

**LAPORAN PRAKTIKUM**  
**MODUL 6**  
**DOUBLE LINKED LIST BAGIAN 1**



**Nama :**

Muhammad Rifqi Al Baqi Ananta (2311104005)

**Dosen :**

Yudha Islami Sulistya, S.Kom., M.Cs.

**PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK**  
**FAKULTAS INFORMATIKA**  
**TELKOM UNIVERSITY PURWOKERTO**  
**2024**

**I. TP**

1. Soal 1

Code:

```

1  #include <iostream>
2  using namespace std;
3
4  // Struktur untuk node dalam Doubly Linked List
5  struct Node {
6      int data;
7      Node* next;
8      Node* prev;
9  };
10
11 // Struktur untuk List
12 struct List {
13     Node* head;
14     Node* tail;
15 };
16
17 // Inisialisasi List kosong
18 void init(List& L) {
19     L.head = nullptr;
20     L.tail = nullptr;
21 }
22
23 // Fungsi untuk menambah elemen di awal list
24 void insertFirst_2311104005(List& L, int nilai) {
25     Node* newNode = new Node;
26     newNode->data = nilai;
27     newNode->next = nullptr;
28     newNode->prev = nullptr;
29
30     if (L.head == nullptr) { // Jika list kosong
31         L.head = newNode;
32         L.tail = newNode;
33     } else { // Jika list tidak kosong
34         newNode->next = L.head;
35         L.head->prev = newNode;
36         L.head = newNode;
37     }
38 }
39
40 // Fungsi untuk menambah elemen di akhir list
41 void insertLast_2311104005(List& L, int nilai) {
42     Node* newNode = new Node;
43     newNode->data = nilai;
44     newNode->next = nullptr;
45     newNode->prev = nullptr;
46
47     if (L.head == nullptr) { // Jika list kosong
48         L.head = newNode;
49         L.tail = newNode;
50     } else { // Jika list tidak kosong
51         L.tail->next = newNode;
52         newNode->prev = L.tail;
53         L.tail = newNode;
54     }
55 }
56
57 // Fungsi untuk menampilkan semua elemen list
58 void tampilkanList(List L) {
59     Node* current = L.head;
60     cout << "DAFTAR ANGGOTA LIST: ";
61     while (current != nullptr) {
62         cout << current->data;
63         if (current->next != nullptr) {
64             cout << " <-> ";
65         }
66         current = current->next;
67     }
68     cout << endl;
69 }
70
71 int main() {
72     List L;
73     init(L);
74
75     int nilai;
76
77     cout << "Masukkan elemen pertama = ";
78     cin >> nilai;
79     insertFirst_2311104005(L, nilai);
80
81     cout << "Masukkan elemen kedua di awal = ";
82     cin >> nilai;
83     insertFirst_2311104005(L, nilai);
84
85     cout << "Masukkan elemen ketiga di akhir = ";
86     cin >> nilai;
87     insertLast_2311104005(L, nilai);
88
89     tampilkanList(L);
90
91     return 0;
92 }

```

Output:

```
Masukkan elemen pertama = 10
Masukkan elemen kedua di awal = 5
Masukkan elemen ketiga di akhir = 20
DAFTAR ANGGOTA LIST: 5 <-> 10 <-> 20
PS D:\TUGAS ITTP\SEMESTER 3\Praktikum Struktur Data\Pertemuan 6\TP_6\output>
```

2. Soal 2

Code:

```

1 #include <iostream>
2 using namespace std;
3
4 struct Node {
5     int data;
6     Node* next;
7     Node* prev;
8 };
9
10 struct List {
11     Node* head;
12     Node* tail;
13 };
14
15 void init(List& L) {
16     L.head = nullptr;
17     L.tail = nullptr;
18 }
19
20 void insertLast_2311104005(List& L, int nilai) {
21     Node* newNode = new Node;
22     newNode->data = nilai;
23     newNode->next = nullptr;
24     newNode->prev = nullptr;
25
26     if (L.head == nullptr) {
27         L.head = newNode;
28         L.tail = newNode;
29     } else {
30         L.tail->next = newNode;
31         newNode->prev = L.tail;
32         L.tail = newNode;
33     }
34 }
35
36 void deleteFirst_2311104005(List& L) {
37     if (L.head == nullptr) {
38         cout << "List kosong" << endl;
39         return;
40     }
41
42     Node* temp = L.head;
43
44     if (L.head == L.tail) { // Jika hanya satu elemen
45         L.head = nullptr;
46         L.tail = nullptr;
47     } else {
48         L.head = L.head->next;
49         L.head->prev = nullptr;
50     }
51
52     delete temp;
53 }
54
55 void deleteLast_2311104005(List& L) {
56     if (L.head == nullptr) {
57         cout << "List kosong" << endl;
58         return;
59     }
60
61     Node* temp = L.tail;
62
63     if (L.head == L.tail) { // Jika hanya satu elemen
64         L.head = nullptr;
65         L.tail = nullptr;
66     } else {
67         L.tail = L.tail->prev;
68         L.tail->next = nullptr;
69     }
70
71     delete temp;
72 }
73
74 void tampilList(List L) {
75     Node* current = L.head;
76     cout << "DAFTAR ANGGOTA LIST: ";
77     while (current != nullptr) {
78         cout << current->data;
79         if (current->next != nullptr) {
80             cout << " <-> ";
81         }
82         current = current->next;
83     }
84     cout << endl;
85 }
86
87 int main() {
88     List L;
89     init(L);
90
91     int nilai;
92
93     cout << "Masukkan elemen pertama = ";
94     cin >> nilai;
95     insertLast_2311104005(L, nilai);
96
97     cout << "Masukkan elemen kedua di akhir = ";
98     cin >> nilai;
99     insertLast_2311104005(L, nilai);
100
101     cout << "Masukkan elemen ketiga di akhir = ";
102     cin >> nilai;
103     insertLast_2311104005(L, nilai);
104
105     cout << "\nList sebelum penghapusan:" << endl;
106     tampilList(L);
107
108     cout << "\nMenghapus elemen pertama dan terakhir..." << endl;
109     deleteFirst_2311104005(L);
110     deleteLast_2311104005(L);
111
112     cout << "\nDaftar anggota list setelah penghapusan:" << endl;
113     tampilList(L);
114
115     return 0;
116 }

