LAPORAN TUGAS BESAR STRUKTUR DATA

Kelas : IF 42 08
Kelompok 12 : Pelanggan vs Menu Makan Online
Anggota :

- **Muhammad Abdurrohman Al Fatih (1301180154)**
- **Muhammad Irsyad Al Ghifary (1301180211)**

Deskripsi studi kasus

Memodelkan pelanggan dengan menu makanan online dan hubungan antara keduanya. Seorang pelanggan bisa memilih lebih dari satu menu, dan satu menu bisa dipilih oleh lebih dari satu pelanggan. Untuk pemodelan menggunakan multi linked-list dengan bentuk relasi M-N.

Penjelasan elemen list

Elemen list kami terdiri dari **Info** yang memiliki tipe data sesuai dengan list yang bersesuaian, dan **Next** yang memiliki tipe data pointer to elemen list yang bersesuaian. Untuk penjelasan **Info** untuk masing-masing list akan dijelaskan berikut :

List Makanan

- **Info** memiliki tipe data struct **InfoFood** yang mempunyai member **NamaMakanan** yang bertipe data string, dan **JumlahPelanggan** yang bertipe data integer
- **NamaMakanan** merepresentasikan nama dari makanan tersebut
- **JumlahPelanggan** merepresentasikan jumlah pelanggan/pembeli dari makanan tersebut

List Pelanggan

- **Info** memiliki tipe data struct **InfoCustomer** yang mempunyai member **NamaPelanggan** yang bertipe data string, dan **JumlahMakanan** yang bertipe data integer
- **NamaPelanggan** merepresentasikan nama dari pelanggan tersebut
- **JumlahMakanan** merepresentasikan jumlah makanan yang dibeli oleh pelanggan tersebut

List Relasi

- **Info** memiliki tipe data struct **InfoRelation** yang mempunyai member **Food** yang bertipe data pointer to element list makanan, dan **Customer** yang bertipe data pointer to element list pelanggan
- **Food** merepresentasikan address sebuah makanan yang berada di list makanan
- **Customer** merepresentasikan address sebuah pelanggan yang berada di list pelanggan

- Hubungan antara **Food** dan **Customer** bisa direpresentasikan sebagai seorang pelanggan yang membeli sebuah makanan atau sebaliknya, sebuah makanan dibeli oleh seorang pelanggan

Penjelasan Abstract Data Type (ADT)

Pendefinisian fungsi-procedure primitiv kami lakukan didalam file header untuk menunjang fitur Template dari C++. Berikut adalah penjelasan ADT untuk masing-masing file header :

Header.h

- Berisi primitiv-primitiv untuk file-file header yang lain

List.h

- Kami menggunakan Single Linked-List dengan dua kepala, yaitu **First** dan **Last**

- **CreateElement** adalah fungsi yang mengembalikan address sebuah element list yang bersesuaian, fungsi ini menerima parameter **X** yang memiliki tipe data **Info** sebuah element list yang bersesuaian. Fungsi ini akan mengalokasikan sebuah elemen list yang bersesuaian dan mengisi **Info** elemen baru tersebut dengan **X**, dan mengisi **Next** dengan NULL, lalu mengembalikan address dari element tersebut.

- **CreateList** adalah procedure (bertipe void). Procedure ini menerima parameter **L** yang memiliki tipe data **List<Data>** dimana **Data** mengacu pada tipe data **Info** sebuah element list yang bersesuaian. Procedure ini akan mengisi **First** dan **Last** dari list tersebut dengan NULL.

- **AddToList** adalah procedure (bertipe void). Procedure ini menerima parameter **L** yang memiliki tipe data **List<Data>** dimana **Data** mengacu pada tipe data **Info** sebuah element list yang bersesuaian dan **A** yang memiliki tipe data **Address** atau pointer to element list yang bersesuaian. Procedure ini akan menambahkan element yang ditunjuk oleh pointer **A** kedalam list **L** dengan algoritma insert last.

- **DeleteFromList** adalah fungsi yang mengembalikan address sebuah element list yang bersesuaian, fungsi ini menerima parameter **L** yang memiliki tipe data **List<Data>** dimana **Data** mengacu pada tipe data **Info** sebuah element list yang bersesuaian dan **A** yang memiliki tipe data **Address** atau pointer to element list yang bersesuaian. Fungsi ini akan melakukan penghapusan element yang ditunjuk oleh pointer **A** dengan algoritma delete first, delete last, dan delete after sesuai dengan posisi element tersebut lalu mengembalikan address dari element tersebut.

