LAPORAN PRAKTIKUM STRUKTUR DATA

PERTEMUAN 5

SINGLE LINKED LIST (BAGIAN KEDUA)

Nama :

Reyner Atira Prasetyo (2311104057)

S1SE-07-02

Dosen :

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

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

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. TOOL

1. Visual Studio Code
2. GCC

III. DASAR TEORI

Searching

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.

IV. GUIDED

1. guidedM5.cpp

```cpp
#include <iostream>

using namespace std;

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

// Fungsi untuk menambahkan elemen baru di 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;
    }

}

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 << "Masukan elemen yang ingin dicari: ";
    cin >> x;

    int result = findElement(head, x);

    if (result == -1){
        cout << "Elemen tidak daat ditemukan dalam linked list" <<
endl;
    } else {
        cout << "Elemen ditemukan pada indeks " << result << endl;
    }

    cout << "Enter element to be deleted: ";
    cin >> x;
    deleteElement(head, x);

    cout << "Elements in linked list after deletion: ";
    display(head);

    return 0;
}
```

Output :

```
Microsoft-MIEngine-Pid-zfyh5x3u.zuo' '--dbgExe=C:\msy
Elemen dalam linked list: 7 5 3 11 14 18
Masukan elemen yang ingin dicari: 11
Elemen ditemukan pada indeks 3
Enter element to be deleted: 14
Elements in linked list after deletion: 7 5 3 11 18
```

V. UNGUIDED

1. singlelist.h

```
#ifndef SINGLELIST_H
```

```h
#define SINGLELIST_H

// define types
typedef int infotype;
typedef struct ElmList* address;

// define structure for ElmList / node
struct ElmList
{
    infotype info;
    address next;
};

// define structure for List
struct List
{
    address First;
};


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(List L, infotype x);
int sumInfo(const List &L);


#endif
```

2. singlelist.cpp

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

using namespace std;


void createList(List &L){
    L.First = nullptr;
}

address alokasi(infotype x){
    address P = new ElmList;
    P->info = x;
    P->next = nullptr;
    return P;
}

void dealokasi(address &P){
    delete P;
    P = nullptr;
}

void printInfo(const List &L) {
    address P = L.First;
    while (P != nullptr) {
```

```
        cout << P->info << " ";
        P = P->next;
    }
    cout << endl;
}

address findElm(List L, infotype x){
    address P = L.First;
    while (P != nullptr)
    {
        if (P->info == x)
        {
            return P; //return alamat node jika ditemukan dan end
function
        }
        P = P->next;
    }
    return nullptr; //return null jika alamat node tidak ditemukan
}

int sumInfo(const List &L) {
    int sum = 0;
    address P = L.First;

    // loop melalui setiap node untuk menambahkan nilai info ke sum
    while (P != nullptr) {
        sum += P->info;
        P = P->next;
    }

    return sum;
}

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

- Fungsi createList: Fungsi ini menginisialisasi list dengan mengatur pointer First ke nullptr, sehingga list awalnya kosong.

- Fungsi alokasi: Fungsi ini mengalokasikan memori untuk elemen list baru. Diberikan nilai x (tipe infotype) sebagai informasi, lalu menginisialisasi pointer next ke nullptr. Fungsi mengembalikan alamat elemen baru (P).

- Fungsi dealokasi: Fungsi ini bertujuan untuk menghapus elemen dari memori dengan menggunakan delete pada alamat P, dan kemudian mengatur P menjadi nullptr untuk mencegah penggunaan memori yang tidak valid.

- Fungsi printInfo: Fungsi ini mencetak semua elemen dalam list. Mulai dari

elemen pertama (First), fungsi mencetak nilai (info) dari setiap elemen hingga akhir list (sampai elemen next bernilai nullptr).

- Fungsi findElm: Fungsi ini mencari elemen dalam list berdasarkan nilai x. Jika elemen dengan nilai x ditemukan, maka fungsi mengembalikan alamat elemen tersebut. Jika tidak ditemukan, fungsi mengembalikan nullptr.

