* newNode = new Node;
        newNode->data = data;
        newNode->prev = nullptr;
        newNode->next = head;

        if (head != nullptr) {
            head->prev = newNode;
        } else {
            tail = newNode; // Jika list kosong, tail juga mengarah ke node baru
        }
        head = newNode;
    }

    // Fungsi untuk menghapus elemen dari depan list
    void deleteNode() {
        if (head == nullptr) {
            return; // Jika list kosong
        }
        Node* temp = head;
        head = head->next;
        if (head != nullptr) {
```

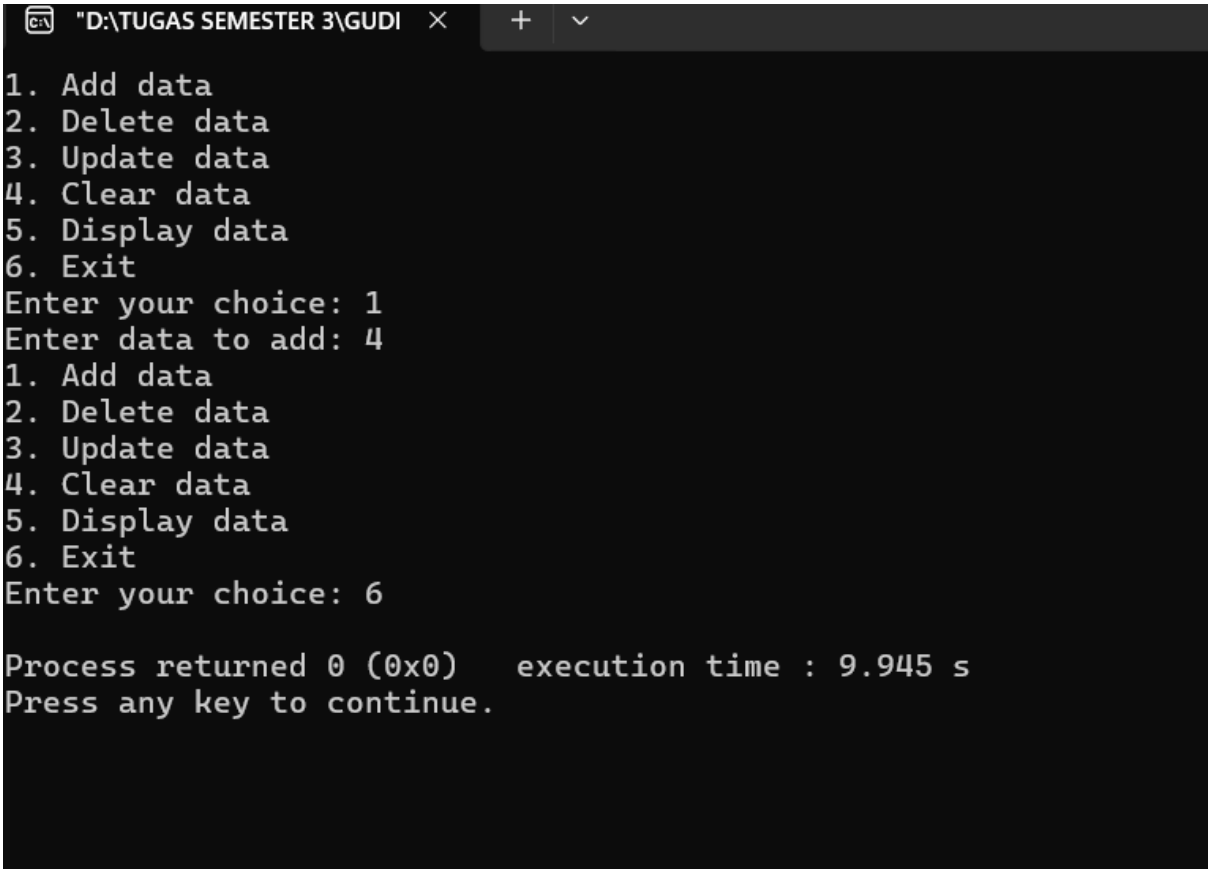
main.cpp

