

LAPORAN PRAKTIKUM
PERTEMUAN 6
DOUBLE LINKED LIST



Nama :

Martryatus Sofia 2311104003

Dosen :

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

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

SOAL TP

Soal 1: Menambahkan Elemen di Awal dan Akhir DLL

Deskripsi Soal:

Buatlah program yang mengizinkan pengguna menambahkan elemen ke dalam DoublyLinked List di awal dan di akhir list.

Instruksi:

1. Implementasikan fungsi `insertFirst` untuk menambahkan elemen di awal list.
2. Implementasikan fungsi `insertLast` untuk menambahkan elemen di akhir list.
3. Tampilkan seluruh elemen dalam list dari depan ke belakang setelah penambahandilakukan.

Contoh Input:

- Input: Masukkan elemen pertama = 10
- Input: Masukkan elemen kedua di awal = 5
- Input: Masukkan elemen ketiga di akhir = 20

Output:

- DAFTAR ANGGOTALIST: 5 <-> 10 <-> 20

<pre>struct Buku { int id; string judul; string penulis; };</pre>	Struktur ini menyimpan informasi tentang buku. Memiliki tiga atribut: id: untuk menyimpan ID buku. judul: untuk menyimpan judul buku. penulis: untuk menyimpan nama penulis buku.
<pre>struct Node { Buku info; Node* next; Node* prev; };</pre>	Struktur ini merepresentasikan node dalam Doubly Linked List. Memiliki dua pointer: next dan prev, yang menunjuk ke node berikutnya dan sebelumnya, serta atribut info yang menyimpan data buku.
<pre>struct List { Node* first;</pre>	Struktur ini menyimpan pointer ke node pertama (first) dan node terakhir (last) dari

<pre>Node* last; };</pre>	linked list.
<pre>void createList(List &L) { L.first = nullptr; L.last = nullptr; }</pre>	Menginisialisasi list dengan mengatur pointer first dan last ke nullptr, menandakan bahwa list kosong.
<pre>Node* alokasi(Buku buku) { Node* newNode = new Node; newNode->info = buku; newNode->next = nullptr; newNode->prev = nullptr; return newNode; }</pre>	<p>Menciptakan dan mengalokasikan node baru dengan data buku yang diberikan.</p> <p>Mengembalikan pointer ke node yang baru dibuat.</p>
<pre>void insertLast(List &L, Node* newNode) { if (L.first == nullptr) { // Jika list kosong L.first = newNode; L.last = newNode; } else { // Jika list tidak kosong L.last->next = newNode; newNode->prev = L.last; L.last = newNode; } }</pre>	<p>Menambahkan node baru di akhir linked list.</p> <p>Jika list kosong, node baru menjadi first dan last.</p> <p>Jika tidak kosong, node baru ditambahkan setelah node terakhir yang ada.</p>
<pre>void printForward(const List &L) { Node* current = L.first; int i = 1; cout << "Daftar Buku (dari awal ke akhir):\n"; while (current != nullptr) { cout << "Buku " << i++ << ":\n"; cout << "ID Buku : " << current->info.id << endl; cout << "Judul : " << current->info.judul << endl; } }</pre>	<p>Menampilkan daftar buku dari node pertama hingga node terakhir.</p> <p>Menggunakan loop untuk mengakses setiap node dan mencetak informasi buku.</p>

<pre> cout << "Penulis : " << current- >info.penulis << endl; current = current->next; } }</pre>	
<pre> void printBackward(const List &L) { Node* current = L.last; int i = 1; cout << "Daftar Buku (dari akhir ke awal):\n"; while (current != nullptr) { cout << "Buku " << i++ << ":\n"; cout << "ID Buku : " << current->info.id << endl; cout << "Judul : " << current->info.judul << endl; cout << "Penulis : " << current- >info.penulis << endl; current = current->prev; } }</pre>	<p>Menampilkan daftar buku dari node terakhir hingga node pertama.</p> <p>Juga menggunakan loop untuk mengakses setiap node secara terbalik dan mencetak informasi buku.</p>
<pre> int main() { List L; createList(L); int jumlahBuku; cout << "Masukkan jumlah buku yang ingin ditambahkan: "; cin >> jumlahBuku; cin.ignore(); for (int i = 0; i < jumlahBuku; ++i) { Buku buku; cout << "\nMasukkan ID Buku : ";</pre>	<p>Program utama dimulai dengan membuat list kosong.</p> <p>Meminta pengguna untuk memasukkan jumlah buku yang ingin ditambahkan.</p> <p>Menggunakan loop untuk mendapatkan informasi buku (ID, judul, penulis) dari pengguna, membuat node baru dengan data tersebut, dan menambahkannya ke list.</p> <p>Setelah semua buku ditambahkan, program menampilkan daftar buku dari depan ke belakang dan dari belakang ke depan.</p>

