LAPORAN PRAKTIKUM
MODULl 5
SINGLE LINKED LIST (BAGIAN KEDUA)

Disusun Oleh:

Muhammad Shafiq Rasuna - 2311104043

Kelas :

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 linked list dengan prototype yang ada

2. Landasan Teori

2.1. 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. Semua fungsi dasar diatas merupakan bagian dari ADT dari singgle linked list, dan aplikasi pada bahasa pemrograman Cp semua ADT tersebut tersimpan dalam file *.c dan file *.h.

3. Guided

1. SINGLE LINKED LIST

Kode C++ mengimplementasikan sebuah linked list menggunakan struktur data dasar. Setiap elemen dalam linked list direpresentasikan oleh struktur Node, yang memiliki dua atribut: data untuk menyimpan nilai (tipe integer) dan next, yang merupakan pointer ke node berikutnya. Terdapat beberapa fungsi penting, termasuk insertFirst untuk menambahkan elemen baru di awal linked list, dan insertLast untuk menambahkan elemen di akhir. Fungsi findElement digunakan untuk mencari nilai tertentu dalam linked list dan mengembalikan indeksnya, sementara display menampilkan semua elemen yang ada. Fungsi deleteElement memungkinkan pengguna untuk menghapus elemen dengan nilai tertentu dari linked list, menangani kasus di mana elemen yang dihapus adalah head. Di dalam fungsi main, linked list diinisialisasi dengan beberapa elemen, kemudian pengguna dapat mencari dan menghapus elemen tertentu. Setelah penghapusan, isi linked list ditampilkan kembali. Kode ini mencakup operasi dasar pada linked list, seperti penambahan, pencarian, penghapusan, dan tampilan elemen.

Kode program :

```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;
}

```

```cpp
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);

    insertFirst(head, tail, 11);
    insertFirst(head, tail, 14);
    insertFirst(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;
}
```

Output program :

```
c:\Users\ASUS\OneDrive\Dokumen\tugas smt  3\Pemograman Struktur Data 3\pertemuan5>cd
emuan5\" && g++ Guided1.cpp -o Guided1 && "c:\Users\ASUS\OneDrive\Dokumen\tugas smt
Elemen dalam linked list: 18 14 11 7 5 3
Masukkan elemen yang ingin dicari: 5
Elemen ditemukan pada indeks 4
Masukkan elemen yang ingin dihapus: 3
Elemen dalam linked list setelah penghapusan: 18 14 11 7 5
```

4. Unguided

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

```
#ifndef SINGLELIST_H
#define SINGLELIST_H

#include <iostream>
using namespace std;

typedef int infotype;
typedef struct ElmList *address;

struct ElmList {
    infotype info;
    address next;
};

struct List {
    address First;
};

// Deklarasi fungsi dan prosedur
void createList(List &L);
address alokasi(infotype x);
void dealokasi(address &P);
void printInfo(List L);
void insertFirst(List &L, address P);

#endif

```

Kemudian buat implementasi ADT Single Linked list pada file “singlelist.cpp”. Adapun isi data

```cpp
#include "singlelist.h"

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(List L) {
    address P = L.First;
    while (P != nullptr) {
        cout << P->info << " ";
        P = P->next;
    }
    cout << endl;
}

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

```

Cobalah hasil implementasi ADT pada file “main.cpp”

```cpp
#include "singlelist.h"

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

    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;
}

```

Output dari kode program :

```
ASUS@LAPTOP-1TBBL9PC MINGW64 ~/OneDrive/Dokumen/tugas smt  3/Pemograman Struktur Data 3/pertemuan5/unguided5
$ g++ main.cpp singlelist.cpp -o program

ASUS@LAPTOP-1TBBL9PC MINGW64 ~/OneDrive/Dokumen/tugas smt  3/Pemograman Struktur Data 3/pertemuan5/unguided5
$ ./program
9 12 8 0 2
```