```
43          head = head->next;
44          if (head != nullptr) {
45              head->prev = nullptr;
46          } else {
47              tail = nullptr; // Jika hanya satu elemen di list
48          }
49          delete temp; // Hapus elemen
50      }
51
52      // Fungsi untuk mengupdate data di list
53      bool update(int oldData, int newData) {
54          Node* current = head;
55          while (current != nullptr) {
56              if (current->data == oldData) {
57                  current->data = newData;
58                  return true; // Jika data ditemukan dan diupdate
59              }
60              current = current->next;
61          }
62          return false; // Jika data tidak ditemukan
63      }
64
65      // Fungsi untuk menghapus semua elemen di list
66      void deleteAll() {
67          Node* current = head;
68          while (current != nullptr) {
69              Node* temp = current;
70              current = current->next;
71              delete temp;
72          }
73          head = nullptr;
74          tail = nullptr;
75      }
76
77      // Fungsi untuk menampilkan semua elemen di list
78      void display() {
79          Node* current = head;
80          while (current != nullptr) {
81              cout << current->data << " ";
82              current = current->next;
83          }
84          cout << endl;
85      }
86  };
```

main.cpp

```
};

int main() {
    DoublyLinkedList list;
    while (true) {
        cout << "1. Add data" << endl;
        cout << "2. Delete data" << endl;
        cout << "3. Update data" << endl;
        cout << "4. Clear data" << endl;
        cout << "5. Display data" << endl;
        cout << "6. Exit" << endl;

        int choice;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1: {
                int data;
                cout << "Enter data to add: ";
                cin >> data;
                list.insert(data);
                break;
            }
            case 2: {
                list.deleteNode();
                break;
            }
            case 3: {
                int oldData, newData;
                cout << "Enter old data: ";
                cin >> oldData;
                cout << "Enter new data: ";
                cin >> newData;
                bool updated = list.update(oldData, newData);
                if (!updated) {
                    cout << "Data not found" << endl;
                }
                break;
            }
            case 4: {
                list.deleteAll();
                break;
            }
            }
            case 5: {
                list.display();
                break;
            }
            case 6: {
                return 0;
            }
            default: {
                cout << "Invalid choice" << endl;
                break;
            }
        }
    }
    return 0;
}
```

Maka Akan Menghasilkan Output



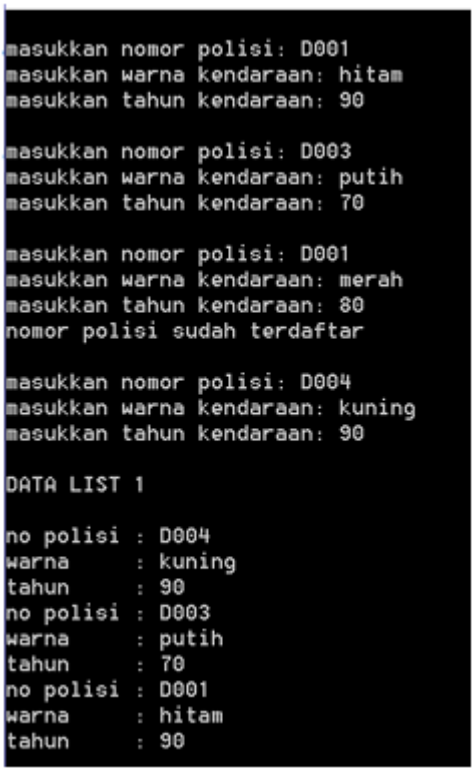
4. Unguided

1. Buatlah ADT *Double Linked list* sebagai berikut di dalam *file* **"doublelist.h"**:

```
Type infotype : kendaraan <
    nopol : string
    warna : string
    thnBuat : integer
>
Type address : pointer to ElmList
Type ElmList <
    info : infotype
    next :address
    prev : address
>

