LAPORAN PRAKTIKUM

MODUL 5

SINGLE LINKED LIST BAGIAN 2

Disusun Oleh:

Muhammad Fathi Rafa -2311104022

SE07A

Dosen :

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM

PURWOKERTO

2024

Soal Tugas Pendahuluan

1. Mencari elemen tertentu dalam sll
   Program:

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

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

// Fungsi untuk menambahkan elemen ke akhir linked list
void insertEnd_2311104022(Node*& head, int value) {
    Node* newNode = new Node;
    newNode->data = value;
    newNode->next = nullptr;

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

   Program dimulai dengan node. Node adalah struktur yang menyimpan data int dan pointer ke node berikutnya. Lalu, masuk pada void yang menerima dua paramater. Void ini berfungsi membuat node baru dengan nilai yang diberikan dan mengatur pointer next ke nullptr. Lalu masuk ke percabangan, jika linked list kosong maka node baru menjadi head. Jika tidak, fungsi ini mencarii node terakhir dan menambahkan node baru di akhir linked list.

```
// Fungsi untuk mencari elemen dalam linked list
void searchElement_2311104022(Node* head, int value) {
    Node* current = head;
    int position = 1;
    bool found = false;

    while (current != nullptr) {
        if (current->data == value) {
            cout << "Elemen ditemukan di posisi: " << position
                 << " dengan alamat: " << current << endl;
            found = true;
            break;
        }
        current = current->next;
        position++;
    }

    if (!found) {
        cout << "Elemen " << value << " tidak ditemukan dalam list." << endl;
    }
}
```

Pada void search berfungsi mencari elemen dalam linked list. Pada void ini ada fungsi while untuk mencari elemen.

```
int main() {
    Node* head = nullptr;

    // Memasukkan 6 elemen ke dalam linked list
    for (int i = 0; i < 6; ++i) {
        int value;
        cout << "Masukkan elemen " << i + 1 << ": ";
        cin >> value;
        insertEnd_2311104022(head, value);
    }

    // Mencari elemen dalam linked list
    int cari;
    cout << "Masukkan elemen yang ingin dicari: ";
    cin >> cari;
    searchElement_2311104022(head, cari);

    return 0;
}
```

Pada program main barulah program dieksekusi. Dimulai dari perulangan untuk menginputkan elemen 1-6. Setelah itu mencari alamat dari elemen baru.

Output:

```
Masukkan elemen 1: 6
Masukkan elemen 2: 5
Masukkan elemen 3: 4
Masukkan elemen 4: 3
Masukkan elemen 5: 2
Masukkan elemen 6: 1
Masukkan elemen yang ingin dicari: 1
Elemen ditemukan di posisi: 6 dengan alamat: 0xfa1b30
```

User diminta menginputkan 6 elemen. Setelah menginputkan, user menginputkan alamat elemen dicari. Setelah selesai mencari alamat akan menampilkan alamat dari elemen yang dicari.

2. Mengurutkan list menggunakan bubble sort

   Program:

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

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

// Fungsi untuk menambahkan elemen ke akhir linked list
void insertEnd_2311104022(Node*& head, int value) {
    Node* newNode = new Node;
    newNode->data = value;
    newNode->next = nullptr;

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

   Pertama, void insert digunankan untuk menambah elemen.

```
void bubbleSortList_2311104022(Node* head) {
    if (head == nullptr) return;

    bool swapped;
    do {
        swapped = false;
        Node* current = head;

        while (current->next != nullptr) {
            if (current->data > current->next->data) {
                swap(current->data, current->next->data);
                swapped = true;
            }
            current = current->next;
        }
    } while (swapped);
}

void printList_2311104022(Node* head) {
    Node* current = head;
    while (current != nullptr) {
        cout << current->data << " ";
        current = current->next;
    }
    cout << endl;
}
```

Pada void ini, semua elemen yang sudah di inputkan akan dilakukan bubble sort untuk mengurutkan. Void printlist ini digunakan untukmencetak elemen sebelum dan sesudah elemen diururtkan.

```
int main() {
    Node* head = nullptr;

    // Memasukkan 5 elemen ke dalam linked list
    for (int i = 0; i < 5; ++i) {
        int value;
        cout << "Masukkan elemen " << i + 1 << ": ";
        cin >> value;
        insertEnd_2311104022(head, value);
    }

    cout << "List sebelum diurutkan: ";
    printList_2311104022(head);

    // Mengurutkan linked list
    bubbleSortList_2311104022(head);

    cout << "List setelah diurutkan: ";
    printList_2311104022(head);

    return 0;
}
```

Pada main ini barulah program akan di eksekusi, mulai dari mneginputkkan elemen lalu , elemen akan diurutkan dengan menggunakan buble

Output:

```
Masukkan elemen 1: 9
Masukkan elemen 2: 8
Masukkan elemen 3: 5
Masukkan elemen 4: 6
Masukkan elemen 5: 7
List sebelum diurutkan: 9 8 5 6 7
List setelah diurutkan: 5 6 7 8 9
```

User diminta menginoutkan 5 elemen. Setelah itu program mengurutkan dengan bubble sort. Setelah selesai, akan menampilkan list sebelum dan sesudah diurutkan.

3. Menambahkan elemen secara terurut

   Program:

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

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

// Fungsi untuk menambahkan elemen ke akhir linked list
void insertEnd_2311104022(Node*& head, int value) {
    Node* newNode = new Node;
    newNode->data = value;
    newNode->next = nullptr;

    if (head == nullptr) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}
void printList_2311104022(Node* head) {
    Node* current = head;
    while (current != nullptr) {
        cout << current->data << " ";
        current = current->next;
    }
    cout << endl;
}
```

```cpp
void insertSorted_2311104022(Node*& head, int value) {
    Node* newNode = new Node;
    newNode->data = value;
    newNode->next = nullptr;

    if (head == nullptr || head->data >= value) {
        newNode->next = head;
        head = newNode;
    } else {
        Node* current = head;
        while (current->next != nullptr && current->next->data < value) {
            current = current->next;
        }
        newNode->next = current->next;
        current->next = newNode;
    }
}
```

Digunakan menempatkan elemen baru dengan benar

```
int main() {
    Node* head = nullptr;

    // Memasukkan 4 elemen terurut secara manual
    for (int i = 0; i < 4; ++i) {
        int value;
        cout << "Masukkan elemen " << i + 1 << ": ";
        cin >> value;
        insertSorted_2311104022(head, value);
    }

    // Menambah elemen baru secara terurut
    int newValue;
    cout << "Masukkan elemen baru yang ingin ditambahkan: ";
    cin >> newValue;
    insertSorted_2311104022(head, newValue);

    cout << "List setelah elemen baru dimasukkan: ";
    printList_2311104022(head);

    return 0;
}
```

Di program main ini barulah program di eksekusi. Dimulai dari menginputkan elemen, lalu menambah elemen baru. Setelah memasukkan elemen baru, elemen akan memanggil void insertsorted untuk mengurutkan elemen baru

Output:

```
Masukkan elemen 1: 9
Masukkan elemen 2: 8
Masukkan elemen 3: 7
Masukkan elemen 4: 6
Masukkan elemen baru yang ingin ditambahkan: 5
List setelah elemen baru dimasukkan: 5 6 7 8 9
```

Pertama, user diminta menginputkan angka acak. setelah menginputkan 4 elemen, user diminta menambahkan elemen baru. Setelah menambahkan, elemen baru akan diurutkan dengan bubble sort lalu mencetak output.