

## **Tugas Besar – CCH1D4 Struktur Data**

### **Kelompok 15**

Ahmad Hammam Muhajir Hanan - 103042400080

Muhammad Haiqal - 103042400013

IF-48-01 PJJ

### **Judul : Data Keluarga**

**Deskripsi :** Implementasikan multi linked-list yang memodelkan data orang tua dan data anak yang saling berelasi satu sama lainnya. Seorang anak memiliki 2 orang tua (1 laki-laki dan 1 perempuan) dan seorang bisa memiliki jumlah anak tertentu.

#### **Fungsionalitas :**

- a. Penambahan data orang tua.
- b. Penambahan data anak.
- c. Penentuan relasi antara orangtua dan anak (baik anak atau orang tua).
- d. Menghapus data orang tua tertentu.
- e. Menghapus data anak tertentu.
- f. Menampilkan data keseluruhan anak beserta orang tuanya.
- g. Menampilkan data anak yang dimiliki oleh orang tua tertentu.
- h. Menampilkan data orang tua tertentu yang memiliki anak tertentu.
- i. Menampilkan data orang tua yang memiliki jumlah anak paling banyak dan juga paling sedikit.

### **Spesifikasi Struktur Data**

Untuk memenuhi Tugas Besar ini, kami kelompok 15 memilih jenis struktur data **M ke N**, dengan data bentukan orang tua sebagai list parent yang memuat list relasi, kemudian data bentukan anak sebagai list child tanpa list relasi.

List parent merupakan tipe bentukan yang memiliki atribut (nama, jenis, usia) dimana nama dan jenis merupakan string yang menunjukkan identitas dan juga perannya sebagai orang tua, dan usia merupakan tipe data integer.

List child merupakan tipe bentukan yang memiliki atribut (nama, kelamin, dan usia) dimana nama dan jenis kelamin merupakan string yang menunjukkan identitas anak, dan usia merupakan tipe data integer.

### **Fungsionalitas yang kami kerjakan**

1. Insert data parent dari depan/belakang
2. Show all data parent
3. Menghapus data parent beserta relasinya
4. Mencari data parent
5. Mencari data child

6. Menambahkan data child
7. Menghubungkan data parent ke data child
8. Menampilkan seluruh data parent beserta childnya
9. Mencari data child pada parent tertentu
10. Menghapus data child pada parent tertentu beserta relasinya
11. Menghitung jumlah data child dari parent tertentu
12. Main program

## ADT (pseudocode)

### **Multi Linked List (M ke N)**

Type infotype\_orangTua : < nama, jenis: string  
                              usia : integer >

Type adr\_orangTua : pointer to elm\_orangTua

Type elm\_orangTua : < info: infotype\_orangTua  
                             next: adr\_orangTua  
                             anak: list\_relation >

Type infotype\_anak: < nama, kelamin: string            co: (Carl, laki-laki, 19)  
                              usia: integer >

Type adr\_anak : pointer to elm\_anak

Type elm\_anak : < info: infotype\_anak  
                             next: adr\_anak >

Type adr\_relation : pointer to elm\_relation

Type elm\_relation < next\_anak: adr\_anak  
                             next : adr\_relation >

