LAPORAN PRAKTIKUM

Modul 05

“SINGLE LINKED LIST (BAGIAN KEDUA)”

Disusun Oleh:

Marvel Sanjaya Setiawan (2311104053)

SE-07-02

Dosen :

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

PURWOKERTO

2024

1. **Tujuan**
    - Memahami dan menggunakan linked list dalam pemrograman.
    - Menguasai operasi dasar linked list.
    - Membuat program dengan linked list.
2. **Landasan Teori**

    SLL (Bagian Kedua)

    Searching adalah proses menemukan node spesifik dalam linked list dengan memeriksa setiap node secara berurutan hingga ditemukan atau mencapai akhir list.

    Operasi searching mendukung operasi lain seperti penyisipan, penghapusan, dan pembaruan data pada node yang telah ditemukan.

Telkom University Purwokerto

3. Guided

```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;
   }
}

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;
    }
}

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, 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;
}
```

```
Elemen dalam linked list: 7 5 3 11 14 18
Masukkan elemen yang ingin dicari: 14
Elemen ditemukan pada indeks 4
```

```
Masukkan elemen yang ingin dihapus: 14
Elemen dalam linked list setelah penghapusan: 7 5 3 11 18
```

Cara kerja program:

1. Inisialisasi: Linked list diinisialisasi dengan head dan tail bernilai nullptr.
2. Penambahan elemen: Elemen-elemen ditambahkan menggunakan insertFirst dan insertLast.
3. Pencarian elemen: Elemen dicari menggunakan findElement.
4. Penghapusan elemen: Elemen dihapus menggunakan deleteElement.
5. Penampilkan elemen: Elemen-elemen ditampilkan menggunakan display.

4. Unguided

1. Membuat ADT Single Linked List.

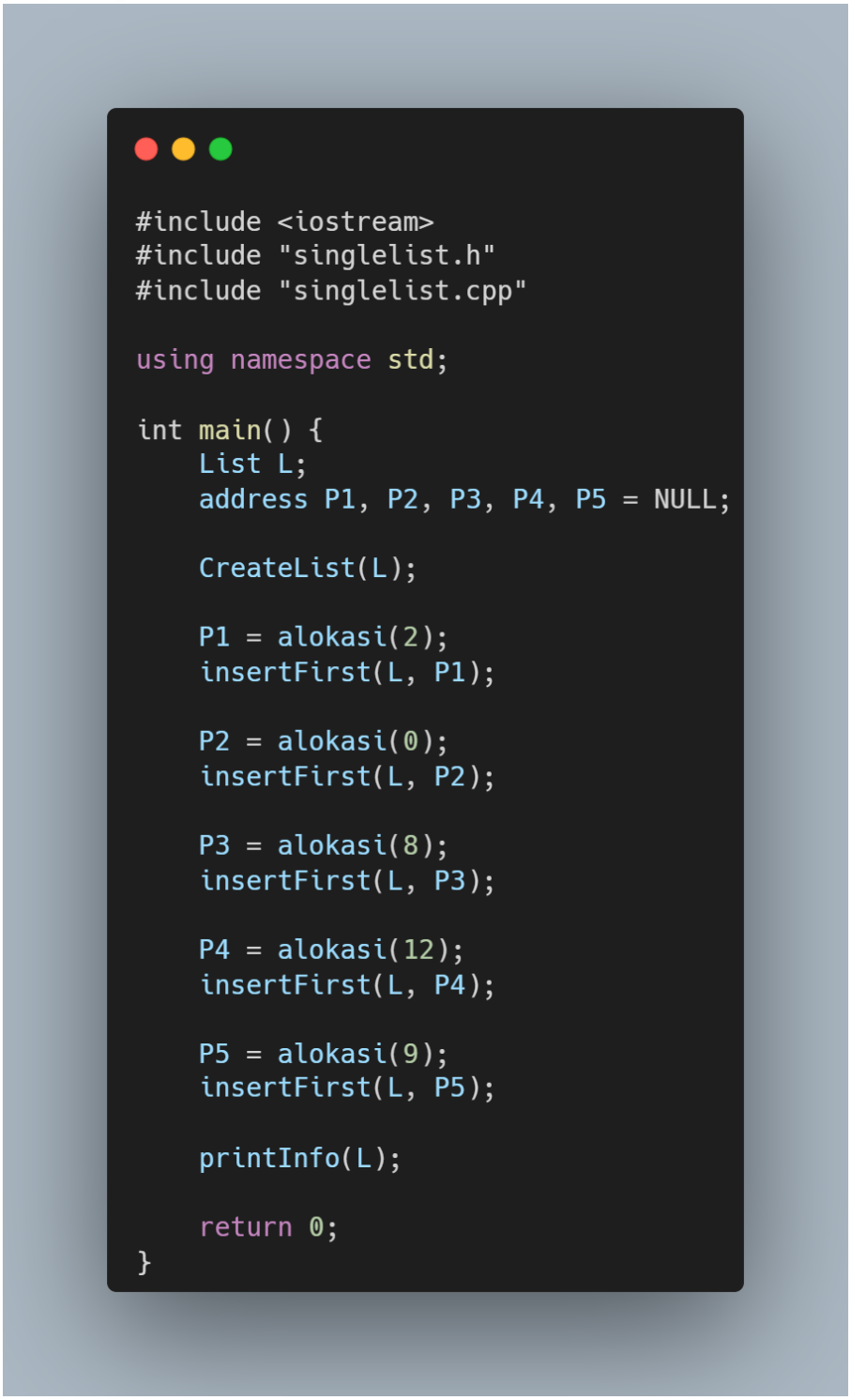
```h
#ifndef SINGLELIST_H
#define SINGLELIST_H

#include <iostream>

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
```

