LAPORAN PRAKTIKUM
Modul 6
Double Linked List

Disusun Oleh:
Jauhar Fajar Zuhair
2311104072
S1SE-07-2

Dosen :
Wahyu Andri Saputra, S.Pd., M.Eng.

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO

2024

Tujuan Praktikum

1. Mengembangkan pemahaman mendalam tentang konsep dan implementasi modul linked list
2. Menerapkan konsep double linked list menggunakan pointer dalam bahasa pemrograman C++

Landasan Teori

Double Linked List merupakan struktur data linier yang terdiri dari sekumpulan elemen dinamis, dimana setiap elemennya memiliki dua buah pointer penghubung (successor). Kedua pointer tersebut berfungsi untuk menunjuk ke elemen sebelumnya (prev) dan elemen sesudahnya (next). Struktur ini memungkinkan traversal data dapat dilakukan dalam dua arah, baik maju maupun mundur.

Komponen utama dalam Double Linked List meliputi:

1. **First** - Pointer yang menunjuk pada elemen pertama dalam list
2. **Last** - Pointer yang menunjuk pada elemen terakhir dalam list
3. **Next** - Pointer pada setiap elemen yang menunjuk ke elemen berikutnya
4. **Prev** - Pointer pada setiap elemen yang menunjuk ke elemen sebelumnya

Karakteristik Double Linked List:

1. Setiap elemen (node) memiliki tiga bagian:
    - Data/Informasi yang disimpan
    - Pointer ke elemen berikutnya (next)
    - Pointer ke elemen sebelumnya (prev)
2. List kosong ditandai dengan First = NULL
3. Elemen terakhir memiliki next = NULL
4. Elemen pertama memiliki prev = NULL

Operasi dasar pada Double Linked List meliputi:

1. **Insert** - Penambahan elemen baru dapat dilakukan di:
    - Awal list (Insert First)
    - Akhir list (Insert Last)
    - Setelah elemen tertentu (Insert After)
    - Sebelum elemen tertentu (Insert Before)
2. **Delete** - Penghapusan elemen dapat dilakukan pada:

- Elemen pertama (Delete First)
- Elemen terakhir (Delete Last)
- Elemen setelah posisi tertentu (Delete After)
- Elemen sebelum posisi tertentu (Delete Before)

3. **Traversal** - Proses penelusuran elemen yang dapat dilakukan dalam dua arah:
    - Maju (menggunakan pointer next)
    - Mundur (menggunakan pointer prev)

Keunggulan Double Linked List dibandingkan Single Linked List adalah kemampuannya untuk melakukan traversal dalam dua arah, yang membuat operasi pencarian dan manipulasi data menjadi lebih fleksibel dan efisien dalam beberapa kasus penggunaan.

Implementasi Double Linked List memerlukan memori yang lebih besar karena setiap node memerlukan tambahan pointer prev, namun memberikan fleksibilitas lebih dalam operasi manipulasi data.

Guided

```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) {
            head->prev = nullptr;
        } else {
            tail = nullptr; // Jika hanya satu elemen di list
        }
        delete temp; // Hapus elemen
    }

    // Fungsi untuk mengupdate data di list
    bool update(int oldData, int newData) {
        Node* current = head;
        while (current != nullptr) {
            if (current->data == oldData) {
                current->data = newData;
                return true; // Jika data ditemukan dan diupdate
            }
            current = current->next;
        }
        return false; // Jika data tidak ditemukan
    }

    // Fungsi untuk menghapus semua elemen di list
    void deleteAll() {
        Node* current = head;
        while (current != nullptr) {
            Node* temp = current;
            current = current->next;
            delete temp;
        }
        head = nullptr;
        tail = nullptr;
    }

    // Fungsi untuk menampilkan semua elemen di list
    void display() {
        Node* current = head;
        while (current != nullptr) {
            cout << current->data << " ";
            current = current->next;
        }
        cout << endl;
    }
};

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;
}
```