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

```cpp
#include <iostream>
#include <vector>

using namespace std;

// Asumsikan infotype adalah integer
int findElm(const vector<int>& L, int x) {
    // Menggunakan algoritma pencarian linear
    for (int i = 0; i < L.size(); ++i) {
        if (L[i] == x) {
            return i; // Mengembalikan indeks jika ditemukan
        }
    }
    return -1; // Mengembalikan -1 jika tidak ditemukan
}

int main() {
    vector<int> myList = {3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, 7, 9, 3, 2, 3, 8, 4, 6, 2, 6, 4, 3, 3, 8, 3, 2, 7, 9, 5};
    int target = 8;

    int result = findElm(myList, target);

    if (result != -1) {
        cout << target << " ditemukan dalam list" << endl;
    } else {
        cout << target << " tidak ditemukan dalam list" << endl;
    }

    return 0;
}
```

Output dari kode program :

```
c:\Users\ASUS\OneDrive\Dokumen\tugas smt  3\Pemograman Struktur Data 3\pertemuan5\unguided5>cd "c:\Users\ASUS\OneDrive\Dokumen\tugas smt  3\Pemograman Struktur D
ata 3\pertemuan5\unguided5\" && g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile && "c:\Users\ASUS\OneDrive\Dokumen\tugas smt  3\Pemograman Struktur Data 3\perte
muan5\unguided5\"tempCodeRunnerFile
8 ditemukan dalam list
```

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

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

// Struktur untuk node dalam linked list
struct Node {
    int data;
    Node* next;
};

// 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) {
        // Jika list kosong, node baru menjadi head dan tail
        head = new_node;
        tail = new_node;
    } else {
        // Jika list tidak kosong, tambahkan node baru di akhir
        tail->next = new_node;
        tail = new_node;
    }
}

// Fungsi untuk menghitung jumlah total elemen dalam linked list
int sumElements(Node* head) {
    int total = 0;
    Node* current = head;
    while (current != nullptr) {
        total += current->data;
        current = current->next;
    }
    return total;
}

// Fungsi untuk menampilkan elemen dalam linked list
void display(Node* node) {
    while (node != nullptr) {
        cout << node->data << " ";
        node = node->next;
    }
    cout << endl;
}

int main() {
    Node* head = nullptr;
    Node* tail = nullptr;

    // Menambahkan elemen ke linked list
    insertLast(head, tail, 9);
    insertLast(head, tail, 12);
    insertLast(head, tail, 8);
    insertLast(head, tail, 0);
    insertLast(head, tail, 2);

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

    // Menghitung total info dari seluruh elemen
    int total = sumElements(head);
    cout << "Total info dari kelima elemen adalah " << total << endl;

    return 0;
}

```

Output dari kode program :

```
c:\Users\ASUS\OneDrive\Dokumen\tugas smt  3\Pemograman Struktur Data 3\pertemuan5\unguided5>cd "c:\Users\ASUS\OneDrive\Dokumen\tugas smt  3\Pemograman Struktur D
ata 3\pertemuan5\unguided5\" && g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile && "c:\Users\ASUS\OneDrive\Dokumen\tugas smt  3\Pemograman Struktur Data 3\perte
muan5\unguided5\"tempCodeRunnerFile
Elemen dalam linked list: 9 12 8 0 2
Total info dari kelima elemen adalah 31
```

5. Kesimpulan

Dalam pemrograman, penggunaan linked list sebagai struktur data dinyatakan sangat efektif untuk menangani operasi-operasi dasar seperti penambahan, pencarian, dan penghapusan elemen. Melalui implementasi ADT Single Linked List, kita dapat memahami bagaimana setiap node terhubung dengan menggunakan pointer, serta bagaimana operasi seperti searching memudahkan manipulasi data. Dengan menciptakan berbagai fungsi seperti insertFirst, insertLast, dan deleteElement, kita dapat mengelola data dengan lebih fleksibel. Selain itu, penambahan fungsi untuk mencari elemen tertentu dan menghitung jumlah total dari semua elemen menunjukkan kemampuan linked list dalam melakukan operasi yang lebih kompleks. Secara keseluruhan, penguasaan konsep dan implementasi linked list sangat penting dalam pengembangan algoritma dan aplikasi berbasis data, memberikan fondasi yang kuat untuk pemrograman yang lebih lanjut.