File singlelist.h

Telkom University Purwokerto

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

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;
    while (P != NULL) {
        std::cout << P->info << " ";
        P = P->next;
    }
    std::cout << std::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;
}
```

file singlelist.cpp



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

using namespace std;

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);

    printInfo(L);

    return 0;
}
```

file main.cpp versi 1

singlelist.h:

- ElmList: Struktur data untuk setiap node dalam linked list, berisi data dan pointer ke node berikutnya.
- List: Struktur data untuk keseluruhan linked list, berisi pointer ke node pertama.
- Fungsi-fungsi prototipe: CreateList, alokasi, dealokasi, printInfo, insertFirst, findElm, sumInfo.

singlelist.cpp:

- CreateList: Membuat linked list kosong.
- alokasi: Membuat node baru dan mengembalikan pointer ke node tersebut.
- dealokasi: Membebaskan memori yang dialokasikan untuk node.
- printInfo: Menampilkan semua elemen dalam linked list.
- insertFirst: Menambahkan elemen baru ke awal linked list.
- findElm: Mencari elemen tertentu dalam linked list.
- sumInfo: Menghitung total nilai semua elemen dalam linked list.

main.cpp versi 1: Menambahkan beberapa elemen ke linked list dan menampilkannya.

2. Menghapus Node pada Linked List

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

using namespace std;

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);

    address foundElement = findElm(L, 8);
    if (foundElement != NULL) {
        cout << "8 ditemukan dalam list" << endl;
    } else {
        cout << "8 tidak ditemukan dalam list" << endl;
    }

    return 0;
}
```

file main.cpp versi 2

main.cpp versi 2: Mencari elemen tertentu dalam linked list.

3. **Mencari dan Menghitung Panjang Linked List**

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

using namespace std;

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);

    int totalInfo = sumInfo(L);
    cout << "Total info dari kelima elemen adalah " << totalInfo << endl;

    return 0;
}
```

file main.cpp versi 3

main.cpp versi 3: Menghitung total nilai semua elemen dalam linked list.

5. Kesimpulan

Praktikum ini berhasil mengimplementasikan struktur data Single Linked List dalam bahasa C++. Melalui berbagai fungsi yang telah dibuat, seperti penambahan, pencarian, penghapusan, dan penjumlahan elemen, pemahaman mengenai konsep dasar linked list semakin mendalam.