1. **Struktur Dasar**:
    - Program menggunakan dua class utama: Node dan DoublyLinkedList
    - Node menyimpan data integer dan dua pointer (prev dan next)
    - DoublyLinkedList mengelola operasi-operasi pada linked list
2. **Class Node**:
    - Memiliki 3 anggota public:
        - data: menyimpan nilai integer
        - prev: pointer ke node sebelumnya
        - next: pointer ke node berikutnya
3. **Class DoublyLinkedList**:
    - Memiliki pointer head dan tail
    - Constructor menginisialisasi head dan tail dengan nullptr
    - Memiliki 5 method utama:
        - insert(): menambah node di awal list
        - deleteNode(): menghapus node dari awal list
        - update(): mengubah nilai data yang ada
        - deleteAll(): menghapus semua node
        - display(): menampilkan semua data
4. **Fungsi-fungsi Utama**:
    - **Insert**:
        - Membuat node baru
        - Mengatur pointer prev dan next
        - Menangani kasus list kosong
        - Menambahkan di depan list
    - **Delete**:
        - Mengecek apakah list kosong
        - Mengatur ulang pointer head
        - Menghapus node pertama
        - Menangani kasus satu elemen
    - **Update**:

- Mencari data yang akan diupdate
- Mengubah nilai jika ditemukan
- Mengembalikan status berhasil/gagal

- **DeleteAll**:
  - Menghapus semua node satu per satu
  - Mereset head dan tail ke nullptr
- **Display**:
  - Menampilkan semua data dari head ke tail

5. **Main Program**:
   - Menggunakan loop while(true) untuk menu interaktif
   - Menyediakan 6 pilihan menu:

a. Tambah data

b. Hapus data

c. Update data

d. Hapus semua data

e. Tampilkan data

f. Keluar program

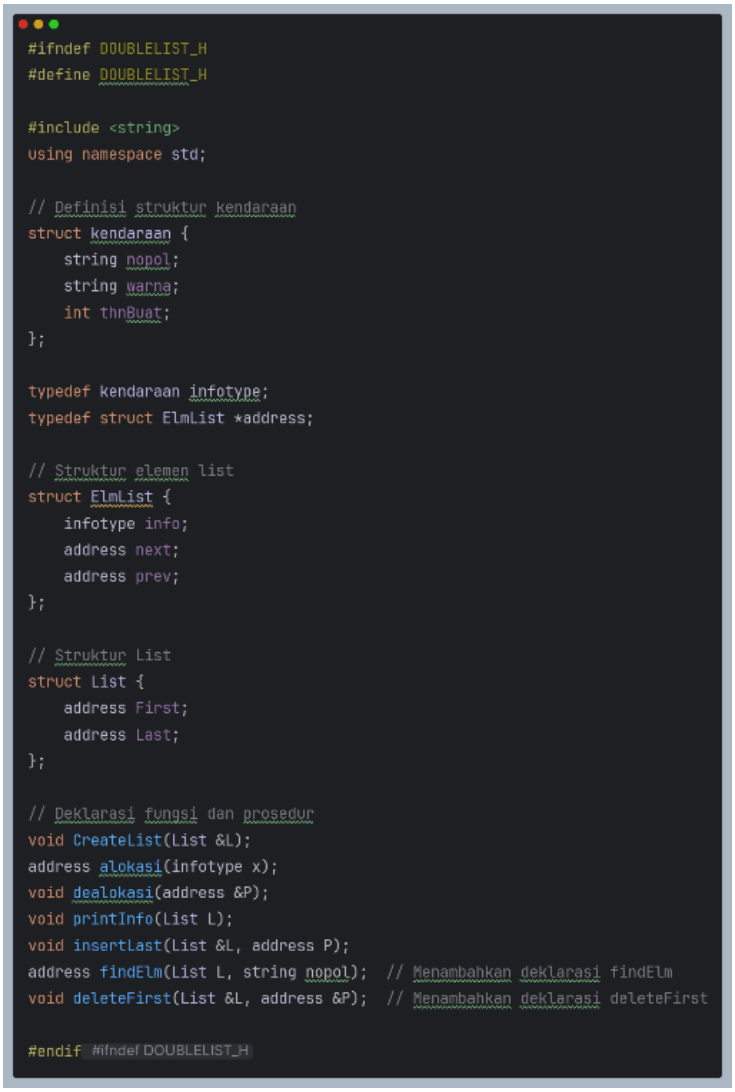
   - Menggunakan switch-case untuk menangani pilihan menu
   - Input/output menggunakan iostream