<pre> cin >> buku.id; cin.ignore(); cout << "Masukkan Judul Buku: "; getline(cin, buku.judul); cout << "Masukkan Penulis : "; getline(cin, buku.penulis); Node* newNode = alokasi(buku); insertLast(L, newNode); } cout << "\n"; printForward(L); cout << "\n"; printBackward(L); return 0; } </pre>	
--	--

OUTPUT :

```

Double_Linked_List\UnGuided\" ; if ($?) { g++ TP_01.cpp -o TP_01 } ; if ($?) { .\TP_01 }
Masukkan elemen pertama = 10
Masukkan elemen kedua di awal = 5
Masukkan elemen ketiga di akhir = 20
DAFTAR ANGGOTA LIST: 5 <-> 10 <-> 20
PS F:\Kuliah\Semester3\Strukdat - praktikum\06_Double_Linked_list\06_Double_Linked_List\UnGuided>

```

CODINGAN VS CODE

```

1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  // Struktur Node untuk menyimpan data buku
7  struct Buku {
8      int id;
9      string judul;
10     string penulis;
11 };
12
13 struct Node {
14     Buku info;
15     Node* next;
16     Node* prev;
17 };
18
19 // Struktur list untuk menyimpan pointer awal dan akhir
20 struct List {
21     Node* first;
22     Node* last;
23 };
24
25 // Membuat list kosong
26 void createlist(List &L) {
27     L.first = nullptr;
28     L.last = nullptr;
29 }
30
31 // Alokasi node baru dengan data buku
32 Node* alokasi(Buku buku) {
33     Node* newNode = new Node;
34     newNode->info = buku;
35     newNode->next = nullptr;
36     newNode->prev = nullptr;
37     return newNode;
38 }
39
40 // Menambah buku di akhir linked list
41 void insertlast(List &L, Node* newNode) {
42     if (L.first == nullptr) { // Jika list kosong
43         L.first = newNode;
44         L.last = newNode;
45     } else { // Jika list tidak kosong
46         L.last->next = newNode;
47         newNode->prev = L.last;
48         L.last = newNode;
49     }
50 }
51
52 // Menampilkan daftar buku dari awal ke akhir
53 void printForward(const List &L) {
54     Node* current = L.first;
55     int i = 1;
56     cout << "Daftar Buku (dari awal ke akhir):\n";
57     while (current != nullptr) {
58         cout << "Buku " << i++ << ":\n";
59         cout << "ID Buku   : " << current->info.id << endl;
60         cout << "Judul     : " << current->info.judul <<
61         endl;
62         cout << "Penulis    : " << current->info.penulis <<
63         endl;
64         current = current->next;
65     }
66 }
67
68 // Menampilkan daftar buku dari akhir ke awal
69 void printBackward(const List &L) {
70     Node* current = L.last;
71     int i = 1;
72     cout << "Daftar Buku (dari akhir ke awal):\n";
73     while (current != nullptr) {
74         cout << "Buku " << i++ << ":\n";
75         cout << "ID Buku   : " << current->info.id << endl;
76         cout << "Judul     : " << current->info.judul <<
77         endl;
78         cout << "Penulis    : " << current->info.penulis <<
79         endl;
80         current = current->prev;
81     }
82 }
83
84 int main() {
85     List L;
86     createlist(L);
87
88     int jumlahBuku;
89     cout << "Masukkan jumlah buku yang ingin ditambahkan: ";
90
91     cin >> jumlahBuku;
92     cin.ignore();
93
94     for (int i = 0; i < jumlahBuku; ++i) {
95         Buku buku;
96         cout << "\nMasukkan ID Buku   : ";
97         cin >> buku.id;
98         cin.ignore();
99         cout << "Masukkan Judul Buku: ";
100        getline(cin, buku.judul);
101        cout << "Masukkan Penulis   : ";
102        getline(cin, buku.penulis);
103
104        Node* newNode = alokasi(buku);
105        insertlast(L, newNode);
106    }
107
108    cout << "\n";
109    printForward(L);
110    cout << "\n";
111    printBackward(L);
112
113    return 0;
114 }
115

