

**SISTEM PENGELOLAAN DATA MEKANIK DAN LAYANAN SERVIS  
BENGKEL**



**Universitas  
Telkom**

**Disusun oleh Kelompok :**

**Yosevin Hendraji / 103092400010**

**Naila Ayu Permatasari/ 103092400061**

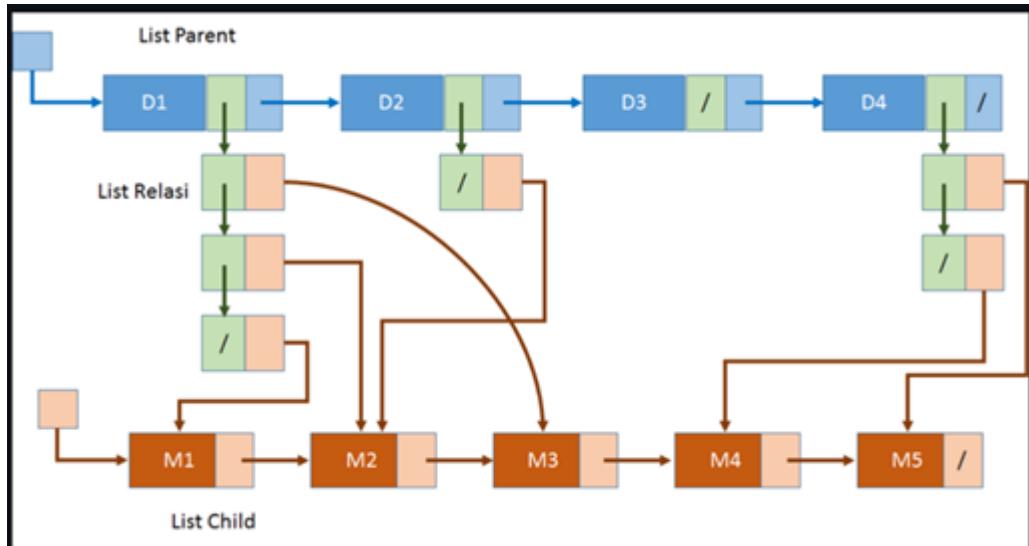
**Fauzan Januar/ 103092400037**

**Mata Kuliah: Struktur Data**

**Prodi Teknologi Informasi, Fakultas Informatika, Universitas Telkom, 2025**

- A. Type MLL : X
- B. Jenis List Parent : Single Linked List
- C. Jenis List Child : Single Linked List
- D. Model MLL
- E. Data Mekanik (Parents)
  - ID Mekanik
  - Nama Mekanik
  - Jam Kerja per Hari (int)
- F. Data Servis (child)
  - ID servis
  - Nama Kendaraan
  - Biaya Servis (int)

G. Spesifikasi Data Multi Linked List Tipe A



1. Sebuah bengkel membutuhkan sistem pendataan untuk mengelola informasi mengenai mekanik dan layanan servis yang mereka kerjakan setiap hari. Saat ini, pencatatan masih dilakukan secara manual sehingga sering terjadi kesalahan dalam pencatatan jam kerja mekanik, jenis servis yang ditangani, serta biaya servis yang harus dibayar pelanggan.

Asistem yang akan dibuat berfungsi untuk menuimpan daftar mekanik (sebagai **parent**) dan daftar servis yang dikerjakan setiap mekanik (sebagai **child**). Setiap mekanik memiliki ID, nama, dan jumlah jam kerja per hari. Sementara itu, setiap layanan servis memiliki ID servis, nama kendaraan yang diservis, dan biaya servis.

Sistem ini diharapkan membantu bengkel dalam:

- Melacak mekanik mana yang menangani servis tertentu.
- Menghitung total biaya servis.
- Mempermudah pelaporan aktivitas harian bengkel.
- Menghindari duplikasi data dan kehilangan catatan.

2.

a. .

```
C tubes.h > •o adrMekanik
1  #ifndef TUBES_H
2  #define TUBES_H
3
4  #include <iostream>
5  using namespace std;
6
7  // ADT Parent, disini kami menggunakan Mekanik sebagai parent
8  struct elmMekanik {
9      string IDMekanik;
10     string namaMekanik;
11     int jamKerja;
12 };
13
```

Penjelasan :

Data mekanik baru berisi ID Mekanik, Nama Mekanik, dan Jam Kerja per hari akan dimasukkan ke dalam **List Mekanik** sebagai **elemen terakhir**.

