

**LAPORAN PRAKTIKUM**

**MODUL 6**

**DOUBLE LINKED LIST (BAGIAN PERTAMA)**



**Disusun Oleh:**

**Rizaldy Aulia Rachman (2311104051)**

**S1SE-07-02**

**Dosen :**

**Wahyu Andi Saputra, S.Pd., M.Eng**

**PROGRAM STUDI S1 SOFTWARE ENGINEERING**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## I. TUJUAN

1. Memahami konsep modul *linked list*.
2. Mengaplikasikan konsep *double linked list* dengan menggunakan *pointer* dan dengan Bahasa C.

## II. LANDASAN TEORI

### 2.1 Double Linked List

**Double Linked List** adalah salah satu bentuk struktur data linked list yang memungkinkan traversal (penjelajahan) dalam dua arah: maju (forward) dan mundur (backward). Berbeda dengan **single linked list** yang hanya memungkinkan penjelajahan ke arah depan, double linked list memiliki dua referensi atau pointer pada setiap node, yaitu ke node **berikutnya** (next) dan **sebelumnya** (prev). Struktur ini memudahkan beberapa operasi seperti penghapusan atau penyisipan elemen dari kedua ujung list.

#### Komponen Utama Double Linked List:

1. **Node**: Setiap elemen dalam double linked list disebut node. Setiap node memiliki tiga komponen:
  - **Info**: Menyimpan informasi atau data (misalnya, dalam latihan, data kendaraan: nomor polisi, warna, dan tahun pembuatan).
  - **Next**: Pointer yang menunjuk ke elemen berikutnya dalam list.
  - **Prev**: Pointer yang menunjuk ke elemen sebelumnya dalam list.
2. **First**: Pointer yang menunjuk ke node pertama dalam list.
3. **Last**: Pointer yang menunjuk ke node terakhir dalam list.

### 2.2 Operasi Utama pada Double Linked List

#### 1. Insert (Sisipkan):

- **Insert First**: Menambahkan node di awal list.
- **Insert Last**: Menambahkan node di akhir list.
- **Insert After**: Menambahkan node setelah node tertentu.
- **Insert Before**: Menambahkan node sebelum node tertentu.

## 2. Delete (Hapus):

- **Delete First:** Menghapus node pertama dari list.
- **Delete Last:** Menghapus node terakhir dari list.
- **Delete After:** Menghapus node setelah node tertentu.
- **Delete Before:** Menghapus node sebelum node tertentu.

## 3. Search (Pencarian):

Pencarian elemen dalam double linked list dilakukan dengan menelusuri elemen dari awal ke akhir atau dari akhir ke awal.

### Keuntungan Double Linked List:

- **Traversal Dua Arah:** Dengan adanya dua pointer (next dan prev), list ini bisa dijelajahi dari depan ke belakang ataupun sebaliknya. Ini memudahkan operasi seperti penghapusan atau penyisipan pada kedua ujung list.
- **Penghapusan Lebih Mudah:** Karena memiliki pointer ke elemen sebelumnya, penghapusan elemen di double linked list lebih mudah daripada di single linked list, di mana kita harus melacak elemen sebelumnya secara manual.

### Kerugian Double Linked List:

- **Penggunaan Memori Lebih Banyak:** Setiap node memerlukan dua pointer (prev dan next), sehingga membutuhkan lebih banyak memori dibandingkan single linked list yang hanya membutuhkan satu pointer.
- **Kompleksitas Kode:** Operasi yang melibatkan manipulasi dua pointer bisa lebih kompleks dan berpotensi menyebabkan kesalahan.

