

LAPORAN PRAKTIKUM
PERTEMUAN 6
DOUBLE LINKED LIST (BAGIAN PERTAMA)



Nama :

Resita Istantia Purwanto (2311104037)

Dosen :

Yudha Islami Sulistya

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

I. TP

Soal 1

Kode:

```
1  #include <iostream>
2
3  Codeium: Refactor | Explain
4  struct Node {
5      int data;
6      Node* next;
7      Node* prev;
8
9      Node(int val) : data(val), next(nullptr), prev(nullptr) {}
10 };
11
12 Codeium: Refactor | Explain
13 class DoublyLinkedList {
14 private:
15     Node* head;
16
17 public:
18     DoublyLinkedList() : head(nullptr) {}
19
20     // Fungsi untuk menambahkan elemen di awal list
21     Codeium: Refactor | Explain | X
22     void insertFirst(int value) {
23         Node* newNode = new Node(value);
24         if (head == nullptr) {
25             head = newNode; // Jika List kosong, node baru menjadi head
26         } else {
27             newNode->next = head;
28             head->prev = newNode;
29             head = newNode; // Update head ke node baru
30         }
31     }
32
33     // Fungsi untuk menambahkan elemen di akhir list
34     Codeium: Refactor | Explain | X
35     void insertLast(int value) {
36         Node* newNode = new Node(value);
37         if (head == nullptr) {
38             head = newNode; // Jika List kosong, node baru menjadi head
39         } else {
40             Node* temp = head;
41             while (temp->next != nullptr) {
42                 temp = temp->next; // Mencari akhir list
43             }
44             temp->next = newNode;
45             newNode->prev = temp; // Hubungkan node baru dengan node terakhir
46         }
47     }
48
49     // Fungsi untuk menampilkan seluruh elemen dalam list
50     Codeium: Refactor | Explain | X
51     void display() {
52         Node* temp = head;
53         std::cout << "DAFTAR ANGGOTA LIST: ";
54         while (temp != nullptr) {
55             std::cout << temp->data;
56             if (temp->next != nullptr) {
57                 std::cout << " <-> ";
58             }
59             temp = temp->next;
60         }
61         std::cout << std::endl;
62     }
63 };
64
65 Codeium: Refactor | Explain | Generate Function Comment | X
66 int main() {
```

```

60 int main() {
61     DoublyLinkedList dll;
62
63     int firstElement, secondElement, thirdElement;
64
65     // Input elemen pertama
66     std::cout << "Masukkan elemen pertama = ";
67     std::cin >> firstElement;
68     dll.insertFirst(firstElement);
69
70     // Input elemen kedua di awal
71     std::cout << "Masukkan elemen kedua di awal = ";
72     std::cin >> secondElement;
73     dll.insertFirst(secondElement);
74
75     // Input elemen ketiga di akhir
76     std::cout << "Masukkan elemen ketiga di akhir = ";
77     std::cin >> thirdElement;
78     dll.insertLast(thirdElement);
79
80     // Tampilkan seluruh elemen
81     dll.display();
82
83     return 0;
84 }

```

Penjelasan:

- Node Structure: Menyimpan data, pointer next, dan prev.
- insertFirst(): Menambah elemen di awal list.
- insertLast(): Menambah elemen di akhir list.
- display(): Menampilkan elemen dari awal hingga akhir dengan tanda <-> di antara elemen.

Output:

```

Masukkan elemen pertama = 10
Masukkan elemen kedua di awal = 5
Masukkan elemen ketiga di akhir = 20
DAFTAR ANGGOTA LIST: 5 <-> 10 <-> 20
PS C:\06_Double_Linked_List_Bagian_1\TP>

```

Penjelasan:

Output diperoleh jika pengguna memasukkan 10 sebagai elemen pertama, 5 sebagai elemen kedua di awal, dan 20 di akhir.

Soal 2

Kode:

```

1  #include <iostream>
2
3  Codeium: Refactor | Explain
4  struct Node {
5      int data;
6      Node* next;
7      Node* prev;
8
9      Node(int val) : data(val), next(nullptr), prev(nullptr) {}
10 };
11
12 Codeium: Refactor | Explain
13 class DoublyLinkedList {
14 private:
15     Node* head;
16
17 public:
18     DoublyLinkedList() : head(nullptr) {}
19
20     // Fungsi untuk menambahkan elemen di akhir List
21     Codeium: Refactor | Explain | X
22     void insertLast(int value) {
23         Node* newNode = new Node(value);
24         if (head == nullptr) {
25             head = newNode; // Jika List kosong, node baru menjadi head
26         } else {
27             Node* temp = head;
28             while (temp->next != nullptr) {
29                 temp = temp->next; // Mencari akhir List
30             }
31             temp->next = newNode;
32             newNode->prev = temp; // Hubungkan node baru dengan node terakhir
33         }
34     }
35 }