Food.h

- **CreateFood** adalah prosedur yang bertugas membuat elementlist berisikan **NamaMakanan** dan **JumlahPelanggan** yang di isi 0 diawal (0 adalah Jumlah Pemesan makanan). Dalam prosedur ini, menggunakan 2 parameter yaitu **L** bertipe List<InfoFood> dan **X** bertipe string sebagai inputan nama makanan. Proses yang berjalan dalam prosedur ini dimulai dari deklarasi variabel **Food** yang bertipe InfoFood, lalu **Food.NamaMakanan** di isi nama makanan dalam variabel **X**, setelah itu **Food.JumlahPelanggan** secara default 0 yang nantinya akan bertambah seiring makanan yang dinputkan dipesan pelanggan. Setelah proses tersebut selesai, dilanjutkan dengan **CreateElement** variabel **Food** dan memasukannya kedalam list **L**.

- **PrintFood** adalah prosedur yang akan meng output kan element-element (nama makanan) yang ada di list makanan. Prosedur ini memiliki satu parameter yaitu **L** bertipe List<InfoFood>. Proses pertama yang dilakukan adalah mengecek apakah list kosong atau tidak. Jika tidak, dilanjut dengan pendeklarasian variabel **P** sebagai pointer to ElementList<InfoFood> dan di assign dengan First(**L**). Setelah itu masuk ke proses Looping sampai **P** sama dengan NULL (sampai element terakhir) dan akan mengoutputkan nama makanan di setiap iterasi. Jika list kosong, akan di outputkan “Makanan Kosong”

- **GetFood** adalah fungsi yang akan mengembalikan address dari sebuah makanan yang dicari jika ditemukan, mengembalikan NULL jika tidak. Fungsi ini memiliki 2 parameter yaitu **L** bertipe List<InfoFood> dan **X** bertipe string yang digunakan sebagai input nama makanan. Proses pertama adalah deklarasi variabel **P** sebagai pointer to ElementList<InfoFood> dan di assign dengan First(**L**). Loop hingga **P** sama dengan NULL atau makanan yang dicari ketemu.

- **ViewFood** adalah prosedur yang akan mengoutputkan Nama, Jumlah Pembeli, dan Pembeli untuk makanan tertentu. Prosedur ini memiliki 2 parameter yaitu **L** bertipe List<InfoRelation> dan Variabel **P** sebagai wadah address makanan. Proses pertama yaitu memastikan bahwa **P** tidak NULL, lalu mengoutputkan **NamaMakanan**, **JumlahPembeli** dan memanggil prosedur **ViewFoodBuyers** yang ada pada Customer.h untuk menampilkan pembeli dari makanan yang dicari. Jika **P** sama dengan NULL, akan di output “Makanan tidak ada!”.

- **ViewAllFood** adalah prosedur yang menampilkan semua data makanan, termasuk nama, jumlah pembeli dan pembeli makanan. Prosedur ini memiliki 2 parameter list, yaitu **LF** bertipe List<InfoFood> dan **LR** bertipe List<InfoRelation>. Proses yang dilakukan adalah deklarasi variabel **P** sebagai pointer to ElementList<InfoFood> dan di assign First(**LF**). Selama **P** belum NULL, akan dilakukan pemanggilan **ViewFood(LR, P)** hingga **P** sama dengan NULL.

- **ViewCustomersFoods** adalah prosedur yang menampilkan makanan yang dipesan oleh customer dengan nama tertentu. Prosedur ini memiliki 2 parameter yaitu **L** bertipe List<InfoRelation> dan variabel **X** sebagai wadah address customer yang akan di cari. Proses yang terjadi pada prosedur ini adalah deklarasi variabel **P** sebagai pointer to ElementList<InfoRelation> dan di assign dengan First(**L**). Selanjutnya melakukan looping sampai **P** sama dengan NULL. pada setiap iterasi, akan dilakukan pengecekan, apabila info dari element yang diperiksa sama dengan inputan yang di simpan dalam variabel **X** maka output, jika tidak lanjutkan looping.

- **SortFood** adalah prosedur sorting yang menggunakan metode selection sort dari besar ke kecil. Jumlah pelanggan menjadi variabel perbandingan. Proses yang dilakukan dalam dalam prosedur ini yang pertama adalah deklarasi variabel **Max** dan **P** sebagai pointer to ElementList<InfoFod>. Variabel **Max** digunakan untuk penyimpan First(**L**) dan variabel **P** digunakan untuk menyimpan satu elemen setelah First(**L**). Lalu deklarasi **LSort** yang digunakan untuk menyimpan element element hasil pengecekan. Setelah variabel yang dibutuhkan terdeklarasi. Selanjutnya adalah **CreateList(LSort)**, setelah itu masuk proses looping yang pertama, yaitu selama List **L** belum kosong, Lakukan inisialisasi **Max** dengan First(**L**) dan **P** dengan satu elemen setelah **Max**, setelah itu masuk ke proses Looping ke dua yaitu membandingkan info jumlah pelanggan dari **P** dan info jumlah pelanggan dari **Max**. jika info **P** lebih besar dari info **Max**, maka address **Max** akan di timpa dengan address **P**. jika tidak, lanjutkan iterasi. Selesai dari looping yang kedua, akan dipanggil prosedur **DeleteFromList(L, Max)** yang ada pada List.h, untuk menghapus address **Max** dari **L** dan menambahkanya ke **LSort**. Setelah selesai sorting, karena parameternya menggunakan input/output **L**, **LSort** yang berisikan data terurut di assign ke variabel **L**.