Type list\_relation < first : adr\_relation >

Type list\_orangTua < first: adr\_orangTua >

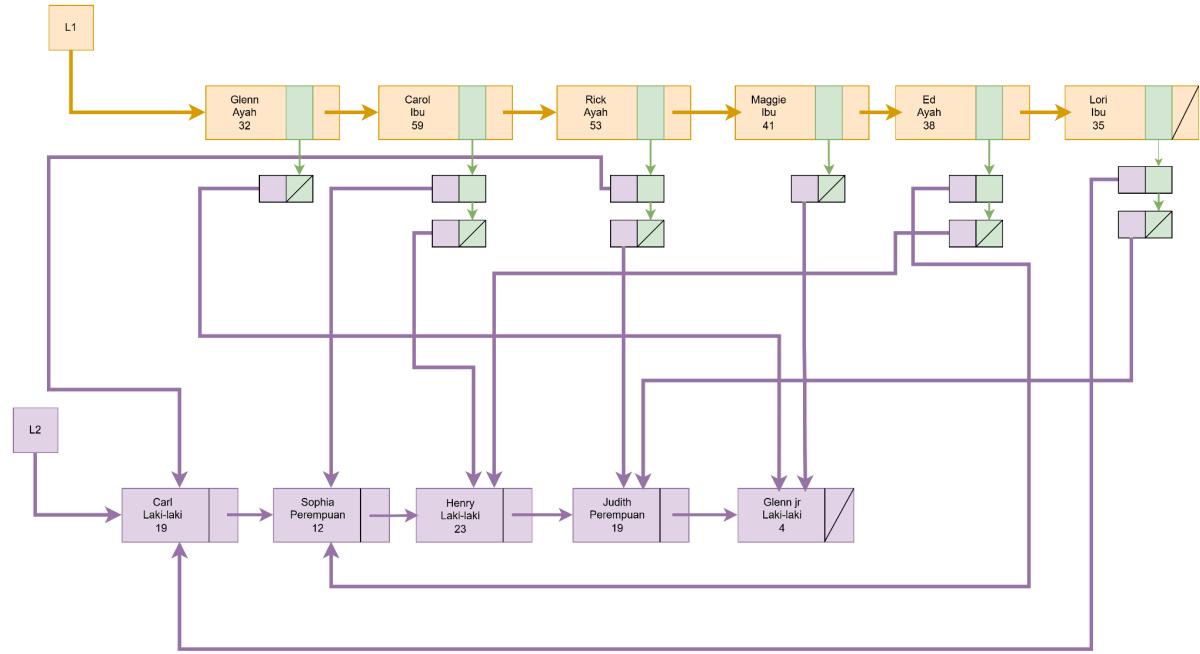
Type list\_anak < first: adr\_anak >

L1: list\_orangTua

L2: list\_anak

**Gambar 1 - ADT Pseudocode**

## Diagram MLL M-N (orang tua dan anak) dengan list\_relation



Gambar 2 - Diagram MLL M-N (orang tua dan anak)

## Pengerjaan Tugas

Seperti yang diperintahkan pada dokumen spesifikasi tugas besar, kami menggunakan IDE Code::Block untuk mengerjakan tugas ini dan menggunakan bahasa pemrograman C++ dengan mengimplementasikan Abstract Data Type (ADT). Untuk detail tugas bisa dilihat pada resource berikut

- Github Link: <https://github.com/iqoll/TugasBesarSTD-K15>
- File.exe (download untuk test program):  
<https://drive.google.com/file/d/1qT5UPIdGyZoZyi2k8KV-tYA2A5QjYCrU/view?usp=sharing>

## Dokumentasi Penggerjaan

### ADT dan Subprogram

MLL.h

#### 1. Struktur dasar Multi-Linked List Parent-Child

```
1 #ifndef MLL_H_INCLUDED
2 #define MLL_H_INCLUDED
3 #include <iostream>
4 using namespace std;
5
6 // forward declaration
7 typedef struct elm_orangTua *adr_orangTua;
8 typedef struct elm_anak *adr_anak;
9 typedef struct elm_relation *adr_relation;
10
11 // struktur untuk list relasi
12 struct list_relation
13 {
14     adr_relation first;
15 };
16
17 // struktur orang tua
18 struct infotype_orangTua
19 {
20     string nama, jenis;
21     int usia;
22 };
23
24 struct elm_orangTua
25 {
26     infotype_orangTua info;
27     adr_orangTua next;
28     list_relation anak;
29 };
30
31 struct list_orangTua
32 {
33     adr_orangTua first;
34 };
35
36 // struktur anak
37 struct infotype_anak
38 {
39     string nama, jenis_kelamin;
40     int usia;
41 };
42
43 struct elm_anak
44 {
45     infotype_anak info;
46     adr_anak next;
47 }
```

Gambar 3.1 - Struktur dasar MLL

#### 2. Handler function

```
67 // handler function
68 void handleInsertParent(list_orangTua &L);
69 void handleInsertChild(list_anak &L);
70 void handleDeleteParent(list_orangTua &L);
71 void handleFindParent(list_orangTua L);
72 void handleFindChild(list_anak L);
73 void handleShowAllParent(list_orangTua L);
74 void handleShowAllChild(list_anak L);
75 void handleCreateRelation(list_orangTua &L_ORTU, list_anak L_ANAK);
76 void handleDeleteRelation(list_orangTua &L_ORTU);
77 void handleShowAllParentAndChild(list_orangTua L_ORTU);
78 void handleFindChildOnParent(list_orangTua L_ORTU);
79 void handleCountChildOfParent(list_orangTua L_ORTU);
80
```

