LAPORAN PRAKTIKUM

MODUL 6

DOUBLE LINKED LIST (BAGIAN PERTAMA)

Nama :

Novita Syahwa Tri Hapsari (2311104007)

Dosen : Yudha Islami

Sulistya,S.Kom,M.Cs

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

Soal 1: Menambahkan Elemen di Awal dan Akhir DLL

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

Instruksi:
1. Implementasikan fungsi `insertFirst` untuk menambahkan elemen di awal list.
2. Implementasikan fungsi `insertLast` untuk menambahkan elemen di akhir list.
3. Tampilkan seluruh elemen dalam list dari depan ke belakang setelah penambahan dilakukan.

Contoh Input:
- Input: Masukkan elemen pertama = 10
- Input: Masukkan elemen kedua di awal = 5
- Input: Masukkan elemen ketiga di akhir = 20

Output:
- DAFTAR ANGGOTA LIST: 5 <-> 10 <-> 20

1.

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

class Node {
public:
    int data;
    Node* prev;
    Node* next;
};
class DoublyLinkedList {
public:
    Node* head;
    Node* tail;

    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }
    void insertFirst_2311104007(int data) {
        Node* newNode = new Node();
        newNode->data = data;
        newNode->prev = nullptr;
        newNode->next = head;

        if (head != nullptr) {
            head->prev = newNode;
        } else {
            tail = newNode;
        }
        head = newNode;
    }
    void insertLast_2311104007(int data) {
        Node* newNode = new Node();
        newNode->data = data;
        newNode->next = nullptr;
        newNode->prev = tail;

        if (tail != nullptr) {
            tail->next = newNode;
        } else {
            head = newNode;
        }
        tail = newNode;
    }

    void display_2311104007() {
        Node* current = head;
        while (current != nullptr) {
            cout << current->data;
            if (current->next != nullptr) {
                cout << " <-> ";
            }
            current = current->next;
        }
        cout << endl;
    }
};
int main() {
    DoublyLinkedList list;
    int data;
    cout << "Masukkan elemen pertama = ";
    cin >> data;
    list.insertFirst_2311104007(data);
    cout << "Masukkan elemen kedua di awal = ";
    cin >> data;
    list.insertFirst_2311104007(data);
    cout << "Masukkan elemen ketiga di akhir = ";
    cin >> data;
    list.insertLast_2311104007(data);
    cout << "DAFTAR ANGGOTA LIST: ";
    list.display_2311104007();

    return 0;
}
```

Hasil Running

```
Masukkan elemen pertama = 10
Masukkan elemen kedua di awal = 5
Masukkan elemen ketiga di akhir = 20
DAFTAR ANGGOTA LIST: 5 <-> 10 <-> 20
PS N:\Telkom University\SEMESTER 3\PRAKTIKUM
```

Penjelasan :

Disini saya sudah menerapkan nim untuk setiap fungsi yang terdapat dalam code program. DoublyLinkedList menyimpan pointer ke head dan tail. Fungsi insertFirst_2311104007 digunakan untuk menambahkan elemen baru di depan list, sedangkan insertLast_2311104007 menambahkan elemen di belakang. Fungsi display_2311104007 mencetak semua elemen di list dengan tanda <-> untuk menunjukkan hubungan antar node.

Soal 2: Menghapus Elemen di Awal dan Akhir DLL

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

Instruksi:
1. Implementasikan fungsi `deleteFirst` untuk menghapus elemen pertama.
2. Implementasikan fungsi `deleteLast` untuk menghapus elemen terakhir.
3. Tampilkan seluruh elemen dalam list setelah penghapusan dilakukan.

Contoh Input:
- Input: Masukkan elemen pertama = 10
- Input: Masukkan elemen kedua di akhir = 15
- Input: Masukkan elemen ketiga di akhir = 20
- Hapus elemen pertama dan terakhir.

Output:
- DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: 15

2.

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

class Node {
public:
    int data;
    Node* prev;
    Node* next;
};

class DoublyLinkedList {
public:
    Node* head;
    Node* tail;

    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }
    void insertFirst_2311104007(int data) {
        Node* newNode = new Node();
        newNode->data = data;
        newNode->prev = nullptr;
        newNode->next = head;

        if (head != nullptr) {
            head->prev = newNode;
        } else {
            tail = newNode;
        }
        head = newNode;
    }
    void insertLast_2311104007(int data) {
        Node* newNode = new Node();
        newNode->data = data;
        newNode->next = nullptr;
        newNode->prev = tail;

        if (tail != nullptr) {
            tail->next = newNode;
        } else {
            head = newNode;
        }
        tail = newNode;
    }
    void deleteFirst_2311104007() {
        if (head == nullptr) {
            cout << "List kosong, tidak ada elemen yang dapat dihapus." << endl;
            return;
        }
        Node* temp = head;
        head = head->next;

        if (head != nullptr) {
            head->prev = nullptr;
        } else {
            tail = nullptr;
        }
        delete temp;
    }
    void deleteLast_2311104007() {
        if (tail == nullptr) {
            cout << "List kosong, tidak ada elemen yang dapat dihapus." << endl;
            return;
        }
        Node* temp = tail;
        tail = tail->prev;

        if (tail != nullptr) {
            tail->next = nullptr;
        } else {
            head = nullptr;
        }
        delete temp;
    }
    void display_2311104007() {
        if (head == nullptr) {
            cout << "List kosong." << endl;
            return;
        }
        Node* current = head;
        while (current != nullptr) {
            cout << current->data;
            if (current->next != nullptr) {
                cout << " <-> ";
            }
```

