LAPORAN PRAKTIKUM

PERTEMUAN 6

DOUBLE LINKED LIST (BAGIAN PERTAMA)

Nama :

Resita Istania Purwanto (2311104037)

Dosen :

Yudha Islami Sulistya

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TP

Soal 1

Kode:

```cpp
#include <iostream>

Codeium: Refactor | Explain
struct Node {
    int data;
    Node* next;
    Node* prev;

    Node(int val) : data(val), next(nullptr), prev(nullptr) {}
};

Codeium: Refactor | Explain
class DoublyLinkedList {
private:
    Node* head;

public:
    DoublyLinkedList() : head(nullptr) {}

    // Fungsi untuk menambahkan elemen di awal list
    Codeium: Refactor | Explain | ×
    void insertFirst(int value) {
        Node* newNode = new Node(value);
        if (head == nullptr) {
            head = newNode; // Jika list kosong, node baru menjadi head
        } else {
            newNode->next = head;
            head->prev = newNode;
            head = newNode; // Update head ke node baru
        }
    }

    // Fungsi untuk menambahkan elemen di akhir list
    Codeium: Refactor | Explain | ×
    void insertLast(int value) {
        Node* newNode = new Node(value);
        if (head == nullptr) {
            head = newNode; // Jika list kosong, node baru menjadi head
        } else {
            Node* temp = head;
            while (temp->next != nullptr) {
                temp = temp->next; // Mencari akhir list
            }
            temp->next = newNode;
            newNode->prev = temp; // Hubungkan node baru dengan node terakhir
        }
    }

    // Fungsi untuk menampilkan seluruh elemen dalam list
    Codeium: Refactor | Explain | ×
    void display() {
        Node* temp = head;
        std::cout << "DAFTAR ANGGOTA LIST: ";
        while (temp != nullptr) {
            std::cout << temp->data;
            if (temp->next != nullptr) {
                std::cout << " <-> ";
            }
            temp = temp->next;
        }
        std::cout << std::endl;
    }
};

Codeium: Refactor | Explain | Generate Function Comment | ×
int main() {
```

```
int main() {
    DoublyLinkedList dll;

    int firstElement, secondElement, thirdElement;

    // Input elemen pertama
    std::cout << "Masukkan elemen pertama = ";
    std::cin >> firstElement;
    dll.insertFirst(firstElement);

    // Input elemen kedua di awal
    std::cout << "Masukkan elemen kedua di awal = ";
    std::cin >> secondElement;
    dll.insertFirst(secondElement);

    // Input elemen ketiga di akhir
    std::cout << "Masukkan elemen ketiga di akhir = ";
    std::cin >> thirdElement;
    dll.insertLast(thirdElement);

    // Tampilkan seluruh elemen
    dll.display();

    return 0;
}
```

Penjelasan:

- Node Structure: Menyimpan data, pointer next, dan prev.
- insertFirst(): Menambah elemen di awal list.
- insertLast(): Menambah elemen di akhir list.
- display(): Menampilkan elemen dari awal hingga akhir dengan tanda <-> di antara elemen.

Output:

```
Masukkan elemen pertama = 10
Masukkan elemen kedua di awal = 5
Masukkan elemen ketiga di akhir = 20
DAFTAR ANGGOTA LIST: 5 <-> 10 <-> 20
PS C:\06_Double_Linked_List_Bagian_1\TP>
```

Penjelasan:

Output diperoleh jika pengguna memasukkan 10 sebagai elemen pertama, 5 sebagai elemen kedua di awal, dan 20 di akhir.

