Telkom University Purwokerto

Aturan Praktikum Struktur Data

1. **Akun GitHub:** Setiap praktikan wajib memiliki akun GitHubyang aktif dan digunakan selama praktikum berlangsung.
2. **Invite Collaborator:** Setiap praktikan diwajibkan untuk menambahkan collaborator di setiap repository
    a. Asisten Praktikum: AndiniNH
    b. AsistenPraktikum:4ldiputra
3. **Repository Praktikum:** Setiap praktikan diwajibkan untuk membuat satu repository di GitHub yang akan digunakan untuk seluruh tugas dan laporan praktikum. Repository ini harus diatur dengan rapi dan sesuai dengan instruksi yang akan diberikan di lampiran.
4. **Penamaan Folder:** Penamaan folderdalam repositoryakan dibahas secara rinci di lampiran. Praktikan wajib mengikuti aturan penamaan yang telah ditentukan.

Nomor	Pertemuan	Penamaan
1	Penganalan Bahasa C++ Bagian Pertama	01_Pengenalan_CPP_Bagian_1
2	Pengenalan Bahasa C++ Bagian Kedua	02_Pengenalan_CPP_Bagian_2
3	Abstract Data Type	03_Abstract_Data_Type
4	Single Linked List Bagian Pertama	04_Single_Linked_List_Bagian_1
5	Single Linked List Bagian Kedua	05_Single_Linked_List_Bagian_2
6	Double Linked List Bagian Pertama	06_Double_Linked_List_Bagian_1
7	Stack	07_Stack
8	Queue	08_Queue
9	Assessment Bagian Pertama	09_Assessment_Bagian_1
10	Tree Bagian Pertama	10_Tree_Bagian_1
11	Tree Bagian Kedua	11_Tree_Bagian_2
12	Asistensi Tugas Besar	12_Asistensi_Tugas_Besar
13	Multi Linked List	13_Multi_Linked_List
14	Graph	14_Graph
15	Assessment Bagian Kedua	15_Assessment_Bagian_2
16	Tugas Besar	16_Tugas_Besar

5. **Jam Praktikum:**
    - Jam masuk praktikum adalah **1 jam lebih lambat** dari jadwal yang tercantum. Sebagai contoh, jika jadwalpraktikumadalah pukul 06.30 - 09.30, maka aturan praktikum akan diatur sebagai berikut:
        - **06.30 - 07.30**: Waktuini digunakan untuk **Tugas Praktikum dan Laporan Praktikum** yang dilakukan di luar laboratorium.
        - **07.30 - 0G.30**: Sesi ini mencakup **tutorial, diskusi, dan kasus problem- solving**. Kegiatan ini berlangsung di dalam laboratorium dengan alokasi waktu sebagai berikut:
            - **60 menit pertama**: Tugas terbimbing.
            - **60 menit kedua**: Tugas mandiri.
6. **Pengumpulan Tugasn Pendahuluan:** Tugas Pendahuluan (TP) wajib dikumpulkan melalui GitHub sesuai dengan format berikut:

    nama_repo/nama_pertemuan/TP_Pertemuan_Ke.md

    Sebagai contoh:

    STD_Yudha_Islalmi_Sulistya_XXXXXXXX/01_Running_Modul/TP_01.md

7. **Pengecekan Tugas Pendahuluan:** Pengumpulan laporan praktikum akan diperiksa **1 hari sebelum praktikum selanjutnya** dimulai. Pastikan tugas telah diunggah tepat waktu untuk menghindari sanksi.

LAPORAN PRAKTIKUM
MODUL 5
SINGLE LINKED LIST (BAGIAN KEDUA)

Disusun Oleh :

Zaenarif Putra ‘Ainurdin – 2311104049

Kelas :

SE-07-02

Dosen :

Wahyu Andi Saputra, S.pd,M.Eng

PROGRAM STUDI SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO

2024

I. TUJUAN

1. Memahami penggunaan linked list dengan pointer operator - operator dalam program.
2. Memahami operasi-operasi dasar dalam linked list.
3. Membuat program dengan menggunakan linked list dengan prototype yang ada

II. LANDASAN TEORI

1. Searching

   Searching merupakan operasi dasar list dengan melakukan aktivitas pencarian terhadap node tertentu. Proses ini berjalan dengan mengunjungi setiap node dan berhenti setelah node yang dicari ketemu. Dengan melakukan operasi searching, operasi-operasi seperti insert after, delete after, dan update akan lebih mudah.