6. **Fitur Keamanan**:
   - Pengecekan list kosong sebelum operasi
   - Penanganan kasus khusus (list kosong, satu elemen)
   - Pembersihan memori yang tepat saat penghapusan

Program ini merupakan implementasi dasar Double Linked List dengan operasi-operasi standar dan interface pengguna yang interaktif melalui menu console.

unguided

doublelist.h (header)



```
#ifndef DOUBLELIST_H
#define DOUBLELIST_H

#include <string>
using namespace std;

// Definisi struktur kendaraan
struct kendaraan {
    string nopol;
    string warna;
    int thnBuat;
};

typedef kendaraan infotype;
typedef struct ElmList *address;

// Struktur elemen list
struct ElmList {
    infotype info;
    address next;
    address prev;
};

// Struktur List
struct List {
    address First;
    address Last;
};

// Deklarasi fungsi dan prosedur
void CreateList(List &L);
address alokasi(infotype x);
void dealokasi(address &P);
void printInfo(List L);
void insertLast(List &L, address P);
address findElm(List L, string nopol);  // Menambahkan deklarasi findElm
void deleteFirst(List &L, address &P);  // Menambahkan deklarasi deleteFirst

#endif #ifndef DOUBLELIST_H
```

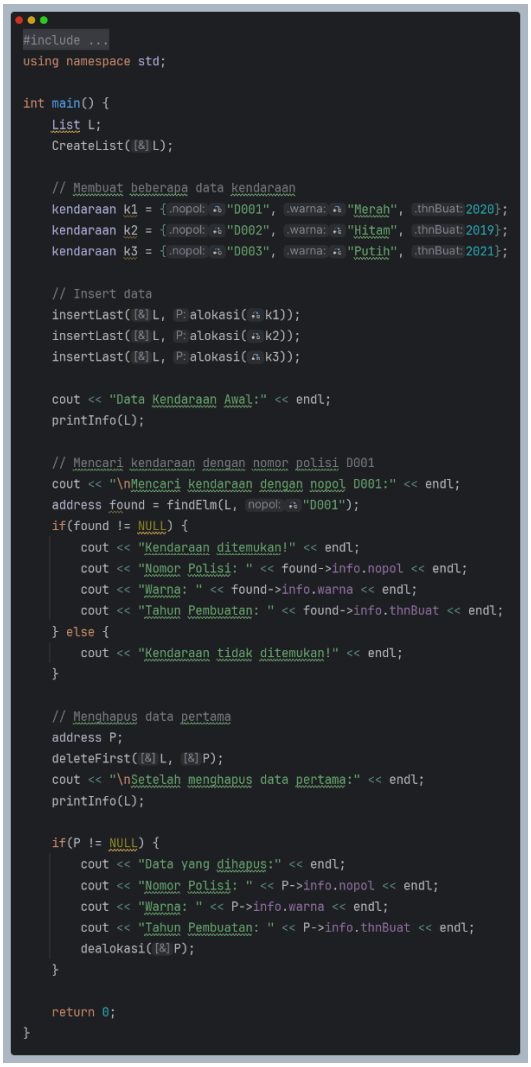
- File header ini berisi deklarasi struktur data dan prototype fungsi yang digunakan
- Komponen utama:
    - Struct kendaraan: menyimpan data kendaraan (nopol, warna, tahun pembuatan)
    - typedef infotype: alias untuk struct kendaraan
    - typedef address: pointer ke struct ElmList
    - Struct ElmList: node list yang berisi info kendaraan dan pointer prev-next
    - Struct List: menyimpan pointer First dan Last untuk manajemen list
- Deklarasi fungsi-fungsi yang akan diimplementasikan di file cpp

