

LAPORAN PRAKTIKUM  
Modul 05  
“SINGLE LINKED LIST (BAGIAN KEDUA)”



Disusun Oleh:  
Faishal Arif Setiawan 2311104066  
Kelas:  
**SE 07 02**  
Dosen :  
WAHYU ANDI SAPUTRA, S.PD.,M.ENG

PROGRAM STUDI S1 SOFTWARE ENGINEERING  
FAKULTAS INFORMATIKA  
TELKOM UNIVERSITY  
PURWOKERTO  
2024

## 1.TUJUAN PRAKTIKUM

- 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.GUIDED

### 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.

```

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

```

menambahkan elemen di awal (insertFirst) dan di akhir (insertLast), mencari elemen tertentu (findElement), menampilkan semua elemen (display), serta menghapus elemen tertentu dari linked list (deleteElement). Struktur Node digunakan untuk membentuk setiap elemen dalam linked list, yang menyimpan data dan pointer ke elemen berikutnya. Dalam fungsi main, beberapa elemen awal ditambahkan, kemudian program meminta pengguna untuk memasukkan elemen yang ingin dicari dan dihapus. Hasil pencarian dan penghapusan ini kemudian ditampilkan.

Outputnya:

```
Elemen dalam linked list: 18 14 11 7 5 3
Masukkan elemen yang ingin dicari: 11
Elemen ditemukan pada indeks 2
Masukkan elemen yang ingin dihapus: 5
Elemen dalam linked list setelah penghapusan: 18 14 11 7 3

Process returned 0 (0x0)   execution time : 16.552 s
Press any key to continue.
```

### 3.UNGUIDED

```

1  #include <iostream>
2  using namespace std;
3
4  // Struktur data untuk elemen list
5  struct Elmlist {
6      int info;
7      Elmlist *next;
8  };
9
10 // Struktur data untuk list
11 struct List {
12     Elmlist *First;
13 };
14
15 // Fungsi alokasi untuk membuat elemen list baru
16 Elmlist* alokasi(int x) {
17     Elmlist *P = new Elmlist;
18     if (P != NULL) {
19         P->info = x;
20         P->next = NULL;
21         return P;
22     } else {
23         return NULL;
24     }
25 }
26
27 // Prosedur untuk dealokasi elemen list
28 void dealokasi(Elmlist *&P) {
29     delete P;
30     P = NULL;
31 }
32
33 // Prosedur untuk membuat list kosong
34 void CreateList(List &L) {
35     L.First = NULL;
36 }
37
38 // Prosedur untuk menampilkan informasi list
39 void printInfo(List L) {
40     Elmlist *P = L.First;
41     while (P != NULL) {
42         cout << P->info << " ";
43         P = P->next;
44     }
45     cout << endl;
46 }
47
48 // Prosedur untuk menambahkan elemen ke awal list
49 void insertFirst(List &L, int P) {
50     Elmlist *baru = alokasi(P);
51     if (baru != NULL) {
52         baru->next = L.First;
53         L.First = baru;
54     }
55 }
56
57 int main() {
58     List L;
59     CreateList(L);
60     insertFirst(L, 2);
61     insertFirst(L, 0);
62     insertFirst(L, 8);
63     insertFirst(L, 12);
64     insertFirst(L, 9);
65
66     printInfo(L);
67
68     return 0;
69 }
70

```

Output:

```


[Running] cd "d:\struktur data pemograman\guided5\" && g++ unguided5no1.cpp -o unguided5no1 && "d:\struktur data pemograman\guided5\"unguided5no1
9 12 8 0 2

