PRAKTIKUM STRUKTUR DATA

TUGAS PENDAHULUAN 04

Single Linked List bagian 02

Nama :

Raihan Sastra Wibyanto (2311104020)

Dosen :

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

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

SOAL TP

Soal 1: Mencari Elemen Tertentu dalam SLL

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

// Struktur Node untuk Single Linked List
struct Node {
    int data;
    Node* next;
};

// Kelas SingleLinkedList untuk menangani operasi-operasi pada SLL
class SingleLinkedList {
private:
    Node* head;

public:
    SingleLinkedList() {
        head = nullptr;
    }

    // Fungsi untuk menambah elemen ke dalam linked list
    void addElement(int value) {
        Node* newNode = new Node();
        newNode->data = value;
        newNode->next = nullptr;

        if (head == nullptr) {
            head = newNode; // Menjadikan elemen pertama
        } else {
            Node* temp = head;
            while (temp->next != nullptr) {
                temp = temp->next;
            }
            temp->next = newNode; // Menambah elemen di akhir list
        }
    }

    // Fungsi untuk mencari elemen dalam linked list
    void searchElement(int value) {
        Node* current = head;  // Inisialisasi pointer current ke head
        int position = 1;      // Inisialisasi posisi ke 1

        // Melakukan pencarian linear pada linked list
        while (current != nullptr) {
            if (current->data == value) {
                // Jika elemen ditemukan, tampilkan alamat dan posisinya
                cout << "Elemen " << value << " ditemukan di alamat: " << current << ", pada posisi ke-" << position << endl;
                return;
            }
            // Bergerak ke node berikutnya dan menambah posisi
            current = current->next;
            position++;
        }

        // Jika elemen tidak ditemukan
        cout << "Elemen " << value << " tidak ditemukan dalam list." << endl;
    }

    // Fungsi untuk menampilkan semua elemen dalam list
    void displayList() {
        Node* temp = head;
        if (head == nullptr) {
            cout << "List kosong." << endl;
            return;
        }
        while (temp != nullptr) {
            cout << temp->data << " -> ";
            temp = temp->next;
        }
        cout << "NULL" << endl;
    }
};

int main() {
    SingleLinkedList list;
    int input, searchValue;

    // Meminta pengguna memasukkan 6 elemen
    cout << "Masukkan 6 elemen integer ke dalam linked list: " << endl;
    for (int i = 0; i < 6; i++) {
        cout << "Elemen " << i+1 << ": ";
        cin >> input;
        list.addElement(input);
    }

    // Menampilkan list
    cout << "\nDaftar elemen dalam list: " << endl;
    list.displayList();

    // Meminta pengguna memasukkan elemen yang ingin dicari
    cout << "\nMasukkan nilai elemen yang ingin dicari: ";
    cin >> searchValue;

    // Mencari elemen dalam list
    list.searchElement(searchValue);

    return 0;
}
```

Outputnya

```
Raihan Sastra W@MSI MINGW64 ~/Documents/Institut Teknologi Telkom Purwokerto/Semester 3/Struktur Data/STD_Raihan_Sastra_Wibyanto_2311104020/05_Single_Linked_List_Bagian2/TP/output (main)
$ ./"soal1.exe"
Masukkan 6 elemen integer ke dalam linked list:
Elemen 1: 1
Elemen 2: 2
Elemen 3: 4
Elemen 4: 5
Elemen 5: 6
Elemen 6: 7

Daftar elemen dalam list:
1 -> 2 -> 4 -> 5 -> 6 -> 7 -> NULL

Masukkan nilai elemen yang ingin dicari: 2
Elemen 2 ditemukan di alamat: 0xfc9ea0, pada posisi ke-2
```

Soal 2: Mengurutkan List Menggunakan Bubble Sort

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

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

void push(Node** head, int new_data) {
    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = (*head);
    (*head) = new_node;
}

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

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

    bool swapped;
    Node* current;
    Node* lastSorted = nullptr;

    do {
        swapped = false;
        current = head;

        while (current->next != lastSorted) {
            if (current->data < current->next->data) {

                int temp = current->data;
                current->data = current->next->data;
                current->next->data = temp;
                swapped = true;
            }
            current = current->next;
        }
        lastSorted = current;
    } while (swapped);
}

int main() {
    Node* head = nullptr;

    push(&head, 1);
    push(&head, 3);
    push(&head, 2);
    push(&head, 4);

    cout << "List before sorting: ";
    printList(head);

    bubbleSort(head);

    cout << "List after sorting in descending order: ";
    printList(head);

    return 0;
}
```

Outputnya

```
Raihan Sastra W@MSI MINGW64 ~/Documents/Institut Teknologi Telkom Pur
wokerto/Semester 3/Struktur Data/STD_Raihan_Sastra_Wibyanto_231110402
0/05_Single_Linked_List_Bagian2/TP/output (main)
$ ./"soal2.exe"
List before sorting: 4 2 3 1
List after sorting in descending order: 4 3 2 1
```

Soal 3: Menambahkan Elemen Secara Terurut

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

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

// Fungsi untuk membuat node baru
Node* createNode(int data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = nullptr;
    return newNode;
}

// Fungsi untuk menyisipkan node baru dalam urutan yang sudah terurut
void insertSorted(Node** head, Node* newNode) {
    Node* current = *head;
    Node* prev = nullptr;
    bool found = false;

    // Menelusuri list untuk menemukan posisi yang tepat bagi nodeBaru
    while (current != nullptr && !found) {
        if (current->data > newNode->data) {
            found = true;
        } else {
            prev = current;
            current = current->next;
        }
    }

    // Menyisipkan node Baru di awal jika harus menjadi kepala baru
    if (prev == nullptr) {
        newNode->next = *head;
        *head = newNode;
    }
    // Menyisipkan node Baru di akhir jika sekarang adalah null
    else if (current == nullptr) {
        prev->next = newNode;
        newNode->next = nullptr;
    }
    // Menyisipkan node Baru di antara sebelumnya dan sekarang
    else {
        prev->next = newNode;
        newNode->next = current;
    }
}

// Fungsi untuk mencetak isi list
void printList(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    Node* head = nullptr;

    // Insert elements in sorted order
    insertSorted(&head, createNode(3));
    insertSorted(&head, createNode(1));
    insertSorted(&head, createNode(4));
    insertSorted(&head, createNode(2));
    insertSorted(&head, createNode(5));

    cout << "Sorted List: ";
    printList(head);

    return 0;
}
```

Outputnya

```
Raihan Sastra W@MSI MINGW64 ~/Documents/Institut Teknologi Telkom Purwokerto/Semester 3/Struktur Data/STD_Raihan_Sastra_Wibyanto_2311104020/05_Single_Linked_List_Bagian2/TP/output (main)
$ ./"soal3.exe"
Sorted List: 1 2 3 4 5
```