```

Soal 2: Menghapus Elemen di Awal dan Akhir DLL

Deskripsi Soal:

Buatlah program yang memungkinkan pengguna untuk menghapus elemen pertama dan elemen terakhir dalam Doubly Linked List.

Instruksi:

1. Implementasikan fungsi `deleteFirst` untuk menghapus elemen pertama.
2. Implementasikan fungsi `deleteLast` untuk menghapus elemen terakhir.
3. Tampilkan seluruh elemen dalam list setelah penghapusan dilakukan.

Contoh Input:

- Input: Masukkan elemen pertama = 10
- Input: Masukkan elemen kedua di akhir = 15
- Input: Masukkan elemen ketiga di akhir = 20
- Hapus elemen pertama dan terakhir.

Output:

- DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: 15

<pre>struct Node { int data; Node* next; Node* prev; };</pre>	<p>Struktur ini merepresentasikan elemen dalam linked list.</p> <p>Memiliki:</p> <p>data: untuk menyimpan nilai integer.</p> <p>next: pointer ke node berikutnya dalam linked list.</p> <p>prev: pointer ke node sebelumnya.</p>
<pre>struct List { Node* first; Node* last; };</pre>	<p>Struktur ini merepresentasikan linked list itu sendiri.</p> <p>Memiliki dua pointer:</p> <p>first: menunjuk ke node pertama dari list.</p>

	last: menunjuk ke node terakhir dari list.
<pre>void createList(List &L) { L.first = nullptr; L.last = nullptr; }</pre>	Menginisialisasi list dengan menetapkan first dan last ke nullptr, menandakan bahwa list kosong.
<pre>Node* alokasi(int data) { Node* newNode = new Node; newNode->data = data; newNode->next = nullptr; newNode->prev = nullptr; return newNode; }</pre>	<p>Menciptakan dan mengalokasikan node baru dengan nilai data yang diberikan.</p> <p>Mengatur pointer next dan prev ke nullptr.</p> <p>Mengembalikan pointer ke node yang baru dibuat.</p>
<pre>void insertLast(List &L, int data) { Node* newNode = alokasi(data); if (L.first == nullptr) { // Jika list kosong L.first = newNode; L.last = newNode; } else { // Jika list tidak kosong newNode->prev = L.last; L.last->next = newNode; L.last = newNode; } }</pre>	<p>Menambahkan elemen baru di akhir linked list.</p> <p>Jika list kosong, node baru menjadi node pertama dan terakhir.</p> <p>Jika tidak kosong, node baru ditambahkan setelah node terakhir yang ada.</p>
<pre>void deleteFirst(List &L) { if (L.first != nullptr) { // Jika list tidak kosong Node* temp = L.first; if (L.first == L.last) { // Jika hanya ada satu elemen L.first = nullptr; L.last = nullptr; } else { L.first = L.first->next;</pre>	<p>Menghapus elemen pertama dari linked list.</p> <p>Jika list tidak kosong:</p> <p>Jika hanya ada satu elemen, set kedua pointer first dan last ke nullptr.</p> <p>Jika lebih dari satu elemen, memindahkan pointer first ke node kedua dan mengatur pointer prev pada node baru first ke nullptr.</p> <p>Menghapus node yang lama untuk mencegah kebocoran memori.</p>