Type List <
    First : address
    Last : address
>
prosedur CreateList( in/out L : List )
fungsi alokasi( x : infotype ) : address
prosedur dealokasi( in/out P : address )
prosedur printInfo( in L : List )
prosedur insertLast( in/out L : List, in P : address )
```

Buatlah implementasi ADT *Double Linked list* pada *file* "doublelist.cpp" dan coba hasil implementasi ADT pada *file* "main.cpp".

Contoh *Output* :



JAWAB:

Kode Program

Langkah pertama membuat project nama LinkedListKendaraan

Lalu menambahkan file header

Dan add file

Pilih Empty File untuk membuat file kosong, beri nama

doublelist.h

doublelist.h | doublelist.cpp | main.cpp

```cpp
#ifndef DOUBLELIST_H
#define DOUBLELIST_H

#include <string>

struct infotype {
    std::string nopol;
    std::string warna;
    int thnBuat;
};

struct ElmList {
    infotype info;
    ElmList* next;
    ElmList* prev;
};

using address = ElmList*;

struct List {
    address First;
    address Last;
};

void CreateList(List &L);
address alokasi(infotype x);
void dealokasi(address &P);
void printInfo(const List &L);
void insertLast(List &L, address P);

#endif

```

Lalu **doublelist.cpp**

doublelist.h | doublelist.cpp | main.cpp

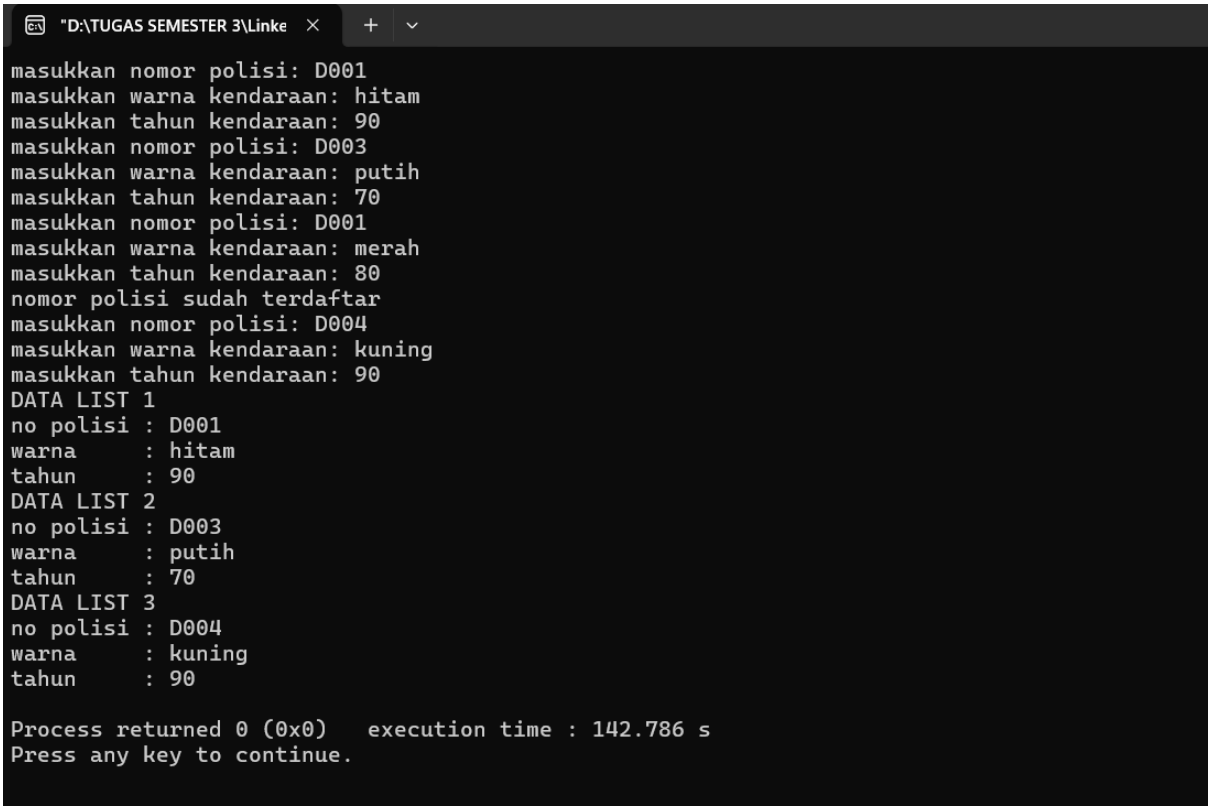
```cpp
#include "doublelist.h"
#include <iostream>

