

LAPORAN PRAKTIKUM
Modul 5
“Single Linked List Bagian Kedua”



Disusun Oleh:
MUHAMMAD RALFI - 2211104054
SE-07-2

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

1. Tujuan

- Memahami penggunaan linked list dengan pointer operator- operator dalam program.
- Memahami operasi-operasi dasar dalam linked list.
- Membuat program dengan menggunakan linked list dengan prototype yang ada

2. Landasan Teori

Searching

Searching adalah operasi dasar pada list yang berfungsi mencari node tertentu dengan mengunjungi setiap node satu per satu hingga node yang dicari ditemukan. Setelah proses pencarian selesai, operasi lain seperti insert after, delete after, dan update dapat dilakukan dengan lebih mudah. Semua fungsi dasar ini merupakan bagian dari Abstract Data Type (ADT) pada Single Linked List dan dalam bahasa pemrograman C, ADT tersebut umumnya disimpan dalam file *.c untuk implementasi dan *.h untuk deklarasi.

3. Guided

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

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

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 << "Element ditemukan pada indeks " << result << endl;

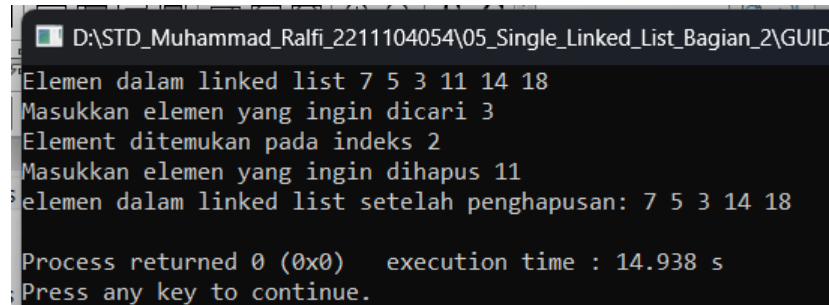
    cout << "Masukkan elemen yang ingin dihapus ";
    cin >> x;
    deleteElement(head, x);

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

    return 0;
}

```

Output:



```

D:\STD_Muhammad_Ralfi_2211104054\05_Single_Linked_List_Bagian_2\GUID
Elemen dalam linked list 7 5 3 11 14 18
Masukkan elemen yang ingin dicari 3
Element ditemukan pada indeks 2
Masukkan elemen yang ingin dihapus 11
Elemen dalam linked list setelah penghapusan: 7 5 3 14 18

Process returned 0 (0x0)   execution time : 14.938 s
Press any key to continue.

```

4. Unguided

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

```

Type infotype : int
Type address  : pointer to ElmList

Type ElmList <
    info : infotype
    next : address
>

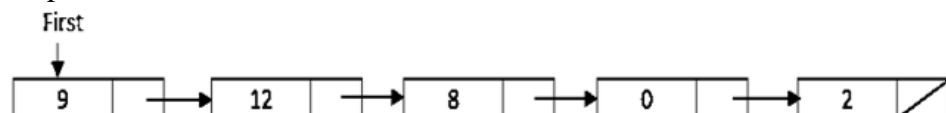
Type List : < First : address >

prosedur CreateList( in/out L : List )
fungsi alokasi( x : infotype ) : address
prosedur dealokasi( in/out P : address )
prosedur printInfo( in L : List )
prosedur insertFirst( in/out L : List, in P : address )

```

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

Adapun isi data



Lalu implementasikan ADT ke file main.cpp

```
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;
}
```

informatics lab

Code:

File Singlelist.h

```
#ifndef SINGLELIST_H
#define SINGLELIST_H

#include <stdio.h>
#include <stdlib.h>

typedef int infotype;
typedef struct elmlist *address;

struct elmlist {
    infotype info;
    address next;
};

struct List {
    address first;
};

