

LAPORAN PRAKTIKUM
Tugas Pendahuluan Modul 06
“Double Linked List”



Disusun Oleh:
Hamidatun Nisa - 21104063
Struktur Data SE07-01

Dosen :
Yudha Islami Sulistya

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM PURWOKERTO
2024

1. Menambahkan Elemen di Awal dan Akhir DLL

Code :

```
06 Double Linked List_Bagian.1 > TP > @-lanhan,1opp > @mang
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 struct Node {
7     int data;
8     Node* next;
9     Node* prev;
10 };
11
12 struct list {
13     Node* head;
14     Node* tail;
15 };
16
17 void createList_21104063(list &L) {
18     L.head = nullptr;
19     L.tail = nullptr;
20 }
21
22 Node* alokasi_21104063(int value) {
23     Node* newNode = new Node;
24     newNode->data = value;
25     newNode->next = nullptr;
26     newNode->prev = nullptr;
27     return newNode;
28 }
29
30 void insertFirst_21104063(list &L, Node* P) {
31     if (L.head == nullptr) {
32         L.head = P;
33         L.tail = P;
34     } else {
35         P->next = L.head;
36         L.head->prev = P;
37         L.head = P;
38     }
39 }
40
41 void insertLast_21104063(list &L, Node* P) {
42     if (L.tail == nullptr) {
43         L.head = P;
44         L.tail = P;
45     } else {
46         P->prev = L.tail;
47         L.tail->next = P;
48         L.tail = P;
49     }
50 }
51
52 void insertSorted_21104063(list &L, Node* P) {
53     if (L.head == nullptr || L.head->data > P->data) {
54         P->next = L.head;
55         L.head->prev = P;
56         L.head = P;
57     } else {
58         Node* current = L.head;
59         while (current->next != nullptr && current->next->data < P->data) {
60             current = current->next;
61         }
62         P->next = current->next;
63         if (current->next != nullptr) {
64             current->next->prev = P;
65         }
66         L.tail = P;
67         current->next = P;
68         P->prev = current;
69     }
70 }
71
72 void deleteFirst_21104063(list &L) {
73     if (L.head == nullptr) {
74         cout << "List kosong, tidak ada elemen yang bisa dihapus" << endl;
75     } else if (L.head == L.tail) {
76         delete L.head;
77         L.head = nullptr;
78         L.tail = nullptr;
79     } else {
80         Node* temp = L.head;
81         L.head = L.head->next;
82         L.head->prev = nullptr;
83         delete temp;
84     }
85 }
86
87 void deleteLast_21104063(list &L) {
88     if (L.tail == nullptr) {
89         cout << "List kosong, tidak ada elemen yang bisa dihapus" << endl;
90     } else if (L.head == L.tail) {
91         delete L.tail;
92         L.head = nullptr;
93         L.tail = nullptr;
94     }
95 }
96
97 void bubbleSortList_21104063(list &L) {
98     do {
99         while (current->next != nullptr) {
100             current = current->next;
101         }
102         while (swapped) {
103             // swapped
104         }
105     } while (current->next != nullptr);
106 }
107
108 void printListFromHead_21104063(list L) {
109     if (L.head == nullptr) {
110         cout << "List kosong." << endl;
111     } else {
112         Node* temp = L.head;
113         while (temp != nullptr) {
114             cout << temp->data;
115             if (temp->next != nullptr) {
116                 cout << " <-> ";
117             }
118             temp = temp->next;
119         }
120         cout << endl;
121     }
122 }
123
124 int main() {
125     list L;
126     createList_21104063(L);
127
128     int value;
129
130     cout << "Masukkan elemen pertama = ";
131     cin >> value;
132     Node* newNode = alokasi_21104063(value);
133     insertFirst_21104063(L, newNode);
134
135     cout << "Masukkan elemen kedua di awal = ";
136     cin >> value;
137     newNode = alokasi_21104063(value);
138     insertFirst_21104063(L, newNode);
139
140     cout << "Masukkan elemen ketiga di akhir = ";
141     cin >> value;
142     newNode = alokasi_21104063(value);
143     insertLast_21104063(L, newNode);
144
145     cout << "Daftar ANGGOTA LIST: ";
146     printListFromHead_21104063(L);
147 }
```

