

Universidade do Minho
Departamento de Informática
Licenciatura em Engenharia Informática

Laboratórios de Informática III

Fase I – Grupo 23

22 novembro 2022

Ana Filipa Cruz Pinto - a96862

Rui Pedro Castro Alves - a100699

Rui Pedro Fernandes Ribeiro - a100816

Índice

Introdução	2
Organização dos Dados	2
Catálogos.....	3
Queries	4
Query 1.....	4
Query 4.....	5
Conclusão.....	5

Introdução

No âmbito da unidade curricular Laboratórios de Informática III foi-nos proposto o desenvolvimento de uma aplicação de prestação de serviços, mais especificamente, de deslocações automóveis, em linguagem C.

Foi-nos informado, pelo enunciado do projeto, que a aplicação correria a partir de três ficheiros .csv que continham a informação necessária sobre os utilizadores, os condutores e as viagens. Partindo desses ficheiros, teríamos que fazer parsing deles e armazená-los em catálogos correspondentes. Além disso, deveria ser, também nesta primeira fase, feito um modo batch que enviaria comandos para a interpretação de comandos, para esta segunda os ler, interpretar e executar através das respetivas queries.

Organização dos Dados

Com vista a uma melhor modularidade da implementação, decidimos dividir o projeto nos seguintes módulos:

- **parse.c** - Este módulo contém o código responsável pela leitura dos ficheiros .csv e armazenamento do user, driver ou ride lida no respetivo catálogo.
- **parse.h** - Header do módulo parse.c.
- **drivers.c** - Este módulo contém o código responsável pela formatação dos condutores. Ou seja, cria a estrutura de um driver.
- **drivers.h** - Header do módulo drivers.c.
- **rides.c** - Este módulo contém o código responsável pela formatação das viagens. Ou seja, cria a estrutura de uma ride.
- **rides.h** - Header do módulo rides.c.
- **user.c** - Este módulo contém o código responsável pela formatação dos utilizadores. Ou seja, cria a estrutura de um user.
- **user.h** - Header do módulo user.c.
- **getters.c** - Este módulo contém o código de auxílio à resolução das queries.
- **getters.h** - Header do módulo getters.c.
- **main.c** - Este módulo contém o código responsável pela interpretação dos comandos e pela indicação do tempo de execução do programa.
- **queries.c** - Este módulo contém o código responsável pela implementação das queries.
- **queries.h** - Header do módulo queries.c.
- **data.c** - Este módulo contém o código responsável pela formatação das datas na forma dd/mm/aaaa.
- **data.h** - Header do módulo data.c.

Catálogos

Neste projeto decidimos implementar os catálogos no módulo parse.c para agilizar a leitura e o armazenamento dos ficheiros .csv numa única função.

- **Catálogo de Utilizadores** - Este catálogo está implementado na função CatUser parseUser. Aqui a linha é lida e, posteriormente, é guardada, com a estrutura definida no módulo user, na hash table do catalogoUser. Para isso, usa com key o username do utilizador.

```
g_hash_table_insert(user, get_username_user(a), a);
```

- **Catálogo de Condutores** - Este catálogo está implementado na função CatDriver parseDrivers. Aqui a linha é lida e, posteriormente, é guardada, com a estrutura definida no módulo drivers, na hash table do catalogoDriver. Para isso, usa como key o id do condutor.

```
g_hash_table_insert(driver, get_id_driver(d), d);
```

- **Catálogo das Viagens** – Este catálogo está implementado na função CatRides parseRides. Aqui a linha é lida e, posteriormente, é guardada, com a estrutura definida no módulo rides, nas hash tables do catalogoRides. No entanto, a implementação desta difere das duas anteriores no ponto em que utilizamos três hash tables. A primeira onde guarda as informações sobre a viagem, usando o id da viagem como key. A segunda onde guarda a informação do utilizador, usando o user como key. E a terceira onde guarda as informações sobre o condutor, usando o driver como key.

```
g_hash_table_insert(rides, get_id_Rides(r), r);  
// key é o id do rides  
g_hash_table_insert(ridesuser, get_user_Rides(r), r);  
// key é o user do rides  
g_hash_table_insert(ridesdriver, get_driver_Rides(r), r);  
// key é o driver do rides
```

