

LAPORAN PRAKTIKUM

Modul 6

DAFTAR BERANTAI GANDA (*DOUBLY LINKED LIST*)



Disusun Oleh:

Adhiansyah Muhammad Pradana Farawowan - 2211104038

S1SE-07-02

Asisten Praktikum:

Aldi Putra

Andini Nur Hidayah

Dosen:

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

PROGRAM STUDI S1 REKAYASAN PERANGKAT LUNAK

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM PURWOKERTO

2024

Setiap elemen memiliki tiga properti, yaitu properti *previous* untuk menunjuk sebelumnya, properti nilai data, dan properti *next* untuk menunjuk setelahnya.

C. Bimbingan (*guided*)

Bimbingan hari ini adalah mengimplementasikan sebuah daftar berantai ganda. Operasi-operasi dasar dijelaskan dengan komentar kode.

```
#include <iostream>

// Elemen akan bertipe data *Node* diimplementasikan dalam kelas C++
// *public* digunakan untuk membuka properti yang bisa diakses bebas
class Node
{
public:
    int data;
    Node *prev;
    Node *next;
};

// Daftar akan bertipe data "DLinkedList"
class DLinkedList
{
public:
    Node *head;
    Node *tail;

    // Konstruktor kelas
    DLinkedList()
    {
        head = nullptr;
        tail = nullptr;
    }

    // Tambah elemen ke awal daftar
    void insert_first(int data)
    {
        Node *new_node = new Node;
        new_node->data = data;

        new_node->prev = nullptr;
        new_node->next = head;

        // *else* akan berlaku jika head (baca: list) sama dengan null
        if (head != nullptr)
        {
            head->prev = new_node;
        }
        else
        {
            tail = new_node;
        }
        head = new_node;

        std::cout << "Insert at first successfully performed" << '\n';
    }
}
```

```

// Hapus elemen pertama
void delete_first()
{
    if (head == nullptr)
    {
        return;
    }
    Node *temp = head;
    head = head->next;
    if (head != nullptr)
    {
        head->prev = nullptr;
    }
    else
    {
        tail = nullptr; // Jika hanya ada satu elemen di daftar
    }

    delete temp;
    std::cout << "Delete at first successfully performed" << '\n';
}

// Perbarui data
bool update(int old_data, int new_data)
{
    Node *current = head;
    while (current != nullptr)
    {
        if (current->data == old_data)
        {
            current->data = new_data;
            return true; // Jika ditemukan
        }
        current = current->next;
    }
    return false; // Jika tidak ditemukan
}

// Hapus semua elemen
void delete_list()
{
    Node *current = head;
    while (current != nullptr)
    {
        Node *temp = current;
        current = current->next;
        delete temp;
    }
    head = nullptr;
    tail = nullptr;

    std::cout << "List has been deleted" << '\n';
}

// Tampilkan semua elemen
void print_list()
{
    Node *current = head;
    while (current != nullptr)
    {
        std::cout << current->data << " ";
        current = current->next;
    }
}

```

```

        std::cout << '\n';
    }
};

int main()
{
    DLinkedList list;

    // Akan membuat program terus berjalan sampai pilihan "6" dimasukkan
    while (true)
    {
        std::cout << "1. Add data" << '\n';
        std::cout << "2. Delete data" << '\n';
        std::cout << "3. Update data" << '\n';
        std::cout << "4. Delete list" << '\n';
        std::cout << "5. Print list" << '\n';
        std::cout << "6. Exit" << '\n'; // Kelur

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

        switch (choice)
        {
            case 1:
            {
                int data;
                std::cout << "Enter data to add: ";
                std::cin >> data;
                list.insert_first(data);
                std::cout << '\n';
                break;
            }
            case 2:
            {
                list.delete_first();
                std::cout << '\n';
                break;
            }
            case 3:
            {
                int old_data, new_data;
                std::cout << "Enter old data: ";
                std::cin >> old_data;
                std::cout << "Enter new data: ";
                std::cin >> new_data;
                bool updated = list.update(old_data, new_data);
                if (!updated)
                {
                    std::cout << "Data not found" << '\n';
                } else {
                    std::cout << "Data successfully updated" << '\n';
                }

                std::cout << '\n';
                break;
            }
            case 4:
            {
                list.delete_list();
                std::cout << '\n';
                break;
            }
        }
    }
}

```

```

        case 5:
        {
            list.print_list();
            std::cout << '\n';
            break;
        }
        case 6:
        {
            return 0;
        }
        default:
        {
            std::cout << "Invalid choice" << '\n';
            std::cout << '\n';
            break;
        }
    }

    return 0;
}

```

Output

```

>a.exe
1. Add data
2. Delete data
3. Update data
4. Delete list
5. Print list
6. Exit
Enter your choice: 1
Enter data to add: 269
Insert at first successfully performed

1. Add data
2. Delete data
3. Update data
4. Delete list
5. Print list
6. Exit
Enter your choice: 1
Enter data to add: 20051
Insert at first successfully performed

1. Add data
2. Delete data
3. Update data
4. Delete list
5. Print list
6. Exit
Enter your choice: 1
Enter data to add: 9001481
Insert at first successfully performed

1. Add data
2. Delete data
3. Update data
4. Delete list
5. Print list
6. Exit
Enter your choice: 3
Enter old data: 269
Enter new data: 2
Data successfully updated

1. Add data
2. Delete data
3. Update data
4. Delete list
5. Print list
6. Exit
Enter your choice: 5
9001481 20051 2

1. Add data
2. Delete data
3. Update data
4. Delete list
5. Print list
6. Exit
Enter your choice:

```

```

1. Add data
2. Delete data
3. Update data
4. Delete list
5. Print list
6. Exit
Enter your choice: 2
Delete at first successfully performed

1. Add data
2. Delete data
3. Update data
4. Delete list
5. Print list
6. Exit
Enter your choice: 5
20051 2

1. Add data
2. Delete data
3. Update data
4. Delete list
5. Print list
6. Exit
Enter your choice: 4
List has been deleted

```

```

1. Add data
2. Delete data
3. Update data
4. Delete list
5. Print list
6. Exit
Enter your choice: 5

1. Add data
2. Delete data
3. Update data
4. Delete list
5. Print list
6. Exit
Enter your choice: 2

1. Add data
2. Delete data
3. Update data
4. Delete list
5. Print list
6. Exit
Enter your choice: 6

```

D. Tugas mandiri (*unguided*)

- 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:

```

masukkan nomor polisi: D001
masukkan warna kendaraan: hitam
masukkan tahun kendaraan: 90

masukkan nomor polisi: D003
masukkan warna kendaraan: putih
masukkan tahun kendaraan: 70

masukkan nomor polisi: D001
masukkan warna kendaraan: merah
masukkan tahun kendaraan: 80
nomor polisi sudah terdaftar

masukkan nomor polisi: D004
masukkan warna kendaraan: kuning
masukkan tahun kendaraan: 90

DATA LIST 1

no polisi : D004
warna     : kuning
tahun     : 90
no polisi : D003
warna     : putih
tahun     : 70
no polisi : D001
warna     : hitam
tahun     : 90

```

Gambar 6-23 Output kasus kendaraan

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

```

Masukkan Nomor Polisi yang dicari : D001

Nomor Polisi : D001
Warna        : hitam
Tahun        : 90

```

Gambar 6-24 Output mencari nomor polisi

- c. 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)

```

Masukkan Nomor Polisi yang akan dihapus : D003
Data dengan nomor polisi D003 berhasil dihapus.

DATA LIST 1

Nomor Polisi : D004
Warna        : kuning
Tahun        : 90
Nomor Polisi : D001
Warna        : hitam
Tahun        : 90

```

Gambar 6-25 Output menghapus data nomor polisi

Jawaban disatukan dalam tiga berkas sesuai instruksi

doublelist.h

```
#ifndef _DOUBLELIST_H
#define _DOUBLELIST_H

#include <iostream>

struct Kendaraan
{
    std::string nopol;
    std::string warna;
    int tahun_buat;
};

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

struct TwoWayList
{
    ElmList *first;
    ElmList *last;
};

void create_list(TwoWayList &L);
ElmList *alloc_elm(Kendaraan x);
void dealloc_elm(ElmList *&P);
void print_info(TwoWayList &L);
void insert_last(TwoWayList &L, ElmList *&P);

ElmList *find_elm(TwoWayList &L, Kendaraan &x);

void delete_first(TwoWayList &L, ElmList *&P);
void delete_last(TwoWayList &L, ElmList *&P);
void delete_after(ElmList *&Prec, ElmList *&P);

#endif
```

doublelist.cpp

```
#include "doublelist.h"

void create_list(TwoWayList &L)
{
    L.first = nullptr;
    L.last = nullptr;
}

ElmList *alloc_elm(Kendaraan x)
{
    ElmList *new_elm = new ElmList;
    new_elm->info = x;
    new_elm->next = nullptr;
    new_elm->prev = nullptr;

    return new_elm;
}
```

```

void dealloc_elm(ElmList *&P)
{
    P->next = nullptr;
    P->prev = nullptr;
    delete P;
}

void print_info(TwoWayList &L)
{
    ElmList *current = L.first;
    while (current != nullptr)
    {
        std::cout << "Nomor polisi: " << current->info.nopol << '\n';
        std::cout << "Warna: " << current->info.warna << '\n';
        std::cout << "Tahun kendaraan: " << current->info.tahun_buat << '\n';
        current = current->next;
    }
}

void insert_last(TwoWayList &L, ElmList *&P)
{
    if (L.first == nullptr)
    {
        L.first = P;
        L.last = P;
    }
    else
    {
        L.last->next = P;
        P->prev = L.last;
        L.last = P;
    }
}

ElmList *find_elm(TwoWayList &L, Kendaraan &x)
{
    ElmList *P = L.first;
    while (P != nullptr)
    {
        if (P->info.nopol == x.nopol)
        {
            return P;
        }
        P = P->next;
    }
    return nullptr;
}

void delete_first(TwoWayList &L, ElmList *&P)
{
    if (L.first != nullptr)
    {
        P = L.first;
        L.first = L.first->next;

        if (L.first != nullptr)
        {
            L.first->prev = nullptr;
        }
        else
        {
            L.last = nullptr;
        }
    }
}

