LAPORAN PRAKTIKUM
Modul 5
SINGLE LINKED LIST (BAGIAN KEDUA)

Disusun Oleh:
Aulia Jasifa Br Ginting 2311104060
S1SE-07-02

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO

2024

1. **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 *likend list* dengan *prototype* yang ada.
2. **Landasan Teori**

    5.1 Searching

    Searching atau pencarian adalah proses untuk menemukan elemen tertentu dalam suatu data atau struktur data, seperti array, vektor, atau daftar (list). 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.

3. **Guided**

    Outputnya:

```
Elemen dalam linked list: 7 5 3 11 14 18
Masukkan elemen yang ingin dicari: 3
Elemen ditemukan pada index 2
Masukkan elemen yang ingin dihapus: 7
Elemen dalam linked list  setelah penghapusan: 5 3 11 14 18
```

    Programnya

Telkom University Purwokerto

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

// Struktur untuk node dalam lonked 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 menambhakan 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 linkend 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 linkend list
void display(Node* node) {
    while (node != nullptr) {
        cout << node->data << " ";
        node = node->next;
    }
    cout << endl;
}

// Fungsi untuk menghapus elemen dari linked 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 likend list" << endl;
    else
        cout << "Elemen ditemukan pada index " << result << endl;

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

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

    return 0;
}

```

4. Unguided

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

Singlelist.h

```h
#ifndef SINGLELIST_H
#define SINGLELIST_H

struct elmlist {
    int info;
    elmlist* next;
};

typedef elmlist* address;
typedef struct {
    address first;
} List;

void createList(List& L);
address alokasi(int x);
void insertFirst(List& L, address P);
void deleteFirst(List& L, address& P);
void printInfo(List L);
void insertAfter(List& L, address& prev, address& P);

#endif
```

Kemudian buat implementasi ADT *Single Likend list* pada *file* "**singlelist.cpp**"

Singlelist.cpp

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

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

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

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

void deleteFirst(List& L, address& P) {
    if (L.first != NULL) {
        P = L.first;
        L.first = L.first->next;
        P->next = NULL;
    }
}

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

void insertAfter(List& L, address& prev, address& P) {
    P->next = prev->next;
    prev->next = P;
}
```

Cobalah hasil implementai ADT pada *file* "**main.cpp**"
Main.cpp

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

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

    // Inisialisasi list kosong
    createList(L);

    // Membuat node-node baru
    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 isi list
    printInfo(L);

    return 0;
}
```

Outputnya

```
PS C:\Users\LENOVO\Documents\STUDYING\
Isi Linked List: 2 0 9 8 12
```

2. Carilah elemen dengan info 8 dengan membuat fungsi baru.
   fungsi findElm(L : *List*, x : infotype) : *address*
   Programnya

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

int main() {
    // Deklarasi variabel untuk menyimpan nilai tiap elemen
    int elemen1 = 9;
    int elemen2 = 12;
    int elemen3 = 8;
    int elemen4 = 0;
    int elemen5 = 2;

    // Menghitung total dari kelima elemen
    int total = elemen1 + elemen2 + elemen3 + elemen4 + elemen5;

    // Menampilkan hasil
    cout << "Total info dari kelima elemen adalah " << total << endl;

    return 0;
}
```

Outputnya

```
PS C:\Users\LENOVO\Documents\STUDYING
8 ditemukan dalam list
```

3. Hitunglah jumlah total info seluruh elemen (9+12+8+0+2=31)

Programnya

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

// Fungsi untuk mencari elemen dalam list
bool findElem(int list[], int size, int x) {
    for (int i = 0; i < size; i++) {
        if (list[i] == x) {
            return true; // Mengembalikan true jika elemen ditemukan
        }
    }
    return false; // Mengembalikan false jika elemen tidak ditemukan
}

int main() {
    // Deklarasi list elemen
    int list[] = {9, 12, 8, 0, 2};
    int size = sizeof(list) / sizeof(list[0]);
    int x = 8; // Elemen yang akan dicari

    // Memanggil fungsi findElem dan menampilkan hasilnya
    if (findElem(list, size, x)) {
        cout << x << " ditemukan dalam list" << endl;
    } else {
        cout << x << " tidak ditemukan dalam list" << endl;
    }

    return 0;
}
```

Outputnya

```
PS C:\Users\LENOVO\Documents\STUDYING\SEMESTER 3\
Total info dari kelima elemen adalah 31
```

5. **Kesimpulan**

   Pada pertumuan kali ini membahas mengenai Searching cara penggunaanya.