

**LAPORAN PRAKTIKUM  
STRUKTUR DATA**

**MODUL VI  
DOUBLY LINKED LIST  
(BAGIAN PERTAMA)**



**Disusun Oleh :**  
NAMA : Afief Amar Purnomo  
NIM : 103112430067

**Dosen**  
FAHRUDIN MUKTI WIBOWO

**PROGRAM STUDI STRUKTUR DATA  
FAKULTAS INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO  
2025**

## A. Dasar Teori

Double linked list adalah perluasan dari single linked list di mana setiap node memiliki dua referensi: satu ke node sebelumnya dan satu ke node berikutnya dalam urutan. Dengan memiliki referensi ke node sebelumnya, double linked list memungkinkan navigasi maju dan mundur, yang membuatnya lebih fleksibel daripada single linked list. Namun, ini juga memerlukan alokasi memori tambahan untuk menyimpan referensi tambahan, sehingga dapat meningkatkan overhead memori.

Secara umum, double linked list sering digunakan dalam berbagai aplikasi seperti navigasi data dua arah, implementasi undo/redo pada aplikasi, serta struktur antrian ganda (deque). Implementasinya dapat ditemukan pada banyak bahasa pemrograman modern seperti C, C++, dan Java.

B. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1

```
⌚ main.cpp > [e] ptr_last
1  #include <iostream>
2  using namespace std;
3
4  struct Node
5  {
6      int data;
7      Node *prev;
8      Node *next;
9  };
10
11 Node *ptr_first = NULL;
12 Node *ptr_last = NULL;
13
Tabnine | Edit | Test | Explain | Document
14 void add_first(int value)
15 {
16     Node *newNode = new Node{value, NULL, ptr_first};
17
18     if (ptr_first == NULL)
19     {
20         ptr_last = newNode;
21     }
22     else
23     {
24         ptr_first -> prev = newNode;
25     }
26     ptr_first = newNode;
27 }
28
Tabnine | Edit | Test | Explain | Document
29 void add_last(int value)
30 {
```

```
31     Node *newNode = new Node{value, ptr_last, NULL};
32
33     if (ptr_last == NULL)
34     {
35         ptr_first = newNode;
36     }
37     else
38     {
39         ptr_last -> next = newNode;
40     }
41     ptr_last = newNode;
42 }
43
Tabnine | Edit | Test | Explain | Document
44 void add_target(int targetValue, int newValue)
45 {
46     Node *current = ptr_first;
47     while(current != NULL && current -> data != targetValue)
48     {
49         current = current -> next;
50     }
51
52     if (current != NULL)
53     {
54         if(current == ptr_last)
55         {
56             add_last(newValue);
57         }
58         else
59         {
60             Node *newNode = new Node {newValue, current, current -> next};
61             current -> next -> prev = newNode;
62             current -> next = newNode;
63         }
64     }
65 }
66
Tabnine | Edit | Test | Explain | Document
67 void view()
68 {
69     Node *current = ptr_first;
70     if (current == NULL)
71     {
72         cout << "List Kosong\n";
73         return;
74     }
75     while (current != NULL)
76     {
77         cout << current -> data << (current -> next != NULL ? " -> " : "");
78         current = current -> next;
79     }
80     cout << endl;
81 }
82
Tabnine | Edit | Test | Explain | Document
83 void delete_first()
84 {
85     if (ptr_first == NULL)
86     {
87         return;
```

```
88     Node *temp = ptr_first;
89
90     if (ptr_first == ptr_last)
91     {
92         ptr_first = NULL;
93         ptr_last = NULL;
94     }
95     else
96     {
97         ptr_first = ptr_first -> next;
98         ptr_first -> prev = NULL;
99     }
100    delete temp;
101 }
102
103 Tabnine | Edit | Test | Explain | Document
104 void delete_last()
105 {
106     if (ptr_last == NULL)
107         return;
108
109     Node *temp = ptr_last;
110
111     if (ptr_first == ptr_last)
112     {
113         ptr_first = NULL;
114         ptr_last = NULL;
115     }
116     else
117     {
118         ptr_last = ptr_last -> prev;
```

```
118     |     ptr_last -> next = NULL;
119     |
120     |     delete temp;
121 }
122
Tabnine | Edit | Test | Explain | Document
123 void delete_target(int targetValue)
124 {
125     Node *current = ptr_first;
126     while(current != NULL && current -> data != targetValue)
127     {
128         current = current -> next;
129     }
130
131     if (current != NULL)
132     {
133         if(current == ptr_first)
134         {
135             delete_first();
136             return;
137         }
138         if(current == ptr_last)
139         {
140             delete_last();
141             return;
142         }
143
144         current -> prev -> next = current -> next;
145         current -> next -> prev = current -> prev;
146         delete current;
```

```
147     }
148 }
149
Tabnine | Edit | Test | Explain | Document
150 void edit_node(int targetValue, int newValue)
151 {
152     Node *current = ptr_first;
153     while(current != NULL && current -> data != targetValue)
154     {
155         current = current -> next;
156     }
157
158     if (current != NULL)
159     {
160         current -> data = newValue;
161     }
162 }
163
Tabnine | Edit | Test | Explain | Document
164 int main()
165 {
166     add_first(10);
167     add_first(5);
168     add_last(20);
169     cout << "Awal\t\t\t: ";
170     view();
171
172     delete_first();
173     cout << "Setelah delete_first\t: ";
174     view();
175     delete_last();
176     cout << "Setelah delete_last\t: ";
177     view();
178
179     add_last(30);
180     add_last(40);
181     cout << "Setelah tambah\t\t: ";
182     view();
183
184     delete_target(30);
185     cout << "Setelah delete_target\t: ";
186     view();
187 }
```