- Fungsi sumInfo: Fungsi ini menghitung jumlah semua nilai elemen dalam list. Dengan menggunakan loop, fungsi menambahkan nilai dari setiap elemen (dengan pointer info) ke variabel sum, dan kemudian mengembalikan hasil jumlah tersebut.

- Fungsi insertFirst: Fungsi ini memasukkan elemen baru (P) ke posisi pertama dalam list. Elemen baru ini menjadi elemen pertama (First) pada list, dan elemen sebelumnya dipindahkan menjadi elemen berikutnya (next).

3. main.cpp

```cpp
#include <iostream>
#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);

    address foundNode = findElm(L, 8);

    if (foundNode != nullptr)
```

```
    {
        cout << "Elemen dengan info 8 ditemukan: " <<foundNode->info
<<endl;
    } else
    {
        cout << "Eleen dengan info 8 tidak ditemukan" <<endl;
    }

    cout << "total info dari semua elemen: " << sumInfo(L) <<endl;


    return 0;
}
```

- Inisialisasi dan Deklarasi: Kode ini dimulai dengan mendeklarasikan variabel L sebagai sebuah list, serta beberapa pointer P1, P2, P3, P4, dan P5 yang diinisialisasi dengan nullptr.

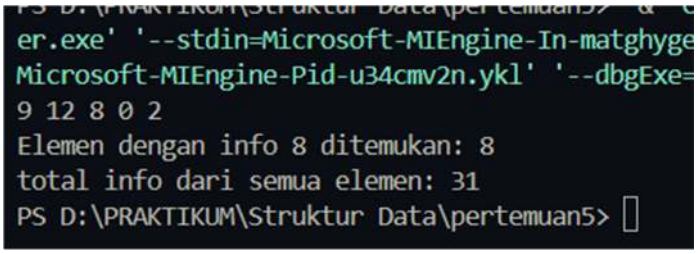
- Membuat List: Fungsi createList(L) digunakan untuk menginisialisasi list L agar menjadi list kosong dengan pointer First diatur ke nullptr.

- Alokasi dan Insert: Setiap elemen baru dialokasikan menggunakan fungsi alokasi(), dan masing-masing elemen tersebut disisipkan di awal list menggunakan fungsi insertFirst().

- Mencetak Isi List: Fungsi printInfo(L) digunakan untuk mencetak semua elemen dalam list L. Hasil cetakan akan menjadi "9 12 8 0 2", karena elemen yang disisipkan terakhir berada di depan.

- Pencarian Elemen: Fungsi findElm(L, 8) digunakan untuk mencari elemen dengan nilai info sebesar 8. Jika elemen ditemukan, program mencetak pesan yang menyatakan elemen tersebut ditemukan, beserta nilainya. Jika tidak ditemukan, program mencetak pesan sebaliknya.
- Menjumlahkan Nilai Elemen: Fungsi sumInfo(L) digunakan untuk menghitung total dari semua nilai info dalam list. Hasilnya dicetak ke layar, yaitu jumlah nilai dari semua elemen.
- Hasil Cetakan: Program ini akan mencetak urutan elemen dalam list, pesan hasil pencarian, dan jumlah total nilai elemen.

  Output :



VI. KESIMPULAN

Pada praktikum ini, saya telah berhasil mengimplementasikan dan mempraktikkan metode pencarian (searching) pada single linked list menggunakan bahasa pemrograman C++. Saya mempelajari bagaimana membuat, mengalokasikan, menambahkan elemen di awal list, mencetak elemen, serta mencari elemen dalam list. Dari hasil implementasi, saya memahami bagaimana struktur data ini bekerja, khususnya dalam menyusun dan mengakses elemen-elemen yang dihubungkan secara berurutan. Dengan menerapkan pencarian menggunakan loop, saya dapat menemukan elemen yang dicari dalam list. Praktikum ini menunjukkan pentingnya penggunaan pointer dalam pengelolaan memori dinamis dan bagaimana teknik dasar pencarian diterapkan pada struktur data linked list.