

**LAPORAN PRAKTIKUM MODUL 5
SINGLE LINKED LIST
BAGIAN 2**



**Disusun Oleh:
Ade Fatkhul Anam 2211104051**

**Asisten Praktikum :
Aldi Putra
Andini Nur Hidayah**

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

**PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024**

MODUL 3

ABSTRAK DATA TYPE

1. Tujuan

- a. Memahami penggunaan linked list dengan pointer operator dalam program.
- b. Memahami operasi dasar dalam linked list.
- c. Membuat program dengan menggunakan linked list dengan prorotype yang ada.

2. Landasan Teori

a. 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.

3. Guided

a. Searching

Source code:

```
#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
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 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) {
        head = new_node;
        tail = new_node;
    } else
        tail->next = new_node;
    tail=new_node;
}
```

```

// Fungsi untuk mencari elemen dalam linked 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 linked 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 << "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 dalam index " << result << endl;

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

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

return 0;
}

```

Output:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS
PS D:\PRAKTIKUM DATA STRUCTURE\LAPRAK\05_Single_Linked_List_BAGIAN KEDUA> cd "d:\PRAKTIKUM DATA STRUCTURE\LAPRAK\05_Single_Linked_List_BAGIAN KEDUA\output"
PS D:\PRAKTIKUM DATA STRUCTURE\LAPRAK\05_Single_Linked_List_BAGIAN KEDUA\output> & .\guidedmodul5.exe
Elemen dalam linked list: 7 5 3 11 14 18
Masukan elemen yang ingin dicari: 11
Elemen ditemukan dalam index 3
Masukan elemen yang ingin dihapus: 14
Elemen dalam linked list setelah penghapusan: 7 5 3 11 18
PS D:\PRAKTIKUM DATA STRUCTURE\LAPRAK\05_Single_Linked_List_BAGIAN KEDUA\output>

```

4. Unguided

- a. Buatlah program C++ untuk membuat sebuah single linked list dengan operasi dasar sebagai berikut:

- Insert Node di Depan: Fungsi untuk menambah node baru di awal linked list.
- Insert Node di Belakang: Fungsi untuk menambah node baru di akhir linked list.
- Cetak Linked List: Fungsi untuk mencetak seluruh isi linked list

Source code:

```
#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;
};

// Prosedur pembuatan list kosong
void createList(List &L);

// Fungsi alokasi elemen baru
address alokasi(infotype x);

// Prosedur dealokasi elemen
void dealokasi(address P);

// Prosedur untuk menampilkan info semua elemen dalam list
void printInfo(List L);

// Prosedur untuk menambahkan elemen di awal list
void insertFirst(List &L, address P);

// Fungsi untuk mencari elemen dengan info tertentu
address findElm(List L, infotype x);

// Fungsi untuk menghitung total info semua elemen
int totalInfo(List L);

#endif
```

Singelist.cpp

```
#include "singlelist.h"

// Prosedur pembuatan list kosong
void createList(List &L) {
    L.first = NULL;
}

// Fungsi alokasi elemen baru
address alokasi(infotype x) {
    address P = new elmlist;
    P->info = x;
    P->next = NULL;
    return P;
}

// Prosedur dealokasi elemen
void dealokasi(address P) {
    delete P;
}

// Prosedur untuk menampilkan info semua elemen dalam list
void printInfo(List L) {
    address P = L.first;
    while (P != NULL) {
        cout << P->info << " ";
        P = P->next;
    }
    cout << endl;
}

// Prosedur untuk menambahkan elemen di awal list
void insertFirst(List &L, address P) {
    P->next = L.first;
    L.first = P;
}

// Fungsi untuk mencari elemen dengan info tertentu
address findElm(List L, infotype x) {
    address P = L.first;
    while (P != NULL) {
        if (P->info == x)
            return P;
        P = P->next;
    }
    return NULL; // Jika tidak ditemukan
}

// Fungsi untuk menghitung total info semua elemen
int totalInfo(List L) {
    address P = L.first;
```