void CreateList(List &L) {
    L.First = nullptr;
    L.Last = nullptr;
}

address alokasi(infotype x) {
    address P = new ElmList;
    P->info = x;
    P->next = nullptr;
    P->prev = nullptr;
    return P;
}

void dealokasi(address &P) {
    delete P;
    P = nullptr;
}

void printInfo(const List &L) {
    address P = L.First;
    int i = 1;
    while (P != nullptr) {
        std::cout << "DATA LIST " << i << std::endl;
        std::cout << "no polisi : " << P->info.nopol << std::endl;
        std::cout << "warna     : " << P->info.warna << std::endl;
        std::cout << "tahun     : " << P->info.thnBuat << std::endl;
        P = P->next;
        i++;
    }
}

void insertLast(List &L, address P) {
    if (L.First == nullptr) {
        L.First = P;
        L.Last = P;
    } else {
        L.Last->next = P;
        P->prev = L.Last;
        L.Last = P;
    }
}

```

Lalu **main.cpp**

doublelist.h | doublelist.cpp | *main.cpp

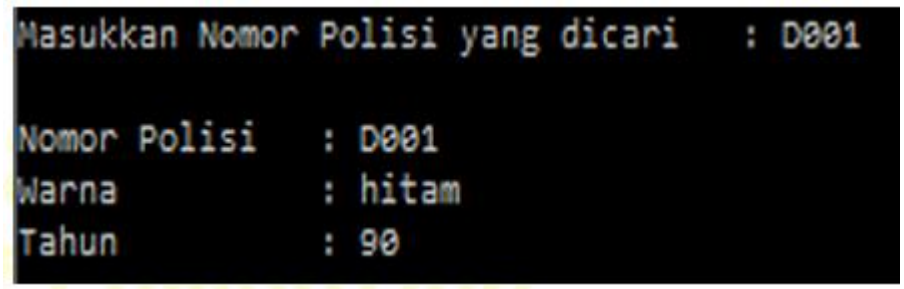
```cpp
#include "doublelist.h"
#include <iostream>

int main() {
    List L;
    CreateList(L);

    infotype x;
    address P;

    for (int i = 0; i < 4; i++) {
        std::cout << "masukkan nomor polisi: ";
        std::cin >> x.nopol;
        std::cout << "masukkan warna kendaraan: ";
        std::cin >> x.warna;
        namespace std {...}tan tahun kendaraan: ";
        std::cin >> x.thnBuat;

        bool found = false;
        address temp = L.First;
        while (temp != nullptr) {
            if (temp->info.nopol == x.nopol) {
                std::cout << "nomor polisi sudah terdaftar" << std::endl;
                found = true;
                break;
            }
            temp = temp->next;
        }
        if (!found) {
            P = alokasi(x);
            insertLast(L, P);
        }
    }

    printInfo(L);

    return 0;
}
```

Maka Akan Menghasilkan Output



2

Carilah elemen dengan nomor polisi D001 dengan membuat fungsi baru. fungsi findElm(L : List, x : infotype) : address



JAWAB

cara bikin fungsi findElm di C++ buat nyari elemen dengan nomor polisi tertentu (misalnya "D001"). Fungsi ini bakalan nerima list (bisa linked list, array, atau vector) sama data yang dicari sebagai inputnya.

Kode Program:

doublelist.h | doublelist.cpp | main.cpp | *main.cpp

```cpp
#include <iostream>
#include <string>

using namespace std;

struct Mobil {
    string nomorPolisi;
    string warna;
    int tahun;
    Mobil* next;
};

Mobil* findElm(Mobil* head, const string& targetNomor) {
    Mobil* current = head;
    while (current != nullptr) {
        if (current->nomorPolisi == targetNomor) {
            return current;
        }
        current = current->next;
    }
    return nullptr;
}

int main() {

    Mobil* mobil1 = new Mobil{"D001", "hitam", 90, nullptr};
    Mobil* mobil2 = new Mobil{"D002", "biru", 85, nullptr};
    Mobil* mobil3 = new Mobil{"D003", "merah", 95, nullptr};

    mobil1->next = mobil2;
    mobil2->next = mobil3;

    string targetNomor;
    cout << "Masukkan Nomor Polisi yang dicari : ";
    cin >> targetNomor;

