Tugas Proceeding

Modul 5 Struktur Data

SINGLE LINKED LIST BAGIAN 2

Disusun Oleh:

Christian Felix Saliman Sugiono (2311104031)

S1SE0701

Dosen:

Yudha Islami Sulistya

Program Studi S1 Software Engineering

Fakultas Informatika

Telkom University

Purwokerto

2024

Tugas Pendahuluan Modul 4

STRUKTUR DATA - Ganjil 2024/2025

"SINGLE LINKED LIST BAGIAN 1"

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

CP (WA):

- Andini (082243700965)
- Imelda (082135374187)

SELAMAT MENGERJAKAN^^

B. Unguided Modul 5

1 Single list mahasiswa, kodingan untuk unguided ini adalah sebagai berikut ini

```cpp
#include <iostream>
#include <string>

using namespace std;

struct Node {
    double nim;
    string nama;
    Node* next;
};

void tambahMahasiswa_2311104031(Node*& head, int nim, string nama) {
    Node* newNode = new Node();
    newNode->nim = nim;
    newNode->nama = nama;
    newNode->next = nullptr;

    if (head == nullptr) {

        head = newNode;
    } else {

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

void cariMahasiswa_2311104031(Node* head, int nimDicari) {
    Node* temp = head;
    while (temp != nullptr) {
        if (temp->nim == nimDicari) {

            cout << "Mahasiswa dengan NIM " << nimDicari << " ditemukan: " << temp->nama << endl;
            return;
        }
        temp = temp->next;
    }

    cout << "Mahasiswa dengan NIM " << nimDicari << " tidak ditemukan" << endl;
}
```

```cpp
void tampilkanMahasiswa_2311104031(Node* head) {
    if (head == nullptr) {
        cout << "Tidak ada data mahasiswa." << endl;
        return;
    }

    Node* temp = head;
    while (temp != nullptr) {
        cout << "NIM: " << temp->nim << ", Nama: " << temp->nama << endl;
        temp = temp->next;
    }
}

int main() {
    Node* head = nullptr;

    tambahMahasiswa_2311104031(head, 23115, "Sarkonus Digimon");
    tambahMahasiswa_2311104031(head, 23120, "Daedulus Strong Arm");
    tambahMahasiswa_2311104031(head, 23130, "Sheldon Plankton");

    cout << "Daftar mahasiswa:" << endl;
    tampilkanMahasiswa_2311104031(head);

    int nimCari;
    cout << "\nMasukkan NIM yang ingin dicari: ";
    cin >> nimCari;
    cariMahasiswa_2311104031(head, nimCari);

    return 0;
}
```

hal pertama yang perlu kita lakukan untuk memulai kodingan ini adalah dengan membuat struct mahasiswa yang Dimana ia berisi double untuk nim dan string untuk Namanya, lalu kita akan membuat method untuk menambahkan mahasiswa dengan void Node*& int nim dan nama yang merupakan variable yang akan ditambahkan kdealam linked list, lalu kita akan membuat method untuk mencari mahasiswa, yang Dimana bila ia menemukan data yang dicari ia akan menampilkan hasilnya, tetapi bila tidak ia akan menampilkan pesan tidak ditemukan, lalu setelah itu method yang selanjutnya adalah untuk menunjukan seluruh data mahasiswa dengan print, tetapi apabila SLL kosong maka akan tampil list kosong. Selanjutnya kita akan masuk ke main untuk penggunaan method method yang baru saja kita buat, namun kita pertama harus mengset headnya sebagai null ptr untuk mendeklarasi list kosong, lalu baru kita dapat mengisi data dengan tambah mahasiswa sebanyak 3 (saya mengambil contoh 3 mahasiswa) dengan nim dan nama yang berbeda, lalu kita akan mengerpint data mahasiswa dengan method tampilMahasiswa, dan ahirnyakita akan membuat search dengan method cariMahasiswa dan hasilnya tergantung apabila nimnya terdapat pada list kita, Hasil output kodinganya adalah sebagai berikut ini:

```
Daftar mahasiswa:
NIM: 23115, Nama: Sarkonus Digimon
NIM: 23120, Nama: Daedulus Strong Arm
NIM: 23130, Nama: Sheldon Plankton

Masukkan NIM yang ingin dicari: 23115
Mahasiswa dengan NIM 23115 ditemukan: Sarkonus Digimon
```

TP modul 5

1 Searching dalam SLL, kodingan yang dapat dibuat untuk menyelesaikan permasalahanya adalah sebagai berikut ini:

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

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

void tambahElemen_2311104031(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 searchElement_2311104031(Node* head, int value) {
    Node* current = head;
    int position = 1;

    while (current != nullptr) {
        if (current->data == value) {
            cout << "Elemen " << value << " ditemukan di alamat " << current << " pada urutan ke " << position << endl;
            return;
        }
        current = current->next;
        position++;
    }
    cout << "Elemen " << value << " tidak ditemukan dalam list." << endl;
}
```

sebagai awalan kita akan membuat struct node sebagai tempat penampungan data dari SLL yang akan kita buat, lalu kita akan akan membuat method untuk menambahkan elemen pada list dengan cara penambahan pada awal list, kemudian kita akan membuat method untuk pencarian pada list SLL yang akan kita masukan datanya pada int main, dan kita akan membuat kondisi apabila data ditemukan dan sebaliknya dengan permisalan if, jika data ditemukan maka ia akan menunjuk pada tmpat dimana data terletak, tetapi apabila tidak maka ia akan mengatakan data tidak ada pada list, dan terakhir kita akan menambahkan method print list agar user tau isi dari list kita. Kemudian kita akan masuk ke dalam main yang dimana di dalam main ini kita akan inisial lisasikan awal node dengan nullptr agar node bernilai kosong/0 kemudian kita akan insertkan nilai dengan perulangan for yang dimana ia akan memasukan nilai yang telah kita set dengan beberapa variabel yang kita tentukan sendiri, lalu kita akan membuat show element untuk menunjukan elemen yang ada pada SLL kita untuk mengetes search function lalukita akan buat, searching list dengan inputan dari user yang menggunakan method searchElement dan out put dari kodingan kita adalah sebagai berikut ini:

```cpp
void showElement_2311104031(Node* head) {
    Node* current = head;
    cout << "Elemen-elemen dalam list: ";
    if (current == nullptr) {
        cout << "List kosong." << endl;
        return;
    }

    while (current != nullptr) {
        cout << current->data << " ";
        current = current->next;
    }
    cout << endl;
}

int main() {
    Node* head = nullptr;
    int elements[] = {10, 20, 30, 40, 50, 60};

    for (int i = 0; i < 6; i++) {
        tambahElemen_2311104031(head, elements[i]);
    }

    showElement_2311104031(head);

    int cari;
    cout << "Masukkan elemen yang ingin dicari: ";
    cin >> cari;

    searchElement_2311104031(head, cari);

    return 0;
}
```

```
Elemen-elemen dalam list: 10 20 30 40 50 60
Masukkan elemen yang ingin dicari: 60
Elemen 60 ditemukan di alamat 0x26468cc1000 pada urutan ke 6
```

2 Bubble sort pada SLL, untuk kodingan ini dapat terlihat seperti berikut ini:

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

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

void tambahElemen_2311104031(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 tampilkanList_2311104031(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}
```

```cpp
void bubbleSortList_2311104031(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);
}

int main() {
    Node* head = nullptr;
    int elements[] = {5, 3, 8, 2, 7};

    for (int i = 0; i < 5; i++) {
        tambahElemen_2311104031(head, elements[i]);
    }

    cout << "List sebelum diurutkan: ";
    tampilkanList_2311104031(head);

    bubbleSortList_2311104031(head);

    cout << "List setelah diurutkan: ";
    tampilkanList_2311104031(head);

    return 0;
}
```

sebagai awalan dari setiap SLL kita akan membuat struct node sebagai penyimpanan dari data yang akan kita urutkan/ masukan pada list lalu langkah keduanya kita akan membuat ethod untuk penambahan elemen, kita hanya akan melakukan insertFirst supaya data menjadi acak yang akan di sorting pada method bubble sort nantinya, kemudian setelah kita membuat insertsion untuk SLL kita akan membuat method untuk menunjukan list, ini berguna untuk menunjukan list sebelum dan setelah di sorting oleh kita menggunakan bubble sort, lalu setelah membuat print list kita akan membuat method sorting menggunakan bubble sort sesuai TP 5, bubble sort sendiri merupakan sorting yang dimana ia menggunakan perbandingan antara 2 elemen dan melakukan pertukaran bila diperlukan. Setelah pembuatan dari sorting langkah selanjutnya dalah kita membuat main yang dimana method method yang baru kita buat akan kita implementasikan untuk menjalankan kodingan, hal pertama adalah mengset head sebagai null untuk deklarasi bahwa list tersebut kosong baru kita membuat array dan elemen yang akan kita masukan, setelah kita memasukan elemen menggunakan method tambahElemen kita akan mengeprint list sebelum di sorting dan setelah di sorting dengan output seperti berikut ini:

```
List sebelum diurutkan: 5 3 8 2 7
List setelah diurutkan: 2 3 5 7 8
```

3 Penambahan elemen secara terurut, kodinganya sebagai berikut ini:

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

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

void insertSorted_2311104031(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;
    }
}
```

```cpp
void tampilkanList_2311104031(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    Node* head = nullptr;
    int elements[] = {1, 3, 5, 7};

    for (int i = 0; i < 4; i++) {
        insertSorted_2311104031(head, elements[i]);
    }

    cout << "List sebelum penambahan: ";
    tampilkanList_2311104031(head);

    int newElement;
    cout << "Masukkan elemen baru yang ingin ditambahkan: ";
    cin >> newElement;

    insertSorted_2311104031(head, newElement);

    cout << "List setelah penambahan: ";
    tampilkanList_2311104031(head);

    return 0;
}
```

sebagai permulaan seperti biasa kita akan membuat struct node untuk penempatan data yang akan kita insertkan pada list, lalu kita akan membuat method insertSorted yang dimana kerjanya adalah mengecek masing masing elemen dan membandingkanya dengan data yang akan di insertkan ke dalam list dan menempatkanya sesuai dengan urutan, lalu kita akan membuat method untuk mengeprint list, yang dimana ia berguna untuk menunjukan list sebelum penambahan data dan sesudahnya, lalu setelah method tersebut kita akan masuk kedalam main yang dimana kita akan mengset head dengan nullptr sebagai deklarasi list kosong, kemudian kita akan membuat array tempat penyimpanan list dengan data yang telah ditentukan oleh kita kemudian kita akan menggunakan method insert untuk memasukan bilangan awal ini dan mengeprinya sebagai list seelum penambahan kemudian kita akan membuat insert dari user sebagai new data yang akan dimasukan kedalam list, setelah membuat insert baru dari user kita akan mengeprint ulang data dan menunjukan newdata setelah di insertkan elemen baru dengan output sebagai berikut ini:

```
List sebelum penambahan: 1 3 5 7
Masukkan elemen baru yang ingin ditambahkan: 4
List setelah penambahan: 1 3 4 5 7
```