Tugas Pendahuluan Modul 5
STRUKTUR DATA - Genap 2024/2025
"Single Linked List Bagian 2"

A. Ketentuan Tugas Pendahuluan

1. Tugas Pendahuluan dikerjakan secara Individu.
2. TP ini bersifat WAJIB, tidak mengerjakan = PENGURANGAN POIN JURNAL / TES ASESMEN.
3. Hanya MENGUMPULKAN tetapi TIDAK MENGERJAKAN = PENGURANGAN POIN JURNAL / TES ASESMEN.
4. Deadline pengumpulan TP Modul 4 adalah Senin, 9 Oktober 2023 pukul 06.00 WIB.
5. TIDAK ADA TOLERANSI KETERLAMBATAN, TERLAMBAT ATAU TIDAK MENGUMPULKAN TP MAKA DIANGGAP TIDAK MENGERJAKAN.
6. DILARANG PLAGIAT (PLAGIAT = E).
7. Kerjakan TP dengan jelas agar dapat dimengerti.
8. File diupload di LMS menggunakan format PDF dengan ketentuan: TP_MOD_[XX]_NIM_NAMA.pdf
9. **SOAL TEORI WAJIB DIKERJAKAN TULIS TANGAN, TIDAK BOLEH DIKETIK!**

CP (WA):

- Raihan (089638482851)
- Kayyisa (085105303555)
- Abiya (082127180662)
- Rio (081210978384)

SELAMAT MENGERJAKAN^^

Nama: Maulana Kaka Halin Widyadhana
Nim: 2311104034
Dosen Pembimbing: Yudha Islami Sulistya

B. Soal Praktek

1. Menambahkan Elemen di Awal dan Akhir DLL

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

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

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

public:
    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    // Menambahkan elemen di awal list
    void insertFirst(int value) {
        Node* newNode = new Node();
        newNode->data = value;
        newNode->prev = nullptr;
        newNode->next = head;

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

        head = newNode;
    }

    // Menambahkan elemen di akhir list
    void insertLast(int value) {
        Node* newNode = new Node();
        newNode->data = value;
        newNode->next = nullptr;
        newNode->prev = tail;

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

        tail = newNode;
    }
```

```
    // Menampilkan elemen dari depan ke belakang
    void display() {
        Node* temp = head;
        cout << "DAFTAR ANGGOTA LIST: ";
        while (temp != nullptr) {
            cout << temp->data;
            if (temp->next != nullptr) {
                cout << " <-> ";
            }
            temp = temp->next;
        }
        cout << endl;
    }
};

int main() {
    DoublyLinkedList dll;

    int value;

    // Input elemen pertama di awal
    cout << "Input: Masukkan elemen pertama = ";
    cin >> value;
    dll.insertFirst(value);

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

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

    // Tampilan seluruh elemen
    dll.display();

    return 0;
}
```

Penjelasan:

Buat struct node yang berfungsi untuk isi data, pointer ke node sebelumnya, dan ke node berikutnya. Kemudian buat “Class DoublyLinkedList” yang didalamnya terdapat 3 method yang masing-masing berfungsi untuk menambahkan elemen di awal, menambahkan elemen di akhir, dan menampilkan elemen dari depan ke belakang. Terakhir untuk “Fungsi main()” yaitu berfungsi untuk menerima input dari pengguna dan menambahkan elemen sesuai instruksi yang kemudian menampilkan hasilnya.

Output:

```
Input: Masukkan elemen pertama = 4
Input: Masukkan elemen kedua di awal = 8
Input: Masukkan elemen ketiga di akhir = 12
DAFTAR ANGGOTA LIST: 8 <-> 4 <-> 12
```

2. Menghapus Elemen di Awal dan Akhir DLL

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

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

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

public:
    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    // Menambahkan elemen di akhir list
    void insertLast(int value) {
        Node* newNode = new Node();
        newNode->data = value;
        newNode->next = nullptr;
        newNode->prev = tail;

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

        tail = newNode;
    }

    // Menghapus elemen pertama
    void deleteFirst() {
        if (head == nullptr) {
            cout << "List kosong, tidak ada elemen yang bisa dihapus." << endl;
            return;
        }

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

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

        delete temp;
    }
```

```
    // Menghapus elemen terakhir
    void deleteLast() {
        if (tail == nullptr) {
            cout << "List kosong, tidak ada elemen yang bisa dihapus." << endl;
            return;
        }

        Node* temp = tail;
        tail = tail->prev;

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

        delete temp;
    }

    // Menampilkan elemen dari depan ke belakang
    void display() {
        Node* temp = head;
        cout << "DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: ";
        while (temp != nullptr) {
            cout << temp->data;
            if (temp->next != nullptr) {
                cout << " <-> ";
            }
            temp = temp->next;
        }
        cout << endl;
    }
};
```

```
int main() {
    DoublyLinkedList dll;
    int value;

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

    // Input elemen kedua
    cout << "Input: Masukkan elemen kedua = ";
    cin >> value;
    dll.insertLast(value);

    // Input elemen ketiga
    cout << "Input: Masukkan elemen ketiga = ";
    cin >> value;
    dll.insertLast(value);

    // Hapus elemen pertama
    dll.deleteFirst();

    // Hapus elemen terakhir
    dll.deleteLast();

    // Tampilan seluruh elemen
    dll.display();

    return 0;
}
```