    Mobil* mobilDitemukan = findElm(mobil1, targetNomor);
    if (mobilDitemukan != nullptr) {
        cout << "Nomor Polisi  : " << mobilDitemukan->nomorPolisi << endl;
        cout << "Warna         : " << mobilDitemukan->warna << endl;
        cout << "Tahun         : " << mobilDitemukan->tahun << endl;
    } else {
        cout << "Nomor Polisi tidak ditemukan." << endl;
    }

    delete mobil1;
    delete mobil2;
    delete mobil3;

    return 0;
}
```

Maka Akan Menghasilkan Output

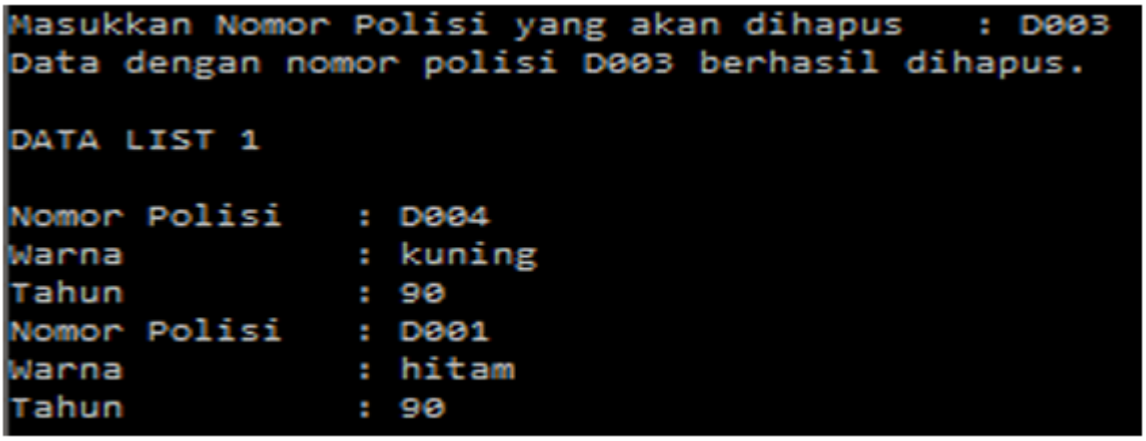
3

Hapus elemen dengan nomor polisi D003 dengan prosedur delete.

- prosedur deleteFirst(in/out L : List, in/out P : address)

- prosedur deleteLast(in/out L : List, in/out P : address)

-prosedur deleteAfter(in Prec : address, in/out: P : address)



JAWAB

Oke, jadi buat hapus elemen dengan nomor polisi tertentu (misalnya "D003") di linked list C++, kita bisa bikin tiga fungsi sederhana:

deleteFirst – Buat hapus elemen pertama di list.

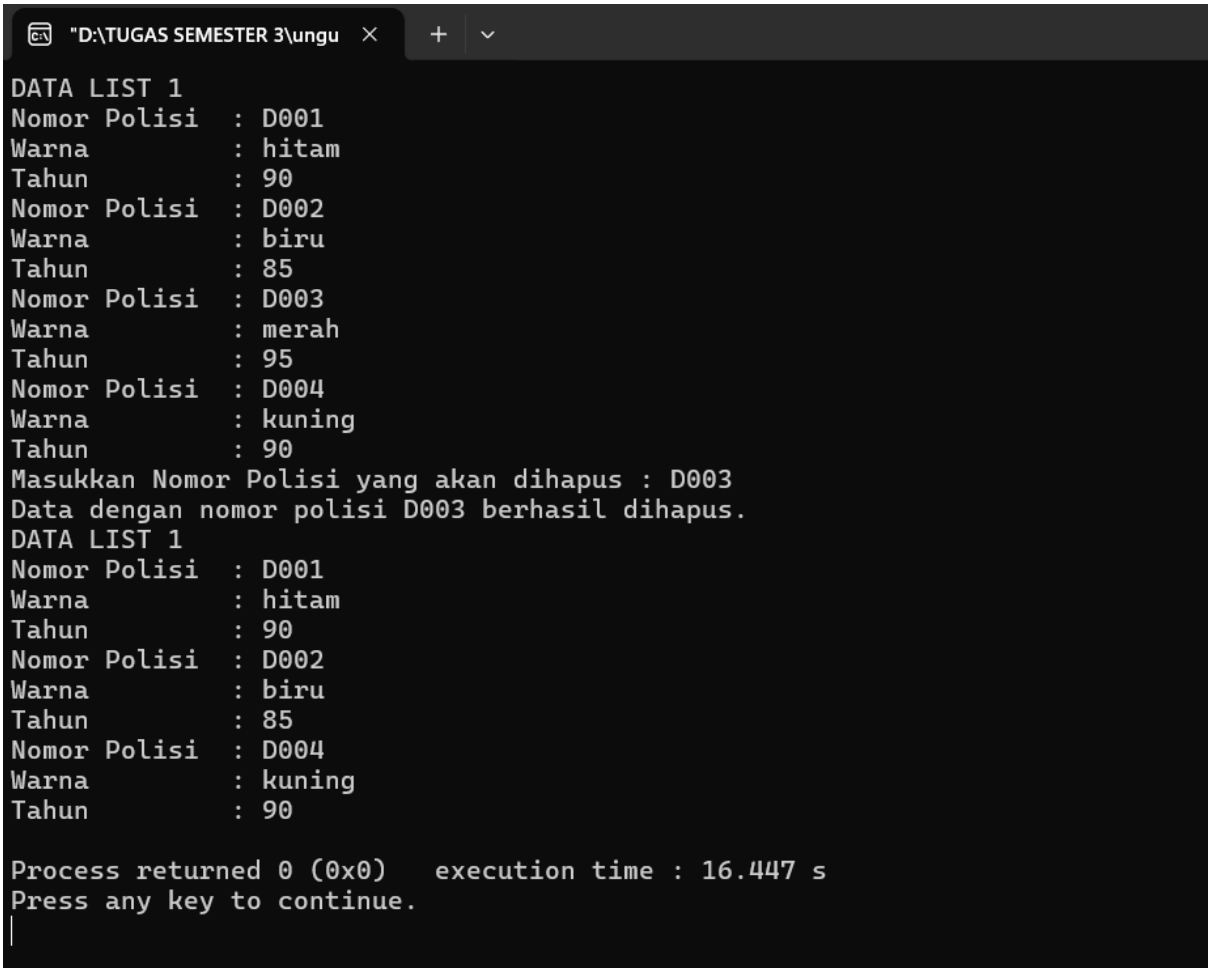
deleteLast – Buat hapus elemen terakhir di list.

deleteAfter – Buat hapus elemen setelah elemen tertentu

doublelist.h | doublelist.cpp | main.cpp | main.cpp | *main.cpp

```cpp
#include <iostream>
#include <string>

using namespace std;

struct Mobil {
    string nomorPolisi;
    string warna;
    int tahun;
    Mobil* next;
};

void deleteFirst(Mobil*& head, Mobil*& p) {
    if (head != nullptr) {
        p = head;
        head = head->next;
        delete p;
    }
}

void deleteLast(Mobil*& head, Mobil*& p) {
    if (head == nullptr) return;

    if (head->next == nullptr) {
        p = head;
        head = nullptr;
        delete p;
    } else {
        Mobil* temp = head;
        while (temp->next->next != nullptr) {
            temp = temp->next;
        }
        p = temp->next;
        temp->next = nullptr;
        delete p;
    }
}
```

Maka outputnya



5. Kesimpulan

Pada praktikum ini, kita telah mempelajari cara membuat dan memanipulasi double linked list menggunakan bahasa C/C++. Dengan memahami double linked list, kita bisa lebih mudah melakukan traversal ke dua arah dan mengelola data secara lebih fleksibel dibandingkan single linked list.