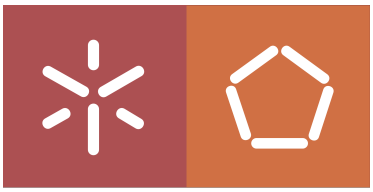

Universidade do Minho
Escola de Engenharia

Laboratórios de Informática III

Trabalho Prático

Fase 1

LEI - 2º Ano - 1º Semestre

Realizado por:
A98695 Lucas Oliveira
A89292 Mike Pinto
A96208 Rafael Gomes

Braga,
22 de novembro de 2022

Conteúdo

1 Introdução

Neste Trabalho Prático da Unidade Curricular de Laboratórios de Informática III, foi-nos proposto desenvolver uma aplicação realizada na *Linguagem C* onde recebemos três ficheiros de entrada, *users.csv, drivers.csv* e *rides.csv*, com informações relativas a utilizadores, condutores e viagens.
Nesta primeira fase fomos desafiados a implementar módulos para a realização de:

- *Parsing* de dados;
- Modo de operação *batch*;
- Implementação de *queries*;
- Catálogos de dados para os três ficheiros de entrada.

2 Abordagem

Para a realização desta primeira fase dividimos os módulos da seguinte forma, para nos podermos guiar e implementar as funções mais facilmente:

- *Users*: Modulo referente a funções e estruturas relacionadas com utilizadores;
- *Drivers*: Modulo referente a funções e estruturas relacionadas com condutores;
- *Rides*: Modulo referente a funções e estruturas relacionadas com as viagens;
- *Statistics*: Modulo referente a funções e estruturas para análise estatística;
- *Parser*: Modulo referente a funções de *parse* e leitura de ficheiros;
- Verifica: Modulo referente a funções de verificação;
- *Hash*: Modulo referente a funções de *Hash*;
- *Queries*: Modulo referente a funções e estruturas usadas em *queries*;

Além disso, dividimos a nossa abordagem em duas fases de desenvolvimento distintas, na primeira fase implementaram-se os módulos principais do programa relativos ao *Parse* das informações e comandos recebidos, à criação dos módulos dos *Users*, *Drivers*, *Rides* e *Hash*, na segunda fase focámo-nos na criação das *queries* 1, 2, 3 e 4 tentando em qualquer uma das fases ter em conta o conceito de modularidade e encapsulamento abordado nas aulas.

3 Estruturas Utilizadas

Durante a execução do trabalho práticos tivemos de adotar diferentes estruturas de dados para enfrentar os diferentes problemas. Foram utilizadas estruturas de dados para catalogar e armazenar as diferentes informações dos ficheiros *.csv*, dos cálculos estatísticos e para a execução das diferentes *queries*, tentando sempre ter em conta a forma de acesso às diferentes informações, utilização de memória e rapidez da sua execução.

3.1 Hash Tables

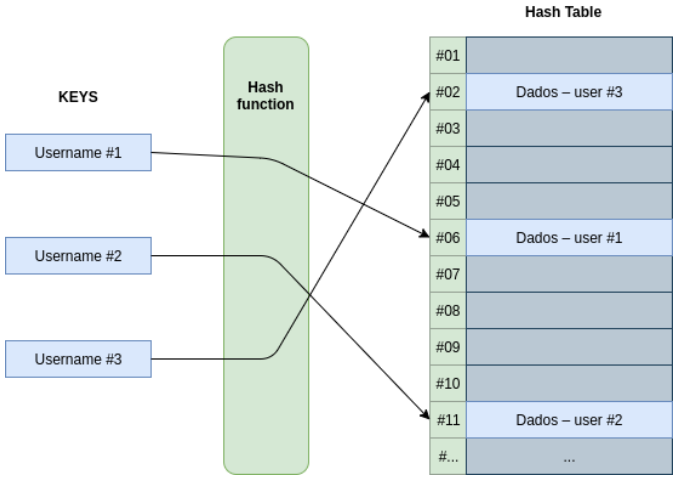

Figura 1: Diagrama a exemplificar como funciona a *Hash Table* dos *Users*.

Para catalogar as diferentes informações dos *Users*, *Drivers* e para armazenar os cálculos estatísticos foram utilizados **Hash Tables**(Figura 1.), pois apesar de utilizarem mais memória possuem uma rápida capacidade de pesquisa das informações que necessitávamos. Para tal utilizamos o algoritmo *djb2 hash* [1] para gerar um hash para as diferentes *KEY's*, que no caso dos *Users* e dos seus cálculos estatísticos são os *usernames* e nos *Drivers* e dos seus cálculos estatísticos são os seus *id's*.