Customer.h

- **CreateCustomer** adalah prosedur yang bertugas membuat elementlist berisikan **NamaPelanggan** dan **JumlahMakanan** yang di isi 0 diawal (0 adalah Jumlah makanan yang dipesan). Dalam prosedur ini, menggunakan 2 parameter yaitu **L** bertipe List<InfoCustomer> dan **X** bertipe string sebagai inputan nama pelanggan. Proses yang berjalan dalam prosedur ini dimulai dari deklarasi variabel **Customer** yang bertipe InfoCustomer, lalu **Customer.NamaPelanggan** di isi nama pelanggan dalam variabel **X**, setelah itu **Customer.JumlahMakanan** secara default 0 yang nantinya akan bertambah seiring pelanggan yang dinputkan memesan makanan. Setelah proses tersebut selesai, dilanjutkan dengan **CreateElement** variabel **Customer** yang akan mengembalikan address dan akan disimpan dalam pointer **C** yang bertipe data ElementList<InfoCustomer>. Proses terakhir adalah memasukan **C** ke list **L**.

- **PrintCustomer** adalah prosedur yang akan meng output kan element-element (nama pelanggan) yang ada di list pelanggan. Prosedur ini memiliki satu parameter yaitu **L** bertipe List<InfoCustomer>. Proses pertama yang dilakukan adalah mengecek apakah list kosong atau tidak. Jika tidak, dilanjut dengan pendeklarasian variabel **P** sebagai pointer to ElementList<InfoCustomer> dan di assign dengan First(**L**). Setelah itu masuk ke proses Looping sampai **P** sama dengan NULL (sampai element terakhir) dan akan mengoutputkan nama pelanggan di setiap iterasi. Jika List Kosong, akan di outputkan “Pelanggan Kosong”

- **GetCustomer** adalah fungsi yang akan mengembalikan address dari sebuah pelanggan yang dicari jika ditemukan, mengembalikan NULL jika tidak. Fungsi ini memiliki 2 parameter yaitu **L** bertipe List<InfoCustomer> dan **X** bertipe String yang digunakan sebagai input nama pelanggan. Proses pertama adalah deklarasi variabel **P** sebagai pointer to ElementList<InfoCustomer> dan di assign dengan First(**L**). Loop hingga **P** sama dengan NULL atau Pelanggan yang dicari ketemu.

- **ViewCustomer** adalah prosedur yang akan mengoutputkan Nama, Jumlah Makanan, dan Makanan untuk pelanggan tertentu. Prosedur ini memiliki 2 parameter yaitu **L** bertipe List<InfoRelation> dan Variabel **P** sebagai wadah address pelanggan. Proses pertama yaitu memastikan bahwa **P** tidak NULL, lalu mengoutputkan **NamaPelanggan**, **JumlahMakanan** dan memanggil prosedur **ViewCustomersFoods** yang ada pada Food.h untuk menampilkan makanan dari pelanggan yang dicari. Jika **P** sama dengan NULL, akan di output “Pelanggan tidak ada!”.

- **ViewAllCustomer** adalah prosedur yang menampilkan semua data pelanggan, termasuk nama, jumlah makanan dan makanan yang dibeli. Prosedur ini memiliki 2 parameter list, yaitu **LC** bertipe List<InfoCustomer> dan **LR** bertipe List<InfoRelation>. Proses yang dilakukan adalah deklarasi variabel **P** sebagai pointer to ElementList<InfoFood> dan di assign First(**LF**). Selama **P** belum NULL, akan dilakukan pemanggilan **ViewFood(LR, P)** hingga **P** sama dengan NULL.

- **ViewFoodsBuyers** adalah prosedur yang menampilkan nama-nama pelanggan yang memesan suatu makanan dengan nama tertentu. Prosedur ini memiliki 2 parameter yaitu **L** bertipe List<InfoRelation> dan variabel **X** sebagai wadah address nama makanan yang akan di cari. Proses yang terjadi pada prosedur ini adalah deklarasi variabel **P** sebagai pointer to ElementList<InfoRelation> dan di

assign dengan first(**L**). Selanjutnya melakukan looping sampai **P** sama dengan NULL. pada setiap iterasi, akan dilakukan pengecekan, apabila info dari element yang diperiksa sama dengan inputan yang di simpan dalam variabel **X** maka output, jika tidak lanjutkan looping.