## III. GUIDED

### 1. Guided

Code:

```

1 #include <iostream>
2 using namespace std;
3
4 class Node {
5 public:
6     int data;
7     Node* prev;
8     Node* next;
9 };
10
11 class DoublyLinkedList {
12 public:
13     Node* head;
14     Node* tail;
15
16     // Constructor untuk inisialisasi head dan tail
17     DoublyLinkedList() {
18         head = nullptr;
19         tail = nullptr;
20     }
21
22     // Fungsi untuk menambahkan elemen di depan list
23     void insert(int data) {
24         Node* newNode = new Node;
25         newNode->data = data;
26         newNode->prev = nullptr;
27         newNode->next = head;
28
29         if (head != nullptr) {
30             head->prev = newNode;
31         } else {
32             tail = newNode; // Jika list kosong, tail juga mengarah ke node baru
33         }
34         head = newNode;
35     }
36
37     // Fungsi untuk menghapus elemen dari depan list
38     void deleteNode() {
39         if (head == nullptr) {
40             return; // Jika list kosong
41         }
42         Node* temp = head;
43         head = head->next;
44         if (head != nullptr) {
45             head->prev = nullptr;
46         } else {
47             tail = nullptr; // Jika hanya satu elemen di list
48         }
49         delete temp; // Hapus elemen
50     }
51
52     // Fungsi untuk mengupdate data di list
53     bool update(int oldData, int newData) {
54         Node* current = head;
55         while (current != nullptr) {
56             if (current->data == oldData) {
57                 current->data = newData;
58                 return true; // Jika data ditemukan dan diupdate
59             }
60             current = current->next;
61         }
62         return false; // Jika data tidak ditemukan
63     }
64
65     // Fungsi untuk menghapus semua elemen di list
66     void deleteAll() {
67         Node* current = head;
68         while (current != nullptr) {
69             Node* temp = current;
70             current = current->next;
71             delete temp;
72         }
73         head = nullptr;
74         tail = nullptr;
75     }
76
77     // Fungsi untuk menampilkan semua elemen di list
78     void display() {
79         Node* current = head;
80         while (current != nullptr) {
81             cout << current->data << " ";
82             current = current->next;
83         }
84         cout << endl;
85     }
86 };
87
88 int main() {
89     DoublyLinkedList list;
90     while (true) {
91         cout << "1. Add data" << endl;
92         cout << "2. Delete data" << endl;
93         cout << "3. Update data" << endl;
94         cout << "4. Clear data" << endl;
95         cout << "5. Display data" << endl;
96         cout << "6. Exit" << endl;
97
98         int choice;
99         cout << "Enter your choice: ";
100         cin >> choice;
101
102         switch (choice) {
103             case 1: {
104                 int data;
105                 cout << "Enter data to add: ";
106                 cin >> data;
107                 list.insert(data);
108                 break;
109             }
110             case 2: {
111                 list.deleteNode();
112                 break;
113             }
114             case 3: {
115                 int oldData, newData;
116                 cout << "Enter old data: ";
117                 cin >> oldData;
118                 cout << "Enter new data: ";
119                 cin >> newData;
120                 bool updated = list.update(oldData, newData);
121                 if (!updated) {
122                     cout << "Data not found" << endl;
123                 }
124                 break;
125             }
126             case 4: {
127                 list.deleteAll();
128                 break;
129             }
130             case 5: {
131                 list.display();
132                 break;
133             }
134             case 6: {
135                 return 0;
136             }
137             default: {
138                 cout << "Invalid choice" << endl;
139                 break;
140             }
141         }
142     }
143     return 0;
144 }