III. GUIDE

1. Guide5M

   a. Syntax

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

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

void insertFirst(Node*& head, Node*& tail, int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = head;
    head = newNode;

    if (tail == nullptr) {
        tail = newNode;
    }
}

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

    if (tail) {
        tail->next = newNode;
    } else {
        head = newNode;
    }
    tail = newNode;
}

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

int findElement(Node* head, int x) {
    int index = 0;
    Node* current = head;
    while (current != nullptr) {
        if (current->data == x) {
            return index;
        }
        current = current->next;
        index++;
    }
    return -1;
}

void deleteElement(Node*& head, int x) {
    Node* current = head;
    Node* previous = nullptr;

    while (current != nullptr && current->data != x) {
        previous = current;
        current = current->next;
    }

    if (current == nullptr) {
        cout << "Elemen tidak ditemukan untuk dihapus." << endl;
        return;
    }

    if (previous == nullptr) {
        head = current->next;
    } else {
        previous->next = current->next;
    }
    delete current;
}

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

    insertFirst(head, tail, 3);
    insertFirst(head, tail, 5);
    insertFirst(head, tail, 7);

    insertLast(head, tail, 11);
    insertLast(head, tail, 14);
    insertLast(head, tail, 18);

    cout << "Elemen dalam linked list: ";
    display(head);

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

    int result = findElement(head, x);

    if (result == -1)
        cout << "Elemen tidak ditemukan dalam linked list" << endl;
    else
        cout << "Elemen ditemukan pada indeks " << result << endl;

    cout << "Masukkan elemen yang ingin dihapus: ";
    cin >> x;
    deleteElement(head, x);

    cout << "Elemen dalam linked list setelah penghapusan: ";
    display(head);

    return 0;
}
```

b. Output

```
PS C:\Users\LENOVO\OneDrive - Telkom University\Documents\ALL Matkul\StrukturData\pertemuan5\guide\output>
'

Elemen dalam linked list: 7 5 3 11 14 18
Masukkan elemen yang ingin dicari: 3
Elemen ditemukan pada indeks 2
Masukkan elemen yang ingin dihapus: 3
Elemen dalam linked list setelah penghapusan: 7 5 11 14 18
PS C:\Users\LENOVO\OneDrive - Telkom University\Documents\ALL Matkul\StrukturData\pertemuan5\guide\output>
```

Telkom University Purwokerto

IV. UNGUIDED

1. Task1

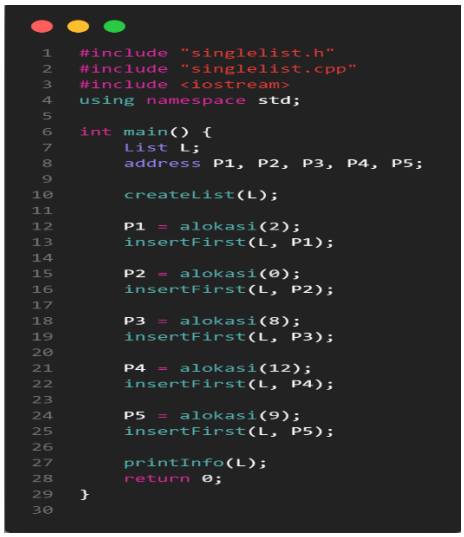
   a. Syntax “singlelist.h”

```h
#ifndef SINGLELIST_H_INCLUDED
#define SINGLELIST_H_INCLUDED

typedef int infotype;
typedef struct ElmList *address;

struct ElmList {
    infotype info;
    address next;
};

struct List {
    address first;
};

void createList(List &L);
address alokasi(infotype x);
void dealokasi(address &P);
void printInfo(List L);
void insertFirst(List &L, address P);
address findElm(List L, infotype x);
int sumInfo(List L);

