LAPORAN UJIAN PRAKTIKUM

Disusun Oleh:
Zhafir Zaidan Avail
S1-SE-07-2

Dosen :
Wahyu Andi Saputra, S.Pd., M.Eng

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

- **PROGRAM**
  - **Doublelinkedlist.cpp**

```
// Nama : Zhafir Zaidan Avail
// NIM : 2311104059
// Kelas : S1-SE-07-02

#include "doublelinkedlist.h"
#include <iostream>
#include <vector>
// Buat node baru
Node* newElement(Mahasiswa data) {
    Node* newNode = new Node;
    newNode->data = data;
    newNode->next = nullptr;
    newNode->prev = nullptr;
    return newNode;
}

// Buat list baru
List newList() {
    List list;
    list.head = nullptr;
    list.tail = nullptr;
    return list;
}

// Cek kosong
bool isEmpty(List list) {
    return list.head == nullptr;
}

// Insert data di awal
void insertFirst(List &list, Node* newNode) {
    if (isEmpty(list)) {
        list.head = newNode;
        list.tail = newNode;
    } else {
        newNode->next = list.head;
        list.head->prev = newNode;
        list.head = newNode;
    }
}

// Insert data setelah node tertentu
void insertAfter(List &list, Mahasiswa data, Node* prevNode) {
    if (prevNode == nullptr) return;
    Node* newNode = newElement(data);
    newNode->next = prevNode->next;
    newNode->prev = prevNode;
    prevNode->next = newNode;
    if (newNode->next != nullptr) {
        newNode->next->prev = newNode;
    } else {
        list.tail = newNode;
    }
}

// Insert data di akhir
void insertLast(List &list, Node* newNode) {
    if (isEmpty(list)) {
        list.head = newNode;
        list.tail = newNode;
    } else {
        list.tail->next = newNode;
        newNode->prev = list.tail;
```

```
        list.tail = newNode;
    }
}

// Hapus node pertama
void deleteFirst(List &list, Node* &deletedNode) {
    if (isEmpty(list)) return;
    deletedNode = list.head;
    list.head = list.head->next;
    if (list.head != nullptr) {
        list.head->prev = nullptr;
    } else {
        list.tail = nullptr;
    }
    delete deletedNode;
}

// Hapus node terakhir
void deleteLast(List &list, Node* &deletedNode) {
    if (isEmpty(list)) return;
    deletedNode = list.tail;
    list.tail = list.tail->prev;
    if (list.tail != nullptr) {
        list.tail->next = nullptr;
    } else {
        list.head = nullptr;
    }
    delete deletedNode;
}

// Hitung panjang list
int length(List list) {
    int count = 0;
    Node* current = list.head;
    while (current != nullptr) {
        count++;
        current = current->next;
    }
    return count;
}

// Cari Node Tertentu
Node* findElement(List list, std::string NIM) {
    Node* current = list.head;
    while (current != nullptr) {
        if (current->data.NIM == NIM) {
            return current;
        }
        current = current->next;
    }
    return nullptr;
}

// Cetak list
void printList(List list) {
    Node* current = list.head;
    while (current != nullptr) {
        std::cout << "Nama: " << current->data.nama << ", NIM: " <<
current->data.NIM
                  << ", Kelas: " << current->data.kelas << ", Nilai
Asesmen: " << current->data.nilaiAsesmen
                  << ", Nilai Praktikum: " << current-
>data.nilaiPraktikum << std::endl;
        current = current->next;
    }
}
```

```
// Masukkan data
void addNData(List &list, int N) {
    for (int i = 0; i < N; i++) {
        Mahasiswa mhs;
        std::cout << "\nMahasiswa ke-" << (i + 1) << ":\n";

        std::cout << "Masukkan Nama: ";
        std::cin.ignore(); // Mengabaikan karakter newline sisa input
sebelumnya
        std::getline(std::cin, mhs.nama);

        std::cout << "Masukkan NIM: ";
        std::getline(std::cin, mhs.NIM);

        std::cout << "Masukkan Kelas: ";
        std::getline(std::cin, mhs.kelas);

        std::cout << "Masukkan Nilai Asesmen: ";
        std::cin >> mhs.nilaiAsesmen;

        std::cout << "Masukkan Nilai Praktikum: ";
        std::cin >> mhs.nilaiPraktikum;

        Node* newNode = newElement(mhs);
        insertLast(list, newNode);
    }
}


// Cari Nilai Tertinggi
Mahasiswa findHighestAsesmen(List list) {
    Node* current = list.head;
    Mahasiswa tertinggi;
    if (current != nullptr) {
        tertinggi = current->data;
        current = current->next;
    }
    while (current != nullptr) {
        if (current->data.nilaiAsesmen > tertinggi.nilaiAsesmen) {
            tertinggi = current->data;
        }
        current = current->next;
    }
    return tertinggi;
}

// Delete NIM duplikat
void removeDuplicates(List &list) {
    Node* current = list.head;
    while (current != nullptr) {
        Node* runner = current->next;
        while (runner != nullptr) {
            if (runner->data.NIM == current->data.NIM) {
                Node* duplicate = runner;
                runner->prev->next = runner->next;
                if (runner->next != nullptr) {
                    runner->next->prev = runner->prev;
                } else {
                    list.tail = runner->prev;
                }
                runner = runner->next;
                delete duplicate;
            } else {
                runner = runner->next;
            }
```

```
        }
        current = current->next;
    }
}
```

