LAPORAN PRAKTIKUM

MODUL 6

DOUBLE LINKED LIST BAGIAN 1

Nama :

Fajar Budiawan (2311104039)

Dosen :

Yudha Islami Sulistya S.Kom, M.Cs

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

TUGAS PENDAHULUAN

1. Buatlah program yang mengizinkan pengguna menambahkan elemen ke dalam Doubly Linked List di awal dan di akhir list.

   Output

```
PS D:\Praktikum Struktur Data\06_Pengenalan_CPP_Bagian_6> cd 'd:\Prakt
enalan_CPP_Bagian_6\TP\output'
PS D:\Praktikum Struktur Data\06_Pengenalan_CPP_Bagian_6\TP\output> &
Masukkan elemen pertama = 10
Masukkan elemen kedua di awal = 15
Masukkan elemen ketiga di akhir = 20
DAFTAR ANGGOTA LIST: 15 <-> 10 <-> 20
PS D:\Praktikum Struktur Data\06_Pengenalan_CPP_Bagian_6\TP\output>
```

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

// Node class untuk Doubly Linked List
class Node {
public:
    int data;
    Node* prev;
    Node* next;

    Node(int value) {
        data = value;
        prev = nullptr;
        next = nullptr;
    }
};

// Doubly Linked List class
class DoublyLinkedList {
private:
    Node* head;

public:
    DoublyLinkedList() {
        head = nullptr;
    }

    // Fungsi untuk menambahkan elemen di awal list
    void insertFirst(int data) {
        Node* newNode = new Node(data);
        if (head == nullptr) { // Jika list kosong
            head = newNode;
        } else {
            newNode->next = head;
            head->prev = newNode;
            head = newNode;
        }
    }

    // Fungsi untuk menambahkan elemen di akhir list
    void insertLast(int data) {
        Node* newNode = new Node(data);
        if (head == nullptr) { // Jika list kosong
            head = newNode;
        } else {
            Node* current = head;
            while (current->next != nullptr) { // Mencari node terakhir
                current = current->next;
            }
            current->next = newNode;
            newNode->prev = current;
        }
    }

    // Fungsi untuk menampilkan seluruh elemen dalam list
    void display() {
        Node* current = head;
        cout << "DAFTAR ANGGOTA LIST: ";
        while (current != nullptr) { // Menampilkan elemen dari depan ke belakang
            cout << current->data;
            if (current->next != nullptr) {
                cout << " <-> ";
            }
            current = current->next;
        }
        cout << endl;
    }
};

// Fungsi utama
int main() {
    DoublyLinkedList dll;

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

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

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

    // Tampilkan seluruh elemen dalam list
    dll.display();

    return 0;
}
```

2. Buatlah program yang memungkinkan pengguna untuk menghapus elemen pertama dan elemen terakhir dalam Doubly Linked List.

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

// Node class untuk Doubly Linked List
class Node {
public:
    int data;
    Node* prev;
    Node* next;

    Node(int value) {
        data = value;
        prev = nullptr;
        next = nullptr;
    }
};

// Doubly Linked List class
class DoublyLinkedList {
private:
    Node* head;

public:
    DoublyLinkedList() {
        head = nullptr;
    }

    // Fungsi untuk menambahkan elemen di akhir list
    void insertLast(int data) {
        Node* newNode = new Node(data);
        if (head == nullptr) { // Jika list kosong
            head = newNode;
        } else {
            Node* current = head;
            while (current->next != nullptr) { // Mencari node terakhir
                current = current->next;
            }
            current->next = newNode;
            newNode->prev = current;
        }
    }

    // Fungsi untuk menghapus elemen pertama
    void deleteFirst() {
        if (head == nullptr) { // Jika list kosong
            cout << "List kosong, tidak ada elemen yang dihapus." << endl;
            return;
        }
        Node* temp = head;
        head = head->next; // Pindahkan head ke node berikutnya
        if (head != nullptr) {
            head->prev = nullptr; // Set prev dari head baru menjadi nullptr
        }
        delete temp; // Hapus node yang lama
    }

    // Fungsi untuk menghapus elemen terakhir
    void deleteLast() {
        if (head == nullptr) { // Jika list kosong
            cout << "List kosong, tidak ada elemen yang dihapus." << endl;
            return;
        }
        if (head->next == nullptr) { // Jika hanya ada satu elemen
            delete head;
            head = nullptr;
            return;
        }
        Node* current = head;
        while (current->next != nullptr) { // Mencari node terakhir
            current = current->next;
        }
        current->prev->next = nullptr; // Set next dari node sebelum terakhir menjadi nullptr
        delete current; // Hapus node terakhir
    }

    // Fungsi untuk menampilkan seluruh elemen dalam list
    void display() {
        Node* current = head;
        cout << "DAFTAR ANGGOTA LIST: ";
        while (current != nullptr) { // Menampilkan elemen dari depan ke belakang
            cout << current->data;
            if (current->next != nullptr) {
                cout << " <-> ";
            }
            current = current->next;
        }
        cout << endl;
    }
};

// Fungsi utama
int main() {
    DoublyLinkedList dll;

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

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

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

    // Tampilkan seluruh elemen dalam list sebelum penghapusan
    cout << "DAFTAR ANGGOTA LIST SEBELUM PENGHAPUSAN: ";
    dll.display();

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

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

    return 0;
}
```