```

```

        P->next = nullptr;
        P->prev = nullptr;
    }
    else
    {
        std::cout << "Elemen pertama tidak ada, penghapusan dibatalkan." <<
'\n';
    }
}

void delete_last(TwoWayList &L, ElmList *&P)
{
    if (L.last != nullptr)
    {
        P = L.last;
        L.last = L.last->prev;
        if (L.last != nullptr)
        {
            L.last->next = nullptr;
        }
        else
        {
            L.first = nullptr;
        }
        P->next = nullptr;
        P->prev = nullptr;
    }
    else
    {
        std::cout << "Elemen terakhir tidak ada, penghapusan dibatalkan." <<
'\n';
    }
}

void delete_after(ElmList *&Prec, ElmList *&P)
{
    if (Prec != nullptr && Prec->next != nullptr)
    {
        P = Prec->next;
        Prec->next = P->next;
        if (P->next != nullptr)
        {
            P->next->prev = Prec;
        }
        P->next = nullptr;
        P->prev = nullptr;
    }
    else
    {
        std::cout << "Elemen selanjutnya kosong, penghapusan dibatalkan." <<
std::endl;
    }
}

```

main.cpp

```
#include "doublelist.cpp"

int main()
{
    TwoWayList L;
    create_list(L);
    Kendaraan vehikel;

    for (int i = 0; i < 4; i = i + 1)
    {
        std::cout << "Masukkan nomor polisi\t\t: ";
        std::cin >> vehikel.nopol;

        std::cout << "Masukkan warna kendaraan\t: ";
        std::cin >> vehikel.warna;

        std::cout << "Masukkan tahun kendaraan\t: ";
        std::cin >> vehikel.tahun_buat;

        if (find_elm(L, vehikel) != nullptr)
        {
            std::cout << "Elemen sudah ada. Diabaikan." << '\n';
            continue;
        }

        ElmList *P = alloc_elm(vehikel);
        insert_last(L, P);
        std::cout << '\n';
    }

    std::cout << "Data kendaraan:" << '\n';
    print_info(L);

    std::cout << '\n';

    Kendaraan k_pengguna;
    std::cout << "Masukkan nopol yang ingin dicari: ";
    std::cin >> k_pengguna.nopol;

    ElmList *elm_result = find_elm(L, k_pengguna);
    if (elm_result != nullptr)
    {
        std::cout << "Kendaraan ditemukan." << '\n';
        std::cout << "Nomor polisi: " << elm_result->info.nopol << '\n';
        std::cout << "Warna kendaraan: " << elm_result->info.warna << '\n';
        std::cout << "Tahun kendaraan: " << elm_result->info.tahun_buat <<
'\n';
    }
    else
    {
        std::cout << "Kendaraan dengan nomor polisi " << k_pengguna.nopol <<
" tidak ditemukan." << '\n';
    }
    std::cout << '\n';

    ElmList *firstt;
    delete_first(L, firstt);
    std::cout << "Setelah elemen pertama hilang:" << '\n';

    print_info(L);

    ElmList *lastt;
```

```

delete_last(L, lastt);
std::cout << "Setelah elemen terakhir hilang:" << '\n';

print_info(L);

if (L.first != nullptr)
{
    Elmlist *afterr;
    delete_after(L.first, afterr);
    std::cout << "Setelah hapus:" << '\n';
    print_info(L);
}
Elmlist *P = L.first;
while (P != nullptr)
{
    Elmlist *next = P->next;
    dealloc_elm(P);
    P = next;
}

return 0;
}

```

Output soal (a)

```

>a.exe
Masukkan nomor polisi      : D001
Masukkan warna kendaraan   : Hitam
Masukkan tahun kendaraan    : 90

Masukkan nomor polisi      : D003
Masukkan warna kendaraan   : Putih
Masukkan tahun kendaraan    : 70

Masukkan nomor polisi      : D001
Masukkan warna kendaraan   : Merah
Masukkan tahun kendaraan    : 80
Elemen sudah ada. Diabaikan.
Masukkan nomor polisi      : D004
Masukkan warna kendaraan   : Kuning
Masukkan tahun kendaraan    : 90

```

Output soal (b)

```

Masukkan nopol yang ingin dicari: D001
Kendaraan ditemukan.
Nomor polisi: D001
Warna kendaraan: Hitam
Tahun kendaraan: 90

```

Output soal (c)

```

Setelah elemen pertama hilang:
Nomor polisi: D003
Warna: Putih
Tahun kendaraan: 70
Nomor polisi: D004
Warna: Kuning
Tahun kendaraan: 90
Setelah elemen terakhir hilang:
Nomor polisi: D003
Warna: Putih
Tahun kendaraan: 70
Elemen selanjutnya kosong, penghapusan dibatalkan.
Setelah hapus:
Nomor polisi: D003
Warna: Putih
Tahun kendaraan: 70

```

Kode sumber lengkap tersedia di direktori [UNGUIDED](#)