Gambar 3.2 - Handler function (MLL.h)

### 3. Main function

```
81 // --- FUNGSI/PROSEDUR UTAMA (PROTOTYPE) ---
82 // 1. Insert data parent dari depan
83 void insertFirstParent(list_orangTua &L, adr_orangTua P);
84 adr_orangTua createNewParent(infotype_orangTua info); // Fungsi pembantu untuk membuat node baru
85
86 // 2. Mencari data parent
87 adr_orangTua findParent(list_orangTua L, string namaParent);
88
89 // 3. Menghapus data parent beserta relasinya
90 // menggunakan deleteLast untuk list parent
91 void deleteParent(list_orangTua &L, string namaParent);
92 |
93 // 4. Menunjukkan semua data parent L1
94 void showAllParent(list_orangTua L);
95
96 // 5. Menambahkan data child dari belakang
97 void insertLastChild(list_anak &L, adr_anak C);
98 adr_anak createNewChild(infotype_anak info); // Fungsi pembantu untuk membuat node baru
99
100 // 6. Mencari data child
101 adr_anak findChild(list_anak L, string namaChild);
102
103 // 7. Menunjukkan semua data child L2
104 void showAllChild(list_anak L);
105
106 // 8. Menghubungkan data parent ke data child
107 adr_relation createNewRelation(adr_anak C); // C adalah pointer ke anak
108 void insertFirstRelation(list_relation &L, adr_relation R);
109
110 // 9. Menampilkan seluruh data parent beserta childnya
111 void showAllParentAndChild(list_orangTua L_Ortu);
112
113 // 10. Mencari data child pada parent tertentu
114 void printChildrenOfParent(list_orangTua L_ORTU, string namaParent);
115
116 // 11. Menghapus data child pada parent tertentu beserta relasinya
117 adr_relation findRelation(adr_orangTua P, string namaAnak);
118 void deleteRelation(list_orangTua &L_ORTU, adr_orangTua P, string namaAnak);
119 // Tambahan: Prosedur untuk menghapus Node Anak di L2 jika semua relasinya sudah dihapus
120 // adr_anak deleteChildNode(list_anak &L_Anak, string namaChild);
121
122 // 12. Menghitung jumlah data child dari parent tertentu
123 int countChildrenOfParent(list_orangTua L_ORTU, string namaParent);
124
125 #endif // MLL_H_INCLUDED
126
```

Gambar 3.3 - Main function (MLL.h)

### 4. Helper function

```
60
61 // procedure inisiasi data keluarga
62 void InisiasiDataKeluarga(list_orangTua &L1, list_anak &L2);
63
64 // main menu
65 int showMainMenu();
```

Gambar 3.4 - Helper function (MLL.h)

## Implementasi Subprogram

MLL.cpp

Beberapa dokumentasi screenshot (lengkapnya di file ZIP yang sudah diupload di LMS)

### 1. Handler function

```
104     // handler function
105     void handleInsertParent(list_orangTua &L)
106     {
107         infotype_orangTua infoBaru;
108         cout << "\n--- TAMBAH ORANG TUA BARU ---\n";
109         cout << "Nama: ";
110         cin.ignore(10000, '\n');
111         getline(cin, infoBaru.nama);
112
113         cout << "Jenis (Ayah/Ibu): ";
114         getline(cin, infoBaru.jenis);
115
116         cout << "usia: ";
117         while (!(cin >> infoBaru.usia))
118         {
119             cout << "Input usia tidak valid. Masukkan angka: ";
120             cin.clear();
121             cin.ignore(10000, '\n');
122         }
123
124         // --- 2. PROSES STRUKTUR DATA (Panggilan ke fungsi MLL inti) ---
125         adr_orangTua P_Baru = createNewParent(infoBaru);
126         insertFirstParent(L, P_Baru);
127
128         cout << "\n[SUCCESS] Orang Tua " << infoBaru.nama << " berhasil ditambahkan." << endl;
129     }
130 }
```