Output :

```
Masukkan elemen pertama = 15
Masukkan elemen kedua di awal = 50
Masukkan elemen ketiga di akhir = 89

DAFTAR ANGGOTA LIST: 50 <-> 15 <-> 89
PS D:\= Collage\Semester 7\Praktikum Struktur Data\STD_Hamidatun_Nisa_21104063>
```

2. Menghapus Elemen di Awal dan Akhir DLL

Code :

```
06_Double_Linked_List_Bagian_1 > TP > latihan_2.cpp > main()
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 struct Node {
7     int data;
8     Node* next;
9     Node* prev;
10 };
11
12 struct list {
13     Node* head;
14     Node* tail;
15 };
16
17 void createlist_21104063(list &L) {
18     L.head = nullptr;
19     L.tail = nullptr;
20 }
21
22 Node* alokasi_21104063(int value) {
23     Node* newNode = new Node;
24     newNode->data = value;
25     newNode->next = nullptr;
26     newNode->prev = nullptr;
27     return newNode;
28 }
29
30 void insertFirst_21104063(list &L, Node* P) {
31     if (L.head == nullptr) {
32         L.head = P;
33         L.tail = P;
34     } else {
35         P->next = L.head;
36         L.head->prev = P;
37         L.head = P;
38     }
39 }
40
41 void insertLast_21104063(list &L, Node* P) {
42     if (L.tail == nullptr) {
43         L.head = P;
44         L.tail = P;
45     } else {
46         P->prev = L.tail;
47         L.tail->next = P;
48         L.tail = P;
49     }
50 }
51
52 void deleteFirst_21104063(list &L) {
53     if (L.head == nullptr) {
54         cout << "List kosong, tidak ada elemen yang dihapus";
55     } else if (L.head == L.tail) {
56         delete L.head;
57         L.head = nullptr;
58         L.tail = nullptr;
59     } else {
60         Node* temp = L.head;
61         L.head = L.head->next;
62         L.head->prev = nullptr;
63         delete temp;
64     }
65 }
66
67 void deleteLast_21104063(list &L) {
68     if (L.tail == nullptr) {
69         cout << "List kosong, tidak ada elemen yang dihapus";
70     } else if (L.head == L.tail) {
71         delete L.tail;
72         L.head = nullptr;
73         L.tail = nullptr;
74     } else {
75         Node* temp = L.tail;
76         L.tail = L.tail->prev;
77         L.tail->next = nullptr;
78         delete temp;
79     }
80 }
81
82 void printListFromHead_21104063(list L) {
83     if (L.head == nullptr) {
84         cout << "List kosong." << endl;
85     } else {
86         Node* temp = L.head;
87         while (temp != nullptr) {
88             cout << temp->data;
89             if (temp->next != nullptr) {
90                 cout << " <-> ";
91             }
92             temp = temp->next;
93         }
94         cout << endl;
95     }
96 }
97
98 int main() {
99     list L;
100     createlist_21104063(L);
101
102     int value;
103
104     cout << "Masukkan elemen pertama = ";
105     cin >> value;
106     Node* newNode = alokasi_21104063(value);
107     insertLast_21104063(L, newNode);
108
109     cout << "Masukkan elemen kedua di akhir = ";
110     cin >> value;
111     newNode = alokasi_21104063(value);
112     insertLast_21104063(L, newNode);
113
114     cout << "Masukkan elemen ketiga di akhir = ";
115     cin >> value;
116     newNode = alokasi_21104063(value);
117     insertLast_21104063(L, newNode);
118
119     cout << "\nDAFTAR ANGGOTA LIST SEBELUM PENGHAPUSAN:";
120     printListFromHead_21104063(L);
121
122     deleteFirst_21104063(L);
123
124     deleteLast_21104063(L);
125
126     cout << "\nDAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN:";
127     printListFromHead_21104063(L);
128
129     return 0;
130 }
```

Output :

```
Masukkan elemen pertama = 90
Masukkan elemen kedua di awal = 12
Masukkan elemen ketiga di akhir = 17

DAFTAR ANGGOTA LIST SEBELUM PENGHAPUSAN: 12 <-> 90 <-> 17

DAFTAR ANGGOTA LIST SETELAH PENGHAPUSAN: 90
PS D:\= Collage\Semester 7\Praktikum Struktur Data\STD_Hamidatun_Nisa_21104063>
```