Soal 2

Kode:

```cpp
#include <iostream>

Codeium: Refactor | Explain
struct Node {
    int data;
    Node* next;
    Node* prev;

    Node(int val) : data(val), next(nullptr), prev(nullptr) {}
};

Codeium: Refactor | Explain
class DoublyLinkedList {
private:
    Node* head;

public:
    DoublyLinkedList() : head(nullptr) {}

    // Fungsi untuk menambahkan elemen di akhir list
    Codeium: Refactor | Explain | ×
    void insertLast(int value) {
        Node* newNode = new Node(value);
        if (head == nullptr) {
            head = newNode; // Jika list kosong, node baru menjadi head
        } else {
            Node* temp = head;
            while (temp->next != nullptr) {
                temp = temp->next; // Mencari akhir list
            }
            temp->next = newNode;
            newNode->prev = temp; // Hubungkan node baru dengan node terakhir
        }
    }

    // Fungsi untuk menghapus elemen pertama dalam list
    Codeium: Refactor | Explain | ×
    void deleteFirst() {
        if (head == nullptr) {
            std::cout << "List kosong, tidak ada elemen yang bisa dihapus." << std::endl;
            return;
        }
        Node* temp = head;
        if (head->next == nullptr) {
            head = nullptr; // Jika hanya ada satu elemen
        } else {
            head = head->next;
            head->prev = nullptr; // Hapus referensi ke node lama
        }
        delete temp; // Hapus node pertama
    }

    // Fungsi untuk menghapus elemen terakhir dalam list
    Codeium: Refactor | Explain | ×
    void deleteLast() {
        if (head == nullptr) {
            std::cout << "List kosong, tidak ada elemen yang bisa dihapus." << std::endl;
            return;
        }
        Node* temp = head;
        if (head->next == nullptr) {
            head = nullptr; // Jika hanya ada satu elemen
        } else {
            while (temp->next != nullptr) {
                temp = temp->next; // Mencari node terakhir
            }
```

```
            temp->prev->next = nullptr; // Putuskan hubungan node terakhir
        }
        delete temp; // Hapus node terakhir
    }

    // Fungsi untuk menampilkan seluruh elemen dalam list
    Codeium: Refactor | Explain | ×
    void display() {
        if (head == nullptr) {
            std::cout << "List kosong." << std::endl;
            return;
        }
        Node* temp = head;
        std::cout << "DAFTAR ANGGOTA LIST: ";
        while (temp != nullptr) {
            std::cout << temp->data;
            if (temp->next != nullptr) {
                std::cout << " <-> ";
            }
            temp = temp->next;
        }
        std::cout << std::endl;
    }
};

Codeium: Refactor | Explain | Generate Function Comment | ×
int main() {
    DoublyLinkedList dll;
```

```
    int firstElement, secondElement, thirdElement;

    // Input elemen pertama
    std::cout << "Masukkan elemen pertama = ";
    std::cin >> firstElement;
    dll.insertLast(firstElement);

    // Input elemen kedua di akhir
    std::cout << "Masukkan elemen kedua di akhir = ";
    std::cin >> secondElement;
    dll.insertLast(secondElement);

    // Input elemen ketiga di akhir
    std::cout << "Masukkan elemen ketiga di akhir = ";
    std::cin >> thirdElement;
    dll.insertLast(thirdElement);

    // Tampilkan list setelah penambahan elemen
    dll.display();

    // Hapus elemen pertama dan terakhir
    dll.deleteFirst();
    dll.deleteLast();

    // Tampilkan list setelah penghapusan
    std::cout << "DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: ";
    dll.display();

    return 0;
}
```

Penjelasan:

- Menambah elemen di akhir (insertLast).
- Menghapus elemen pertama (deleteFirst).
- Menghapus elemen terakhir (deleteLast).
- Menampilkan isi list (display).

Output:

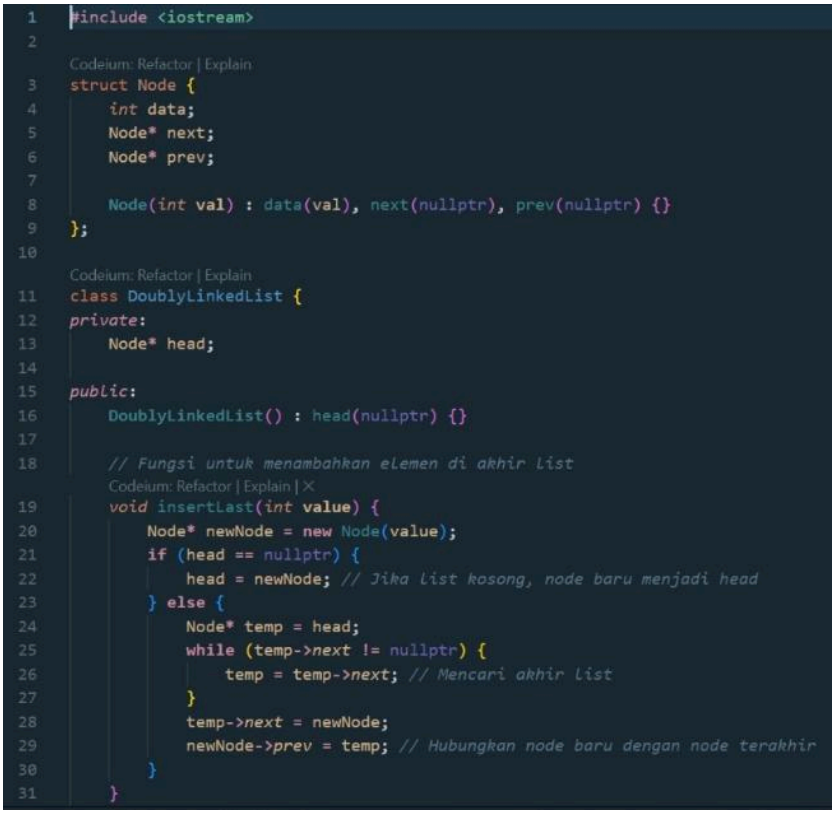
```
Masukkan elemen pertama = 10
Masukkan elemen kedua di akhir = 15
Masukkan elemen ketiga di akhir = 20
DAFTAR ANGGOTA LIST: 10 <-> 15 <-> 20
DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: DAFTAR ANGGOTA LIST: 15
PS C:\06_Double_Linked_List_Bagian_1\TP>
```

Penjelasan: Pengguna memasukkan tiga elemen ke list, kemudian list ditampilkan.

Setelah itu, elemen pertama dan terakhir dihapus, dan list ditampilkan lagi.

Soal 3

Kode:



```cpp
#include <iostream>

Codeium: Refactor | Explain
struct Node {
    int data;
    Node* next;
    Node* prev;

    Node(int val) : data(val), next(nullptr), prev(nullptr) {}
};

Codeium: Refactor | Explain
class DoublyLinkedList {
private:
    Node* head;

public:
    DoublyLinkedList() : head(nullptr) {}

    // Fungsi untuk menambahkan elemen di akhir list
    Codeium: Refactor | Explain | ×
    void insertLast(int value) {
        Node* newNode = new Node(value);
        if (head == nullptr) {
            head = newNode; // Jika list kosong, node baru menjadi head
        } else {
            Node* temp = head;
            while (temp->next != nullptr) {
                temp = temp->next; // Mencari akhir list
            }
            temp->next = newNode;
            newNode->prev = temp; // Hubungkan node baru dengan node terakhir
        }
    }
```

```cpp
    // Fungsi untuk menampilkan elemen dari depan ke belakang
    Codeium: Refactor | Explain | ×
    void displayForward() {
        if (head == nullptr) {
            std::cout << "List kosong." << std::endl;
            return;
        }
        Node* temp = head;
        std::cout << "Daftar elemen dari depan ke belakang: ";
        while (temp != nullptr) {
            std::cout << temp->data;
            if (temp->next != nullptr) {
                std::cout << " <-> ";
            }
            temp = temp->next;
        }
        std::cout << std::endl;
    }

    // Fungsi untuk menampilkan elemen dari belakang ke depan
    Codeium: Refactor | Explain | ×
    void displayBackward() {
        if (head == nullptr) {
            std::cout << "List kosong." << std::endl;
            return;
        }
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next; // Mencari node terakhir
        }
        std::cout << "Daftar elemen dari belakang ke depan: ";
        while (temp != nullptr) {
            std::cout << temp->data;
```

```
            if (temp->prev != nullptr) {
                std::cout << " <-> ";
            }
            temp = temp->prev;
        }
        std::cout << std::endl;
    }
};

Codeium: Refactor | Explain | Generate Function Comment | X
int main() {
    DoublyLinkedList dll;

    int element;

    // Input 4 elemen ke dalam list
    for (int i = 1; i <= 4; i++) {
        std::cout << "Masukkan elemen ke-" << i << ": ";
        std::cin >> element;
        dll.insertLast(element);
    }

    // Tampilkan elemen dari depan ke belakang
    dll.displayForward();

    // Tampilkan elemen dari belakang ke depan
    dll.displayBackward();

    return 0;
}
```