- **SortCustomer** adalah prosedur sorting yang menggunakan metode selection sort dari besar ke kecil. Jumlah makanan yang dipesan oleh customer menjadi variabel perbandingan. Proses yang dilakukan dalam dalam prosedur ini yang pertama adalah deklarasi variabel **Max** dan **P** sebagai pointer to ElementList<InfoCustomer>. Variabel **Max** digunakan untuk penyimpan First(**L**) dan variabel **P** digunakan untuk menyimpan satu elemen setelah First(**L**). Lalu deklarasi **LSort** yang digunakan untuk menyimpan element element hasil pengecekan. Setelah variabel yang dibutuhkan terdeklarasi. Selanjutnya adalah **CreateList(LSort)**, setelah itu masuk proses looping yang pertama, yaitu selama List **L** belum kosong, Lakukan inisialisasi **Max** dengan First(**L**) dan **P** dengan satu elemen setelah **Max**, setelah itu masuk ke proses Looping ke dua yaitu membandingkan info jumlah makanan dari **P** dan info jumlah makanan dari **Max**. jika info **P** lebih besar dari info **Max**, maka address **Max** akan di timpa dengan address **P**. jika tidak, lanjutkan iterasi. Selesai dari looping yang kedua, akan dipanggil prosedur **DeleteFromList(L, Max)** yang ada pada List.h, untuk menghapus address **Max** dari **L** dan menambahkanya ke **LSort**. Setelah selesai sorting, karena parameternya menggunakan input/output **L**, **LSort** yang berisikan data terurut di assign ke variabel **L**.

Relation.h

- **CreateRelation** adalah procedure (bertipe void). Procedure ini menerima parameter **L** yang memiliki tipe data List<InfoRelation> , **F** yang memiliki tipe data pointer to ElementList<InfoFood> yang merepresentasikan address dari suatu makanan, dan **C** yang memiliki tipe data pointer to ElementList<InfoCustomer> yang merepresentasikan address dari suatu pelanggan. Proses yang berjalan dalam prosedur ini dimulai dari deklarasi variabel **Relation** yang bertipe InfoRelation, lalu **Relation.Food** di isi dengan address **F**, setelah itu **Relation.Customer** di isi dengan address **C**. Setelah proses tersebut selesai, dilanjutkan dengan menambahkan jumlah pelanggan dari element yang ditunjuk oleh address **F** dan juga menambahkan jumlah makanan dari element yang ditunjuk oleh address **C**. Selanjutnya akan dilakukan **CreateElement** variabel **Relation** yang akan mengembalikan address dan akan disimpan dalam pointer **R** yang bertipe data ElementList<InfoRelation>. Proses terakhir adalah memasukan **R** ke list **L**.

- **GetRelation** adalah fungsi yang akan mengembalikan address dari suatu relasi yang akan dicari. Fungsi ini memiliki 3 parameter yaitu **L** bertipe List<InfoRelasi>, variabel **F** sebagai wadah address food yang dicari, dan variabel **C** sebagai

wadah address customer yang dicari. Fungsi ini bisa berlaku dua arah, maksudnya, hanya perlu memasukan salah satu dari address yang dibutuhkan, bisa address food saja, atau address customer saja. Proses yang pertama adalah deklarasi variabel **P** sebagai pointer to ElementList<InfoRelation>. Selanjutnya adalah melakukan Looping sampai **P** sama dengan NULL atau **F** sama denggan **Info(P).Food** atau **C** sama dengan **Info(P).Customer**. Jika tidak ditemukan relasi yang dicari, akan mengembalikan NULL.

UI.h

- **Continue** adalah procedure (bertipe void). Procedure ini tidak menerima parameter. Procedure ini akan memanggil fungsi bawaan **system** dengan parameter **pause** yang bertujuan untuk mempause tampilan program.

- **Print** adalah procedure (bertipe void). Procedure ini menerima parameter **X** yang bertipe string dan mempunyai nilai default string kosong (“”), dan **End** yang memiliki tipe data string yang mempunyai nilai default **\n** atau new line. Procedure ini akan meng outputkan string sesuai urutan : **X** -> **End**.