Queries

Query 1

- Esta query tem a sua resolução dividida em 2, a sua especificação para os utilizadores e a sua especificação para os condutores.
- Para resolver a primeira, começamos por, no módulo getters, implementar as funções:
 - `get_name_list_user` e `get_gender_list_user` que procuram no catálogo de utilizadores o nome e o género, respetivamente, do utilizador;
 - `get_idade_list_User` calcula a idade do utilizador, acedendo ao catálogo dos utilizadores e invocando a função `build_data` do módulo `data`;
 - `get_avaliacao_media_user` calcula a avaliação média do utilizador, acedendo ao catálogo das viagens e invocando as funções `get_user_Rides` e `get_score_user_Rides` do módulo `rides`;
 - `get_numero_viagens_user` calcula o número de viagens do utilizador, acedendo ao catálogo das viagens e invocando a função `get_user_Rides` do módulo `rides`;
 - `get_total_gasto` calcula o valor total gasto pelo utilizador, acedendo aos três catálogos e invocando as funções `get_driver_Rides`, `get_user_Rides`, `get_distance_Rides` e `get_tip_Rides` do módulo `rides` e a função `get_car_class_driver` do módulo `drivers`.

Já no módulo `queries`, se o estado da conta do utilizador estiver “active”, a função procura os parâmetros pedidos no catálogo dos utilizadores através das funções definidas anteriormente no módulo `getters` e faz print dos mesmos.

- Para resolver a segunda especificação, começamos por, no módulo `getters`, implementar as funções:
 - `get_name_list_driver` e `get_gender_list_driver` que procuram no catálogo de condutores o nome e o género, respetivamente, do condutor;
 - `get_idade_list_driver` que é implementada da mesma forma que a função `get_idade_list_User` diferindo no facto da procura ser feita no catálogo dos condutores;
 - `get_avaliacao_media` calcula a avaliação média do condutor, acedendo ao catálogo das viagens e invocando as funções `get_driver_Rides` e `get_score_driver_Rides` do módulo `rides`;
 - `get_numero_viagens` calcula o número de viagens do condutor, acedendo ao catálogo das viagens e invocando a função `get_driver_Rides` do módulo `rides`;
 - `get_total_euferido` calcula o valor total ganho, acedendo aos três catálogos e invocando as funções `get_driver_Rides`, `get_distance_Rides` e `get_tip_Rides` do módulo `rides` e a função `get_car_class_driver` do módulo `drivers`.

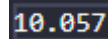
A implementação no módulo `queries` é, idêntica à da primeira especificação, com a diferença do catálogo consultado ser o dos condutores.

Petra Pacheco;F;51;2.667;6;78.230

Figura 1: Resultados Obtidos com `query1()`

Query 4

- Esta query, em semelhança com a query 1, começa por definir, no módulo getters, a função `get_preco_medio_city` que, acedendo aos três catálogos e invocando as funções `get_driver_Rides`, `get_city_Rides` e `get_distance_Rides` do módulo rides e a função `get_car_class_driver` do módulo drivers, calcula o preço médio das viagens, sem gorjeta, numa determinada cidade.
- Posteriormente, no módulo queries, a função `query4`, acedendo aos catálogos dos condutores e das viagens, chama a função `get_preco_medio_city`, anteriormente definida, e faz print do seu resultado.



10.057

Figura 2: Resultados Obtidos com query4()

Conclusão

Dado o problema proposto, no nosso ponto de vista, a resolução por nós implementada cumpre os principais objetivos desta primeira fase. Isto dado que consolidamos as nossas competências na linguagem de programação C, fizemos uma divisão eficiente e clara dos módulos do nosso projeto, respeitando, assim, a modularidade estudada na unidade curricular, e todos os módulos implementam o seu código de forma encapsulada e propicia a alterações. Para o armazenamento da informação recolhida pelo parsing dos dados dos ficheiros .csv, foi por nós usado hash tables por considerarmos o método mais eficiente e rápido na procura dos dados para a resolução das queries propostas.

Todavia, não conseguimos, por falta de tempo, fazer a terceira query proposta. Este é um dos aspetos que queremos melhorar na próxima fase do projeto. Além disso, pretendemos, também, melhorar a qualidade do código implementado.