```

```

33 // Fungsi untuk menghapus elemen pertama dalam List
34 Codeium: Refactor | Explain | X
35 void deleteFirst() {
36     if (head == nullptr) {
37         std::cout << "List kosong, tidak ada elemen yang bisa dihapus." << std::endl;
38         return;
39     }
40     Node* temp = head;
41     if (head->next == nullptr) {
42         head = nullptr; // Jika hanya ada satu elemen
43     } else {
44         head = head->next;
45         head->prev = nullptr; // Hapus referensi ke node lama
46     }
47     delete temp; // Hapus node pertama
48 }
49
50 // Fungsi untuk menghapus elemen terakhir dalam List
51 Codeium: Refactor | Explain | X
52 void deleteLast() {
53     if (head == nullptr) {
54         std::cout << "List kosong, tidak ada elemen yang bisa dihapus." << std::endl;
55         return;
56     }
57     Node* temp = head;
58     if (head->next == nullptr) {
59         head = nullptr; // Jika hanya ada satu elemen
60     } else {
61         while (temp->next != nullptr) {
62             temp = temp->next; // Mencari node terakhir
63         }
64         temp->next = nullptr;
65     }
66 }

```

```

62         temp->prev->next = nullptr; // Putuskan hubungan node terakhir
63     }
64     delete temp; // Hapus node terakhir
65 }
66
67 // Fungsi untuk menampilkan seluruh elemen dalam List
68 void display() {
69     if (head == nullptr) {
70         std::cout << "List kosong." << std::endl;
71         return;
72     }
73     Node* temp = head;
74     std::cout << "DAFTAR ANGGOTA LIST: ";
75     while (temp != nullptr) {
76         std::cout << temp->data;
77         if (temp->next != nullptr) {
78             std::cout << " <-> ";
79         }
80         temp = temp->next;
81     }
82     std::cout << std::endl;
83 }
84 };
85
86 int main() {
87     DoublyLinkedList dll;
88
89     int firstElement, secondElement, thirdElement;
90
91     // Input elemen pertama
92     std::cout << "Masukkan elemen pertama = ";
93     std::cin >> firstElement;
94     dll.insertLast(firstElement);
95
96     // Input elemen kedua di akhir
97     std::cout << "Masukkan elemen kedua di akhir = ";
98     std::cin >> secondElement;
99     dll.insertLast(secondElement);
100
101     // Input elemen ketiga di akhir
102     std::cout << "Masukkan elemen ketiga di akhir = ";
103     std::cin >> thirdElement;
104     dll.insertLast(thirdElement);
105
106     // Tampilkan list setelah penambahan elemen
107     dll.display();
108
109     // Hapus elemen pertama dan terakhir
110     dll.deleteFirst();
111     dll.deleteLast();
112
113     // Tampilkan list setelah penghapusan
114     std::cout << "DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: ";
115     dll.display();
116
117     return 0;
118 }

```

Penjelasan:

- Menambah elemen di akhir (insertLast).
- Menghapus elemen pertama (deleteFirst).
- Menghapus elemen terakhir (deleteLast).
- Menampilkan isi list (display).

Output:

```

Masukkan elemen pertama = 10
Masukkan elemen kedua di akhir = 15
Masukkan elemen ketiga di akhir = 20
DAFTAR ANGGOTA LIST: 10 <-> 15 <-> 20
DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: DAFTAR ANGGOTA LIST: 15
PS C:\06_Double_Linked_List_Bagian_1\TP>

```

Penjelasan: Pengguna memasukkan tiga elemen ke list, kemudian list ditampilkan.

Setelah itu, elemen pertama dan terakhir dihapus, dan list ditampilkan lagi.