## Screenshots Output

```
● PS D:\Kuliah\Semester 3\MATKUL\Praktikum Struktur Data\Praktikum\Modul_6\Guide> g++ main.cpp -o main
● PS D:\Kuliah\Semester 3\MATKUL\Praktikum Struktur Data\Praktikum\Modul_6\Guide> ./main
Awal          : 5 <-> 10 <-> 20
Setelah delete_first : 10 <-> 20
Setelah delete_last  : 10
Setelah tambah      : 10 <-> 30 <-> 40
Setelah delete_target: 10 <-> 40
○ PS D:\Kuliah\Semester 3\MATKUL\Praktikum Struktur Data\Praktikum\Modul_6\Guide> █
```

Deskripsi:

Program di atas merupakan implementasi struktur data double linked list dalam bahasa C++ yang memungkinkan penyimpanan dan pengelolaan data secara dinamis dua arah. Setiap node dalam list memiliki tiga komponen, yaitu data, pointer ke node sebelumnya (prev), dan pointer ke node berikutnya (next). Program ini mendefinisikan beberapa fungsi seperti add\_first untuk menambahkan node di awal list, add\_last untuk menambahkan node di akhir list, add\_target untuk menambahkan node baru setelah node dengan nilai tertentu, serta delete\_first, delete\_last, dan delete\_target untuk menghapus node di posisi awal, akhir, atau berdasarkan nilai tertentu. Selain itu, terdapat fungsi edit\_node untuk mengubah data node tertentu dan view untuk menampilkan seluruh isi list secara berurutan. Pada fungsi main, program menambahkan data 10, 5, dan 20, kemudian menampilkan isi list menggunakan fungsi view, dilanjutkan dengan operasi penghapusan dan penambahan kembali beberapa data seperti 30 dan 40. Sebagai contoh, setelah menambahkan data 30 dan 40, hasil tampilan list adalah “10 <-> 30 <-> 40”, yang menunjukkan bahwa data disimpan secara berurutan dan dapat ditelusuri maju maupun mundur, sesuai karakteristik dari double linked list.

C. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Unguided 1 (soal 1)

(file doublylist.h)

```
C doublylist.h > □ Kendaraan > ⌂ warna
1  #ifndef DOUBLYLIST_H
2  #define DOUBLYLIST_H
3  #include <iostream>
4  #include <string>
5  using namespace std;
6
7  struct Kendaraan {
8      string nopol;
9      string warna;
10     int thnbuat;
11 };
12
13 struct ElmList;
14 typedef ElmList* address;
15
16 struct ElmList {
17     Kendaraan info;
18     address next;
19     address prev;
20 };
21
22 struct List {
23     address first;
24     address last;
25 };
26
27 void createList(List &L);
28 address alokasi(Kendaraan x);
29 void dealokasi(address &p);
30 void insertFirst(List &L, address P);
31 void printInfo(List L);
32 bool isExist(List L, string nopol);
33 address findElm(List L, string x);
34 void deleteFirst(List &L, address &P);
35 void deleteLast(List &L, address &P);
36 void deleteAfter(address Prec, address &P);
37
38 #endif
```

(file doublylist.cpp)

```
⌚ doublylist.cpp > ...
1   #include "Doublylist.h"
2
3   Tabnine | Edit | Test | Explain | Document
4   void createList(List &L) {
5       L.first = NULL;
6       L.last = NULL;
7   }
8
9   Tabnine | Edit | Test | Explain | Document
10  address alokasi(Kendaraan x) {
11      address P = new ElmList;
12      P->info = x;
13      P->next = NULL;
14      P->prev = NULL;
15      return P;
16  }
17
18  Tabnine | Edit | Test | Explain | Document
19  void dealokasi(address &p) {
20      delete p;
21      p = NULL;
22  }
23
24  Tabnine | Edit | Test | Explain | Document
25  void insertFirst(List &L, address P) {
26      if (L.first == NULL) {
27          L.first = P;
28          L.last = P;
29      } else {
30          P->next = L.first;
31          L.first->prev = P;
32          L.first = P;
33      }
34  }
```

```
29     }
30 }
31
32     Tabnine | Edit | Test | Explain | Document
32 bool isExist(List L, string nopol) {
33     address P = L.first;
34     while (P != NULL) {
35         if (P->info.nopol == nopol)
36             return true;
37         P = P->next;
38     }
39     return false;
40 }
41
41     Tabnine | Edit | Test | Explain | Document
42 void printInfo(List L) {
43     cout << "\nDATA LIST 1\n";
44     address P = L.first;
45     while (P != NULL) {
46         cout << "Nomor Polisi : " << P->info.nopol << endl;
47         cout << "Warna      : " << P->info.warna << endl;
48         cout << "Tahun      : " << P->info.thnbuat << endl;
49         P = P->next;
50     }
51 }
52
52     Tabnine | Edit | Test | Explain | Document
53 address findElm(List L, string x) {
54     address P = L.first;
55     while (P != NULL) {
```

```

56         if (P->info.nopol == x) {
57             cout << "\nMasukkan Nomor Polisi yang dicari : " << x << endl;
58             cout << "Nomor Polisi : " << P->info.nopol << endl;
59             cout << "Warna      : " << P->info.warna << endl;
60             cout << "Tahun      : " << P->info.thnbuat << endl;
61             return P;
62         }
63         P = P->next;
64     }
65     cout << "Data dengan nomor polisi " << x << " tidak ditemukan.\n";
66     return NULL;
67 }
68
Tabnine | Edit | Test | Explain | Document
69 void deleteFirst(List &L, address &P) {
70     if (L.first != NULL) {
71         P = L.first;
72         if (L.first == L.last) {
73             L.first = NULL;
74             L.last = NULL;
75         } else {
76             L.first = P->next;
77             L.first->prev = NULL;
78             P->next = NULL;
79         }
80     }
81 }
82
83 void deleteLast(List &L, address &P) {
84     if (L.last != NULL) {
85         P = L.last;
86         if (L.first == L.last) {
87             L.first = NULL;
88             L.last = NULL;
89         } else {
90             L.last = P->prev;
91             L.last->next = NULL;
92             P->prev = NULL;
93         }
94     }
95 }
96
Tabnine | Edit | Test | Explain | Document
97 void deleteAfter(address Prec, address &P) {
98     if (Prec != NULL && Prec->next != NULL) {
99         P = Prec->next;
100        Prec->next = P->next;
101        if (P->next != NULL)
102            P->next->prev = Prec;
103        P->next = NULL;
104        P->prev = NULL;
105    }
106 }
107

