LAPORAN PRAKTIKUM
MODUL 6
DOUBLE LINKED LIST BAGIAN 1

Nama :

Muhammad Rifqi Al Baqi Ananta (2311104005)

Dosen :

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

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSTY PURWOKERTO

2024

I. TP

1. Soal 1

   Code:

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

// Struktur untuk node dalam Doubly Linked List
struct Node {
    int data;
    Node* next;
    Node* prev;
};

// Struktur untuk List
struct List {
    Node* head;
    Node* tail;
};

// Inisialisasi List kosong
void init(List& L) {
    L.head = nullptr;
    L.tail = nullptr;
}

// Fungsi untuk menambah elemen di awal list
void insertFirst_2311104005(List& L, int nilai) {
    Node* newNode = new Node;
    newNode->data = nilai;
    newNode->next = nullptr;
    newNode->prev = nullptr;

    if (L.head == nullptr) {  // Jika list kosong
        L.head = newNode;
        L.tail = newNode;
    } else {  // Jika list tidak kosong
        newNode->next = L.head;
        L.head->prev = newNode;
        L.head = newNode;
    }
}

// Fungsi untuk menambah elemen di akhir list
void insertLast_2311104005(List& L, int nilai) {
    Node* newNode = new Node;
    newNode->data = nilai;
    newNode->next = nullptr;
    newNode->prev = nullptr;

    if (L.head == nullptr) {  // Jika list kosong
        L.head = newNode;
        L.tail = newNode;
    } else {  // Jika list tidak kosong
        L.tail->next = newNode;
        newNode->prev = L.tail;
        L.tail = newNode;
    }
}

// Fungsi untuk menampilkan semua elemen list
void tampilList(List L) {
    Node* current = L.head;
    cout << "DAFTAR ANGGOTA LIST: ";
    while (current != nullptr) {
        cout << current->data;
        if (current->next != nullptr) {
            cout << " <-> ";
        }
        current = current->next;
    }
    cout << endl;
}

int main() {
    List L;
    init(L);

    int nilai;

    cout << "Masukkan elemen pertama = ";
    cin >> nilai;
    insertFirst_2311104005(L, nilai);

    cout << "Masukkan elemen kedua di awal = ";
    cin >> nilai;
    insertFirst_2311104005(L, nilai);

    cout << "Masukkan elemen ketiga di akhir = ";
    cin >> nilai;
    insertLast_2311104005(L, nilai);

    tampilList(L);

    return 0;
}
```

Output:

```
Masukkan elemen pertama = 10
Masukkan elemen kedua di awal = 5
Masukkan elemen ketiga di akhir = 20
DAFTAR ANGGOTA LIST: 5 <-> 10 <-> 20
PS D:\TUGAS ITTP\SEMESTER 3\Praktikum Struktur Data\Pertemuan 6\TP_6\output>
```

2. Soal 2
   Code:

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

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

struct List {
    Node* head;
    Node* tail;
};

void init(List& L) {
    L.head = nullptr;
    L.tail = nullptr;
}

void insertLast_2311104005(List& L, int nilai) {
    Node* newNode = new Node;
    newNode->data = nilai;
    newNode->next = nullptr;
    newNode->prev = nullptr;

    if (L.head == nullptr) {
        L.head = newNode;
        L.tail = newNode;
    } else {
        L.tail->next = newNode;
        newNode->prev = L.tail;
        L.tail = newNode;
    }
}

void deleteFirst_2311104005(List& L) {
    if (L.head == nullptr) {
        cout << "List kosong" << endl;
        return;
    }

    Node* temp = L.head;

    if (L.head == L.tail) {  // Jika hanya satu elemen
        L.head = nullptr;
        L.tail = nullptr;
    } else {
        L.head = L.head->next;
        L.head->prev = nullptr;
    }

    delete temp;
}

void deleteLast_2311104005(List& L) {
    if (L.head == nullptr) {
        cout << "List kosong" << endl;
        return;
    }

    Node* temp = L.tail;

    if (L.head == L.tail) {  // Jika hanya satu elemen
        L.head = nullptr;
        L.tail = nullptr;
    } else {
        L.tail = L.tail->prev;
        L.tail->next = nullptr;
    }

    delete temp;
}

void tampilList(List L) {
    Node* current = L.head;
    cout << "DAFTAR ANGGOTA LIST: ";
    while (current != nullptr) {
        cout << current->data;
        if (current->next != nullptr) {
            cout << " <-> ";
        }
        current = current->next;
    }
    cout << endl;
}

int main() {
    List L;
    init(L);

    int nilai;

    cout << "Masukkan elemen pertama = ";
    cin >> nilai;
    insertLast_2311104005(L, nilai);

    cout << "Masukkan elemen kedua di akhir = ";
    cin >> nilai;
    insertLast_2311104005(L, nilai);

    cout << "Masukkan elemen ketiga di akhir = ";
    cin >> nilai;
    insertLast_2311104005(L, nilai);

    cout << "\nList sebelum penghapusan:" << endl;
    tampilList(L);

    cout << "\nMenghapus elemen pertama dan terakhir..." << endl;
    deleteFirst_2311104005(L);
    deleteLast_2311104005(L);

    cout << "\nDaftar anggota list setelah penghapusan:" << endl;
    tampilList(L);

    return 0;
}
```