Gambar 3.5 - Implementasi Handler function (MLL.cpp)

### 2. Main function

```
345
346     void insertFirstParent(list_orangTua &L, adr_orangTua P)
347     {
348         // Kasus 1: List kosong
349         if (L.first == NULL)
350         {
351             L.first = P;
352         }
353         // Kasus 2: List tidak kosong
354         else
355         {
356             P->next = L.first;
357             L.first = P;
358         }
359     }
360
361     adr_orangTua createNewParent(infotype_orangTua info)
362     {
363         adr_orangTua P = new elm_orangTua;
364
365         P->info = info;
366         P->next = NULL;
367         P->anak.first = NULL;
368
369         return P;
370     }
371 }
```

Gambar 3.6 - Implementasi Main function (MLL.cpp)

### 3. Helper function

```
6 // inisiasi data keluarga (parent - children)
7 void InisiasiDataKeluarga(list_orangTua &L1, list_anak &L2)
8 {
9     L1.first = NULL;
10    L2.first = NULL;
11
12    // ayah
13    adr_orangTua P1 = new elm_orangTua;
14    P1->info = {"Rick", "Ayah", 53};
15    P1->next = NULL;
16    P1->anak.first = NULL;
17
18    // ibu
19    adr_orangTua P2 = new elm_orangTua;
20    P2->info = {"Lori", "Ibu", 35};
21    P2->next = NULL;
22    P2->anak.first = NULL;
23
24    // hubungkan parent-parent
25    L1.first = P1;
26    P1->next = P2;
27
28    // anak
29    adr_anak C1 = new elm_anak;
30    C1->info = {"Carl", "Laki-laki", 19};
31    C1->next = NULL;
32
33    L2.first = C1;
34
35    // relasi 1
36    adr_relation R1 = new elm_relation;
37    R1->next_anak = C1;
38    R1->next = NULL;
39
40    P1->anak.first = R1;
41
42    // relasi 2
43    adr_relation R2 = new elm_relation;
44    R2->next_anak = C1;
45    R2->next = NULL;
46
47    P2->anak.first = R2;
48
49 }
```

Gambar 3.7 - Implementasi Helper function (MLL.cpp)

## Main Program

main.cpp

```
1 #include <iostream>
2 #include "MLL.h"
3
4 using namespace std;
5
6 int main()
7 {
8     list_orangTua L_ORTU;
9     list_anak L_ANAK;
10
11    // menginisiasi list dengan seorang ayah, ibu, dan satu anak
12    InisiasiDataKeluarga(L_ORTU, L_ANAK);
13
14    int pilihanMenu;
15    do
16    {
17        pilihanMenu = showMainMenu();
18
19        switch (pilihanMenu)
20        {
21            case 1:
22                // Panggil InsertFirstParent
23                handleInsertParent(L_ORTU);
24                break;
25            case 2:
26                // Panggil findParent
27                handleFindParent(L_ORTU);
28                break;
29            case 3:
30                // Panggil deleteParent
31                handleDeleteParent(L_ORTU);
32                break;
33            case 4:
34                // panggil showAllParent
35                handleShowAllParent(L_ORTU);
36                break;
37            case 5:
38                // Panggil insertLastChild
39                handleInsertChild(L_ANAK);
40                break;
41            case 6:
42                // Panggil findChild
43                handleFindChild(L_ANAK);
44                break;
45            case 7:
46                // Panggil showAllChild
47                handleShowAllChild(L_ANAK);
48                break;
49            case 8:
50        }
51    } while (pilihanMenu != 9);
52 }
```

Gambar 3.8 - Main program (main.cpp)

```
48         break;
49     case 8:
50         // Panggil HubungkanParentToChild
51         handleCreateRelation(L_ORTU, L_ANAK);
52         break;
53     case 9:
54         // Panggil deleteRelation
55         handleDeleteRelation(L_ORTU);
56         break;
57     case 10:
58         // Panggil ShowAllParentAndChild
59         handleShowAllParentAndChild(L_ORTU);
60         break;
61     case 11:
62         // Panggil findChildOnParent
63         handleFindChildOnParent(L_ORTU);
64         break;
65     case 12:
66         // Panggil countChildOfParent
67         handleCountChildOfParent(L_ORTU);
68         break;
69     case 0:
70         cout << "Terima kasih telah menggunakan program." << endl;
71         break;
72     default:
73         cout << "Pilihan tidak valid. Silakan coba lagi." << endl;
74         break;
75     }
76
77     if (pilihanMenu != 0)
78     {
79         cout << "\nTekan ENTER untuk melanjutkan...";
80         cin.ignore();
81         cin.get();
82     }
83
84 } while (pilihanMenu != 0);
85
86
87 }
```