```

        int total = 0;
        while (P != NULL) {
            total += P->info;
            P = P->next;
        }
        return total;
    }

```

Main.cpp

```

#include "singlelist.h"

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

    // Buat list kosong
    createList(L);

    // Alokasi dan insert elemen ke dalam list
    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);

    // Tampilkan semua elemen dalam list
    printInfo(L);

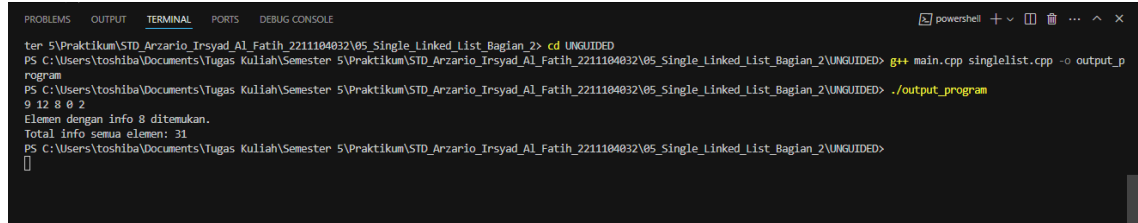
    // Mencari elemen dengan info 8
    address found = findElm(L, 8);
    if (found != NULL) {
        cout << "Elemen dengan info 8 ditemukan." << endl;
    } else {
        cout << "Elemen dengan info 8 tidak ditemukan." << endl;
    }

    // Menghitung total info semua elemen
    int total = totalInfo(L);
    cout << "Total info semua elemen: " << total << endl;

    return 0;
}

```

Output:



```
PS C:\Praktikum\STD_Arzario_Irsyad_Al_Fatih_2211104032\05_Single_Linked_List_Bagian_2\UNGUIDED> cd UNGUIDED
PS C:\Users\toshiba\Documents\Tugas Kuliah\Semester 5\Praktikum\STD_Arzario_Irsyad_Al_Fatih_2211104032\05_Single_Linked_List_Bagian_2\UNGUIDED> g++ main.cpp singlelist.cpp -o output_program
PS C:\Users\toshiba\Documents\Tugas Kuliah\Semester 5\Praktikum\STD_Arzario_Irsyad_Al_Fatih_2211104032\05_Single_Linked_List_Bagian_2\UNGUIDED> ./output_program
Elemen dengan info 8 ditemukan.
Total info semua elemen: 31
PS C:\Users\toshiba\Documents\Tugas Kuliah\Semester 5\Praktikum\STD_Arzario_Irsyad_Al_Fatih_2211104032\05_Single_Linked_List_Bagian_2\UNGUIDED>
```

5. Kesimpulan

Pada praktikum modul 5 ini, kita telah mempelajari konsep dasar tentang *single linked list*, termasuk operasi dasar seperti penambahan elemen, pencarian, dan penghapusan elemen. Praktikum ini dimulai dengan pemahaman teoretis mengenai penggunaan pointer dalam linked list, di mana kita belajar bagaimana menyusun dan mengelola struktur data tersebut. Melalui implementasi kode dalam bahasa C++, kami berhasil membuat fungsi-fungsi seperti `insertFirst`, `insertLast`, `findElement`, dan `deleteElement`, yang memberikan kemampuan untuk mengelola elemen dalam linked list dengan efisien. Pengalaman praktikum ini mengajarkan kami pentingnya memahami struktur data dinamis, yang sangat bermanfaat dalam pengembangan perangkat lunak. Untuk lebih memahami konsep yang telah dipelajari, kami diminta untuk membuat program C++ untuk *single linked list* dengan fungsi-fungsi dasar, termasuk:

- Insert Node di Depan: Menambahkan node baru di awal linked list.
- Insert Node di Belakang: Menambahkan node baru di akhir linked list.
- Cetak Linked List: Mencetak seluruh isi linked list.

Melalui tugas ini, kami ditantang untuk menerapkan pengetahuan yang telah dipelajari dan mengembangkan keterampilan pemrograman dalam mengelola data menggunakan linked list.