LAPORAN PRAKTIKUM

Modul 06

“DOUBLE LINKED LIST”

Disusun Oleh:

Muhammad Widya Tirta Cahyatama – 21104047

S1SE07-01

Dosen:

Yudha Islami Sulistya, S.Kom., M.Cs

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

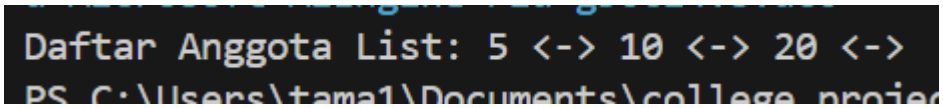
TELKOM UNIVERSITY PURWOKERTO

2024

Soal 1

```cpp
#include <iostream>

using namespace std;

struct Node {
    int data;
    Node* prev;
    Node* next;
};

class DoublyLinkedList {
public:
    Node* head;

    DoublyLinkedList() {
        head = nullptr;
    }

    void insertFirst(int data) {
        Node* newNode = new Node;
        newNode->data = data;
        newNode->prev = nullptr;
        newNode->next = head;

        if (head != nullptr) {
            head->prev = newNode;
        }
        head = newNode;
    }

    void insertLast(int data) {
        Node* newNode = new Node;
        newNode->data = data;
        newNode->prev = nullptr;
        newNode->next = nullptr;

        if (head == nullptr) {
            head = newNode;
        } else {
            Node* current = head;
            while (current->next != nullptr) {
                current = current->next;
            }
            current->next = newNode;
            newNode->prev = current;
        }
    }

    void display() {
        Node* current = head;
        while (current != nullptr) {
            cout << current->data << " <-> ";
            current = current->next;
        }
        cout << endl;
    }
};

int main() {
    DoublyLinkedList list;

    int element1 = 10;
    int element2 = 5;
    int element3 = 20;

    list.insertFirst(element1);
    list.insertFirst(element2);
    list.insertLast(element3);

    cout << "Daftar Anggota List: ";
    list.display();

    return 0;
}
```

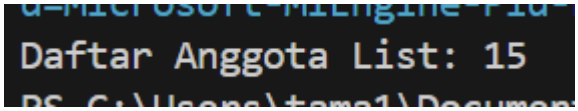
Output:



Soal 2

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

struct Node {
    int data;
    Node* prev;
    Node* next;
};

class DoublyLinkedList {
public:
    Node* head;

    DoublyLinkedList() { head = nullptr; }

    void insertLast(int data) {
        Node* newNode = new Node();
        newNode->data = data;
        newNode->next = nullptr;

        if (head == nullptr) {
            newNode->prev = nullptr;
            head = newNode;
            return;
        }

        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }

    void deleteFirst() {
        if (head == nullptr) return;

        Node* temp = head;
        head = head->next;

        if (head != nullptr) {
            head->prev = nullptr;
        }

        delete temp;
    }

    void deleteLast() {
        if (head == nullptr) return;

        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }

        if (temp->prev != nullptr) {
            temp->prev->next = nullptr;
        } else {
            head = nullptr;
        }

        delete temp;
    }

    void display() {
        Node* temp = head;
        while (temp != nullptr) {
            cout << temp->data;
            if (temp->next != nullptr) cout << " <-> ";
            temp = temp->next;
        }
        cout << endl;
    }
};

int main() {
    DoublyLinkedList dll;
    dll.insertLast(10);
    dll.insertLast(15);
    dll.insertLast(20);

    dll.deleteFirst();
    dll.deleteLast();

    cout << "Daftar Anggota List: ";
    dll.display();

    return 0;
}
```

Output:



Soal 3

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

struct Node {
    int data;
    Node* prev;
    Node* next;
};

class DoublyLinkedList {
public:
    Node* head;

    DoublyLinkedList() { head = nullptr; }

    void insertLast(int data) {
        Node* newNode = new Node();
        newNode->data = data;
        newNode->next = nullptr;

        if (head == nullptr) {
            newNode->prev = nullptr;
            head = newNode;
            return;
        }

        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }

    void displayForward() {
        Node* temp = head;
        while (temp != nullptr) {
            cout << temp->data;
            if (temp->next != nullptr) cout << " <-> ";
            temp = temp->next;
        }
        cout << endl;
    }

    void displayBackward() {
        Node* temp = head;
        if (temp == nullptr) return;

        while (temp->next != nullptr) {
            temp = temp->next;
        }

        while (temp != nullptr) {
            cout << temp->data;
            if (temp->prev != nullptr) cout << " <-> ";
            temp = temp->prev;
        }
        cout << endl;
    }
};

int main() {
    DoublyLinkedList dll;
    dll.insertLast(1);
    dll.insertLast(2);
    dll.insertLast(3);
    dll.insertLast(4);

    cout << "Daftar elemen dari depan ke belakang: ";
    dll.displayForward();

    cout << "Daftar elemen dari belakang ke depan: ";
    dll.displayBackward();

    return 0;
}
```

Output:

```
Daftar elemen dari depan ke belakang: 1 <-> 2 <-> 3 <-> 4
Daftar elemen dari belakang ke depan: 4 <-> 3 <-> 2 <-> 1
```