3. Menampilkan Elemen dari Depan ke Belakang dan Sebaliknya

Code :

```
06_Double_Linked_List_Bagian_1 TP > @ lathun_3.cpp > ...
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 struct Node {
7     int data;
8     Node* next;
9     Node* prev;
10 };
11
12 struct List {
13     Node* head;
14     Node* tail;
15 };
16
17 void createList_21104063(List &L) {
18     L.head = nullptr;
19     L.tail = nullptr;
20 }
21
22 Node* alokasi_21104063(int value) {
23     Node* newNode = new Node;
24     newNode->data = value;
25     newNode->next = nullptr;
26     newNode->prev = nullptr;
27     return newNode;
28 }
29
30 void insertFirst_21104063(List &L, Node* P) {
31     if (L.head == nullptr) {
32         L.head = P;
33         L.tail = P;
34     } else {
35         P->next = L.head;
36         L.head->prev = P;
37         L.head = P;
38     }
39 }
40
41 void insertLast_21104063(List &L, Node* P) {
42     if (L.tail == nullptr) {
43         L.head = P;
44         L.tail = P;
45     } else {
46         P->prev = L.tail;
47         L.tail->next = P;
48         L.tail = P;
49     }
50 }
51
52 void deleteFirst_21104063(List &L) {
53     if (L.head == nullptr) {
54         cout << "List kosong, tidak ada elemen yang bisa dihapus";
55     } else if (L.head == L.tail) {
56         delete L.head;
57         L.head = nullptr;
58         L.tail = nullptr;
59     } else {
60         Node* temp = L.head;
61         L.head = L.head->next;
62         L.head->prev = nullptr;
63         delete temp;
64     }
65 }
66
67 void deleteLast_21104063(List &L) {
68     if (L.tail == nullptr) {
69         cout << "List kosong, tidak ada elemen yang bisa dihapus";
70     } else if (L.head == L.tail) {
71         delete L.tail;
72         L.head = nullptr;
73         L.tail = nullptr;
74     } else {
75         Node* temp = L.tail;
76         L.tail = L.tail->prev;
77         L.tail->next = nullptr;
78         delete temp;
79     }
80 }
81
82 void printListFromHead_21104063(List L) {
83     if (L.head == nullptr) {
84         cout << "List kosong." << endl;
85     } else {
86         Node* temp = L.head;
87         while (temp != nullptr) {
88             cout << temp->data;
89             if (temp->next != nullptr) {
90                 cout << " <-> ";
91             }
92             temp = temp->next;
93         }
94         cout << endl;
95     }
96 }
97
98 void printListFromTail_21104063(List L) {
99     if (L.tail == nullptr) {
100         cout << "List kosong." << endl;
101     } else {
102         Node* temp = L.tail;
103         while (temp != nullptr) {
104             cout << temp->data;
105             if (temp->prev != nullptr) {
106                 cout << " <-> ";
107             }
108             temp = temp->prev;
109         }
110         cout << endl;
111     }
112 }
113
114 int main() {
115     List L;
116     createList_21104063(L);
117     int value;
118
119     // Input empat elemen secara berurutan
120     for (int i = 1; i <= 4; i++) {
121         cout << "Masukkan elemen ke-" << i << " : ";
122         cin >> value;
123         Node* newNode = alokasi_21104063(value);
124         insertLast_21104063(L, newNode);
125     }
126
127     // Cetak list dari depan ke belakang
128     cout << "Daftar elemen dari depan ke belakang: ";
129     printListFromHead_21104063(L);
130
131     // Cetak list dari belakang ke depan
132     cout << "Daftar elemen dari belakang ke depan: ";
133     printListFromTail_21104063(L);
134
135     return 0;
136 }
```

Output :

```
Masukkan elemen ke-1 = 90
Masukkan elemen ke-2 = 54
Masukkan elemen ke-3 = 35
Masukkan elemen ke-4 = 75

Daftar elemen dari depan ke belakang: 90 <-> 54 <-> 35 <-> 75

Daftar elemen dari belakang ke depan: 75 <-> 35 <-> 54 <-> 90
PS D:\= Collage\Semester 7\Praktikum Struktur Data\STD_Hamidatun_Nisa_21104063>
```