```

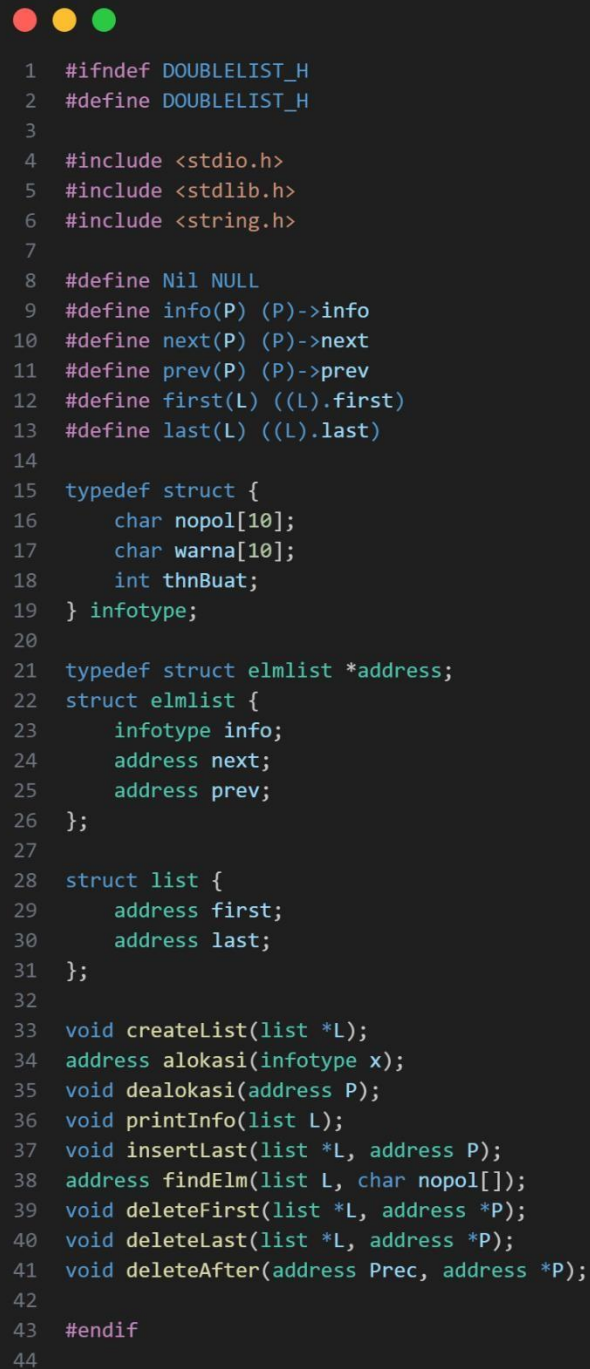
Output:

```
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 1
Enter data to add: 25
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 3
Enter old data: 25
Enter new data: 11
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
11
```

#### IV. UNGUIDED

##### 1. Doublelist.h

Code:



```

1  #ifndef DOUBLELIST_H
2  #define DOUBLELIST_H
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <string.h>
7
8  #define Nil NULL
9  #define info(P) (P)->info
10 #define next(P) (P)->next
11 #define prev(P) (P)->prev
12 #define first(L) ((L).first)
13 #define last(L) ((L).last)
14
15 typedef struct {
16     char nopol[10];
17     char warna[10];
18     int thnBuat;
19 } infotype;
20
21 typedef struct elmlist *address;
22 struct elmlist {
23     infotype info;
24     address next;
25     address prev;
26 };
27
28 struct list {
29     address first;
30     address last;
31 };
32
33 void createList(list *L);
34 address alokasi(infotype x);
35 void dealokasi(address P);
36 void printInfo(list L);
37 void insertLast(list *L, address P);
38 address findElm(list L, char nopol[]);
39 void deleteFirst(list *L, address *P);
40 void deleteLast(list *L, address *P);
41 void deleteAfter(address Prec, address *P);
42
43 #endif
44