Gambar 3.9 - main program part 2 (main.cpp)

## Menjalankan Program

### Main Menu

```
C:\Users\Muhammad Haiqal\I X + ▾
=====
APLIKASI DATA KELUARGA (MLL M:N)
=====
:: MANAJEMEN DATA ORANG TUA (L1) ::

1. Tambah Data Orang Tua Baru (Insert First)
2. Cari Data Orang Tua
3. Hapus Data Orang Tua & Semua Relasinya
4. Tunjukkan List Orang Tua

:: MANAJEMEN DATA ANAK (L2) ::

5. Tambah Data Anak Baru (Insert Last)
6. Cari Data Anak
7. Tunjukkan List Anak

:: MANAJEMEN RELASI ::

8. Hubungkan Orang Tua ke Anak (Buat Relasi)
9. Hapus Relasi Tertentu (Putus Hubungan Ortu-Anak)

:: PELAPORAN & ANALISIS ::

10. Tampilkan Seluruh Data Keluarga (Ortu & Anak)
11. Cari Anak dari Orang Tua Tertentu
12. Hitung Jumlah Anak dari Orang Tua Tertentu

0. KELUAR DARI PROGRAM

Masukkan Pilihan Anda (0-12): |
```

Gambar 4.1 - Main Menu

## Menjalankan Fungsi

### Manajemen Data Orang Tua (L1)

#### 1. Menambahkan Data Orang Tua

```
0. KELUAR DARI PROGRAM
=====
Masukkan Pilihan Anda (0-12): 1

--- TAMBAH ORANG TUA BARU ---
Nama: Glenn
Jenis (Ayah/Ibu): Ayah
usia: 36

[SUCCESS] Orang Tua Glenn berhasil ditambahkan.

Tekan ENTER untuk melanjutkan...|
```

Gambar 4.2 - InsertFirstParent

2. Mencari Data Orang Tua

```
0. KELUAR DARI PROGRAM
=====
Masukkan Pilihan Anda (0-12): 2

--- CARI DATA ORANG TUA ---
Masukkan Nama Orang Tua yang dicari: Rick

[SUCCESS] Orang Tua ditemukan!
Nama: Rick
Jenis: Ayah
Usia: 53 tahun
Anak: 1. Carl

Tekan ENTER untuk melanjutkan...|
```

Gambar 4.3 - findParent

3. Menghapus Data Orang Tua & Semua Relasinya

```
0. KELUAR DARI PROGRAM
=====
Masukkan Pilihan Anda (0-12): 3

--- HAPUS DATA ORANG TUA ---
Masukkan Nama Orang Tua yang akan dihapus: Rick
[SUCCESS] Orang Tua Rick beserta semua relasinya berhasil dihapus.

Tekan ENTER untuk melanjutkan...|
```

Gambar 4.4 - deleteParent

4. Menunjukkan List Orang Tua

```
0. KELUAR DARI PROGRAM
=====
Masukkan Pilihan Anda (0-12): 4
=====

DAFTAR SELURUH ORANG TUA (L1)
=====

1. Nama      : Glenn
   Jenis     : Ayah
   Usia      : 36 tahun
=====

2. Nama      : Lori
   Jenis     : Ibu
   Usia      : 35 tahun
=====

Total Orang Tua dalam List: 2
=====

Tekan ENTER untuk melanjutkan...|
```

Gambar 4.5 - showAllParent

## Manajemen Data Anak (L2)

### 5. Menambah Data Anak Baru (Insert Last)

```
0. KELUAR DARI PROGRAM
=====
Masukkan Pilihan Anda (0-12): 5

--- TAMBAH ANAK BARU ---
Nama: Judith
Jenis kelamin: Perempuan
Usia: 12

[SUCCESS] Anak Judith berhasil ditambahkan.

Tekan ENTER untuk melanjutkan...|
```