<pre> L.first->prev = nullptr; } delete temp; } else { cout << "List kosong, tidak ada elemen yang bisa dihapus." << endl; } } </pre>	
<pre> void deleteLast(List &L) { if (L.last != nullptr) { // Jika list tidak kosong Node* temp = L.last; if (L.first == L.last) { // Jika hanya ada satu elemen L.first = nullptr; L.last = nullptr; } else { L.last = L.last->prev; L.last->next = nullptr; } delete temp; } else { cout << "List kosong, tidak ada elemen yang bisa dihapus." << endl; } } </pre>	<p>Menghapus elemen terakhir dari linked list.</p> <p>Jika list tidak kosong:</p> <p>Jika hanya ada satu elemen, set kedua pointer first dan last ke nullptr.</p> <p>Jika lebih dari satu elemen, memindahkan pointer last ke node sebelumnya dan mengatur pointer next pada node baru last ke nullptr.</p>
<pre> void printForward(const List &L) { Node* current = L.first; cout << "DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: "; if (current == nullptr) { cout << "List kosong." << endl; return; } } </pre>	<p>Menampilkan semua elemen dari linked list dari awal ke akhir.</p> <p>Jika list kosong, mencetak pesan yang sesuai.</p> <p>Menggunakan loop untuk mengakses setiap node dan mencetak nilai data.</p>

<pre>while (current != nullptr) { cout << current->data; if (current->next != nullptr) { cout << " <-> "; } current = current->next; } cout << endl; }</pre>	
---	--

OUTPUT CODINGAN

```
Masukkan elemen pertama = 10  
Masukkan elemen kedua di akhir = 15  
Masukkan elemen ketiga di akhir = 20  
DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: 15  
PS F:\Kuliah\Semester3\Strukdat - praktikum\06_Double_Linked_list\06_Double_Linked_List\UnGuided>
```

CODINGAN VS CODE

```

1 #include <iostream>
2
3 using namespace std;
4
5 // Struktur Node untuk menyimpan data elemen
6 struct Node {
7     int data;
8     Node* next;
9     Node* prev;
10 };
11
12 // Struktur List untuk menyimpan pointer awal dan akhir
13 struct List {
14     Node* first;
15     Node* last;
16 };
17
18 // Membuat list kosong
19 void createList(List &l) {
20     l.first = nullptr;
21     l.last = nullptr;
22 }
23
24 // Alokasi node baru dengan data elemen
25 Node* alokasi(int data) {
26     Node* newNode = new Node;
27     newNode->data = data;
28     newNode->next = nullptr;
29     newNode->prev = nullptr;
30     return newNode;
31 }
32
33 // Menambahkan elemen di akhir Linked List
34 void insertLast(List &l, int data) {
35     Node* newNode = alokasi(data);
36     if (l.first == nullptr) { // Jika list kosong
37         l.first = newNode;
38         l.last = newNode;
39     } else { // Jika list tidak kosong
40         newNode->prev = l.last;
41         l.last->next = newNode;
42         l.last = newNode;
43     }
44 }
45
46 // Menghapus elemen pertama dari linked list
47 void deleteFirst(List &l) {
48     if (l.first == nullptr) { // Jika list tidak kosong
49         Node* temp = l.first;
50         if (l.first == l.last) {
51             // Jika hanya ada satu elemen
52             l.first = nullptr;
53             l.last = nullptr;
54         } else {
55             l.first = l.first->next;
56             l.first->prev = nullptr;
57         }
58         delete temp;
59     } else {
60         cout << "
61         List kosong, tidak ada elemen yang bisa dihapus." << endl;
62     }
63 }
64
65 // Menghapus elemen terakhir dari linked list
66 void deleteLast(List &l) {
67     if (l.last != nullptr) { // Jika list tidak kosong
68         Node* temp = l.last;
69         if (l.first == l.last) {
70             // Jika hanya ada satu elemen
71             l.first = nullptr;
72             l.last = nullptr;
73         } else {
74             l.last = l.last->prev;
75             l.last->next = nullptr;
76         }
77         delete temp;
78     } else {
79         cout << "
80         List kosong, tidak ada elemen yang bisa dihapus." << endl;
81     }
82 }
83
84 // Menampilkan semua elemen dari depan ke belakang
85 void printForward(const List &l) {
86     Node* current = l.first;
87     cout << "DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: ";
88     if (current == nullptr) {
89         cout << "List kosong." << endl;
90         return;
91     }
92     while (current != nullptr) {
93         cout << current->data;
94         if (current->next != nullptr) {
95             cout << " <> ";
96         }
97         current = current->next;
98     }
99     cout << endl;
100 }
101
102 int main() {
103     List l;
104     createList(l);
105
106     // Input dan penambahan elemen sesuai instruksi soal
107     int data;
108
109     cout << "Masukkan elemen pertama = ";
110     cin >> data;
111     insertLast(l, data);
112
113     // Penambahan elemen pertama di akhir (atau awal karena
114     // list kosong)
115
116     cout << "Masukkan elemen kedua di akhir = ";
117     cin >> data;
118     insertLast(l, data);
119
120     // Penambahan elemen ketiga di akhir
121
122     // Hapus elemen pertama dan terakhir
123     deleteFirst(l);
124     deleteLast(l);
125
126     // Tampilkan list setelah penghapusan
127     printForward(l);
128
129     return 0;
130 }
131
132