#endif
```

Syntax “singlelist.cpp”

```cpp
#include "singlelist.h"
#include <iostream>
using namespace std;

void createList(List &L) {
    L.first = NULL;
}

address alokasi(infotype x) {
    address P = new ElmList;
    P->info = x;
    P->next = NULL;
    return P;
}

void dealokasi(address &P) {
    delete P;
}

void printInfo(List L) {
    address P = L.first;
    cout << "Isi List: ";
    while (P != NULL) {
        cout << P->info << " ";
        P = P->next;
    }
    cout << endl;
}

void insertFirst(List &L, address P) {
    P->next = L.first;
    L.first = P;
}

address findElm(List L, infotype x) {
    address P = L.first;
    while (P != NULL) {
        if (P->info == x) {
            return P;
        }
        P = P->next;
    }
    return NULL;
}

int sumInfo(List L) {
    int total = 0;
    address P = L.first;
    while (P != NULL) {
        total += P->info;
        P = P->next;
    }
    return total;
}
```

Syntax “main.cpp”



```cpp
#include "singlelist.h"
#include "singlelist.cpp"
#include <iostream>
using namespace std;

int main() {
    List L;
    address P1, P2, P3, P4, P5;

    createList(L);

    P1 = alokasi(2);
    insertFirst(L, P1);

    P2 = alokasi(0);
    insertFirst(L, P2);

    P3 = alokasi(8);
    insertFirst(L, P3);

    P4 = alokasi(12);
    insertFirst(L, P4);

    P5 = alokasi(9);
    insertFirst(L, P5);