3.2 Array

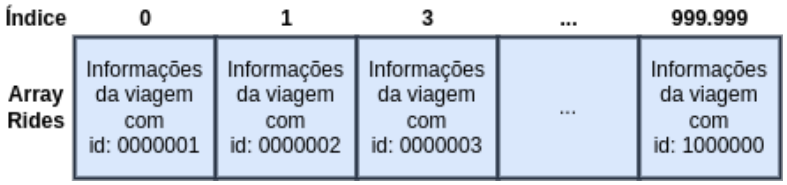

Figura 2: Diagrama a exemplificar o *Array* das *Rides*.

Ao contrário das informações dos *Users* e *Drivers*, para as *Rides* utilizamos um **Array**, pois como iríamos aceder as suas informações de forma sequencial e os *id's* das viagens já estavam ordenadas por ordem crescente, um *Array* seria a melhor opção, pois o *id* de uma viagem iria corresponder ao seu índice no *Array* e não necessitaríamos de alocar tanta memória como em *Hash Tables*.

3.3 Listas Ligadas

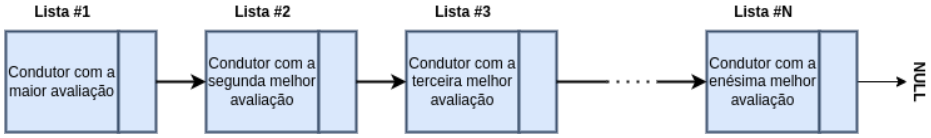

Figura 3: Diagrama a exemplificar a estrutura de dados da *querie* 2 com os N condutores com maior avaliação.

Nas *queries* 2 e 3 surgiu a necessidade de criar uma estrutura que conseguíssemos ordenar de forma fácil e rápida, para isso usamos **Listas Ligadas** que apesar de no pior caso terem *Time Complexity* igual a O(N), só iríamos guardar na estrutura os N utilizadores/condutores pedidos no comando de execução da *querie*.