Soal 3

Kode:

```
1  #include <iostream>
2
3  Codeium: Refactor | Explain
4  struct Node {
5      int data;
6      Node* next;
7      Node* prev;
8
9      Node(int val) : data(val), next(nullptr), prev(nullptr) {}
10 };
11
12 Codeium: Refactor | Explain
13 class DoublyLinkedList {
14 private:
15     Node* head;
16
17 public:
18     DoublyLinkedList() : head(nullptr) {}
19
20     // Fungsi untuk menambahkan elemen di akhir list
21     Codeium: Refactor | Explain | X
22     void insertLast(int value) {
23         Node* newNode = new Node(value);
24         if (head == nullptr) {
25             head = newNode; // Jika list kosong, node baru menjadi head
26         } else {
27             Node* temp = head;
28             while (temp->next != nullptr) {
29                 temp = temp->next; // Mencari akhir list
30             }
31             temp->next = newNode;
32             newNode->prev = temp; // Hubungkan node baru dengan node terakhir
33         }
34     }
35
36     // Fungsi untuk menampilkan elemen dari depan ke belakang
37     Codeium: Refactor | Explain | X
38     void displayForward() {
39         if (head == nullptr) {
40             std::cout << "List kosong." << std::endl;
41             return;
42         }
43         Node* temp = head;
44         std::cout << "Daftar elemen dari depan ke belakang: ";
45         while (temp != nullptr) {
46             std::cout << temp->data;
47             if (temp->next != nullptr) {
48                 std::cout << " <-> ";
49             }
50             temp = temp->next;
51         }
52         std::cout << std::endl;
53     }
54
55     // Fungsi untuk menampilkan elemen dari belakang ke depan
56     Codeium: Refactor | Explain | X
57     void displayBackward() {
58         if (head == nullptr) {
59             std::cout << "List kosong." << std::endl;
60             return;
61         }
62         Node* temp = head;
63         while (temp->next != nullptr) {
64             temp = temp->next; // Mencari node terakhir
65         }
66         std::cout << "Daftar elemen dari belakang ke depan: ";
67         while (temp != nullptr) {
68             std::cout << temp->data;
```

```

64         if (temp->prev != nullptr) {
65             std::cout << " <-> ";
66         }
67         temp = temp->prev;
68     }
69     std::cout << std::endl;
70 }
71 };
72
73 Codeium: Refactor | Explain | Generate Function Comment | X
74 int main() {
75     DoublyLinkedList dll;
76
77     int element;
78
79     // Input 4 elemen ke dalam list
80     for (int i = 1; i <= 4; i++) {
81         std::cout << "Masukkan elemen ke-" << i << ": ";
82         std::cin >> element;
83         dll.insertLast(element);
84     }
85
86     // Tampilkan elemen dari depan ke belakang
87     dll.displayForward();
88
89     // Tampilkan elemen dari belakang ke depan
90     dll.displayBackward();
91
92     return 0;
93 }

```

Penjelasan:

- insertLast: Fungsi untuk menambahkan node baru di akhir list.
- displayForward: Menampilkan elemen dari node pertama ke node terakhir.
- displayBackward: Menampilkan elemen dari node terakhir ke node pertama.

Output:

```

Masukkan elemen ke-1: 1
Masukkan elemen ke-2: 2
Masukkan elemen ke-3: 3
Masukkan elemen ke-4: 4
Daftar elemen dari depan ke belakang: 1 <-> 2 <-> 3 <-> 4
Daftar elemen dari belakang ke depan: 4 <-> 3 <-> 2 <-> 1
PS C:\06_Double_Linked_List_Bagian_1\TP>

```

Penjelasan: Pengguna memasukkan empat elemen ke dalam list. Setelah itu, list ditampilkan dua kali, pertama dari depan ke belakang dan kedua dari belakang ke depan.

II. UNGUIDED

Kode:


```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  Codeium: Refactor | Explain
6  struct Node {
7      int idBuku;
8      string judulBuku;
9      string penulisBuku;
10     Node* prev;
11     Node* next;
12 };
13
14 Codeium: Refactor | Explain
15 class DoubleLinkedList {
16 private:
17     Node* head;
18     Node* tail;
19 public:
20     DoubleLinkedList() : head(nullptr), tail(nullptr) {}
21
22     // Fungsi untuk menambahkan buku di akhir Linked List
23     Codeium: Refactor | Explain | X
24     void tambahBukuAkhir(int id, const string& judul, const string& penulis) {
25         Node* newNode = new Node{id, judul, penulis, nullptr, nullptr};
26         if (head == nullptr) {
27             head = tail = newNode;
28         } else {
29             tail->next = newNode;
30             newNode->prev = tail;
31             tail = newNode;
32         }
33         cout << "Buku berhasil ditambahkan!" << endl;
34     }
35
36     // Fungsi untuk menampilkan buku dari awal ke akhir
37     Codeium: Refactor | Explain | X
38     void tampilBukuDariAwal() const {
39         if (head == nullptr) {
40             cout << "Daftar buku kosong." << endl;
41             return;
42         }
43         Node* temp = head;
44         cout << "Daftar Buku dari Awal ke Akhir:" << endl;
45         while (temp != nullptr) {
46             cout << "ID Buku: " << temp->idBuku << ", Judul: " << temp->judulBuku
47             << ", Penulis: " << temp->penulisBuku << endl;
48             temp = temp->next;
49         }
50     }
51
52     // Fungsi untuk menampilkan buku dari akhir ke awal
53     Codeium: Refactor | Explain | X
54     void tampilBukuDariAkhir() const {
55         if (tail == nullptr) {
56             cout << "Daftar buku kosong." << endl;
57             return;
58         }
59         Node* temp = tail;
60         cout << "Daftar Buku dari Akhir ke Awal:" << endl;
61         while (temp != nullptr) {
62             cout << "ID Buku: " << temp->idBuku << ", Judul: " << temp->judulBuku
63             << ", Penulis: " << temp->penulisBuku << endl;
64             temp = temp->prev;
65         }
66     }
67 };
68
69 Codeium: Refactor | Explain | Generate Function Comment | X
70 int main() {
71     DoubleLinkedList daftarBuku;
72     int pilihan, id;
73     string judul, penulis;
74
75     do {
76         cout << "\nMenu:\n1. Tambah Buku\n2. Tampil Buku dari Awal ke Akhir\n3. Tampil Buku dari Akhir ke Awal\n0. Keluar\n";
77         cout << "Pilihan: ";
78         cin >> pilihan;
79
80         switch (pilihan) {
81             case 1:
82                 cout << "Masukkan ID Buku: ";
83                 cin >> id;
84                 cin.ignore();
85                 cout << "Masukkan Judul Buku: ";
86                 getline(cin, judul);
87                 cout << "Masukkan Penulis Buku: ";
88                 getline(cin, penulis);
89                 daftarBuku.tambahBukuAkhir(id, judul, penulis);
90                 break;
91             case 2:
92                 daftarBuku.tampilBukuDariAwal();
93                 break;
94             case 3:
95                 daftarBuku.tampilBukuDariAkhir();
96                 break;
97             case 0:
98                 return 0;
99             default:
100                cout << "Pilihan tidak valid. Silakan pilih kembali.\n";
101                continue;
102            }
103        } while (pilihan != 0);
104    }