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

#endif
```

File Singlelist.cpp

```
#include "singlelist.h"

void createList(List *L) {
    L->first = NULL;
}
```

```
address alokasi(infotype x) {
    address P = (address)malloc(sizeof(struct elm1ist));
    if (P != NULL) {
        P->info = x;
        P->next = NULL;
    }
    return P;
}

void dealokasi(address P) {
    free(P);
}

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

void printInfo(List L) {
    address P = L.first;
    while (P != NULL) {
        printf("%d", P->info);
        P = P->next;
        if (P != NULL) {
            printf(" ");
        }
    }
    printf("\n");
}
```

File Main.cpp

```
#include "singlelist.h"

int main() {
    List L;
    createList(&L);

    // Membuat dan menambahkan elemen ke dalam list
    address P1 = alokasi(2);
    insertFirst(&L, P1);

    address P2 = alokasi(0);
    insertFirst(&L, P2);

    address P3 = alokasi(8);
    insertFirst(&L, P3);

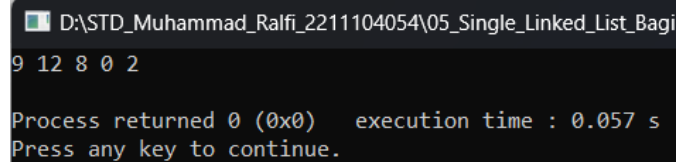
    address P4 = alokasi(12);
    insertFirst(&L, P4);

    address P5 = alokasi(9);
    insertFirst(&L, P5);
}
```

```
    printInfo(L);

    return 0;
}
```

Output:



```
D:\STD_Muhammad_Ralfi_2211104054\05_Single_Linked_List_Bagi
9 12 8 0 2
Process returned 0 (0x0)   execution time : 0.057 s
Press any key to continue.
```

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

Code:

File Singlelist.h

```
// Deklarasi fungsi
void createList(List *L);
address alokasi(infotype x);
void dealokasi(address P);
void insertFirst(List *L, address P);
void printInfo(List L);
address findElm(List L, infotype x);
#endif
```

File Singlelist.cpp

```
address findElm(List L, infotype x) {
    address P = L.first;
    while (P != NULL) {
        if (P->info == x) {
            return P;
        }
        P = P->next;
    }
    return NULL;
}
```

File Main.cpp

```
// Mencari elemen dengan nilai info 8
infotype searchValue = 8;
address found = findElm(L, searchValue);
if (found != NULL) {
    printf("%d ditemukan di dalam list.\n", searchValue);
} else {
    printf("%d tidak ditemukan di dalam list.\n",
searchValue);
}
```

Output:

```
D:\STD_Muhammad_Ralfi_2211104054\05_Single_Linked_List_Bagian
8 ditemukan di dalam list.

Process returned 0 (0x0)   execution time : 0.061 s
Press any key to continue.
```

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

Code:

File Main.cpp

```
// Menghitung jumlah total info elemen dalam list
int total = 0;
address P = L.first;
while (P != NULL) {
    total += P->info;
    P = P->next;
}
printf("Total info dari kelima elemen adalah: %d\n",
total);
```

Output:

```
D:\STD_Muhammad_Ralfi_2211104054\05_Single_Linked_List_Bagian
Total info dari kelima elemen adalah: 31

Process returned 0 (0x0)   execution time : 0.065 s
Press any key to continue.
```

5. Kesimpulan

-Implementasi Single Linked List berupa struktur data yang terdiri dari elemen-elemen list yang terhubung berurutan menggunakan pointer. Dalam bahasa C, deklarasi fungsi dasar seperti `createList`, `insertFirst`, `printInfo`, dan `findElm` ditempatkan di file header (.h), sementara implementasinya ada di file source (.c). Fungsi `findElm` memungkinkan pencarian elemen dengan mengunjungi setiap node satu per satu. Setelah elemen ditemukan, operasi lain seperti penambahan, penghapusan, atau pembaruan elemen dapat dilakukan lebih mudah. Praktikum ini membantu mahasiswa memahami pengelolaan memori dan penggunaan pointer dalam manipulasi struktur data dinamis.