4 Desenvolvimento

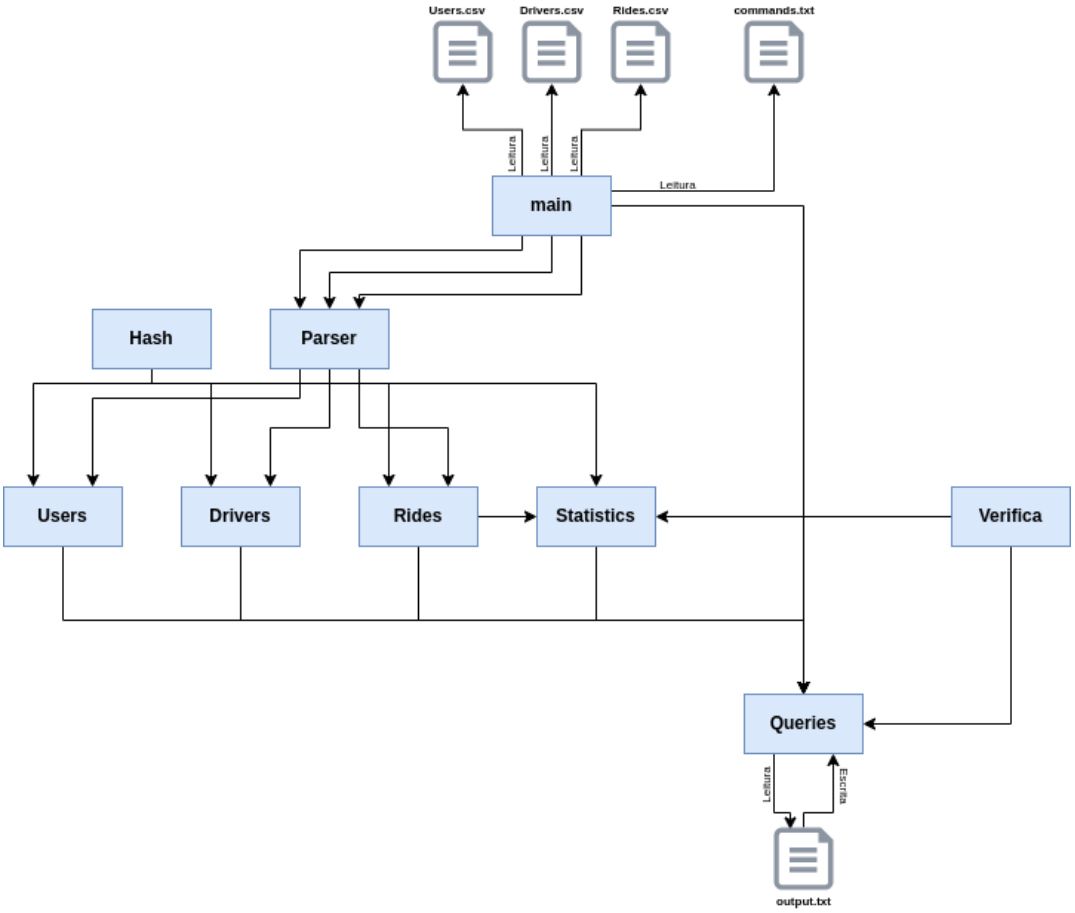

Figura 4: Diagrama a exemplificar o uso de cada módulo.

4.1 Primeira fase desenvolvimento - Módulos Principais

Na primeira fase de desenvolvimento desenvolveu-se a estrutura principal do programa, a leitura e *parse* dos ficheiros *.csv*, o desenvolvimento do catálogo de utilizadores, condutores e viagens.

4.1.1 Leitura e *parsing* dos ficheiros recebidos como argumento

O nosso primeiro objetivo era a implementação do *modo batch* onde o utilizador passa ao programa dois argumentos, o primeiro sendo o caminho para a pasta onde estão os ficheiros *.csv* e o segundo argumento um ficheiro de texto em que cada linha específica um comando para uma determinada *querie*.
Assim, com o auxílio das funções *fopen* e *fgets* foram lidos linha a linha os ficheiros *users.csv, drivers.csv e rides.csv*. Para cada linha as informações foram separadas por ";"para serem armazenadas.

Para o ficheiro de texto que contem os comandos para as *queries* foi também usado o auxílio das funções *fopen* e *fgets* lendo linha a linha os comandos, separando cada argumento por "espaço"sendo o primeiro argumento relativo ao número de cada *querie* e o segundo o respetivo comando passado para as diferentes *queries*.

4.1.2 Módulo *User* e módulo *Drivers*

Feito o modulo de *parsing* foram criadas estruturas para armazenar as diferentes informações dos utilizadores e condutores, tendo a estrutura dos utilizadores parâmetros para armazenar o *username, nome, género, data de nascimento, data de criação de conta, modo de pagamento e estado da conta* e a estrutura dos condutores os parâmetros *id, nome, data de nascimento, classe do carro, matricula, cidade, data de criação da conta e estado da conta*.

Para armazenar estas informações foi alocada memoria para a criação de ***Hash Tables*** onde cada utilizador/condutor é armazenado, criando assim um catálogo de utilizadores e um catálogo de condutores dando como KEY no caso dos utilizadores o *username* e no caso dos condutores o seu *id*.

4.1.3 Módulo *Rides*

Para armazenar as informações das viagens(*Rides*) foi criada uma estrutura que armazena as informações recebidos através do *parsing* do ficheiro, com os parâmetros *id da viagem, data da viagem, condutor, utilizador, cidade, distancia, avaliação do utilizador, avaliação do condutor, gorjeta e comentário*.
Para armazenar estas informações foi alocada memoria para a criação de um ***Array*** onde cada viagem é armazenada criando assim um catálogo de viagens onde cada id da viagem é um índice no *Array* por ordem crescente.

4.2 Segunda fase desenvolvimento - Queries e Módulos de estatística

4.2.1 Módulos de estatística

Previamente à criação das diferentes *queries* foi criado um módulo de estatística para armazenar cálculos estatísticos dos utilizadores, condutores e cidades em diferentes estruturas.
Para tal foi percorrido sequencialmente o *Array* de catálogo de viagens retirando parâmetros para os utilizadores como *username, numero de viagens, avaliação total, total gasto, distância total, data da viagem mais recente*, para os condutores foram retirados os parâmetros como *id, numero de viagens, avaliação total, total auferido e data da viagem mais recente* e para as estatísticas das cidades *cidade, numero de viagens, distância total e total gasto*.

