

# Laboratórios de Informática III – 2022/2023

## Fase 2

### Grupo 24

**João Pedro Baptista (a100705), João Rodrigues (a100896), Mateus Martins (a100645)**

#### 1. Encapsulamento

Para a concretização do trabalho (fase 2) tivemos que encontrar uma estratégia melhor para o encapsulamento, pois o da primeira fase era algo escasso.

Para tal, decidimos implementar uma série de funções que foram inseridas nos módulos dos catálogos de cada CSV. Tais funções permitem obter as características de cada User, Driver ou Ride sem comprometer a privacidade das estruturas.

Começamos por remover as estruturas dos ficheiros “.h”, deixando lá apenas os “typedef”, transferindo as mesmas para os ficheiros “.c” correspondentes. Posteriormente, optamos por criar funções tais como “int get\_n\_drivers()”, “bool is\_driver(char \*id)” para que não fosse necessário os módulos restantes acederem ao catálogo em si.

Para finalizar, foi necessário trocar os momentos em que as queries acediam aos catálogos diretamente, pelas funções agora criadas, e remover os catálogos dos argumentos das mesmas.

#### 2. Verificação de Dados

Em relação à verificação de dados, encontramos alguns contratempos, pois no enunciado era referido que os score\_user e score\_driver poderiam ter valores decimais e iguais a zero. Porém, posteriormente, foi-nos dito que teriam de ser, tais como a distância, inteiros e maiores que 0.

Implementamos as funções “isDvalid”, “isUvalid” e “isRvalid” para verificar se um dado Driver, User ou Ride seriam válidos para adicionar ao catálogo correspondente. Finalmente, precisamos de os inserir nas funções “inserir\_drivers”, “inserir\_users” e “inserir\_rides” para que apenas fosse inserido nos respetivos catálogos os elementos válidos.

#### 3. Memória (Datasets Maiores)

Para resolver alguns problemas de memória que tínhamos, tais como “memory exceeded”, decidimos implementar arrays realocáveis (usando realloc). As arrays dos catálogos começavam com capacidade para apenas 2 elementos e, quando fosse necessário, o realloc interviria para alocar memória para mais elementos, duplicando a capacidade das arrays em questão.

Outra estratégia que utilizamos para diminuir a memória usada do programa foi diminuir o tamanho de cada elemento. Exemplificando: diminuimos o tamanho do ID de 15 para 11 characters. O mesmo processo foi utilizado para as datas, entre outros.

#### **4. Modo Interativo**

Para a realização do modo Interativo, adicionamos uma condição na função main que verifica quando é que o programa é executado sem argumentos (`argc == 1`). Quando o programa é executado nesse modo, já não cria a pasta “Resultados”.

Inicialmente, o programa irá “pedir” que o utilizador digite o caminho para os ficheiros CSV, para possibilitar a chamada das funções que os inserem nos respetivos catálogos.

Decidimos também adicionar um novo argumento às queries (mode). Este argumento permitirá distinguir se será utilizado o modo Batch (0) ou o modo Interativo (1).

Criamos um módulo adicional referente ao modo Interativo que contém o funcionamento idêntico ao input do modo Batch, porém é passado como argumento “mode” 1, ao invés de 0 do modo Batch, para que apenas imprima no terminal as respostas pretendidas quando o programa se encontra no modo Interativo.

Para terminar, quando o utilizador pretende sair do programa, apenas precisa de digitar “exit”.

Em relação à paginação dos resultados longos, não conseguimos implementar essa feature, pois demos mais importância a outros aspetos.

#### **5. Queries**

As queries realizadas nesta fase (Q7, Q8 e Q9) são muito parecidas com as queries anteriores.

Em particular, a Q7 é idêntica à Q2, necessitando apenas de adicionar uma condição para selecionar condutores apenas da cidade pretendida.

Em relação às Q8 e Q9, as estratégias utilizadas foram similares às restantes, apenas necessitando de uma pequena adaptação para a respetiva Query.

Retardamos a ordenação das queries Q2, Q3, Q7, Q8 e Q9 para a 2ª Fase pois já estávamos com falta de tempo. O pensamento para a mesma foi similar para todas as queries. Usamos o quicksort da biblioteca C, aplicando uma função auxiliar para cada exercício que apenas ordenava dois elementos, sendo que a função incluída já ordenava a array independentemente do tipo de dados.

#### **6. Testes Funcionais e de Desempenho**

Implementamos um módulo “tests.c” que realiza os mesmos testes do site “li3.di.uminho.pt”. A partir de cada Dataset, verifica cada input do mesmo e imprime no ecrã “Test 12 passed”, sendo 12 a linha de input. Mostra também o tempo de execução de cada query, independentemente de o teste ter passado ou não.

Para tal, fizemos uma função “compare” que, utilizando um loop while com 2 fgets, verifica se cada linha dos dois ficheiros são iguais, até chegar ao fim dos mesmos.