

LAPORAN PRAKTIKUM
Modul 5
Single Linked List Bagian 2



Disusun Oleh :
Satria Ariq Adelard Dompas/2211104033
SE 06 2

Asisten Praktikum :
Aldi Putra
Andini Nur Hidayah

Dosen Pengampu :
Wahyu Andi Saputra

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024

1. Tujuan

- Mahasiswa dapat memahami penggunaan linked list dengan pointer operator dalam program.
- Mahasiswa dapat memahami operasi dasar dalam linked list.
- Mahasiswa dapat membuat program dengan menggunakan linked list dengan prototype yang disediakan.

2. Landasan Teori

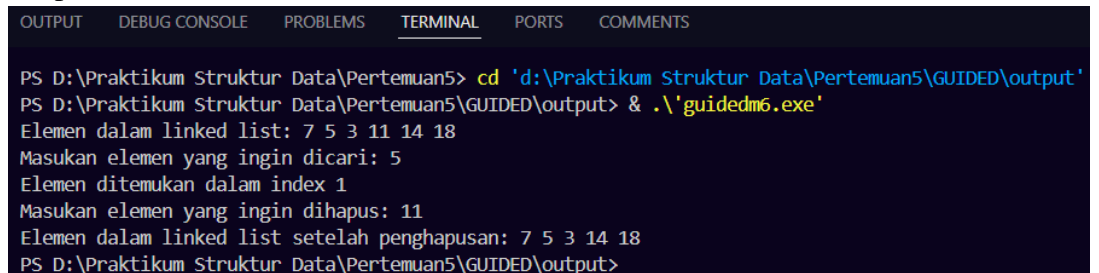
a. Searching

Pencarian merupakan operasi daftar dasar dengan melakukan operasi pencarian pada node tertentu. Proses ini berjalan dengan mengunjungi setiap node dan berhenti ketika menemukan node yang dicari. Dengan melakukan operasi pencarian, operasi seperti menyisipkan nanti, menghapus nanti, dan memperbarui menjadi lebih mudah.

3. Guided

a. Searching

Output



```
OUTPUT  DEBUG CONSOLE  PROBLEMS  TERMINAL  PORTS  COMMENTS
PS D:\Praktikum Struktur Data\Pertemuan5> cd 'd:\Praktikum Struktur Data\Pertemuan5\GUIDED\output'
PS D:\Praktikum Struktur Data\Pertemuan5\GUIDED\output> & .\'guidedm6.exe'
Elemen dalam linked list: 7 5 3 11 14 18
Masukan elemen yang ingin dicari: 5
Elemen ditemukan dalam index 1
Masukan elemen yang ingin dihapus: 11
Elemen dalam linked list setelah penghapusan: 7 5 3 14 18
PS D:\Praktikum Struktur Data\Pertemuan5\GUIDED\output>
```

Source Code

```

#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
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;
}

```

4. Unguided

a. Soal 1

Source Code

singlelist.h

```
#ifndef SINGLELIST_H
#define SINGLELIST_H

#include <iostream>

using namespace std;

typedef int infotype;
typedef struct elmlist *address;

struct elmlist {
    infotypea info;
    address next;
};

struct List {
    address first;
};

// Prosedur pembuatan list kosong
void createList(List &L);

// Fungsi alokasi elemen baru
address alokasi(infotype x);

// Prosedur dealokasi elemen
void dealokasi(address P);

// Prosedur untuk menampilkan info semua elemen dalam list
void printInfo(List L);

// Prosedur untuk menambahkan elemen di awal list
void insertFirst(List &L, address P);

// Fungsi untuk mencari elemen dengan info tertentu
address findElm(List L, infotype x);

// Fungsi untuk menghitung total info semua elemen
int totalInfo(List L);
```

singlelist.cpp

```
#include "singlelist.h"

// Prosedur pembuatan list kosong
void createList(List &L) {
    L.first = NULL;
}

// Fungsi alokasi elemen baru
address alokasi(infotype x) {
    address P = new elmList;
    P->info = x;
    P->next = NULL;
    return P;
}

// Prosedur dealokasi elemen
void dealokasi(address P) {
    delete P;
}

// Prosedur untuk menampilkan info semua elemen dalam list
void printInfo(List L) {
    address P = L.first;
    while (P != NULL) {
        cout << P->info << " ";
        P = P->next;
    }
    cout << endl;
}

// Prosedur untuk menambahkan elemen di awal list
void insertFirst(List &L, address P) {
    P->next = L.first;
    L.first = P;
}

// Fungsi untuk mencari elemen dengan info tertentu
address findElm(List L, infotype x) {
    address P = L.first;
    while (P != NULL) {
        if (P->info == x)
            return P;
        P = P->next;
    }
    return NULL; // Jika tidak ditemukan
}

// Fungsi untuk menghitung total info semua elemen
int totalInfo(List L) {
    address P = L.first;
    int total = 0;
    while (P != NULL) {
        total += P->info;
        P = P->next;
    }
    return total;
}
```

main.cpp

```
#include "singlelist.h"

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

    // Buat list kosong
    createList(L);

    // Alokasi dan insert 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);

    // Tampilkan semua elemen dalam list
    printInfo(L);

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

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

    return 0;
}
```

Output

```
OUTPUT  DEBUG CONSOLE  PROBLEMS  TERMINAL  PORTS  COMMENTS
PS D:\Praktikum Struktur Data\Pertemuan5> cd UNGUIDED
PS D:\Praktikum Struktur Data\Pertemuan5\UNGUIDED> g++ main.cpp singlelist.cpp -o output_program
PS D:\Praktikum Struktur Data\Pertemuan5\UNGUIDED> ./output_program
9 12 8 0 2
Elemen dengan info 8 ditemukan.
Total info semua elemen: 31
PS D:\Praktikum Struktur Data\Pertemuan5\UNGUIDED> █
```

Deskripsi

Codingan diatas merupakan implementasi dari struktur data Single Linked List dalam C++. Struktur data ini terdiri dari elemen-elemen yang saling terhubung di mana setiap elemen menyimpan sebuah nilai integer (`info`) dan pointer (`next`) ke elemen berikutnya.

- createList: Menginisialisasi list kosong dengan menetapkan pointer `first` ke `NULL`.
- alokasi: Membuat elemen baru dengan nilai `info` tertentu dan mengembalikan alamat dari elemen tersebut.
- dealokasi: Menghapus elemen yang telah dialokasikan dari memori.
- printInfo: Menampilkan semua elemen dalam list dengan mencetak nilai `info` tiap elemen.
insertFirst: Menambahkan elemen baru di awal list.
- findElm: Mencari elemen dengan nilai `info` tertentu dalam list dan mengembalikan alamatnya jika ditemukan.
- totalInfo: Menghitung total dari semua nilai `info` elemen dalam list.

Di bagian `main`:

- Dibuat list kosong dan diisi dengan beberapa elemen menggunakan fungsi `alokasi` dan `insertFirst`.
- List ditampilkan dengan `printInfo`.
- Dilakukan pencarian elemen dengan nilai `info` 8 menggunakan `findElm`.
- Total semua nilai `info` elemen dihitung dengan `totalInfo`.