Jika list masih kosong, elemen mekanik baru akan menjadi elemen pertama.

Jika list sudah terisi, proses akan menelusuri node mekanik hingga node terakhir kemudian menambahkan elemen mekanik baru di posisi paling akhir.

b.

```
14 //Untuk childnya, kami menggunakan Servis sebagai child
15 struct elmServis {
16     string IDServis;
17     string namaKendaraan;
18     int biayaServis;
19 };
20
21 // Deklarasi forward
22 struct nodeMekanik;
23 struct nodeServis;
24 struct nodeRelasi;
25
```

Penjelasan:

Data **servis baru** yang berisi ID Servis, Nama Kendaraan, dan Biaya Servis akan dimasukkan ke dalam **List Servis** sebagai **elemen terakhir**.

Jika list servis masih kosong, data servis baru akan menjadi elemen pertama.

Jika list sudah terisi, proses akan menelusuri node servis hingga elemen terakhir, kemudian menambahkan elemen servis baru di bagian akhir list.

Setelah data servis berhasil dimasukkan, data tersebut **dapat langsung direlasikan** ke mekanik yang dipilih (jika ditempatkan melalui menu relasi).

Ketika relasi dibuat, maka pada parent (mekanik) yang dipilih jumlah layanan/servis yang ditanganinya akan **bertambah 1**.

c.

```
42 //Node relasi yang menghubungkan Mekanik dan Servis
43 struct nodeRelasi {
44     adrServis servis; // pointer ke servis
45     adrRelasi nextRelasi; // pointer ke relasi selanjutnya
46 };
47 
```

Penjelasan:

Relasi dibuat dengan menghubungkan satu mekanik dengan satu data servis yang dipilih.

Sistem akan mencari mekanik berdasarkan ID kemudian mencari data servis berdasarkan ID. Jika keduanya ditemukan, maka node relasi baru dibuat dan ditambahkan sebagai elemen terakhir pada **list relasi di dalam node mekanik tersebut**.

d.

```
//fungsi delete
void deleteMekanik(listMekanik &LM, string IDMekanik);
void deleteServis(listServis &LS, string IDServis);
void deleteRelasi(adrMekanik &M, string IDServis);
```

Penjelasan:

Data mekanik akan dicari berdasarkan ID Mekanik. Jika ditemukan, node mekanik tersebut akan dihapus dari list mekanik. Semua **relasi yang dimiliki mekanik tersebut juga otomatis dihapus** karena relasi berada di dalam parent.

**Penjelasan :**

Data servis dicari berdasarkan ID Servis dalam List Servis. Jika ditemukan, data servis dihapus dari list. Selain itu, seluruh relasi pada setiap mekanik yang menunjuk ke servis tersebut juga akan dihapus agar tidak ada pointer liar.

**Penjelasan :**

Relasi dihapus dengan cara menelusuri list relasi pada mekanik tertentu. Jika ditemukan relasi yang mengarah ke servis yang dipilih, node relasi tersebut dilepas dari list tanpa menghapus data mekanik atau data servisnya.

e.

```
//fungsi untuk menemukan data data
adrMekanik findMekanik(listMekanik LM, string IDMekanik);
adrServis findServis(listServis LS, string IDServis);
```

**Penjelasan :**

Pencarian dilakukan di List Servis berdasarkan **ID Servis**. Jika data servis ditemukan, pointer dikembalikan; jika tidak, hasilnya NULL.

**Penjelasan :**

Sistem menelusuri relasi pada mekanik tertentu untuk menemukan apakah mekanik tersebut memiliki relasi ke suatu servis tertentu (berdasarkan ID Servis). Jika ditemukan, pointer relasi dikembalikan; jika tidak, hasil NULL.

f.

```
//fungsi untuk menampilkan data
void showAllMekanik(listMekanik LM);
void showAllServis(listServis LS);
void showAllRelasi(listMekanik LM);
```

**Penjelasan :**

Semua mekanik yang berada di list akan ditampilkan secara berurutan, mulai dari elemen pertama hingga terakhir. Informasi yang ditampilkan adalah: ID Mekanik, Nama Mekanik, dan Jam Kerja.

**Penjelasan :**

List Servis ditelusuri dari elemen pertama hingga terakhir dan setiap data ditampilkan lengkap: ID Servis, Nama Kendaraan, dan Biaya Servis.

g.

```
void showServisdariMekanik(listMekanik LM);
void showMekanikdariServis(listMekanik LM, string IDServis);
```

**Tujuan Fungsi :**

Menampilkan **semua data Servis (child)** yang berelasi dengan **setiap Mekanik (parent)** pada list.