```

(file main.cpp)

```
⌚ main.cpp > ⚙️ main()
1 #include "Doublylist.h"
2
3 Tabnine | Edit | Test | Explain | Document
4 int main() {
5     List L;
6     createList(L);
7     Kendaraan K;
8     char lagi;
9
10    do {
11        cout << "\nmasukkan nomor polisi : ";
12        cin >> K.nopol;
13
14        if (isExist(L, K.nopol)) {
15            cout << "nomor polisi sudah terdaftar\n";
16        } else {
17            cout << "masukkan warna kendaraan : ";
18            cin >> K.warna;
19            cout << "masukkan tahun kendaraan : ";
20            cin >> K.thnbuat;
21
22            address P = alokasi(K);
23            insertFirst(L, P);
24
25            cout << "\nTambah data lagi? (y/n): ";
26            cin >> lagi;
27        } while (lagi == 'y' || lagi == 'Y');
28
29        printInfo(L);
30
31        cout << "\n";
```

```
32     string cari = "D001";
33     findElm(L, cari);
34
35     cout << "\n";
36     string hapus = "D003";
37     cout << "Masukkan Nomor Polisi yang akan dihapus : " << hapus << endl;
38
39     address P = L.first, Prec = NULL;
40     bool found = false;
41
42     while (P != NULL) {
43         if (P->info.nopol == hapus) {
44             found = true;
45             break;
46         }
47         Prec = P;
48         P = P->next;
49     }
50
51     if (found) {
52         address delNode = NULL;
53         if (P == L.first) {
54             deleteFirst(L, delNode);
55         } else if (P == L.last) {
56             deleteLast(L, delNode);
57         } else {
58             deleteAfter(Prec, delNode);
59         }
60         cout << "Data dengan nomor polisi " << hapus << " berhasil dihapus.\n";
61         dealokasi(delNode);
62     } else {
63         cout << "Data tidak ditemukan.\n";
64     }
65
66     printInfo(L);
67
68     return 0;
69 }
```

## Screenshots Output

### 1) Tambah data

```
PS D:\Kuliah\Semester 3\MATKUL\Praktikum Struktur Data\Praktikum\Modul_6\Unguide> g++ main.cpp doublylist.cpp
● PS D:\Kuliah\Semester 3\MATKUL\Praktikum Struktur Data\Praktikum\Modul_6\Unguide> ./a.exe

masukkan nomor polisi : D001
masukkan warna kendaraan : Merah
masukkan tahun kendaraan : 90

Tambah data lagi? (y/n): y

masukkan nomor polisi : D003
masukkan warna kendaraan : Hitam
masukkan tahun kendaraan : 70

Tambah data lagi? (y/n): y

masukkan nomor polisi : D001
nomor polisi sudah terdaftar

Tambah data lagi? (y/n): y

masukkan nomor polisi : D004
masukkan warna kendaraan : Putih
masukkan tahun kendaraan : 90

Tambah data lagi? (y/n): n
```

```
DATA LIST 1
Nomor Polisi : D004
Warna       : Putih
Tahun        : 90
Nomor Polisi : D003
Warna       : Hitam
Tahun        : 70
Nomor Polisi : D001
Warna       : Merah
Tahun        : 90
```

### 2) Cari elemen dengan nomor polisi D001

```
Masukkan Nomor Polisi yang dicari : D001
Nomor Polisi : D001
Warna       : Merah
Tahun        : 90
```

3) Hapus elemen dengan nomor polisi D003

```
Masukkan Nomor Polisi yang akan dihapus : D003
Data dengan nomor polisi D003 berhasil dihapus.

DATA LIST 1
Nomor Polisi : D004
Warna        : Putih
Tahun         : 90
Nomor Polisi : D001
Warna        : Merah
Tahun         : 90
```

Deskripsi:

Program di atas merupakan program dalam bahasa C++ yang menggunakan struktur data double linked list untuk menyimpan, menampilkan, mencari, dan menghapus data kendaraan secara dinamis. File doublylist.h berisi deklarasi struktur Kendaraan yang menyimpan data berupa nomor polisi, warna, dan tahun pembuatan, serta struktur ElmList yang memiliki tiga komponen utama yaitu data kendaraan (info), pointer ke elemen berikutnya (next), dan pointer ke elemen sebelumnya (prev). Struktur List menyimpan alamat elemen pertama (first) dan elemen terakhir (last). File ini juga berisi deklarasi fungsi-fungsi seperti createList, alokasi, insertFirst, deleteFirst, deleteLast, deleteAfter, isExist, dan findElm. Implementasi dari fungsi-fungsi tersebut terdapat pada file doublylist.cpp, di mana createList digunakan untuk membuat list kosong, alokasi untuk membuat node baru, insertFirst untuk menambahkan data di awal list, isExist untuk memastikan nomor polisi tidak duplikat, serta printInfo untuk menampilkan seluruh data kendaraan. Sementara itu, file main.cpp berisi program utama yang meminta pengguna untuk memasukkan data kendaraan seperti nomor polisi, warna, dan tahun pembuatan, kemudian menambahkannya ke dalam list menggunakan fungsi insertFirst. Program juga dapat mencari data kendaraan berdasarkan nomor polisi menggunakan findElm dan menghapus data tertentu dengan deleteFirst, deleteLast, atau deleteAfter sesuai posisi data. Sebagai contoh, ketika pengguna memasukkan data kendaraan dengan nomor polisi D001, D002, dan D003, kemudian menghapus D003, maka program akan menampilkan daftar kendaraan yang tersisa secara berurutan, menunjukkan bagaimana double linked list bekerja untuk menyimpan dan mengelola data dua arah dengan efisien.

#### D. Kesimpulan

Berdasarkan hasil praktikum pada modul Doubly Linked List, dapat disimpulkan bahwa penggunaan struktur data ini memungkinkan pengelolaan data secara dinamis dengan kemampuan navigasi dua arah baik maju maupun mundur berkat adanya pointer prev dan next pada setiap node. Hal ini membuat operasi seperti penambahan, penghapusan, serta pencarian data menjadi lebih fleksibel dibandingkan Singly Linked List yang hanya dapat diakses satu arah.

Melalui bagian guided, mahasiswa memahami konsep dasar implementasi Doubly Linked List mulai dari pembuatan node, penyisipan data di awal, akhir, atau setelah elemen tertentu, hingga proses penghapusan dan penelusuran data. Sedangkan pada bagian unguided, konsep tersebut diterapkan dalam program pengelolaan data kendaraan, di mana pengguna dapat menambahkan, mencari, serta menghapus data kendaraan berdasarkan nomor polisi.

Sebagai contoh, ketika pengguna menambahkan data kendaraan dengan nomor polisi D001, D002, dan D003, lalu menghapus D003, program akan menampilkan daftar kendaraan yang tersisa (D001 <-> D002). Hal ini menunjukkan bahwa Doubly Linked List bekerja secara efisien dalam mengelola data yang membutuhkan fleksibilitas dua arah. Secara keseluruhan, praktikum ini memperkuat pemahaman tentang manipulasi pointer, alokasi memori dinamis, serta pentingnya struktur data Doubly Linked List dalam pengembangan program yang efisien dan terstruktur.

#### E. Referensi

Memahami Doubly Linked List dalam Struktur Data (2024). Daismabali.com.

[https://daismabali.com/artikel\\_detail/63/1/Memahami-Doubly-Linked-List-dalam-Struktu-Data-dengan-mudah.html](https://daismabali.com/artikel_detail/63/1/Memahami-Doubly-Linked-List-dalam-Struktu-Data-dengan-mudah.html)

RumahCoding. (2024). Linked list: Pengertian dan implementasi dasar. Rumah Coding.

<https://rumahcoding.co.id/linked-list-pengertian-dan-implementasi-dasar/>