```



```

90         daftarBuku.tampilBukuDariAkhir();
91         break;
92     case 0:
93         cout << "Program selesai." << endl;
94         break;
95     default:
96         cout << "Pilihan tidak valid." << endl;
97     }
98 } while (pilihan != 0);
99
100 return 0;
101 }
102

```

Penjelasan:

- Struktur Node: Menyimpan data buku dengan atribut idBuku, judulBuku, dan penulisBuku
- Kelas DoubleLinkedList:
 - Fungsi tambahBukuAkhir: Menambahkan buku di akhir list.
 - Fungsi tampilBukuDariAwal: Menampilkan buku dari head hingga tail.
 - Fungsi tampilBukuDariAkhir: Menampilkan buku dari tail hingga head.
- Menu Program: Memungkinkan pengguna memilih untuk menambahkan buku, menampilkan buku dari awal ke akhir, atau dari akhir ke awal.

Output:

```

Menu:
1. Tambah Buku
2. Tampil Buku dari Awal ke Akhir
3. Tampil Buku dari Akhir ke Awal
0. Keluar
Pilihan: 1
Masukkan ID Buku: 101
Masukkan Judul Buku: Double Linked List Bagian 1
Masukkan Penulis Buku: Resi
Buku berhasil ditambahkan!

Menu:
1. Tambah Buku
2. Tampil Buku dari Awal ke Akhir
3. Tampil Buku dari Akhir ke Awal
0. Keluar
Pilihan: 1
Masukkan ID Buku: 102
Masukkan Judul Buku: Double Linked List
Masukkan Penulis Buku: Tania
Buku berhasil ditambahkan!

Menu:
1. Tambah Buku
2. Tampil Buku dari Awal ke Akhir
3. Tampil Buku dari Akhir ke Awal
0. Keluar

```

```

Pilihan: 2
Daftar Buku dari Awal ke Akhir:
ID Buku: 101, Judul: Double Linked List Bagian 1, Penulis: Resi
ID Buku: 102, Judul: Double Linked List, Penulis: Tania

Menu:
1. Tambah Buku
2. Tampil Buku dari Awal ke Akhir
3. Tampil Buku dari Akhir ke Awal
0. Keluar
Pilihan: 3
Daftar Buku dari Akhir ke Awal:
ID Buku: 102, Judul: Double Linked List, Penulis: Tania
ID Buku: 101, Judul: Double Linked List Bagian 1, Penulis: Resi

Menu:
1. Tambah Buku
2. Tampil Buku dari Awal ke Akhir
3. Tampil Buku dari Akhir ke Awal
0. Keluar
Pilihan: 0
Program selesai.
PS C:\06_Double_Linked_List_Bagian_1\UNGUIDED>

```

Penjelasan:

- Tambah Buku: Pengguna memasukkan ID, judul, dan penulis buku. Buku ditambahkan di akhir daftar.
- Tampil dari Awal ke Akhir: Menampilkan semua buku dari yang pertama hingga terakhir.
- Tampil dari Akhir ke Awal: Menampilkan semua buku dari yang terakhir hingga pertama.
- Keluar Program: Memilih "0" untuk keluar dari program.