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 menunjukan 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

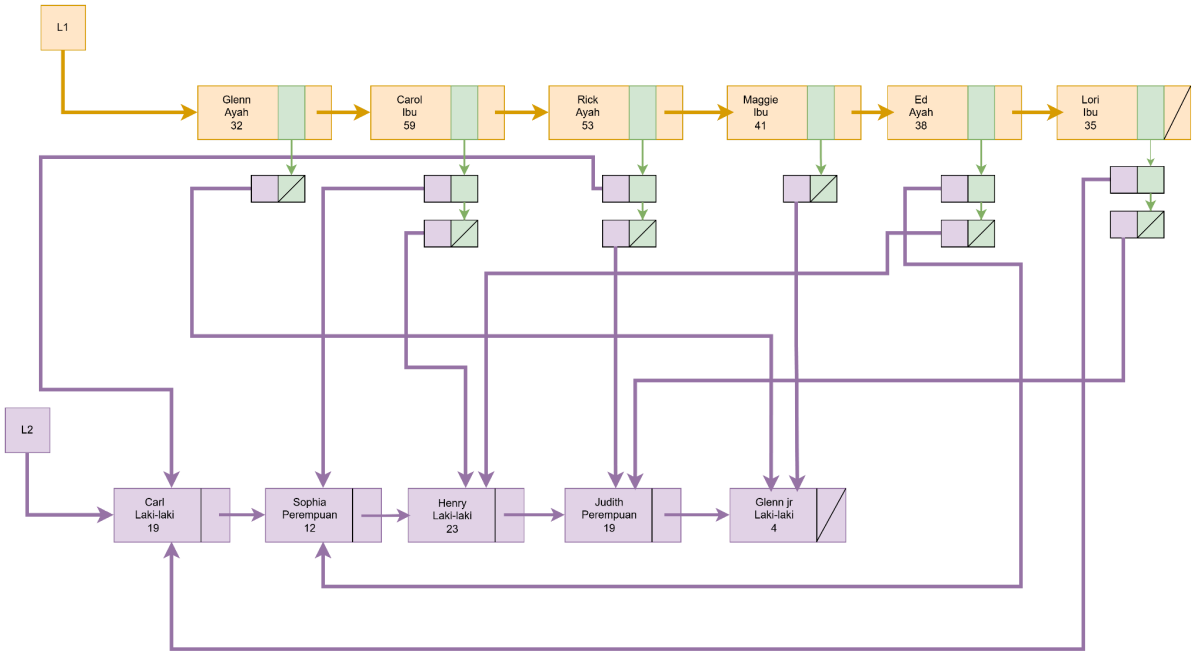
6. Menambahkan data child
7. Menghubungkan data parent ke data child
8. Menampilkan seluruh data parent beserta childnya
9. Mencari data child pada parent tertentu
10. Menghapus data child pada parent tertentu beserta relasinya
11. Menghitung jumlah data child dari parent tertentu
12. Main program

ADT (pseudocode)

```
Multi Linked List (M ke N)

Type infotype_orangTua : < nama, jenis: string          co: (Rick, Ayah, 53)
                           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

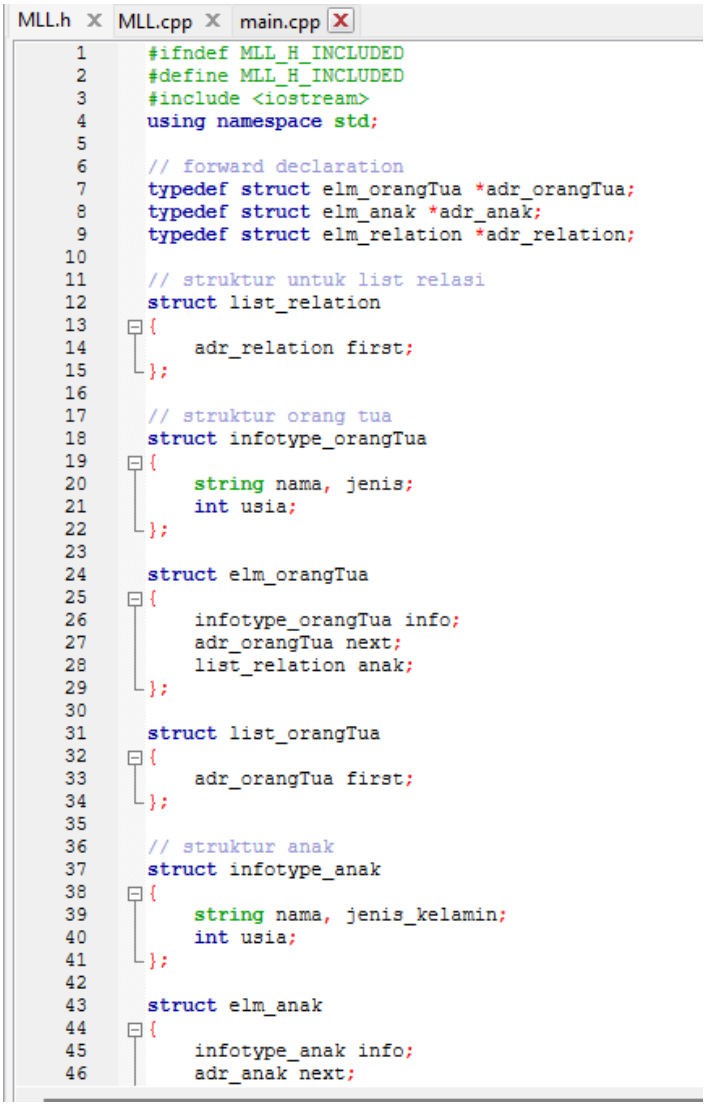
- Github Link: https://github.com/iqoll/TugasBesarSTD-K15
- File.exe (download untuk test program): https://drive.google.com/file/d/1qT5UPIdGyZoZyi2k8KV-tYA2A5QjYCrU/view?usp=sharing

Dokumentasi Pengerjaan

ADT dan Subprogram

MLL.h

1. Struktur dasar Multi-Linked List Parent-Child




