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. Mencari Elemen Tertentu dalam SLL

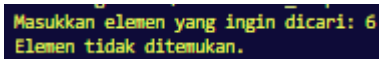
```cpp
#include <iostream>
using namespace std;

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

// Fungsi menambahkan elemen ke dalam linked list
void insert(Node*& head, int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = head;
    head = newNode;
}

// Fungsi mencari elemen dalam linked list
void searchElement(Node* head, int target) {
    Node* current = head;
    int position = 1; // Posisi dimulai dari 1
    bool found = false;

    // Linear search dalam linked list
    while (current != nullptr) {
        if (current->data == target) {
            cout << "Elemen ditemukan pada posisi: " << position << endl;
            cout << "Alamat elemen: " << current << endl;
            found = true;
            break;
        }
        current = current->next;
        position++;
    }

    // Jika elemen tidak ditemukan
    if (!found) {
        cout << "Elemen tidak ditemukan." << endl;
    }
}

int main() {
    Node* head = nullptr;

    // Memasukkan 6 elemen ke dalam list
    insert(head, 5);
    insert(head, 10);
    insert(head, 15);
    insert(head, 20);
    insert(head, 25);
    insert(head, 30);

    // Meminta input dari pengguna
    int target;
    cout << "Masukkan elemen yang ingin dicari: ";
    cin >> target;

    // Mencari elemen dalam list
    searchElement(head, target);

    return 0;
}
```

Penjelasan:

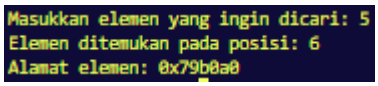
Buat fungsi insert yang digunakan untuk menambah elemen ke dalam linked list dan kemudian menambahkan elemen baru pada awal list. Lalu buat fungsi searchElement untuk mencari elemen tertentu pada linked list dengan linear search dan terdapat percabangan. Jika elemen ditemukan, maka posisi dan alamat memori elemen ditampilkan. Jika tidak, pesan elemen tidak akan ditemukan akan muncul.

Output:

Jika tidak ditemukan:



Jika ditemukan:



2. Mengurutkan List Menggunakan Bubble Sort

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

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

// Menambahkan elemen ke dalam linked list
void insert(Node*& head, int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = head;
    head = newNode;
}

// Menukar nilai dua node
void swap(Node* a, Node* b) {
    int temp = a->data;
    a->data = b->data;
    b->data = temp;
}

// Melakukan Bubble Sort pada linked list
void bubbleSortList(Node* head) {
    if (head == nullptr) {
        return; // Jika list kosong, tidak perlu sorting
    }

    bool swapped;
    Node* ptr1;
    Node* lptr = nullptr;

    // Perulangan hingga tidak ada pertukaran
    do {
        swapped = false;
        ptr1 = head;

        while (ptr1->next != lptr) {
            if (ptr1->data > ptr1->next->data) {
                swap(ptr1, ptr1->next); // Tukar data jika urutannya salah
                swapped = true;
            }
            ptr1 = ptr1->next;
        }
        lptr = ptr1; // Elemen terakhir sudah berada pada posisi yang benar
    } while (swapped);
}
```

```
51
52  // Menampilkan elemen-elemen dalam linked list
53  void displayList(Node* head) {
54      Node* current = head;
55      while (current != nullptr) {
56          cout << current->data << " ";
57          current = current->next;
58      }
59      cout << endl;
60  }
61
62  int main() {
63      Node* head = nullptr;
64
65      // Memasukkan 5 elemen ke dalam list
66      insert(head, 3);
67      insert(head, 1);
68      insert(head, 4);
69      insert(head, 2);
70      insert(head, 5);
71
72      cout << "List sebelum sorting: ";
73      displayList(head);
74
75      // Mengurutkan list menggunakan Bubble Sort
76      bubbleSortList(head);
77
78      cout << "List setelah sorting: ";
79      displayList(head);
80
81      return 0;
82  }
```

Penjelasan:

Membuat fungsi insert untuk menambah elemen kemudian membuat fungsi swap untuk menukar elemen. Lalu buat fungsi bubbleSortList untuk mengurutkan linked list dengan metode bubble sort. Proses dilakukan dengan perulangan hingga tidak ada pertukaran lagi. Terakhir buat fungsi displayList untuk menampilkan elemen dalam linked list.

Output:

```
List sebelum sorting: 5 2 4 1 3
List setelah sorting: 1 2 3 4 5
```

3. Menambahkan elemen secara terurut

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

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

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

// Menampilkan elemen-elemen dalam linked list
void displayList(Node* head) {
    Node* current = head;
    while (current != nullptr) {
        cout << current->data << " ";
        current = current->next;
    }
    cout << endl;
}

// Menambahkan elemen secara terurut ke dalam linked list
void insertSorted(Node*& head, int value) {
    Node* newNode = createNode(value);

    // Jika list kosong atau elemen baru lebih kecil dari elemen pertama
    if (head == nullptr || head->data >= value) {
        newNode->next = head;
        head = newNode;
    } else {
        // Mencari posisi yang tepat untuk memasukkan elemen baru
        Node* current = head;
        while (current->next != nullptr && current->next->data < value) {
            current = current->next;
        }
        // Menyisipkan elemen baru di antara current dan current->next
        newNode->next = current->next;
        current->next = newNode;
    }
}
```

```
48
49  int main() {
50      Node* head = nullptr;
51
52      // Memasukkan 4 elemen ke dalam list
53      insertSorted(head, 10);
54      insertSorted(head, 5);
55      insertSorted(head, 20);
56      insertSorted(head, 15);
57
58      cout << "List setelah penambahan elemen secara terurut: ";
59      displayList(head);
60
61      // Meminta pengguna memasukkan elemen tambahan untuk disisipkan
62      int newElement;
63      cout << "Masukkan elemen baru yang ingin ditambahkan: ";
64      cin >> newElement;
65
66      // Menambahkan elemen baru ke dalam list secara terurut
67      insertSorted(head, newElement);
68
69      cout << "List setelah elemen baru ditambahkan: ";
70      displayList(head);
71
72      return 0;
73  }
```

Penjelasan:

Pertama buat fungsi createNode untuk membuat node baru dengan nilai yang akan dimasukkan oleh pengguna. Kemudian buat fungsi displayList untuk menampilkan elemen pada linked list. Setelah itu buat fungsi insertSorted untuk menambahkan elemen baru pada linked list dengan terurut kemudian buat main untuk bagian utama program.

Output:

```
List setelah penambahan elemen secara terurut: 5 10 15 20
Masukkan elemen baru yang ingin ditambahkan: 13
List setelah elemen baru ditambahkan: 5 10 13 15 20
```