    printInfo(L);
    return 0;
}
```

b. Penjelasan Syntax

* Syntax singlelist.h

File *.h* digunakan sebagai pendefinisian dari struktur data dan fungsi yang akan digunakan pada program user. Contohnya yaitu dengan menggunakan linked list.

#ifndefSINGLELIST_H_INCLUDED,#defineSINGLELIST_H_INCLUDED, #endif : digunakan untuk mencegah file header dimasukkan lebih dari sekali pada program yang ingin dibuat. Jika user memasukkan lebih dari sekali, maka akan menyebabkan kesalahan pada saat melakukan kompilasi.

typedef int infotype; : Kita membuat nama baru, infotype, yang berarti int. Ini akan digunakan untuk menyimpan data dalam elemen daftar. Sedangkan typedef struct ElmList *address; : Kita mendefinisikan address sebagai pointer ke elemen daftar (ElmList). Ini membantu kita merujuk ke elemen-elemen dalam daftar.

struct ElmList { infotype info; address next; }; : Digunakan sebagai mempresentasikan satu element dalam linked list. Yang dimana terdapat komponen yaitu infotype info; digunakan untuk menyimpan data (seperti angka), dan address next; digunakan untuk menyimpan alamat elemen berikutnya dalam daftar. Ini memungkinkan kita untuk menghubungkan elemen-elemen.

struct List { address first; }; : Berikut merupakan struktur yang dimana untuk mempresentasikan keseluruhan pada linked list. address first;: Menyimpan alamat elemen pertama dalam daftar. Ini penting agar kita bisa mulai mengakses elemen-elemen dalam daftar.

Fungsi createList(List &L) membuat daftar baru yang kosong, sedangkan

alokasi(infotype x) mengalokasikan memori untuk elemen baru dengan nilai x. Jika elemen tidak diperlukan, kita bisa menggunakan dealokasi(address &P) untuk menghapusnya dari memori. Untuk mencetak semua data dalam daftar, kita menggunakan printInfo(List L). Fungsi insertFirst(List &L, address P) menambahkan elemen baru di awal daftar, dan findElm(List L, infotype x) mencari elemen dengan nilai x. Terakhir, sumInfo(List L) menghitung total nilai dalam daftar. Fungsi-fungsi ini memudahkan berbagai operasi pada linked list.

* Syntax singlelist.cpp

#include "singlelist.h": Baris ini menyertakan file header singlelist.h, yang berisi definisi struktur data dan fungsi-fungsi terkait linked list.

using namespace std;: Baris ini memungkinkan kita untuk menggunakan elemen dari namespace std tanpa harus menuliskan std:: setiap kali, seperti cout dan cin.

void createList(List &L): Fungsi ini digunakan untuk membuat daftar baru. Di dalamnya, L.first diatur menjadi NULL, yang berarti daftar tersebut kosong.

address alokasi(infotype x): Fungsi ini mengalokasikan memori untuk elemen baru. Di dalam fungsi, objek ElmList baru dibuat, nilai x disimpan dalam info, dan next diatur menjadi NULL. Fungsi ini mengembalikan alamat elemen baru yang telah dibuat.

void dealokasi(address &P): Fungsi ini digunakan untuk menghapus elemen dari memori. Parameter P adalah alamat dari elemen yang akan dihapus, dan fungsi ini menggunakan delete untuk membebaskan memori yang digunakan oleh elemen tersebut.

void printInfo(List L): Fungsi ini mencetak semua data yang ada dalam daftar. Dengan menggunakan loop while, fungsi ini mengiterasi melalui setiap elemen dalam daftar, mencetak nilai info dari setiap elemen hingga mencapai akhir daftar (ketika P menjadi NULL).

void insertFirst(List &L, address P): Fungsi ini menambahkan elemen baru di awal daftar. Elemen baru yang ditunjuk oleh P dihubungkan ke elemen pertama yang ada, dan kemudian L.first diupdate untuk menunjuk ke elemen baru ini.

address findElm(List L, infotype x): Fungsi ini mencari elemen dalam daftar yang memiliki nilai sama dengan x. Jika elemen ditemukan, alamat elemen tersebut dikembalikan; jika tidak, fungsi ini mengembalikan NULL.

int sumInfo(List L): Fungsi ini menghitung total dari semua nilai dalam daftar. Dengan menggunakan loop while, fungsi ini menjumlahkan nilai info dari setiap elemen hingga mencapai akhir daftar, dan mengembalikan totalnya.

* Syntax main.cpp

#include "singlelist.h" dan #include "singlelist.cpp": Baris ini menyertakan file header singlelist.h dan implementasi singlelist.cpp. File header berisi definisi struktur data dan deklarasi fungsi, sedangkan file implementasi berisi kode untuk fungsi-fungsi tersebut.

using namespace std;: Baris ini memungkinkan kita untuk menggunakan elemen dari namespace std tanpa harus menuliskan std:: setiap kali, seperti cout.

int main() {: Ini adalah titik masuk dari program. Fungsi main adalah tempat dimana inti program dimulai.

List L;: Di sini, kita mendeklarasikan sebuah variabel L dari tipe List, yang merupakan struktur data untuk linked list.

address P1, P2, P3, P4, P5;: Kita mendeklarasikan beberapa variabel bertipe address, yang akan digunakan untuk menyimpan alamat dari elemen-elemen yang akan ditambahkan ke dalam linked list.

createList(L);: Fungsi ini dipanggil untuk menginisialisasi daftar L. Setelah pemanggilan ini, L.first akan diset ke NULL, menandakan bahwa daftar saat ini kosong.

P1 = alokasi(2);: Fungsi alokasi dipanggil untuk membuat elemen baru dengan nilai 2. Alamat elemen baru ini disimpan dalam variabel P1.

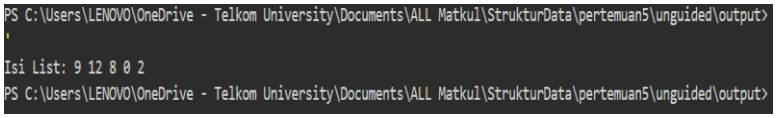
insertFirst(L, P1);: Fungsi ini digunakan untuk menambahkan elemen yang ditunjuk oleh P1 ke awal daftar L.

Langkah 8 dan 9 diulang untuk P2, P3, P4, dan P5: Elemen-elemen baru dengan nilai 0, 8, 12, dan 9 dibuat dan ditambahkan ke awal daftar dengan cara yang sama. Setiap kali, elemen baru ditambahkan di depan elemen yang sudah ada, sehingga urutan elemen dalam daftar adalah 9, 12, 8, 0, dan 2.

printInfo(L);: Fungsi ini dipanggil untuk mencetak semua nilai yang ada dalam daftar L. Fungsi ini akan mengiterasi melalui setiap elemen dalam daftar dan mencetak nilai info dari setiap elemen.

return 0;: Baris ini menandakan bahwa program selesai dijalankan dengan sukses. Nilai 0 biasanya menunjukkan bahwa tidak ada kesalahan yang terjadi selama eksekusi.

c. Output



2. Task2 elemen dengan info 8 dengan membuat fungsi baru.

   a. Syntax