```

Soal 3: Menampilkan Elemen dari Depan ke Belakang dan Sebaliknya

Deskripsi Soal: Buatlah program yang memungkinkan pengguna memasukkan beberapa elemen ke dalam Doubly Linked List. Setelah elemen dimasukkan, tampilkan seluruh elemen dalam list dari depan ke belakang, kemudian dari belakang ke depan.

Instruksi:

1. Implementasikan fungsi untuk menampilkan elemen dari depan ke belakang.
2. Implementasikan fungsi untuk menampilkan elemen dari belakang ke depan.
3. Tambahkan 4 elemen ke dalam list dan tampilkan elemen tersebut dalam dua arah.

Contoh Input:

- Input: Masukkan 4 elemen secara berurutan: 1, 2, 3, 4

Output:

- Daftar elemen dari depan ke belakang: 1 <-> 2 <-> 3 <-> 4
- Daftar elemen dari belakang ke depan: 4 <-> 3 <-> 2 <-> 1

HASIL CODINGAN VS CODE

```
Masukkan 4 elemen secara berurutan:
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
```

```

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    Node(int val) {
11        data = val;
12        next = nullptr;
13        prev = nullptr;
14    }
15 };
16
17 // Kelas untuk Doubly Linked List
18 class DoublyLinkedList {
19 private:
20     Node* head;
21
22 public:
23     DoublyLinkedList() {
24         head = nullptr;
25     }
26
27     // Fungsi untuk menambahkan elemen di akhir List
28     void append(int data) {
29         Node* newNode = new Node(data);
30         if (!head) { // Jika List kosong
31             head = newNode;
32             return;
33         }
34
35         Node* last = head;
36         while (last->next) { // Menemukan node terakhir
37             last = last->next;
38         }
39
40         last->next = newNode; // Menghubungkan node baru
41         newNode->prev = last;
42         // Mengatur prev untuk node baru
43     }
44
45     // Fungsi untuk menampilkan elemen dari depan ke belakang
46     void displayForward() {
47         Node* current = head;
48         cout << "Daftar elemen dari depan ke belakang: ";
49         while (current) {
50             cout << current->data;
51             if (current->next) {
52                 cout << " <-> ";
53             }
54             current = current->next;
55         }
56         cout << endl;
57     }
58
59     // Fungsi untuk menampilkan elemen dari belakang ke depan
60     void displayBackward() {
61         Node* current = head;
62         if (!current) return;
63
64         // Menemukan node terakhir
65         while (current->next) {
66             current = current->next;
67         }
68
69         cout << "Daftar elemen dari belakang ke depan: ";
70         while (current) {
71             cout << current->data;
72             if (current->prev) {
73                 cout << " <-> ";
74             }
75             current = current->prev;
76         }
77         cout << endl;
78     }
79 };
80
81 // Fungsi utama
82 int main() {
83     DoublyLinkedList dll;
84
85     // Input dari pengguna
86     cout << "Masukkan 4 elemen secara berurutan:" << endl;
87     for (int i = 0; i < 4; ++i) {
88         int element;
89         cout << "Masukkan elemen ke-" << (i + 1) << ": ";
90         cin >> element;
91         dll.append(element);
92     }
93
94     // Menampilkan elemen
95     dll.displayForward();
96     // Tampilkan dari depan ke belakang
97     dll.displayBackward();
98     // Tampilkan dari belakang ke depan
99
100    return 0;
101 }

```