- **Main.cpp**

```
//Zhafir Zaidan Avail
//2311104059
// Kelas : S1-SE-07-02

#include "doublelinkedlist.h"
#include <iostream>
#include <string>

// Fungsi utama
int main() {
    List list = newList(); // Inisialisasi linked list
    int N;

    std::cout << "Jumlah Data Mahasiswa: ";
    std::cin >> N;

    // Memasukkan data mahasiswa
    for (int i = 0; i < N; ++i) {
        Mahasiswa mhs;
        std::cout << "\nMahasiswa ke-" << (i + 1) << ":\n";

        std::cout << "Masukan Nama: ";
        std::cin.ignore(); // Membersihkan buffer
        std::getline(std::cin, mhs.nama);

        std::cout << "Masukan NIM: ";
        std::getline(std::cin, mhs.NIM);

        std::cout << "Masukan Kelas: ";
        std::getline(std::cin, mhs.kelas);

        std::cout << "Masukan Nilai Assesmen: ";
        std::cin >> mhs.nilaiAsesmen;

        std::cout << "Masukan Nilai Praktikum: ";
        std::cin >> mhs.nilaiPraktikum;

        // Menambahkan data mahasiswa ke akhir list
        Node* newNode = newElement(mhs);
        insertLast(list, newNode);
    }

    std::cout << "\nData mahasiswa dalam list:\n";
    printList(list);

    Mahasiswa tertinggi = findHighestAsesmen(list);
    std::cout << "\nMahasiswa dengan nilai asesmen tertinggi:\n";
    std::cout << "Nama: " << tertinggi.nama << ", NIM: " <<
tertinggi.NIM
              << ", Kelas: " << tertinggi.kelas << ", Nilai Asesmen: "
<< tertinggi.nilaiAsesmen
              << ", Nilai Praktikum: " << tertinggi.nilaiPraktikum <<
std::endl;

    removeDuplicates(list);
    std::cout << "\nSetelah pengecekan duplikat:\n";
    printList(list);

    return 0;
}
```

- **Doublelinkedlist.h**

```
//Zhafir Zaidan Avail
//2311104059
// Kelas : S1-SE-07-02

#ifndef DOUBLELIST_H
#define DOUBLELIST_H
#include <vector>
#include <string>
struct Mahasiswa {
    std::string nama;
    std::string NIM;
    std::string kelas;
    float nilaiAsesmen;
    float nilaiPraktikum;
};
struct Node {
    Mahasiswa data;
    Node* next;
    Node* prev;
};
struct List {
    Node* head;
    Node* tail;
};
Node* newElement(Mahasiswa data);
List newList();
bool isEmpty(List list);
void insertFirst(List &list, Node* newNode);
void insertAfter(List &list, Mahasiswa data, Node* prevNode);
void insertLast(List &list, Node* newNode);
void deleteFirst(List &list, Node* &deletedNode);
void deleteLast(List &list, Node* &deletedNode);
int length(List list);
Node* findElement(List list, std::string NIM);
void printList(List list);
void addNData(List &list, int N);
Mahasiswa findHighestAsesmen(List list);
void removeDuplicates(List &list);

#endif
```

- **OUTPUT**

```
Jumlah Data Mahasiswa: 1

Mahasiswa ke-1:
Masukan Nama: Zhafir Zaidan Avail
Masukan NIM: 2311104059
Masukan Kelas: S1-SE-07-02
Masukan Nilai Assesmen: 40
Masukan Nilai Praktikum: 50

Data mahasiswa dalam list:
Nama: Zhafir Zaidan Avail, NIM: 2311104059, Kelas: S1-SE-07-02, Nilai Asesmen: 40, Nilai Praktikum: 50

Mahasiswa dengan nilai asesmen tertinggi:
Nama: Zhafir Zaidan Avail, NIM: 2311104059, Kelas: S1-SE-07-02, Nilai Asesmen: 40, Nilai Praktikum: 50

Setelah pengecekan duplikat:
Nama: Zhafir Zaidan Avail, NIM: 2311104059, Kelas: S1-SE-07-02, Nilai Asesmen: 40, Nilai Praktikum: 50

Process returned 0 (0x0)   execution time : 41.658 s
Press any key to continue.
```