Penjelasan:

- insertLast: Fungsi untuk menambahkan node baru di akhir list.
- displayForward: Menampilkan elemen dari node pertama ke node terakhir.
- displayBackward: Menampilkan elemen dari node terakhir ke node pertama.

Output:

```
Masukkan elemen ke-1: 1
Masukkan elemen ke-2: 2
Masukkan elemen ke-3: 3
Masukkan elemen ke-4: 4
Daftar elemen dari depan ke belakang: 1 <-> 2 <-> 3 <-> 4
Daftar elemen dari belakang ke depan: 4 <-> 3 <-> 2 <-> 1
PS C:\06_Double_Linked_List_Bagian_1\TP>
```

Penjelasan: Pengguna memasukkan empat elemen ke dalam list. Setelah itu, list ditampilkan dua kali, pertama dari depan ke belakang dan kedua dari belakang ke depan.

II. UNGUIDED

Kode:

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

Codeium: Refactor | Explain
struct Node {
    int idBuku;
    string judulBuku;
    string penulisBuku;
    Node* prev;
    Node* next;
};

Codeium: Refactor | Explain
class DoubleLinkedList {
private:
    Node* head;
    Node* tail;

public:
    DoubleLinkedList() : head(nullptr), tail(nullptr) {}

    // Fungsi untuk menambahkan buku di akhir linked list
    Codeium: Refactor | Explain | ×
    void tambahBukuAkhir(int id, const string& judul, const string& penulis) {
        Node* newNode = new Node{id, judul, penulis, nullptr, nullptr};
        if (head == nullptr) {
            head = tail = newNode;
        } else {
            tail->next = newNode;
            newNode->prev = tail;
            tail = newNode;
        }
        cout << "Buku berhasil ditambahkan!" << endl;
    }

    // Fungsi untuk menampilkan buku dari awal ke akhir
    Codeium: Refactor | Explain | ×
    void tampilBukuDariAwal() const {
        if (head == nullptr) {
            cout << "Daftar buku kosong." << endl;
            return;
        }
        Node* temp = head;
        cout << "Daftar Buku dari Awal ke Akhir:" << endl;
        while (temp != nullptr) {
            cout << "ID Buku: " << temp->idBuku << ", Judul: " << temp->judulBuku
                 << ", Penulis: " << temp->penulisBuku << endl;
            temp = temp->next;
        }
    }

    // Fungsi untuk menampilkan buku dari akhir ke awal
    Codeium: Refactor | Explain | ×
    void tampilBukuDariAkhir() const {
        if (tail == nullptr) {
            cout << "Daftar buku kosong." << endl;
            return;
        }
        Node* temp = tail;
        cout << "Daftar Buku dari Akhir ke Awal:" << endl;
        while (temp != nullptr) {
            cout << "ID Buku: " << temp->idBuku << ", Judul: " << temp->judulBuku
                 << ", Penulis: " << temp->penulisBuku << endl;
            temp = temp->prev;
        }
    }
};

Codeium: Refactor | Explain | Generate Function Comment | ×
int main() {
    DoubleLinkedList daftarBuku;
    int pilihan, id;
    string judul, penulis;

    do {
        cout << "\nMenu:\n1. Tambah Buku\n2. Tampil Buku dari Awal ke Akhir\n3. Tampil Buku dari Akhir ke Awal\n0. Keluar\n";
        cout << "Pilihan: ";
        cin >> pilihan;

        switch (pilihan) {
            case 1:
                cout << "Masukkan ID Buku: ";
                cin >> id;
                cin.ignore();
                cout << "Masukkan Judul Buku: ";
                getline(cin, judul);
                cout << "Masukkan Penulis Buku: ";
                getline(cin, penulis);
                daftarBuku.tambahBukuAkhir(id, judul, penulis);
                break;
            case 2:
                daftarBuku.tampilBukuDariAwal();
                break;
            case 3:
```