doublelist.cpp(implementation)

```cpp
#include ...
using namespace std;

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

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

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

void printInfo(List L) {
    address P = L.First;
    if(P == NULL) {
        cout << "List Kosong" << endl;
        return;
    }

    while(P != NULL) {
        cout << "Nomor Polisi: " << P->info.nopol << endl;
        cout << "Warna: " << P->info.warna << endl;
        cout << "Tahun Pembuatan: " << P->info.thnBuat << endl;
        cout << "------------------------" << endl;
        P = P->next;
    }
}

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

address findElm(List L, string nopol) {
    address P = L.First;
    while(P != NULL) {
        if(P->info.nopol == nopol) {
            return P;
        }
        P = P->next;
    }
    return NULL;
}

void deleteFirst(List &L, address &P) {
    if(L.First == NULL) {
        P = NULL;
        return;
    }

    P = L.First;
    if(L.First == L.Last) {
        L.First = NULL;
        L.Last = NULL;
    } else {
        L.First = L.First->next;
        L.First->prev = NULL;
        P->next = NULL;
    }
}
```

- Berisi implementasi dari fungsi-fungsi yang dideklarasikan di header
- Fungsi-fungsi utama:
    - CreateList: Inisialisasi list kosong (First dan Last = NULL)
    - alokasi: Membuat node baru dan mengisi data kendaraan
    - dealokasi: Menghapus node dari memori
    - printInfo: Menampilkan seluruh data kendaraan dalam list
    - insertLast: Menambah node di akhir list
    - findElm: Mencari kendaraan berdasarkan nomor polisi
    - deleteFirst: Menghapus node pertama dari list

main.cpp(main)



```
#include ...
using namespace std;

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

    // Membuat beberapa data kendaraan
    kendaraan k1 = {.nopol = "D001", .warna = "Merah", .thnBuat = 2020};
    kendaraan k2 = {.nopol = "D002", .warna = "Hitam", .thnBuat = 2019};
    kendaraan k3 = {.nopol = "D003", .warna = "Putih", .thnBuat = 2021};

    // Insert data
    insertLast(&L, P: alokasi(k1));
    insertLast(&L, P: alokasi(k2));
    insertLast(&L, P: alokasi(k3));

    cout << "Data Kendaraan Awal:" << endl;
    printInfo(L);

    // Mencari kendaraan dengan nomor polisi D001
    cout << "\nMencari kendaraan dengan nopol D001:" << endl;
    address found = findElm(L, nopol: "D001");
    if(found != NULL) {
        cout << "Kendaraan ditemukan!" << endl;
        cout << "Nomor Polisi: " << found->info.nopol << endl;
        cout << "Warna: " << found->info.warna << endl;
        cout << "Tahun Pembuatan: " << found->info.thnBuat << endl;
    } else {
        cout << "Kendaraan tidak ditemukan!" << endl;
    }

    // Menghapus data pertama
    address P;
    deleteFirst(&L, &P);
    cout << "\nSetelah menghapus data pertama:" << endl;
    printInfo(L);

    if(P != NULL) {
        cout << "Data yang dihapus:" << endl;
        cout << "Nomor Polisi: " << P->info.nopol << endl;
        cout << "Warna: " << P->info.warna << endl;
        cout << "Tahun Pembuatan: " << P->info.thnBuat << endl;
        dealokasi(&P);
    }

    return 0;
}
```

- Program utama yang mendemonstrasikan penggunaan double linked list
- Alur program:
    a. Membuat list kosong
    b. Membuat 3 data kendaraan (k1, k2, k3)
    c. Memasukkan data ke dalam list menggunakan insertLast
    d. Menampilkan data awal
    e. Mencari kendaraan dengan nopol "D001"
    f. Menghapus data pertama
    g. Menampilkan hasil setelah penghapusan
- Menunjukkan penggunaan berbagai fungsi yang telah diimplementasikan