**Tujuan Fungsi :**

Menampilkan **data Mekanik (parent)** yang menangani **servis tertentu (child)** berdasarkan IDServis.

```
C tubes.h > •○ adrMekanik
1  #ifndef TUBES_H
2  #define TUBES_H
3
4  #include <iostream>
5  using namespace std;
6
7  // ADT Parent, disini kami menggunakan Mekanik sebagai parent
8  struct elmMekanik {
9      string IDMekanik;
10     string namaMekanik;
11     int jamKerja;
12 };
13
14 //Untuk childnya, kami menggunakan Servis sebagai child
15 struct elmServis {
16     string IDServis;
17     string namaKendaraan;
18     int biayaServis;
19 };
20
21 // Deklarasi forward
22 struct nodeMekanik;
23 struct nodeServis;
24 struct nodeRelasi;
25
26 typedef nodeMekanik* adrMekanik;
27 typedef nodeServis* adrServis;
28 typedef nodeRelasi* adrRelasi;
29
30 //Node parent yaitu Mekanik
31 struct nodeMekanik {
32     elmMekanik infoMekanik;
33     adrMekanik nextMekanik;
34     adrRelasi firstRelasi; // pointer ke relasi
35 };
36
37 //Node child yaitu Servis.
38 struct nodeServis {
39     elmServis infoServis;
40     adrServis nextServis; //ini adalah pointer ke servis selanjutnya
41 };
42
43 //Node relasi yang menghubungkan Mekanik dan Servis
44 struct nodeRelasi {
45     adrServis servis; // pointer ke servis
46     adrRelasi nextRelasi; // pointer ke relasi selanjutnya
47 };
48
49 //List struktur
50 struct listMekanik {
51     adrMekanik firstMekanik;
```

```
49 //List struktur
50 struct listMekanik {
51     adrMekanik firstMekanik;
52 };
53
54 struct listServis {
55     adrServis firstServis;
56 };
57
58 //Fungsi-fungsi
59 //Disini kai menggunakan singkatan agar tidak memakan waktu dan tempat
60 //LM = List Mekanik
61 //LS = List Servis
62
63 //ini fungsi untuk membuat list mekanik dan servis
64 void createListMekanik(listMekanik &LM);
65 void createListServis(listServis &LS);
66
67 //fungsi untuk membuat elemen mekanik, servis, dan relasi
68 adrMekanik createElmMekanik(elmMekanik dataMekanik);
69 adrServis createElmServis(elmServis dataServis);
70 adrRelasi createElmRelasi(adrServis servis);
71
72 //fungsi untuk menemukan data data
73 adrMekanik findMekanik(listMekanik LM, string IDMekanik);
74 adrServis findServis(listServis LS, string IDServis);
75
76 //fungsi untuk menampilkan data
77 void showAllMekanik(listMekanik LM);
78 void showAllServis(listServis LS);
79 void showAllRelasi(listMekanik LM);
80
81 void showServisdariMekanik(listMekanik LM);
82 void showMekanikdariServis(listMekanik LM, string IDServis);
83
84 //fungsi delete
85 void deleteMekanik(listMekanik &LM, string IDMekanik);
86 void deleteServis(listServis &LS, string IDServis);
87 void deleteRelasi(adrMekanik &M, string IDServis);
88
89 #endif
```

```
tubes.cpp > createListServis(listServis &)
1  #include "tubes.h"
2  #include <iostream>
3  using namespace std;
4
5  //untuk buat list mekanik
6  void createListMekanik(listMekanik &LM) {
7      LM.firstMekanik = nullptr;
8  }
9
10 //untuk buat list servis
11 void createListServis(listServis &LS) {
12     LS.firstServis = nullptr;
13 }
```

#### H. O X : 35% Yosevin

- Membuat data mekanik
- Membuat fungsi insert data mekanik
- Membuat fungsi edit & delete mekanik
- Pengujian fungsi mekanik

#### o Y : 35% Naila

- Membuat data servis
- Membuat fungsi insert data servis
- Membuat fungsi edit & delete servis
- Membuat relasi mekanik–servis

#### o Z : 30% Fauzan

- Membuat fungsi menampilkan semua data (mekanik beserta servis)
- Membuat fungsi pencarian mekanik & servis
- Integrasi semua menu
- Dokumentasi & penyusunan laporan

#### I. Bukti responsi tugas besar bersama asdos, asprak, ataupun dosen

