LAPORAN PRAKTIKUM
Modul 5
Single Linked List(2)

Disusun Oleh:
Jauhar Fajar Zuhair
2311104072
S1SE-07-2

Dosen :
Wahyu Andri Saputra, S.Pd., M.Eng.

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO

2024

Tujuan Praktikum

1. Menguasai implementasi pointer dan operator dalam struktur linked list
2. Memahami operasi fundamental linked list seperti insert, delete, dan traversal
3. Mengimplementasikan program linked list sesuai dengan prototype yang ditentukan

Landasan Teori

Searching adalah operasi fundamental dalam linked list yang berfungsi untuk menemukan node spesifik dalam struktur data. Mekanismenya melibatkan traversal sekuensial, dimana proses pencarian akan menelusuri setiap node hingga menemukan node target yang dicari. Keberadaan operasi searching sangat penting karena menjadi dasar untuk operasi-operasi lanjutan seperti:

- Insert after (penyisipan setelah node tertentu)
- Delete after (penghapusan setelah node tertentu)
- Update (pembaruan nilai node)

Dalam implementasinya menggunakan bahasa C++, seluruh operasi dasar ini merupakan bagian dari Abstract Data Type (ADT) single linked list yang terbagi dalam dua file:

- File header (.h) untuk deklarasi
- File implementasi (.cpp) untuk definisi fungsi

Guided

```cpp
#include <iostream>
using namespace std;
struct Node {
    int data;
    Node* next;
};

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;
    }
}
void insertLast(Node*& head, Node*& tail, int new_data) {
    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = nullptr;  // Use nullptr instead of NULL

    if (head == nullptr) {
        head = new_node;
        tail = new_node;
    } else {
        tail->next = new_node;
        tail = new_node;
    }
}
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;
}
void display(Node* node) {
    while (node != nullptr) {
        cout << node->data << " ";
        node = node->next;
    }
    cout << endl;
}

void deleteElement(Node*& head, Node*& tail, int x) {  // Add tail parameter
    if (head == nullptr) {
        cout << "Linked list Kosong" << endl;
        return;
    }
    if (head->data == x) {
        Node* temp = head;
        head = head->next;
        delete temp;
        if (head == nullptr) {  // Update tail if list becomes empty
            tail = nullptr;
        }
        return;
    }
    Node* current = head;
    while (current->next != nullptr) {
        if (current->next->data == x) {
            Node* temp = current->next;
            current->next = current->next->next;
            if (temp == tail) {  // Update tail if last element is deleted
                tail = current;
            }
            delete temp;
            return;
        }
        current = current->next;
    }
    cout << "Elemen " << x << " tidak ditemukan dalam linked list" << endl;
}
int main() {
    Node* head = nullptr;
    Node* tail = nullptr;

    insertFirst(&head, &tail, new_data:3);
    insertFirst(&head, &tail, new_data:5);
    insertFirst(&head, &tail, new_data:7);
    insertLast(&head, &tail, new_data:11);
    insertLast(&head, &tail, new_data:14);
    insertLast(&head, &tail, new_data:18);

    cout << "Elemen dalam linkedlist: ";
    display(head);
    int x;
    cout << "Masukan 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 << "Masukan Elemen yang ingin dihapus: ";
    cin >> x;
    deleteElement(&head, &tail, x);  // Pass tail to deleteElement
    cout << "Elemen dalam linked list setelah penghapusan: ";
    display(head);

    // Free memory before exiting
    while (head != nullptr) {
        Node* temp = head;
        head = head->next;
        delete temp;
    }

    return 0;
}
```

Struktur utama program menggunakan struct Node yang memiliki dua komponen:

- data: menyimpan nilai integer
- next: pointer yang menunjuk ke node berikutnya

Program menggunakan dua pointer utama:

- head: menunjuk ke node pertama
- tail: menunjuk ke node terakhir Penggunaan kedua pointer ini memungkinkan operasi yang lebih efisien, terutama untuk penyisipan di akhir list.

Fungsi-fungsi utama dalam program:

1. insertFirst(): Menyisipkan node baru di awal list
2. insertLast(): Menyisipkan node baru di akhir list
3. findElement(): Mencari posisi (indeks) suatu nilai dalam list
4. display(): Menampilkan seluruh isi list
5. deleteElement(): Menghapus node dengan nilai tertentu

Dalam fungsi main(), program mendemonstrasikan penggunaan linked list dengan:

1. Membuat list kosong
2. Menyisipkan beberapa nilai (3, 5, 7 di awal dan 11, 14, 18 di akhir)
3. Menampilkan isi list
4. Meminta input untuk pencarian elemen
5. Meminta input untuk penghapusan elemen
6. Menampilkan hasil setelah penghapusan
7. Membersihkan memori sebelum program berakhir

unguided

singlelist.h (header)

```h
#ifndef SINGLELIST_H
#define SINGLELIST_H

#include ...

typedef int infotype;
typedef struct ElmList *address;

struct ElmList {
    infotype info;
    address next;
};

struct List {
    address First;
};

// Deklarasi prosedur dan fungsi
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 #ifndef SINGLELIST_H
```