- **GetInput** adalah fungsi yang mengembalikan string berdasar apa yang diinputkan oleh user nantinya. Fungsi ini menerima parameter **X** yang bertipe string dan mempunyai nilai default string kosong (“”). Fungsi ini akan memanggil **Print(X, “”)** untuk mengoutputkan **X** ke program lalu meminta inputan dari user dengan menggunakan fungsi **getline** yang akan membaca inputan user dalam 1 baris lalu menyimpannya di variabel **Temp** yang memiliki tipe data string dan mengembalikannya.

- **ClearScreen** adalah procedure (bertipe void). Procedure ini tidak menerima parameter. Procedure ini akan memanggil fungsi bawaan **system** dengan parameter **cls** yang bertujuan untuk menghapus/mengkosongkan tampilan program.

- **MenuUser** adalah prosedur menu yang akan ditampilkan jika user menginput selain “**ADMIN**” yang otomatis akan menjadi **NamaCustomer**. Prosedur ini memiliki 4 parameter yaitu **LC** sebagai List<InfoCustomer>, **LF** sebagai List<InfoFood>, **LR** sebagai List<InfoRelation> dan **Nama** guna menampung inputan user di Main.cpp. Proses pertama yaitu memeriksa apakah list food kosong, jika kosong, akan output “Warung belum buka!”, jika tidak kosong, dilanjutkan pengecekan kedua, yaitu apakah nama yang dinput sudah ada atau tidak didalam list customer, karena jika ada, akan ada dua nama customer yang sama dan memungkinkan tertukarnya pesanan wkwkw. Maka dari itu akan output “Maaf, sudah ada nama pelanggan yang sama”, jika tidak ada yang sama akan berlanjut keproses selanjutnya yaitu ditampilkan sapaan “Selamat datang “ dan inputan user (Nama user) “ di AKMJ Caffe”. Setelah itu memanggil fungsi

CreateCustomer yang di assign ke variabel **C** untuk dijiadikan sebuah ElementList<InfoCustomer>. Lalu ditampilkan menu makanan yang sudah terurut dari yang paling favorit. Selanjutnya user diminta untuk memilih makanannya dan di assign ke variabel **Makanan**. User diperbolehkan memesan lebih dari satu menu, maka dari itu ada looping yang terus meminta user untuk memesan makanan, selama inputan user bukan “x”. Setelah selesai memesan akan ada output “Terimakasih OwO”.

- **MenuAdmin** prosedur yang akan dipanggil jika user menginputkan “**ADMIN**”. Prosedur ini berisikan fitur fitur yang hanya bisa dilakukan ADMIN dan prosedur ini memiliki 3 parameter, yaitu **LC** sebagai List<InfoCustomer>, **LF** sebagai List<InfoFood>, **LR** sebagai List<InfoRelation>. Dalam menu ADMIN, ada 3 fitur utama yaitu Input Menu, Lihat Pesanan dan Delete Menu dan “0” untuk mengakhiri menu ini. Selanjutnya diminta inputan pilihan user yang akan disimpan di variabel **Decision**.
    - Jika **Decision** “1”, maka Input Menu
    - Diminta nama makanan yang akan diinput ke Menu
    - Masuk proses looping dengan tahap pertama memeriksa apakah nama makanan sudah ada dalam ListFood atau tidak. Jika sudah, akan dioutputkan “Maaf makanan sudah ada”. Jika belum maka dibuat Element<InfoFood> dengan info nama makanan yang tadi di input, lalu kembali ADMIN diminta menginputkan kembali jika ingin menambah menu. Jika tidak, cukup input “x” untuk mengakhiri input menu.

    - Jika **Decision** “2”, maka Lihat pesanan.
    - Dalam fitur ini diberikan pilihan lagi, yaitu melihat pesanan dari makanan atau dari pelanggan, dan “0” untuk kembali ke menu sebelumnya. Lalu Admin diminta input pilihan yang akan disimpan dalam **Decision2**.
        - Jika **Decision2** “1”, pertama akan ditampilkan pilihan lagi, ALL, untuk menampilkan semua pesanan, FAV untuk menampilkan makanan yang paling banyak di pesan, dan input “Nama Makanan” untuk menampilkan pesanan tertentu berdasarkan nama makanan.
        - Masuk ke proses looping yang akan meminta ADMIN memasukan input sesuai perintah sebelumnya. Jika “ALL” akan ditampilkan semua pesanan. Jika “FAV” akan di tampilkan pesanan dengan jumlah terbanyak yang memesan. Jika Nama Makanan tertentu akan ditampilkan berapa banyak yang memesan makana tersebut. Loop akan berhenti jika inputan “x” atau “ALL”
        - Jika **Decision2** “2”, pertama akan ditampilkan pilihan lagi, ALL, untuk menampilkan semua pelanggan, FAV untuk menampilkan pelanggan yang paling banyak memesan, dan input “Nama