Foi então alocada memoria suficiente para a criação de ***Hash Tables*** para guardar as várias informações recolhidas onde para a *Hash Table* das estatísticas dos utilizadores foi usado o *username* como *KEY*, para a *Hash Table* dos condutores foi usado o *id* como *KEY* e finalmente para as cidades foi usado o nome da cidade como *KEY*

4.2.2 Querie 1

Na *querie* 1 era pedido para listar o resumo do perfil de um utilizador/condutor, para tal era preciso saber a sua avaliação média, número total de viagens e total gasto/auferido.

Inicialmente na execução da *querie* é verificado se o *id/username* pedido é uma *string* vazia, caso fosse era criado um ficheiro de *output* vazio e terminava a execução da *querie*, caso contrario verificamos, com o auxílio da função *verifica_id* se a *string* recebida é um *username* para um utilizador ou um *id* de um condutor.

Posteriormente é procurado as informações desse utilizador/condutor nos seus catálogos e nas suas estruturas de cálculo estatístico e escrito o resultado num ficheiro de texto de *output*, se esse utilizador/condutor não existisse é criado um ficheiro de output vazio e caso não existissem informações estatísticas relativas ao utilizador/condutor pedido é gerado um ficheiro de *output* com viagens, avaliação média e total gasto tudo a zeros.

4.2.3 Querie 2

Na *querie* 2 era pedido para listas os N condutores com maior avaliação média, para tal precisávamos de criar uma estrutura que conseguíssemos organizar informação de forma fácil e rápida e que tivesse um melhor desempenho em pesquisa sequencial, optando assim por listas ligadas(ver figura 3).

A estratégia consiste no seguinte: pegamos num *driver* correspondente à estrutura *Hash Table statistics drivers* e introduzimos numa lista de tamanho N, onde N é o número de condutores que queremos listar.
Se a lista estiver vazia o *driver* é apenas introduzido, se a lista já conter algum elemento verifica qual dos dois utilizadores possui maior avaliação média, em caso de empate é verificado qual dos condutores tem a viagem mais recente e se houver empate novamente é desempatado usando o *id* do condutor. Caso o condutor que tentamos inserir na lista não cumprir nenhum destas condições e a lista já tiver N condutores simplesmente não o introduzimos na lista, caso contrário inserimos o condutor e descartamos o último que estava na lista, e ainda verificado se o condutor possui uma conta ativa, caso contrario não é inserido na lista.

No final devolve a lista completa de N elementos pedidos, guardando o resultado num ficheiro de *output* e fazendo *free* da estrutura criada para as listas ligadas.

4.2.4 Querie 3

Nesta *querie* era-nos pedido para listar os N utilizadores com maior distância viajada e de forma análoga à *querie* 2 foi usado uma lista ligada.

A estratégia adotada foi semelhante à *querie* 2, era comparado a distância total dos utilizadores, em caso de empate era usado a data da viagem mais recente e se houvesse novamente empate era comparado os *username* de forma crescente, criando assim uma lista com N utilizadores. No final é guardado o resultado da lista num ficheiro de *output* sendo feito o *free* da estrutura criada para as listas ligadas

4.2.5 Querie 4

Na *querie* 4 era pedido para listar o preço médio das viagens numa determinada cidade(*city*). Para tal usamos a mesma estratégia implementada na *querie* 1, ou seja, recorrendo à *Hash Tables statistics City*.

A seguir verificamos se a *string* recebida como argumento é nula, caso seja criamos um ficheiro de *output* vazio e terminamos a execução da *querie*, caso contrario usamos a string como *KEY* para a *Hash Table statistics City*.
Se não for encontrada informações sobre a cidade na estrutura que armazena as estatísticas das cidades então, é criado um ficheiro de *output* vazio terminando a execução da *querie*, se encontrar informações estatísticas fazemos então o cálculo do preço médio através do total gasto na cidade e do número de viagens realizadas e escrevemos o resultado num ficheiro de *output* terminando a execução da *querie*.

5 Conclusão

Na realização desta fase do trabalho prático de Laboratórios de Informatica III tentamos aplicar os conhecimentos lecionados nas aulas numa vertente prática.

Tentamos ser críticos na fase de escolha de estruturas de dados adequadas ao seu contexto de utilização e quais as vantagens e desvantagens.
Acreditamos que tenhamos superado positivamente todas as dificuldades encontradas utilizando as estruturas suprarreferidas e conseguido realizar todas as tarefas propostas e desafios que nos foram apresentados.

6 Referências

[1] Filip Stanis. *008 - DJB2 hash*. URL: `https://theartincode.stanis.me/008-djb2/`.