LAPORAN PRAKTIKUM

MODUL 5

SINGLE LINKED LIST (BAGIAN KEDUA)

Disusun Oleh:

Muhammad Ikhsan Al Hakim (2311104064)

S1SE-07-02

Dosen :

Wahyu Andi Saputra, S.Pd., M.Eng

PROGRAM STUDI S1 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

2.1 Searching

Searching atau pencarian dalam pemrograman C++ adalah proses menemukan suatu elemen spesifik dalam sebuah struktur data, seperti array atau linked list. Elemen yang dicari bisa berupa angka, karakter, atau bahkan struktur data yang lebih kompleks..

Semua fungsi dasar diatas merupakan bagian dari ADT dari singgle *linked list*, dan aplikasi pada bahasa pemrograman Cp semua ADT tersebut tersimpan dalam *file* *.c dan *file* *.h

III. GUIDED

Codingan Searching Latihan Minggu ke-5

Code:

```cpp
#include <iostream>

using namespace std;

// Struktur untuk node dalam linked list

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

// Fungsi untuk menambahkan elemen baru ke awal linked list 11
void insertFirst(Node*& head, Node*& tail, int new_data){
   Node* new_node = new Node();
   new_node->data= new_data;
   new_node->next = head;
   head = new_node;

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

// fungsi untuk menambahkan elemen baru ke akhir linked list
void insertLast(Node*& head, Node*& tail, int new_data){
    Node*new_node = new Node ();
    new_node->data = new_data;
    new_node->next = nullptr;

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

// fungsi untuk mencari elemen dalam linked list
int findElement(Node* head, int x){
    Node* current = head;
    int index = 0;

    while(current != nullptr){
        if (current->data == x){
            return index;
        }
        current = current->next;
        index++;
    }
    return -1;
}

// fungsi untuk menampilkan elemen dalam linked list
void display(Node* node){
    while (node != nullptr){
        cout << node->data << " ";
        node = node->next;
    }
    cout << endl;
}

// fungsi untuk menghapus elemen dari lnked list
void deleteELement(Node*& head, int x){
    if (head == nullptr){
        cout << "Linked List kosong" << endl;
        return;
    }

    if (head->data == x){
        Node* temp = head;
        head = head->next;
        delete temp;
        return;
    }

    Node* current = head;
    while(current -> next != nullptr){
        if(current->next->data == x){
            Node* temp = current->next;
            current->next = current->next->next;
            delete temp;
            return;
        }
        current = current->next;
    }
}

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;
}
```

Output:

```
Elemen dalam linked list: 7 5 3 11 14 18
Masukkan elemen yang ingin dicari: 11
Elemen ditemukan pada indeks 3
Masukkan elemen yang ingin dihapus: 3
Elemen dalam linked list setelah penghapusan: 7 5 11 14 18
PS C:\Praktikum Struktur data\pertemuan5>
```

IV. UNGUIDED

Buatlah ADT *Single Linked list* sebagai berikut di dalam *file* **"singlelist.h"**.

Jawaban:

a) Singlelist.

h

Code:

```h
#ifndef SINGLELIST_H
#define SINGLELIST_H

// Definisikan infotype sebagai int dan address sebagai pointer ke ElmList
typedef int infotype;
typedef struct ElmList *address;

// Definisikan struktur untuk setiap elemen (node)
struct ElmList {
    infotype info;    // Menyimpan nilai dari node
    address next;     // Pointer ke node berikutnya
};

// Definisikan struktur untuk List
struct List {
    address First;    // Menunjuk ke node pertama dalam list
};

// Deklarasi fungsi dan prosedur
void createList(List &L);
address alokasi(infotype x);
void dealokasi(address &P);
void printInfo(const List &L);
void insertFirst(List &L, address P);
address findElm(const List &L, infotype x);
int totalInfo(const List &L);

#endif
```

b) Singlelist

Code:

```cpp
#include <iostream>
#include "singlelist.h"

using namespace std;

// Membuat list kosong
void createList(List &L) {
    L.First = nullptr;
}

// Mengalokasikan node baru dengan nilai info tertentu
address alokasi(infotype x) {
    address P = new ElmList;
    P->info = x;
    P->next = nullptr;
    return P;
}

// Dealokasi atau membebaskan memori dari node
void dealokasi(address &P) {
    delete P;
    P = nullptr;
}

// Menampilkan info dari semua elemen dalam list
void printInfo(const List &L) {
    address P = L.First;
    while (P != nullptr) {
        cout << P->info << " ";
        P = P->next;
    }
    cout << endl;
}

// Menyisipkan node baru di awal list
void insertFirst(List &L, address P) {
    P->next = L.First;
    L.First = P;
}

// Mencari elemen dengan nilai tertentu
address findElm(const List &L, infotype x) {
    address P = L.First;
    while (P != nullptr && P->info != x) {
        P = P->next;
    }
    return P;
}

// Menghitung total info dari semua elemen
int totalInfo(const List &L) {
    int total = 0;
    address P = L.First;
    while (P != nullptr) {
        total += P->info;
        P = P->next;
    }
    return total;
}

```

c) Main

Code:

```cpp
#include <iostream>
#include "singlelist.h"

using namespace std;

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

    // Membuat list kosong
    createList(L);

    // Menyisipkan beberapa elemen ke dalam list
    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);

    // Menampilkan semua elemen dalam list
    cout << "Isi List: ";
    printInfo(L);

    // Mencari elemen dengan info 8
    address found = findElm(L, 8);
    if (found != nullptr) {
        cout << "Elemen dengan info 8 ditemukan: " << found->info << endl;
    } else {
        cout << "Elemen dengan info 8 tidak ditemukan" << endl;
    }

    // Menghitung total info dari semua elemen
    int total = totalInfo(L);
    cout << "Total nilai semua elemen: " << total << endl;

    return 0;
}
```

Output Main:

```
Isi List: 9 12 8 0 2
Elemen dengan info 8 ditemukan: 8
Total nilai semua elemen: 31
PS C:\Praktikum Struktur data\pertemuan5>
```