Customer” untuk menampilkan pesanan tertentu berdasarkan nama pelanggan.
    - Masuk ke proses looping yang akan meminta ADMIN memasukan input sesuai perintah sebelumnya. Jika “ALL” akan ditampilkan semua pelanggan. Jika “FAV” akan di tampilkan pelanggan dengan jumlah pesanan terbanyak. Jika Nama Pelanggan tertentu akan ditampilkan berapa banyak yang dipesan Pelanggan tersebut. Loop akan berhenti jika inputan “x” atau “ALL”
- Jika **Decision** “3”, maka Delete Menu
- Pertama, adalah menampilkan list menu yang ada
- Setelah itu ADMIN diminta memasukan Nama makanan yang akan di hapus.
- Lalu akan masuk proses looping dimana loop akan berhenti ketika nama makanan yang di input adalah “x”.
- Setelah input nama makanan, nama tersebut akan di cari dalam List Food. apabila ditemukan address akan dimasukkan ke variabel **F** lalu dilanjutkan ke proses selanjutnya, jika tidak, akan di output kan “maaf “ + nama makanan + “ tidak ada”. Dan akan diminta input nama makanan kembali.
- Proses selanjutnya setelah input adalah mencari relasi di list relasi dengan address food yang ingin dihapus.
- Semua relasi yang memiliki menu sama denggan address food yang ada di variabel **F** akan di hapus juga dengan jumlah pemesan akan berkurang.
- Setelah semua relasi dengan address **F** dihapus, barulah dilakukan penghapusan **F** dari list food.

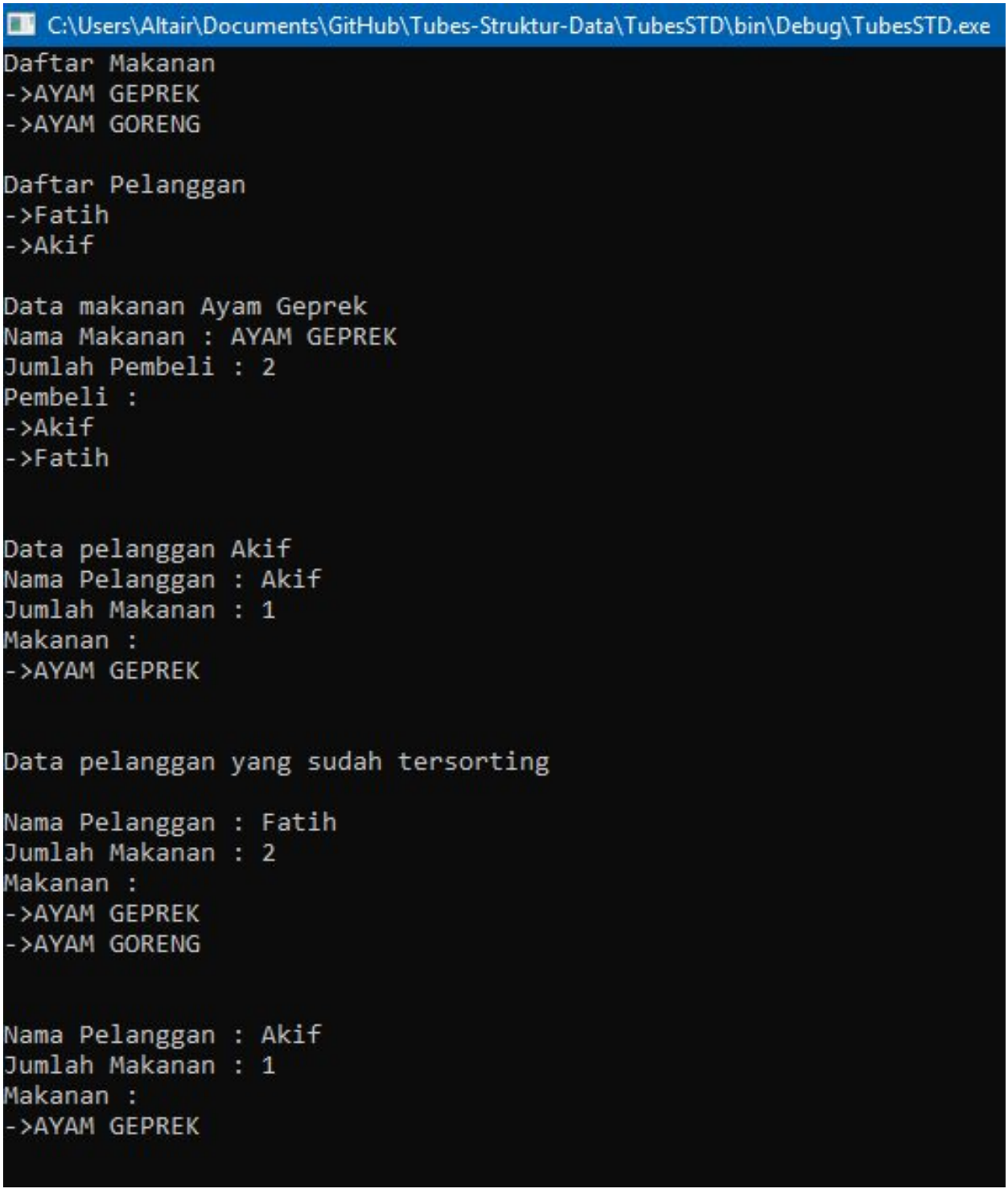
Program untuk mengetest aplikasi

```
void Testing(List<InfoFood> LF, List<InfoCustomer> LC, List<InfoRelation> LR) {
    /* Membuat makanan dengan nama AYAM GEPREK */
    if(GetFood(LF, "AYAM GEPREK") == NULL) {
        CreateFood(LF, "AYAM GEPREK");
    }

    /* Membuat makanan dengan nama AYAM GORENG */
    if(GetFood(LF, "AYAM GORENG") == NULL) {
        CreateFood(LF, "AYAM GORENG");
    }

    Print("Daftar Makanan");
    PrintFood(LF);

    /* Membuat pelanggan dengan nama Fatih */
    if(GetCustomer(LC, "Fatih") == NULL) {
        CreateCustomer(LC, "Fatih");
    }

    /* Membuat pelanggan dengan nama Akif */
    if(GetCustomer(LC, "Akif") == NULL) {
        CreateCustomer(LC, "Akif");
    }

    Print("\nDaftar Pelanggan");
    PrintCustomer(LC);

    auto Fatih = GetCustomer(LC, "Fatih");
    auto Akif = GetCustomer(LC, "Akif");
    auto Geprek = GetFood(LF, "AYAM GEPREK");
    auto Goreng = GetFood(LF, "AYAM GORENG");

    /* Menambahkan hubungan antara AYAM GEPREK dengan Akif */
    CreateRelation(LR, Geprek, Akif);
    /* Menambahkan hubungan antara AYAM GEPREK dengan Fatih */
    CreateRelation(LR, Geprek, Fatih);
    /* Menambahkan hubungan antara AYAM GORENG dengan Fatih */
    CreateRelation(LR, Goreng, Fatih);

    Print("\nData makanan Ayam Geprek");
    ViewFood(LR, Geprek);

    Print("\nData pelanggan Akif");
    ViewCustomer(LR, Akif);

    SortCustomer(LC);
    SortFood(LF);

    Print("\nData pelanggan yang sudah tersorting");
    ViewAllCustomer(LC, LR);
```