Penjelasan:

Struct Node, struktur ini berisikan data, pointer ke node sebelumnya dan berikutnya. Kemudian terdapat juga class doublylinkedlist yang kali ini berisikan 4 Method. Method pertama berfungsi untuk menambahkan elemen di akhir. Untuk Method kedua berfungsi untuk menghapus elemen pertama. Selanjutnya method kedua yang berfungsi untuk menghapus elemen terakhir. Kemudian method keempat yang berfungsi untuk menampilkan elemen dari depan ke belakang. Kemudian yang terakhir adalah Fungsi Main(), fungsi ini menerima input, menambah elemen, menghapus elemen pertama dan terakhir, serta menampilkan hasil.

Output:

```
Input: Masukkan elemen pertama = 7
Input: Masukkan elemen kedua = 2
Input: Masukkan elemen ketiga = 0
DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: 2
```

3. Menampilkan Elemen dari Depan ke Belakang dan Sebaliknya

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

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

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

public:
    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    // Menambahkan elemen di akhir list
    void insertLast(int value) {
        Node* newNode = new Node();
        newNode->data = value;
        newNode->next = nullptr;
        newNode->prev = tail;

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

        tail = newNode;
    }

    // Menampilkan elemen dari depan ke belakang
    void displayForward() {
        Node* temp = head;
        cout << "Daftar elemen dari depan ke belakang: ";
        while (temp != nullptr) {
            cout << temp->data;
            if (temp->next != nullptr) {
                cout << " <-> ";
            }
            temp = temp->next;
        }
        cout << endl;
    }
```

```
51
52      // Menampilkan elemen dari belakang ke depan
53      void displayBackward() {
54          Node* temp = tail;
55          cout << "Daftar elemen dari belakang ke depan: ";
56          while (temp != nullptr) {
57              cout << temp->data;
58              if (temp->prev != nullptr) {
59                  cout << " <-> ";
60              }
61              temp = temp->prev;
62          }
63          cout << endl;
64      }
65  };
66
67  int main() {
68      DoublyLinkedList dll;
69      int value;
70
71      // Memasukkan 4 elemen ke dalam list
72      cout << "Input: Masukkan 4 elemen secara berurutan:\n";
73      for (int i = 0; i < 4; ++i) {
74          cout << "Elemen ke-" << i+1 << " = ";
75          cin >> value;
76          dll.insertLast(value);
77      }
78
79      // Menampilkan elemen dari depan ke belakang
80      dll.displayForward();
81
82      // Menampilkan elemen dari belakang ke depan
83      dll.displayBackward();
84
85      return 0;
86  }
```

Penjelasan:

Buat Struct node yang berisikan data, pointer ke node sebelumnya dan berikutnya. Setelah itu buat class doublylinkedlist yang didalamnya terdapat 3 method. Ketiga method tersebut masing-masing berfungsi untuk menambah elemen di akhir, menampilkan elemen dari depan ke belakang, dan menampilkan elemen dari belakang ke depan. Kemudian seperti biasa buat fungsi main() yang akan menerima input dan menambahkannya ke dalam list kemudian menampilkannya.

Output:

```
Input: Masukkan 4 elemen secara berurutan:
Elemen ke-1 = 4
Elemen ke-2 = 8
Elemen ke-3 = 12
Elemen ke-4 = 16
Daftar elemen dari depan ke belakang: 4 <-> 8 <-> 12 <-> 16
Daftar elemen dari belakang ke depan: 16 <-> 12 <-> 8 <-> 4
```