```

```

[Done] exited with code=0 in 1.532 seconds

```



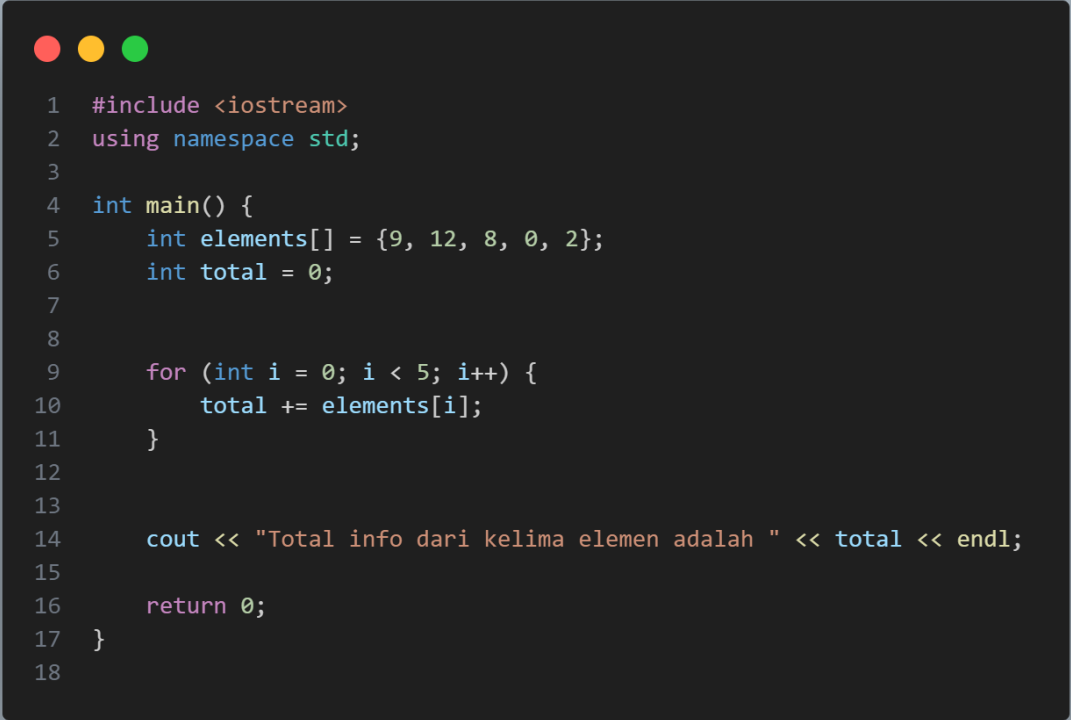
```

1  #include <iostream>
2  using namespace std;
3
4  // Struktur Node untuk Single Linked List
5  struct Node {
6      int info;
7      Node *next;
8  };
9
10 // Fungsi untuk membuat node baru
11 Node* newNode(int data) {
12     Node* node = new Node;
13     node->info = data;
14     node->next = NULL;
15     return node;
16 }
17
18 // Fungsi untuk menemukan node dengan info tertentu
19 Node* findElm(Node* head, int x) {
20     Node* current = head;
21     while (current != NULL) {
22         if (current->info == x) {
23             return current;
24         }
25         current = current->next;
26     }
27     return NULL; // Node tidak ditemukan
28 }
29
30 int main() {
31     // Membuat Singly Linked List
32     Node* head = newNode(1);
33     head->next = newNode(2);
34     head->next->next = newNode(3);
35     head->next->next->next = newNode(4);
36     head->next->next->next->next = newNode(5);
37     head->next->next->next->next->next = newNode(6);
38     head->next->next->next->next->next->next = newNode(7);
39     head->next->next->next->next->next->next->next = newNode(8);
40
41     // Mencari node dengan info 8
42     Node* foundNode = findElm(head, 8);
43
44     if (foundNode != NULL) {
45         cout << "8 ditemukan dalam list" << endl;
46     } else {
47         cout << "8 tidak ditemukan dalam list" << endl;
48     }
49
50     return 0;
51 }

```

Output:

```
[Running] cd "d:\struktur data pemograman\guided5\" && g++ unguided5no2.cpp -o unguided5no2 && "d:\struktur data pemograman\guided5\"unguided5no2
8 ditemukan dalam list
[Done] exited with code=0 in 3.01 seconds
```



```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int elements[] = {9, 12, 8, 0, 2};
6      int total = 0;
7
8
9      for (int i = 0; i < 5; i++) {
10         total += elements[i];
11     }
12
13
14     cout << "Total info dari kelima elemen adalah " << total << endl;
15
16     return 0;
17 }
18
```

Output:

```
[Running] cd "d:\struktur data pemograman\guided5\" && g++ unguided5no3.cpp -o unguided5no3 && "d:\struktur data pemograman\guided5\"unguided5no3
Total info dari kelima elemen adalah 31
[Done] exited with code=0 in 2.013 seconds
```

## 4.KESIMPULAN

melalui praktikum ini, kami mempelajari konsep dasar linked list, termasuk operasi penambahan elemen di awal dan akhir, pencarian elemen, menampilkan isi linked list, dan penghapusan elemen tertentu. Linked list memungkinkan pengelolaan data secara dinamis dengan alokasi memori yang efisien, dan setiap operasi dasar memanfaatkan konsep pointer untuk mengakses serta memodifikasi node. Operasi pencarian (searching) menjadi dasar untuk operasi lanjut seperti insert after, delete after, dan update, yang memperlihatkan fleksibilitas linked list dalam pengelolaan data