Output

```
PS D:\Praktikum Struktur Data\06_Pengenalan_CPP_Bagian_6\TP\output> cd 'd:\Prakt
ta\06_Pengenalan_CPP_Bagian_6\TP\output'
PS D:\Praktikum Struktur Data\06_Pengenalan_CPP_Bagian_6\TP\output> & .\'Soal_2.
Masukkan elemen pertama = 10
Masukkan elemen kedua di akhir = 15
Masukkan elemen ketiga di akhir = 20
DAFTAR ANGGOTA LIST SEBELUM PENGHAPUSAN: DAFTAR ANGGOTA LIST: 10 <-> 15 <-> 20
DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: DAFTAR ANGGOTA LIST: 15
PS D:\Praktikum Struktur Data\06_Pengenalan_CPP_Bagian_6\TP\output>
```

3. Buatlah program yang memungkinkan pengguna memasukkan beberapa elemen ke dalam Doubly Linked List. Setelah elemen dimasukkan, tampilkan seluruh elemen dalam list dari depan ke belakang, kemudian dari belakang ke depan.

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

// Node class untuk Doubly Linked List
class Node {
public:
    int data; // Data yang disimpan dalam node
    Node* prev; // Pointer ke node sebelumnya
    Node* next; // Pointer ke node berikutnya

    // Constructor untuk menginisialisasi node
    Node(int value) {
        data = value;
        prev = nullptr;
        next = nullptr;
    }
};

// Doubly Linked List class
class DoublyLinkedList {
private:
    Node* head; // Pointer ke node pertama

public:
    // Constructor untuk menginisialisasi list
    DoublyLinkedList() {
        head = nullptr; // List dimulai dengan head yang kosong
    }

    // Fungsi untuk menambahkan elemen di akhir list
    void insertLast(int data) {
        Node* newNode = new Node(data); // Buat node baru
        if (head == nullptr) { // Jika list kosong
            head = newNode; // Node baru menjadi head
        } else {
            Node* current = head; // Mulai dari head
            while (current->next != nullptr) { // Cari node terakhir
                current = current->next;
            }
            current->next = newNode; // Hubungkan node terakhir dengan node baru
            newNode->prev = current; // Set prev dari node baru
        }
    }

    // Fungsi untuk menampilkan elemen dari depan ke belakang
    void displayForward() {
        Node* current = head; // Mulai dari head
        cout << "Daftar elemen dari depan ke belakang: ";
        while (current != nullptr) { // Selama current tidak null
            cout << current->data; // Tampilkan data
            if (current->next != nullptr) {
                cout << " <-> "; // Tampilkan panah jika ada node berikutnya
            }
            current = current->next; // Pindah ke node berikutnya
        }
        cout << endl;
    }

    // Fungsi untuk menampilkan elemen dari belakang ke depan
    void displayBackward() {
        if (head == nullptr) { // Jika list kosong
            cout << "List kosong." << endl;
            return;
        }

        Node* current = head; // Mulai dari head
        // Mencari node terakhir
        while (current->next != nullptr) {
            current = current->next;
        }

        cout << "Daftar elemen dari belakang ke depan: ";
        while (current != nullptr) { // Selama current tidak null
            cout << current->data; // Tampilkan data
            if (current->prev != nullptr) {
                cout << " <-> "; // Tampilkan panah jika ada node sebelumnya
            }
            current = current->prev; // Pindah ke node sebelumnya
        }
        cout << endl;
    }
};

// Fungsi utama
int main() {
    DoublyLinkedList dll; // Buat objek dari DoublyLinkedList

    // Input 4 elemen
    for (int i = 0; i < 4; i++) {
        int element;
        cout << "Masukkan elemen ke-" << (i + 1) << ": ";
        cin >> element; // Ambil input dari pengguna
        dll.insertLast(element); // Tambahkan elemen ke list
    }

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

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

    return 0;
}
```

Output

```
PS D:\Praktikum Struktur Data\06_Pengenalan_CPP_Bagian_6> cd 'd:\Prak
enalan_CPP_Bagian_6\TP\output'
PS D:\Praktikum Struktur Data\06_Pengenalan_CPP_Bagian_6\TP\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 D:\Praktikum Struktur Data\06_Pengenalan_CPP_Bagian_6\TP\output>
```