

LAPORAN PRAKTIKUM

MODUL 5

SINGLE LINKED LIST (BAGIAN KEDUA)



Disusun Oleh:

Rizaldy Aulia Rachman (2311104051)

S1SE-07-02

Dosen :

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

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. 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 *single linked list*, dan aplikasi pada bahasa pemrograman C++ semua ADT tersebut tersimpan dalam *file *.c* dan *file *.h*

III. GUIDED

Codingan Searching Latihan Minggu ke-5

Code:

```

1  #include <iostream>
2
3  using namespace std;
4
5  // Struktur untuk node dalam linked list
6
7  struct Node {
8      int data;
9      Node* next;
10 };
11
12 // Fungsi untuk menambahkan elemen baru ke awal linked list
13 void InsertFirst(Node*& head, Node*& tail, int new_data){
14     Node* new_node = new Node();
15     new_node->data = new_data;
16     new_node->next = head;
17     head = new_node;
18
19     if (tail == nullptr) {
20         tail = new_node;
21     }
22 }
23
24 // fungsi untuk menambahkan elemen baru ke akhir linked list
25 void InsertLast(Node*& head, Node*& tail, int new_data){
26     Node* new_node = new Node ();
27     new_node->data = new_data;
28     new_node->next = nullptr;
29
30     if (head == nullptr){
31         head = new_node;
32         tail = new_node;
33     } else {
34         tail->next = new_node;
35         tail = new_node;
36     }
37 }
38
39 // fungsi untuk mencari elemen dalam linked list
40 int findElement(Node* head, int x){
41     Node* current = head;
42     int index = 0;
43
44     while(current != nullptr){
45         if (current->data == x){
46             return index;
47         }
48         current = current->next;
49         index++;
50     }
51     return -1;
52 }
53
54 // fungsi untuk menampilkan elemen dalam linked list
55 void display(Node* node){
56     while (node != nullptr){
57         cout << node->data << " ";
58         node = node->next;
59     }
60     cout << endl;
61 }
62
63 // fungsi untuk menghapus elemen dari linked list
64 void deleteElement(Node*& head, int x){
65     if (head == nullptr){
66         cout << "Linked List kosong" << endl;
67         return;
68     }
69
70     if (head->data == x){
71         Node* temp = head;
72         head = head->next;
73         delete temp;
74         return;
75     }
76
77     Node* current = head;
78     while(current->next != nullptr){
79         if(current->next->data == x){
80             Node* temp = current->next;
81             current->next = current->next->next;
82             delete temp;
83             return;
84         }
85         current = current->next;
86     }
87 }
88
89 int main()
90 {
91     Node* head = nullptr;
92     Node* tail = nullptr;
93
94     insertFirst(head, tail, 3);
95     insertFirst(head, tail, 5);
96     insertFirst(head, tail, 7);
97
98     insertLast(head, tail, 11);
99     insertLast(head, tail, 14);
100    insertLast(head, tail, 18);
101
102    cout << "Elemen dalam linked list: ";
103    display(head);
104
105    int x;
106    cout << "Masukkan elemen yang ingin dicari: ";
107    cin >> x;
108
109    int result = findElement(head, x);
110
111    if (result == -1)
112        cout << "Elemen tidak ditemukan dalam linked list" << endl;
113    else
114        cout << "Elemen ditemukan pada indeks " << result << endl;
115
116    cout << "Masukkan elemen yang ingin dihapus: ";
117    cin >> x;
118    deleteElement(head, x);
119
120    cout << "Elemen dalam linked list setelah penghapusan: ";
121    display(head);
122
123    return 0;
124 }

```

Output:

```
Elemen dalam linked list: 7 5 3 11 14 18
Masukkan elemen yang ingin dicari: 11
Elemen ditemukan pada indeks 3
Masukkan elemen yang ingin dihapus: 3
Elemen dalam linked list setelah penghapusan: 7 5 11 14 18
PS C:\Praktikum Struktur data\pertemuan5>
```