```

### Output:

```
Masukkan elemen pertama = 10
Masukkan elemen kedua di akhir = 13
Masukkan elemen ketiga di akhir = 15

List sebelum penghapusan:
DAFTAR ANGGOTA LIST: 10 <-> 13 <-> 15

Menghapus elemen pertama dan terakhir...

Daftar anggota list setelah penghapusan:
DAFTAR ANGGOTA LIST: 13
PS D:\TUGAS ITTP\SEMESTER 3\Praktikum Struktur Data\Pertemuan 6\TP_6\output>
```

### 3. Soal 3

#### Code:

```

1  #include <iostream>
2  using namespace std;
3
4  struct Node {
5      int data;
6      Node* next;
7      Node* prev;
8  };
9
10 struct List {
11     Node* head;
12     Node* tail;
13 };
14
15 void init(List& L) {
16     L.head = nullptr;
17     L.tail = nullptr;
18 }
19
20 void insertLast_2311104005(List& L, int nilai) {
21     Node* newNode = new Node;
22     newNode->data = nilai;
23     newNode->next = nullptr;
24     newNode->prev = nullptr;
25
26     if (L.head == nullptr) {
27         L.head = newNode;
28         L.tail = newNode;
29     } else {
30         L.tail->next = newNode;
31         newNode->prev = L.tail;
32         L.tail = newNode;
33     }
34 }
35
36 void tampilDepanBelakang_2311104005(List L) {
37     Node* current = L.head;
38     cout << "Daftar elemen dari depan ke belakang: ";
39     while (current != nullptr) {
40         cout << current->data;
41         if (current->next != nullptr) {
42             cout << " <-> ";
43         }
44         current = current->next;
45     }
46     cout << endl;
47 }
48
49 void tampilBelakangDepan_2311104005(List L) {
50     Node* current = L.tail;
51     cout << "Daftar elemen dari belakang ke depan: ";
52     while (current != nullptr) {
53         cout << current->data;
54         if (current->prev != nullptr) {
55             cout << " <-> ";
56         }
57         current = current->prev;
58     }
59     cout << endl;
60 }
61
62 int main() {
63     List L;
64     init(L);
65
66     cout << "Masukkan 4 elemen secara berurutan:" << endl;
67     for(int i = 1; i <= 4; i++) {
68         int nilai;
69         cout << "Masukkan elemen ke-" << i << ": ";
70         cin >> nilai;
71         insertLast_2311104005(L, nilai);
72     }
73
74     cout << endl;
75     tampilDepanBelakang_2311104005(L);
76     tampilBelakangDepan_2311104005(L);
77
78     return 0;
79 }

```

Output:

```
Masukkan 4 elemen secara berurutan:
Masukkan elemen ke-1: 10
Masukkan elemen ke-2: 15
Masukkan elemen ke-3: 20
Masukkan elemen ke-4: 25

Daftar elemen dari depan ke belakang: 10 <-> 15 <-> 20 <-> 25
Daftar elemen dari belakang ke depan: 25 <-> 20 <-> 15 <-> 10
PS D:\TUGAS ITTP\SEMESTER 3\Praktikum Struktur Data\Pertemuan 6\TP_6\output>
```

## II. PENJELASAN

1. Program ini mengimplementasikan struktur data Doubly Linked List dengan fokus pada operasi penambahan elemen. Pengguna dapat memasukkan tiga nilai, dengan dua nilai pertama ditambahkan di awal list menggunakan fungsi `insertFirst_2311104005()`, dan nilai ketiga ditambahkan di akhir list menggunakan fungsi `insertLast_2311104005()`. Program kemudian menampilkan seluruh elemen list yang terhubung dengan tanda "<->", menunjukkan bahwa setiap node memiliki koneksi dua arah.
2. Program ini merupakan pengembangan dari Doubly Linked List dengan penambahan operasi penghapusan. Program meminta pengguna memasukkan tiga nilai yang ditambahkan ke akhir list, kemudian melakukan operasi penghapusan elemen pertama (`deleteFirst_2311104005()`) dan elemen terakhir (`deleteLast_2311104005()`). Program menampilkan kondisi list sebelum dan sesudah penghapusan, memperlihatkan perubahan struktur data setelah operasi penghapusan dilakukan.
3. Program ini mendemonstrasikan kemampuan traversal dua arah dalam Doubly Linked List. Pengguna diminta memasukkan empat nilai yang disimpan dalam list, kemudian program menampilkan elemen-elemen tersebut dalam dua arah: dari depan ke belakang (`tampilDepanBelakang_2311104005()`) dan dari belakang ke depan (`tampilBelakangDepan_2311104005()`). Ini menunjukkan keunggulan Doubly Linked List yang memungkinkan navigasi dalam dua arah melalui penggunaan pointer `prev` dan `next`.