Output:

```
Masukkan elemen pertama = 10
Masukkan elemen kedua di akhir = 13
Masukkan elemen ketiga di akhir = 15

List sebelum penghapusan:
DAFTAR ANGGOTA LIST: 10 <-> 13 <-> 15

Menghapus elemen pertama dan terakhir...

Daftar anggota list setelah penghapusan:
DAFTAR ANGGOTA LIST: 13
PS D:\TUGAS ITTP\SEMESTER 3\Praktikum Struktur Data\Pertemuan 6\TP_6\output>
```

3. Soal 3

   Code:

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

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

struct List {
    Node* head;
    Node* tail;
};

void init(List& L) {
    L.head = nullptr;
    L.tail = nullptr;
}

void insertLast_2311104005(List& L, int nilai) {
    Node* newNode = new Node;
    newNode->data = nilai;
    newNode->next = nullptr;
    newNode->prev = nullptr;

    if (L.head == nullptr) {
        L.head = newNode;
        L.tail = newNode;
    } else {
        L.tail->next = newNode;
        newNode->prev = L.tail;
        L.tail = newNode;
    }
}

void tampilDepanBelakang_2311104005(List L) {
    Node* current = L.head;
    cout << "Daftar elemen dari depan ke belakang: ";
    while (current != nullptr) {
        cout << current->data;
        if (current->next != nullptr) {
            cout << " <-> ";
        }
        current = current->next;
    }
    cout << endl;
}

void tampilBelakangDepan_2311104005(List L) {
    Node* current = L.tail;
    cout << "Daftar elemen dari belakang ke depan: ";
    while (current != nullptr) {
        cout << current->data;
        if (current->prev != nullptr) {
            cout << " <-> ";
        }
        current = current->prev;
    }
    cout << endl;
}

int main() {
    List L;
    init(L);

    cout << "Masukkan 4 elemen secara berurutan:" << endl;
    for(int i = 1; i <= 4; i++) {
        int nilai;
        cout << "Masukkan elemen ke-" << i << ": ";
        cin >> nilai;
        insertLast_2311104005(L, nilai);
    }

    cout << endl;
    tampilDepanBelakang_2311104005(L);
    tampilBelakangDepan_2311104005(L);

    return 0;
}
```

Output:

```
Masukkan 4 elemen secara berurutan:
Masukkan elemen ke-1: 10
Masukkan elemen ke-2: 15
Masukkan elemen ke-3: 20
Masukkan elemen ke-4: 25

Daftar elemen dari depan ke belakang: 10 <-> 15 <-> 20 <-> 25
Daftar elemen dari belakang ke depan: 25 <-> 20 <-> 15 <-> 10
PS D:\TUGAS ITTP\SEMESTER 3\Praktikum Struktur Data\Pertemuan 6\TP_6\output>
```

II. PENJELASAN

1. Program ini mengimplementasikan struktur data Doubly Linked List dengan fokus pada operasi penambahan elemen. Pengguna dapat memasukkan tiga nilai, dengan dua nilai pertama ditambahkan di awal list menggunakan fungsi insertFirst_2311104005(), dan nilai ketiga ditambahkan di akhir list menggunakan fungsi insertLast_2311104005(). Program kemudian menampilkan seluruh elemen list yang terhubung dengan tanda "<->", menunjukkan bahwa setiap node memiliki koneksi dua arah.

2. Program ini merupakan pengembangan dari Doubly Linked List dengan penambahan operasi penghapusan. Program meminta pengguna memasukkan tiga nilai yang ditambahkan ke akhir list, kemudian melakukan operasi penghapusan elemen pertama (deleteFirst_2311104005()) dan elemen terakhir (deleteLast_2311104005()). Program menampilkan kondisi list sebelum dan sesudah penghapusan, memperlihatkan perubahan struktur data setelah operasi penghapusan dilakukan.

3. Program ini mendemonstrasikan kemampuan traversal dua arah dalam Doubly Linked List. Pengguna diminta memasukkan empat nilai yang disimpan dalam list, kemudian program menampilkan elemen-elemen tersebut dalam dua arah: dari depan ke belakang (tampilDepanBelakang_2311104005()) dan dari belakang ke depan (tampilBelakangDepan_2311104005()). Ini menunjukkan keunggulan Doubly Linked List yang memungkinkan navigasi dalam dua arah melalui penggunaan pointer prev dan next.