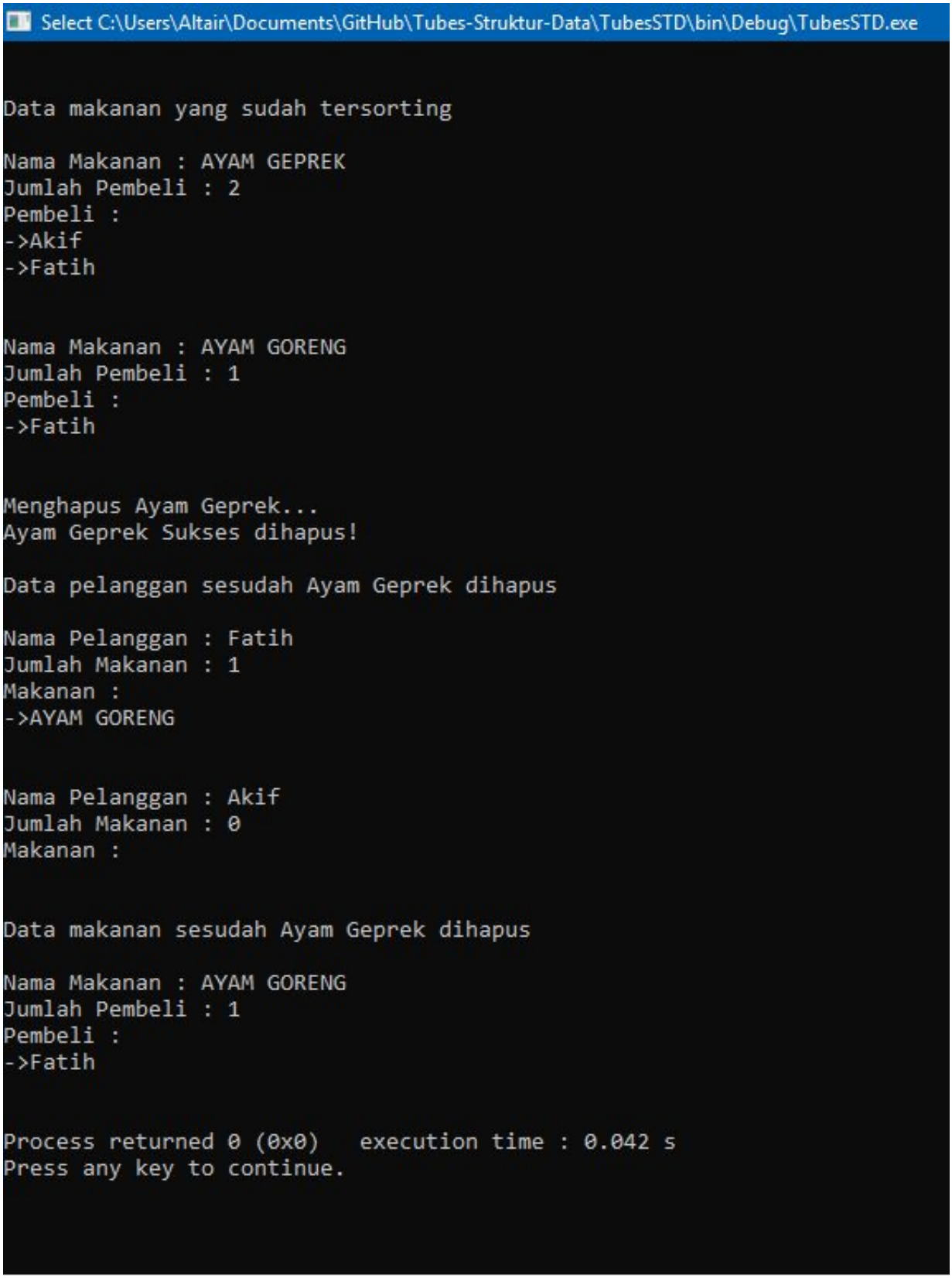
Program untuk mengetest aplikasi

```
    Print("\nData makanan yang sudah tersorting");
    ViewAllFood(LF, LR);

    Print("\nMenghapus AYAM GEPREK...");
    auto R = GetRelation(LR, Geprek, NULL);
    while(R != NULL) {
        Info(Info(R).Customer).JumlahMakanan--;
        DeleteFromList(LR, R);
        R = GetRelation(LR, Geprek, NULL);
    }
    DeleteFromList(LF, Geprek);
    Print("AYAM GEPREK Sukses dihapus!");

    Print("\nData pelanggan sesudah AYAM GEPREK dihapus");
    ViewAllCustomer(LC, LR);

    Print("\nData makanan sesudah AYAM GEPREK dihapus");
    ViewAllFood(LF, LR);
}
```