```

## 2. Doublelist.cpp Code:

```

1  #include "doublelist.h"
2
3  void createList(list *L) {
4      first(*L) = Nil;
5      last(*L) = Nil;
6  }
7
8  address alokasi(infotype x) {
9      address P = (address)malloc(sizeof(struct elmList));
10     if (P != Nil) {
11         strcpy(info(P).nopol, x.nopol);
12         strcpy(info(P).warna, x.warna);
13         info(P).thnBuat = x.thnBuat;
14         next(P) = Nil;
15         prev(P) = Nil;
16     }
17     return P;
18 }
19
20 void dealokasi(address P) {
21     free(P);
22 }
23
24 void printInfo(list L) {
25     address P = first(L);
26     while (P != Nil) {
27         printf("NoPol: %s, Warna: %s, Tahun: %d\n", info(P).nopol, info(P).warna, info(P).thnBuat);
28         P = next(P);
29     }
30 }
31
32 void insertLast(list *L, address P) {
33     if ((first(*L) == Nil) {
34         first(*L) = P;
35         last(*L) = P;
36     } else {
37         prev(P) = last(*L);
38         next(last(*L)) = P;
39         last(*L) = P;
40     }
41 }
42
43 address findElm(list L, char nopol[]) {
44     address P = first(L);
45     while (P != Nil) {
46         if (strcmp(info(P).nopol, nopol) == 0) {
47             return P;
48         }
49         P = next(P);
50     }
51     return Nil;
52 }
53
54 void deleteFirst(list *L, address *P) {
55     *P = first(*L);
56     if (first(*L) != Nil) {
57         if (next(first(*L)) == Nil) {
58             first(*L) = Nil;
59             last(*L) = Nil;
60         } else {
61             first(*L) = next(*P);
62             prev(first(*L)) = Nil;
63         }
64         next(*P) = Nil;
65         prev(*P) = Nil;
66     }
67 }
68
69 void deleteLast(list *L, address *P) {
70     *P = last(*L);
71     if (last(*L) != Nil) {
72         if (prev(last(*L)) == Nil) {
73             first(*L) = Nil;
74             last(*L) = Nil;
75         } else {
76             last(*L) = prev(*P);
77             next(last(*L)) = Nil;
78         }
79         next(*P) = Nil;
80         prev(*P) = Nil;
81     }
82 }
83
84 void deleteAfter(address Prec, address *P) {
85     if (Prec != Nil && next(Prec) != Nil) {
86         *P = next(Prec);
87         if (next(*P) != Nil) {
88             next(Prec) = next(*P);
89             prev(next(*P)) = Prec;
90         } else {
91             next(Prec) = Nil;
92         }
93         next(*P) = Nil;
94         prev(*P) = Nil;
95     }
96 }
97

```

### 3. Main.cpp Code:

```

1  #include "doublelist.h"
2
3  int main() {
4      list L;
5      infotype kendaraan;
6      address P;
7
8      createlist(&L);
9
10     // Input kendaraan 1
11     strcpy(kendaraan.nopol, "D001");
12     strcpy(kendaraan.warna, "Hitam");
13     kendaraan.thnBuat = 2015;
14     P = alokasi(kendaraan);
15     insertLast(&L, P);
16
17     // Input kendaraan 2
18     strcpy(kendaraan.nopol, "D002");
19     strcpy(kendaraan.warna, "Merah");
20     kendaraan.thnBuat = 2018;
21     P = alokasi(kendaraan);
22     insertLast(&L, P);
23
24     // Input kendaraan 3
25     strcpy(kendaraan.nopol, "D003");
26     strcpy(kendaraan.warna, "Putih");
27     kendaraan.thnBuat = 2020;
28     P = alokasi(kendaraan);
29     insertLast(&L, P);
30
31     // Cetak semua data
32     printf("Data Kendaraan:\n");
33     printInfo(L);
34
35     // Cari kendaraan dengan NoPol D001
36     P = findElm(L, "D001");
37     if (P != Nil) {
38         printf("Kendaraan dengan NoPol D001 ditemukan: Warna %s, Tahun %d\n", info(P).warna, info(P).thnBuat);
39     } else {
40         printf("Kendaraan dengan NoPol D001 tidak ditemukan\n");
41     }
42
43     // Hapus kendaraan dengan NoPol D003
44     deletelast(&L, &P);
45     printf("Kendaraan dengan NoPol D003 telah dihapus\n");
46
47     // Cetak ulang data
48     printf("Data Kendaraan setelah penghapusan:\n");
49     printInfo(L);
50
51     return 0;
52 }
53

```

## Output Program:

```

Data Kendaraan:
NoPol: D001, Warna: Hitam, Tahun: 2015
NoPol: D002, Warna: Merah, Tahun: 2018
NoPol: D003, Warna: Putih, Tahun: 2020
Kendaraan dengan NoPol D001 ditemukan: Warna Hitam, Tahun 2015
Kendaraan dengan NoPol D003 telah dihapus
Data Kendaraan setelah penghapusan:
NoPol: D001, Warna: Hitam, Tahun: 2015
NoPol: D002, Warna: Merah, Tahun: 2018
PS C:\Praktikum Struktur data\pertemuan6>

```