Gambar 4.6 - insertLastChild

### 6. Mencari Data Anak

```
0. KELUAR DARI PROGRAM
=====
Masukkan Pilihan Anda (0-12): 6

--- CARI DATA ANAK ---
Masukkan Nama Anak yang dicari: Carl

[SUCCESS] Anak ditemukan!
Nama: Carl
Jenis Kelamin: Laki-laki
Usia: 19 tahun

Tekan ENTER untuk melanjutkan...|
```

Gambar 4.7 - findChild

7. Menunjukkan List Anak

```
0. KELUAR DARI PROGRAM
=====
Masukkan Pilihan Anda (0-12): 7
=====
DAFTAR SELURUH ANAK (L2)
=====

1. Nama : Carl
Jenis Kelamin : Laki-laki
Usia : 19 tahun

2. Nama : Judith
Jenis Kelamin : Perempuan
Usia : 12 tahun

=====
Total Anak dalam List: 2
=====

Tekan ENTER untuk melanjutkan...|
```

Gambar 4.8 - showAllChild

### Manajemen Relasi

8. Membuat Hubungan Orang Tua ke Anak (Buat Relasi)

```
0. KELUAR DARI PROGRAM
=====
Masukkan Pilihan Anda (0-12): 8
=====
--- BUAT RELASI ORANG TUA - ANAK ---
Masukkan Nama Orang Tua: Lori
Masukkan Nama Anak: Judith
[SUCCESS] Relasi antara Lori dan Judith berhasil dibuat.

Tekan ENTER untuk melanjutkan...|
```

Gambar 4.9 - createRelation

9. Menghapus Relasi Tertentu (Putus Hubungan Ortu-Anak)

```
0. KELUAR DARI PROGRAM
=====
Masukkan Pilihan Anda (0-12): 9
=====
--- HAPUS RELASI ORANG TUA - ANAK ---
Masukkan Nama Orang Tua: Lori
Masukkan Nama Anak yang relasinya akan dihapus: Judith
[SUCCESS] Relasi Orang Tua Lori dengan Anak Judith berhasil dihapus.

Tekan ENTER untuk melanjutkan...|
```

Gambar 4.10 - deleteRelation

## Pelaporan & Analisis

10. Tampilkan Seluruh Data Keluarga (Ortu & Anak)

```
0. KELUAR DARI PROGRAM
=====
Masukkan Pilihan Anda (0-12): 10
=====
LAPORAN SELURUH DATA KELUARGA
=====

--- [1] ORANG TUA ---
Nama: Glenn
Jenis (Ayah/Ibu): Ayah
Usia: 36
-----
ANAK YANG BERELASI:
[INFO] Tidak memiliki relasi anak.

--- [2] ORANG TUA ---
Nama: Lori
Jenis (Ayah/Ibu): Ibu
Usia: 35
-----
ANAK YANG BERELASI:
1. Carl (Laki-laki, 19 th)
-----
Total Anak: 1
=====

Tekan ENTER untuk melanjutkan...|
```

Gambar 4.11 - showAllParentAndChild

11. Mencari Anak dari Orang Tua Tertentu

```
0. KELUAR DARI PROGRAM
=====
Masukkan Pilihan Anda (0-12): 11
=====
--- CARI ANAK DARI ORANG TUA TERTENTU ---
Masukkan Nama Orang Tua: Lori

--- DAFTAR ANAK DARI ORANG TUA: Lori ---
- Nama: Carl, Jenis Kelamin: Laki-laki, Usia: 19
Total anak yang ditemukan: 1

Tekan ENTER untuk melanjutkan...|
```

Gambar 4.12 - findChildOnParent

12. Menghitung Jumlah Anak dari Orang Tua Tertentu

```
0. KELUAR DARI PROGRAM
=====
Masukkan Pilihan Anda (0-12): 12
--- HITUNG JUMLAH ANAK DARI ORANG TUA TERTENTU ---
Masukkan Nama Orang Tua: Lori
[RESULT] Orang Tua Lori memiliki total 1 anak.
Tekan ENTER untuk melanjutkan...|
```

Gambar 4.13 - countChildOfParent