Penjelasan: Pada Code ini punya banyak fitur. Dengan menggunakan fungsi insertFirst_2311104007, elemen ditambahkan ke depan, insertLast_2311104007 menambahkannya ke belakang, dan deleteFirst_2311104007 dan deleteLast_2311104007 menghapus elemen pertama dan terakhir. Fungsi main disini meminta memasukkan tiga elemen (dan fungsi "display_2311104007" menampilkan semua elemen. Daftar anggota ditampilkan kembali setelah elemen pertama dan terakhir dihapus. Untuk menambah, menghapus, dan menampilkan elemen dalam daftar yang terhubung dua kali.

Hasil Running:

```
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: 15
PS N:\Telkom University\SEMESTER 3\PRAKTIKUM
```

```
            current = current->next;
        }
        cout << endl;
    }
};

int main() {
    DoublyLinkedList list;
    int data;
    cout << "Masukkan elemen pertama = ";
    cin >> data;
    list.insertFirst_2311104007(data);
    cout << "Masukkan elemen kedua di akhir = ";
    cin >> data;
    list.insertLast_2311104007(data);
    cout << "Masukkan elemen ketiga di akhir = ";
    cin >> data;
    list.insertLast_2311104007(data);
    cout << "Dafftar Anggota List: ";
    list.display_2311104007();
    list.deleteFirst_2311104007();
    list.deleteLast_2311104007();
    cout << "Daftar Anggota List Setelah Dihapus: ";
    list.display_2311104007();
    return 0;
}
```

3. **Soal 3:** Menampilkan Elemen dari Depan ke Belakang dan Sebaliknya

Deskripsi Soal: 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.

Instruksi:

1. Implementasikan fungsi untuk menampilkan elemen dari depan ke belakang.
2. Implementasikan fungsi untuk menampilkan elemen dari belakang ke depan.
3. Tambahkan 4 elemen ke dalam list dan tampilkan elemen tersebut dalam dua arah.

Contoh Input:

- Input: Masukkan 4 elemen secara berurutan: 1, 2, 3, 4

Output:

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

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

class Node {
public:
    int data;
    Node* prev;
    Node* next;
};

class DoublyLinkedList {
public:
    Node* head;
    Node* tail;

    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }
    void insertLast_2311104007(int data) {
        Node* newNode = new Node();
        newNode->data = data;
        newNode->next = nullptr;
        newNode->prev = tail;

        if (tail != nullptr) {
            tail->next = newNode;
        } else {
            head = newNode;
        }
        tail = newNode;
    }
    void PrintDepan_2311104007() {
        if (head == nullptr) {
            cout << "List kosong." << endl;
            return;
        }
        Node* current = head;
        while (current != nullptr) {
            cout << current->data;
            if (current->next != nullptr) {
                cout << " <-> ";
            }
            current = current->next;
        }
        cout << endl;
    }
    void PrintBelakang_2311104007() {
        if (tail == nullptr) {
            cout << "List kosong." << endl;
            return;
        }
        Node* current = tail;
        while (current != nullptr) {
            cout << current->data;
            if (current->prev != nullptr) {
                cout << " <-> ";
            }
            current = current->prev;
        }
        cout << endl;
    }
};

int main() {
    DoublyLinkedList list;
    int data;
    cout << "Masukkan 4 elemen secara berurutan: ";
    for (int i = 0; i < 4; i++) {
        cin >> data;
        list.insertLast_2311104007(data);
    }
    cout << "Daftar elemen dari depan ke belakang: ";
    list.PrintDepan_2311104007();
    cout << "Daftar elemen dari belakang ke depan: ";
    list.PrintBelakang_2311104007();

    return 0;
}
```

Penjelasan:

Pada code program disini program meminta pengguna untuk memasukkan empat elemen secara berurutan, yang kemudian dimasukkan di akhir list. Setelah itu, PrintDepan_2311104007 menampilkan elemen dari depan ke belakang, dan PrintBelakang_2311104007 menampilkan elemen dari belakang ke depan. Program ini dengan sederhana menunjukkan cara menambahkan elemen di akhir list serta menampilkannya dalam dua arah.

Hasil Running :

```
Masukkan 4 elemen secara berurutan: 1 2 3 4
Daftar elemen dari depan ke belakang: 1 <-> 2 <-> 3 <-> 4
Daftar elemen dari belakang ke depan: 4 <-> 3 <-> 2 <-> 1
PS N:\Telkom University\SEMESTER 3\PRAKTIKUM STRUKTUR DATA\
```