Hasil test


C:\Users\Altair\Documents\GitHub\Tubes-Struktur-Data\TubesSTD\bin\Debug\TubesSTD.exe
Daftar Makanan
->AYAM GEPREK
->AYAM GORENG
Daftar Pelanggan
->Fatih
->Akif
Data makanan Ayam Geprek
Nama Makanan : AYAM GEPREK
Jumlah Pembeli : 2
Pembeli :
->Akif
->Fatih
Data pelanggan Akif
Nama Pelanggan : Akif
Jumlah Makanan : 1
Makanan :
->AYAM GEPREK
Data pelanggan yang sudah tersorting
Nama Pelanggan : Fatih
Jumlah Makanan : 2
Makanan :
->AYAM GEPREK
->AYAM GORENG
Nama Pelanggan : Akif
Jumlah Makanan : 1
Makanan :
->AYAM GEPREK

HASIL TEST


Select C:\Users\Altair\Documents\GitHub\Tubes-Struktur-Data\TubesSTD\bin\Debug\TubesSTD.exe
Data makanan yang sudah tersorting
Nama Makanan : AYAM GEPREK
Jumlah Pembeli : 2
Pembeli :
->Akif
->Fatih
Nama Makanan : AYAM GORENG
Jumlah Pembeli : 1
Pembeli :
->Fatih
Menghapus Ayam Geprek...
Ayam Geprek Sukses dihapus!
Data pelanggan sesudah Ayam Geprek dihapus
Nama Pelanggan : Fatih
Jumlah Makanan : 1
Makanan :
->AYAM GORENG
Nama Pelanggan : Akif
Jumlah Makanan : 0
Makanan :
Data makanan sesudah Ayam Geprek dihapus
Nama Makanan : AYAM GORENG
Jumlah Pembeli : 1
Pembeli :
->Fatih
Process returned 0 (0x0) execution time : 0.042 s
Press any key to continue.

‘