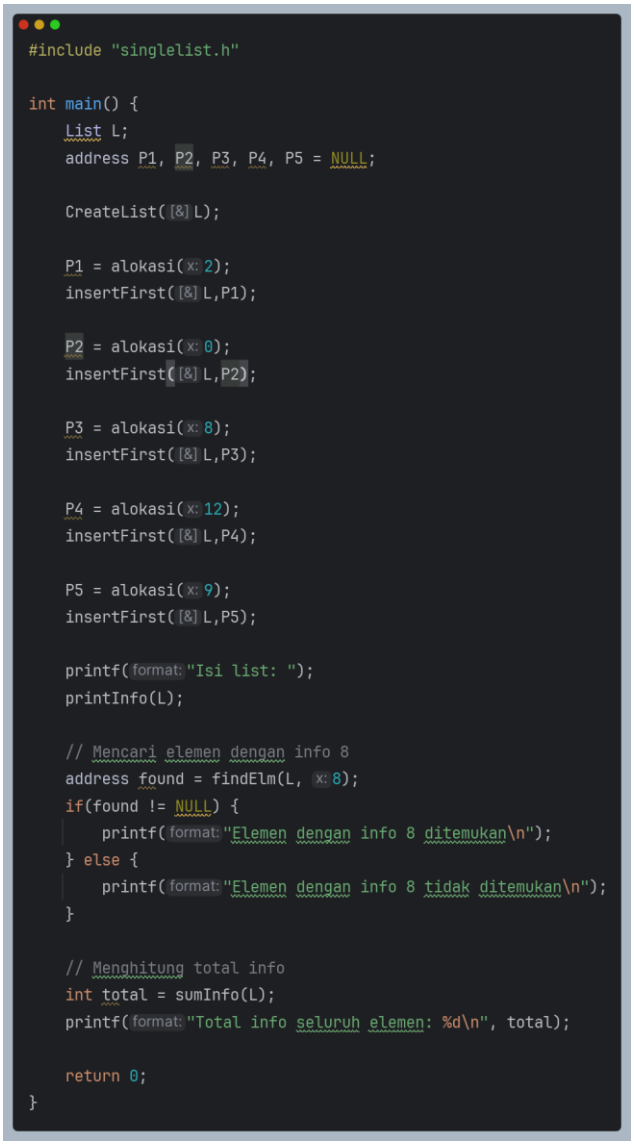
- File header ini berisi deklarasi struktur data dan prototype fungsi yang digunakan
- Mendefinisikan tipe data:
    - infotype: alias untuk tipe data integer
    - address: pointer ke struktur ElmList
- Memiliki dua struktur utama:
    - ElmList: node yang berisi info (data) dan pointer next
    - List: struktur yang menyimpan pointer First (head dari list)
- Mendeklarasikan prototype fungsi-fungsi yang akan diimplementasikan

Singlelist.cpp(implementation)

```cpp
#include "singlelist.h"

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;
    P = NULL;
}

void printInfo(List L) {
    address P = L.First;
    while(P != NULL) {
        printf(format: "%d ", P->info);
        P = P->next;
    }
    printf(format: "\n");
}

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 sum = 0;
    address P = L.First;
    while(P != NULL) {
        sum += P->info;
        P = P->next;
    }
    return sum;
}
```

- Berisi implementasi dari fungsi-fungsi yang dideklarasikan di header
- Fungsi-fungsi yang diimplementasikan:
    - CreateList: Inisialisasi list kosong
    - alokasi: Membuat node baru dengan data tertentu
    - dealokasi: Menghapus node dari memori
    - printInfo: Menampilkan seluruh isi list
    - insertFirst: Menyisipkan node di awal list
    - findElm: Mencari node dengan nilai tertentu
    - sumInfo: Menghitung total nilai dalam list

main.cpp(main)



```
#include "singlelist.h"

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

    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);

    printf("Isi list: ");
    printInfo(L);

    // Mencari elemen dengan info 8
    address found = findElm(L, 8);
    if(found != NULL) {
        printf("Elemen dengan info 8 ditemukan\n");
    } else {
        printf("Elemen dengan info 8 tidak ditemukan\n");
    }

    // Menghitung total info
    int total = sumInfo(L);
    printf("Total info seluruh elemen: %d\n", total);

    return 0;
}
```

- File yang berisi fungsi main untuk menjalankan program
- Mendemonstrasikan penggunaan linked list dengan:
    - Membuat list kosong
    - Menyisipkan 5 nilai (2, 0, 8, 12, 9) ke dalam list
    - Menampilkan isi list
    - Mencari elemen dengan nilai 8
    - Menghitung dan menampilkan total nilai dalam list