```
                daftarBuku.tampilBukuDariAkhir();
                break;
            case 0:
                cout << "Program selesai." << endl;
                break;
            default:
                cout << "Pilihan tidak valid." << endl;
        }
    } while (pilihan != 0);

    return 0;
}
```

Penjelasan:

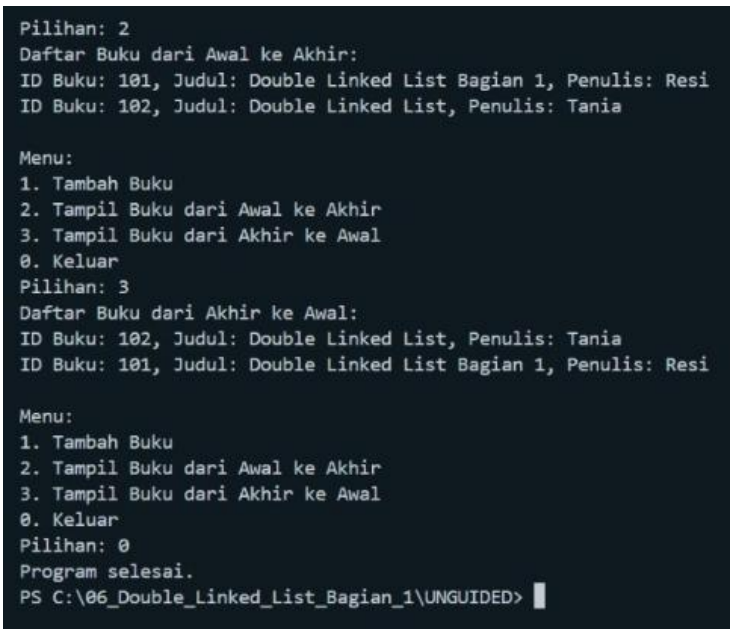
a. Struktur Node: Menyimpan data buku dengan atribut idBuku, judulBuku, dan penulisBuku
b. Kelas DoubleLinkedList:
    - Fungsi tambahBukuAkhir: Menambahkan buku di akhir list.
    - Fungsi tampilBukuDariAwal: Menampilkan buku dari head hingga tail.
    - Fungsi tampilBukuDariAkhir: Menampilkan buku dari tail hingga head.
c. Menu Program: Memungkinkan pengguna memilih untuk menambahkan buku, menampilkan buku dari awal ke akhir, atau dari akhir ke awal.

Output:

```
Menu:
1. Tambah Buku
2. Tampil Buku dari Awal ke Akhir
3. Tampil Buku dari Akhir ke Awal
0. Keluar
Pilihan: 1
Masukkan ID Buku: 101
Masukkan Judul Buku: Double Linked List Bagian 1
Masukkan Penulis Buku: Resi
Buku berhasil ditambahkan!

Menu:
1. Tambah Buku
2. Tampil Buku dari Awal ke Akhir
3. Tampil Buku dari Akhir ke Awal
0. Keluar
Pilihan: 1
Masukkan ID Buku: 102
Masukkan Judul Buku: Double Linked List
Masukkan Penulis Buku: Tania
Buku berhasil ditambahkan!

Menu:
1. Tambah Buku
2. Tampil Buku dari Awal ke Akhir
3. Tampil Buku dari Akhir ke Awal
0. Keluar
```



Penjelasan:

- Tambah Buku: Pengguna memasukkan ID, judul, dan penulis buku. Buku ditambahkan di akhir daftar.
- Tampil dari Awal ke Akhir: Menampilkan semua buku dari yang pertama hingga terakhir.
- Tampil dari Akhir ke Awal: Menampilkan semua buku dari yang terakhir hingga pertama.
- Keluar Program: Memilih "0" untuk keluar dari program.