```
address hasil = findElm(L, 8);
    if (hasil != NULL) {
        cout << "Elemen 8 berhasil ditemukan!" << endl;
    } else {
        cout << "Elemen 8 tidak ditemukan!" << endl;
    }
```

   b. Penjelasan Syntax

   address hasil = findElm(L, 8);: Memanggil fungsi findElm untuk mencari elemen dengan nilai 8 dalam daftar L. Alamat elemen yang ditemukan disimpan dalam variabel hasil, atau NULL jika tidak ditemukan.

   if (hasil != NULL) {: Memeriksa apakah hasil tidak NULL, yang berarti elemen ditemukan.

   cout << "Elemen 8 berhasil ditemukan!" << endl;: Jika elemen ditemukan, mencetak pesan bahwa elemen 8 berhasil ditemukan.

   } else {: Jika hasil adalah NULL, maka blok ini akan dieksekusi.

   cout << "Elemen 8 tidak ditemukan!" << endl;: Jika elemen tidak ditemukan, mencetak pesan bahwa elemen 8 tidak ditemukan.

   c. Output

```
PS C:\Users\LENOVO\OneDrive - Telkom University\Documents\ALL Matkul\StrukturData\pertemuan5\unguided\output>
'

Elemen 8 berhasil ditemukan!
PS C:\Users\LENOVO\OneDrive - Telkom University\Documents\ALL Matkul\StrukturData\pertemuan5\unguided\output>
```

3. Task3 jumlah total info seluruh elemen

   a. Syntax

```
int total = sumInfo(L);
    cout << "Total semua elemen: " << total << endl;
```

b. Penjelasan Syntax

int total = sumInfo(L);: Memanggil fungsi sumInfo untuk menghitung jumlah semua elemen dalam linked list L dan menyimpan hasilnya dalam variabel total.

cout << "Total semua elemen: " << total << endl;: Mencetak pesan "Total semua elemen: " diikuti dengan nilai total ke layar.

c. Output

```
PS C:\Users\LENOVO\OneDrive - Telkom University\Documents\ALL Matkul\StrukturData\pertemuan5\unguided\output>
'

Total semua elemen: 31
PS C:\Users\LENOVO\OneDrive - Telkom University\Documents\ALL Matkul\StrukturData\pertemuan5\unguided\output>
```

4. Berikut Output full *main.cpp* jika semuanya disatukan :

```
Isi List: 9 12 8 0 2
Elemen 8 berhasil ditemukan!
Total semua elemen: 31
PS C:\Users\LENOVO\OneDrive - Telkom University\Documents\ALL Matkul\StrukturData\pertemuan5\unguided\output>
```

V. KESIMPULAN

Pada praktikum modul ke-5 ini membahas tentang Single Linked List dengan fokus pada operasi searching dan implementasi ADT (Abstract Data Type). Materi utama yang dibahas meliputi operasi searching untuk mencari node tertentu dalam linked list, serta implementasi ADT Single Linked List yang terdiri dari komponen infotype (integer) dan pointer. Selain itu, modul ini juga mencakup operasi dasar seperti createList, alokasi, dealokasi, insertFirst, dan printInfo. Melalui latihan praktikum, mahasiswa diminta untuk membuat implementasi ADT Single Linked List dan mengaplikasikan operasi-operasi dasar seperti pencarian elemen dan perhitungan total info. Tujuan akhirnya adalah untuk memahami penggunaan linked list dengan pointer dan operasi-operasi dasarnya dalam pemrograman.