```
#ifndef MLL_H_INCLUDED
#define MLL_H_INCLUDED
#include <iostream>
using namespace std;

// forward declaration
typedef struct elm_orangTua *adr_orangTua;
typedef struct elm_anak *adr_anak;
typedef struct elm_relation *adr_relation;

// struktur untuk list relasi
struct list_relation
{
    adr_relation first;
};

// struktur orang tua
struct infotype_orangTua
{
    string nama, jenis;
    int usia;
};

struct elm_orangTua
{
    infotype_orangTua info;
    adr_orangTua next;
    list_relation anak;
};

struct list_orangTua
{
    adr_orangTua first;
};

// struktur anak
struct infotype_anak
{
    string nama, jenis_kelamin;
    int usia;
};

struct elm_anak
{
    infotype_anak info;
    adr_anak next;
```

Gambar 3.1 - Struktur dasar MLL

2. Handler function

```
// handler function
void handleInsertParent(list_orangTua &L);
void handleInsertChild(list_anak &L);
void handleDeleteParent(list_orangTua &L);
void handleFindParent(list_orangTua L);
void handleFindChild(list_anak L);
void handleShowAllParent(list_orangTua L);
void handleShowAllChild(list_anak L);
void handleCreateRelation(list_orangTua &L_ORTU, list_anak L_ANAK);
void handleDeleteRelation(list_orangTua &L_ORTU);
void handleShowAllParentAndChild(list_orangTua L_ORTU);
void handleFindChildOnParent(list_orangTua L_ORTU);
void handleCountChildOfParent(list_orangTua L_ORTU);
```

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

```
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  }
```

Gambar 3.5 - Implementasi Handler function (MLL.cpp)

2. Main function

```
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  }
```

Gambar 3.6 - Implementasi Main function (MLL.cpp)

3. Helper function

```cpp
// inisiasi data keluarga (parent - children)
void InisiasiDataKeluarga(list_orangTua &L1, list_anak &L2)
{
    L1.first = NULL;
    L2.first = NULL;

    // ayah
    adr_orangTua P1 = new elm_orangTua;
    P1->info = {"Rick", "Ayah", 53};
    P1->next = NULL;
    P1->anak.first = NULL;

    // ibu
    adr_orangTua P2 = new elm_orangTua;
    P2->info = {"Lori", "Ibu", 35};
    P2->next = NULL;
    P2->anak.first = NULL;

    // hubungkan parent-parent
    L1.first = P1;
    P1->next = P2;

    // anak
    adr_anak C1 = new elm_anak;
    C1->info = {"Carl", "Laki-laki", 19};
    C1->next = NULL;

    L2.first = C1;

    // relasi 1
    adr_relation R1 = new elm_relation;
    R1->next_anak = C1;
    R1->next = NULL;

    P1->anak.first = R1;

    // relasi 2
    adr_relation R2 = new elm_relation;
    R2->next_anak = C1;
    R2->next = NULL;

    P2->anak.first = R2;
}

```

Gambar 3.7 - Implementasi Helper function (MLL.cpp)

Main Program

main.cpp

MLL.h | MLL.cpp | main.cpp

```cpp
#include <iostream>
#include "MLL.h"

using namespace std;

int main()
{
    list_orangTua L_ORTU;
    list_anak L_ANAK;

    // menginisasi list dengan seorang ayah, ibu, dan satu anak
    InisiasiDataKeluarga(L_ORTU, L_ANAK);

    int pilihanMenu;
    do
    {
        pilihanMenu = showMainMenu();

        switch (pilihanMenu)
        {
        case 1:
            // Panggil InsertFirstParent
            handleInsertParent(L_ORTU);
            break;
        case 2:
            // Panggil findParent
            handleFindParent(L_ORTU);
            break;
        case 3:
            // Panggil deleteParent
            handleDeleteParent(L_ORTU);
            break;
        case 4:
            // panggil showAllParent
            handleShowAllParent(L_ORTU);
            break;
        case 5:
            // Panggil insertLastChild
            handleInsertChild(L_ANAK);
            break;
        case 6:
            // Panggil findChild
            handleFindChild(L_ANAK);
            break;
        case 7:
            // Panggil showAllChild
            handleShowAllChild(L_ANAK);
            break;
        case 8:
```

Gambar 3.8 - Main program (main.cpp)

```
            break;
        case 8:
            // Panggil HubungkanParentToChild
            handleCreateRelation(L_ORTU, L_ANAK);
            break;
        case 9:
            // Panggil deleteRelation
            handleDeleteRelation(L_ORTU);
            break;
        case 10:
            // Panggil ShowAllParentAndChild
            handleShowAllParentAndChild(L_ORTU);
            break;
        case 11:
            // Panggil findChildOnParent
            handleFindChildOnParent(L_ORTU);
            break;
        case 12:
            // Panggil countChildOfParent
            handleCountChildOfParent(L_ORTU);
            break;
        case 0:
            cout << "Terima kasih telah menggunakan program." << endl;
            break;
        default:
            cout << "Pilihan tidak valid. Silakan coba lagi." << endl;
            break;
        }

        if (pilihanMenu != 0)
        {
            cout << "\nTekan ENTER untuk melanjutkan...";
            cin.ignore();
            cin.get();
        }

    } while (pilihanMenu != 0);
    return 0;
}
```

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