IV. UNGUIDED

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

Jawaban:

a) Singlelist.h

Code:

```
1  #ifndef SINGLELIST_H
2  #define SINGLELIST_H
3
4  // Definisikan infotype sebagai int dan address sebagai pointer ke ElmList
5  typedef int infotype;
6  typedef struct ElmList *address;
7
8  // Definisikan struktur untuk setiap elemen (node)
9  struct ElmList {
10     infotype info;    // Menyimpan nilai dari node
11     address next;     // Pointer ke node berikutnya
12 };
13
14 // Definisikan struktur untuk List
15 struct List {
16     address First;    // Menunjuk ke node pertama dalam list
17 };
18
19 // Deklarasi fungsi dan prosedur
20 void createList(List &L);
21 address alokasi(infotype x);
22 void dealokasi(address &P);
23 void printInfo(const List &L);
24 void insertFirst(List &L, address P);
25 address findElm(const List &L, infotype x);
26 int totalInfo(const List &L);
27
28 #endif
29
```

b) Singlelist

Code:

```

1  #include <iostream>
2  #include "singlelist.h"
3
4  using namespace std;
5
6  // Membuat list kosong
7  void createlist(List &L) {
8      L.First = nullptr;
9  }
10
11 // Mengalokasikan node baru dengan nilai info tertentu
12 address alokasi(infotype x) {
13     address P = new Elmlist;
14     P->info = x;
15     P->next = nullptr;
16     return P;
17 }
18
19 // Dealokasi atau membebaskan memori dari node
20 void dealokasi(address &P) {
21     delete P;
22     P = nullptr;
23 }
24
25 // Menampilkan info dari semua elemen dalam list
26 void printInfo(const List &L) {
27     address P = L.First;
28     while (P != nullptr) {
29         cout << P->info << " ";
30         P = P->next;
31     }
32     cout << endl;
33 }
34
35 // Menyisipkan node baru di awal list
36 void insertFirst(List &L, address P) {
37     P->next = L.First;
38     L.First = P;
39 }
40
41 // Mencari elemen dengan nilai tertentu
42 address findElm(const List &L, infotype x) {
43     address P = L.First;
44     while (P != nullptr && P->info != x) {
45         P = P->next;
46     }
47     return P;
48 }
49
50 // Menghitung total info dari semua elemen
51 int totalInfo(const List &L) {
52     int total = 0;
53     address P = L.First;
54     while (P != nullptr) {
55         total += P->info;
56         P = P->next;
57     }
58     return total;
59 }
60

```

c) Main

Code:

```
1 #include <iostream>
2 #include "singlelist.h"
3
4 using namespace std;
5
6 int main() {
7     List L;
8     address P1, P2, P3, P4, P5 = nullptr;
9
10    // Membuat list kosong
11    createList(L);
12
13    // Menyisipkan beberapa elemen ke dalam list
14    P1 = alokasi(2);
15    insertFirst(L, P1);
16
17    P2 = alokasi(0);
18    insertFirst(L, P2);
19
20    P3 = alokasi(8);
21    insertFirst(L, P3);
22
23    P4 = alokasi(12);
24    insertFirst(L, P4);
25
26    P5 = alokasi(9);
27    insertFirst(L, P5);
28
29    // Menampilkan semua elemen dalam list
30    cout << "Isi List: ";
31    printInfo(L);
32
33    // Mencari elemen dengan info 8
34    address found = findElm(L, 8);
35    if (found != nullptr) {
36        cout << "Elemen dengan info 8 ditemukan: " << found->info << endl;
37    } else {
38        cout << "Elemen dengan info 8 tidak ditemukan" << endl;
39    }
40
41    // Menghitung total info dari semua elemen
42    int total = totalInfo(L);
43    cout << "Total nilai semua elemen: " << total << endl;
44
45    return 0;
46 }
47
```

Output Main:

```
Isi List: 9 12 8 0 2
Elemen dengan info 8 ditemukan: 8
Total nilai semua elemen: 31
PS C:\Praktikum Struktur data\pertemuan5>
```