Relatório do projeto de L.I.3
Fase 1

Universidade do Minho
Escola de Engenharia

Realizado por:

-Luís Ferreira a91672 | @NopeGuy

-Bernardo Lima a93258 | @HBernaH

-Pedro Sequeira a91660 | @pedrusphantom

2022/23

Índice

Explicação do Programa:

Como método de avaliação para a cadeira de L.I.3, foi-nos pedido para realizarmos um *parser* de ficheiros, que depois de captar a informação nos ficheiros de entrada (drivers.csv, riders.csv, users.csv) conforme o input do utilizador (commands.txt) pode executar uma série de queries que respondem a várias questões estatísticas em relação ao *dataset* inserido que serão posteriormente exportadas para a pasta de saída com os vários resultados esperados.

Parsing:

Como seria de esperar, foi-nos necessário desenvolver certas funções capazes de captar a informação dos ficheiros .csv para estruturas de dados que depois serão lidas pelo programa.

Essas funções funcionam á base de fgets() que pegam na informação do nosso buffer que provém inicialmente do ficheiro “commands.txt” e enviam através da função executeQueries() a informação de quais queries a executar.

Após isso, damos *load* ao(s) ficheiro(s) necessário(s) de dados para que a sua informação seja analisada e guardada nos

vários catálogos (definidos como *Binary Trees*) para que depois possam ser acedidas e sua informação processada para as estatísticas necessárias.

Catálogos:

Após a indicação do path dos ficheiros, são enviados os mesmo para o catalogo.c aonde as várias *trees* necessárias são criadas e preenchidas com novamente a ajuda de fgets() para se captar a informação e transformada para o tipo necessário.

Primeiramente os catálogos são inicializados com valor NULL para que depois ao executar as funções getUsers/Drivers/Rides() eles possam ser completados e posteriormente inseridos através da setUsers/Drivers/Rides().

Dentro de cada catálogo referente a cada ficheiro .csv, estão definidas as árvores balanceadas com todos os parâmetros que estão presentes em cada ficheiro (como nome, aniversário, cidade, etc.), que através da função buildUsers/Drivers/Rides(), são verificados os valores incompatíveis ou ilegíveis e inseridos numa árvore temporária os válidos. De seguinte, os valores são enviados para a estrutura final aonde através de *getters* possam ser enviados os dados para serem processados pelos restantes ficheiros.

Queries(1,4,5):

Quando as árvores já estão verificadas e preenchidas, podemos passar os valores para a queries.c aonde serão tratados de modo a responder ás várias queries.
Para além disto, no ficheiro estão escritas as várias funções de verificação e de manutenção das árvores que são chamadas por outros ficheiros.

Nesta primeira fase, decidimos começar por implementar a query 1, 4 e 5.

Na query 1, aonde através de um ID é nos enviada a informação do utilizador ou condutor não encontramos muitos problemas, visto que a sua aplicação é relativamente simples. Para começar separamos os dois casos possíveis, no primeiro caso o ID é um número inteiro e corresponde a um condutor, no segundo o ID é uma *string* com o nome do utilizador. De seguida o valor é comparado com os vários possíveis do catálogo previamente criado até existir uma correspondência de forma a verificar a existência do utilizador ou condutor. No final caso tenha existido um *hit*, são obtidos os outros valores referentes ao ID em questão e calculada a avaliação média e o número de viagens para que seja escrito no ficheiro de output toda a informação referente ao utilizador ou condutor e libertada a memória alocada para a realização da query.

Na query 4, foi nos pedido o preço médio das viagens de uma determinada cidade, então procuramos através do g_tree_foreach a cidade em questão e após catalogarmos somamos os preços e dividimos pelo número de viagens.

Para finalizar, na query 5 tivemos de calcular o preço médio das viagens num determinado espaço de tempo, e usando a

mesma lógica que na query 4, usou-se a libraria time.h para comparar as datas introduzidas e verificar se estão dentro do *timeframe* indicado para consequentemente calcular a média.

Conclusão:

Este trabalho permitiu-nos aprimorar os nosso conhecimentos em parsing de dados, encapsulamento e utilização da glib.

Conseguimos também ganhar bastante aptidão em criação de estruturas de dados e debugging devido aos vários segfault's que obtivemos até conseguir um projeto funcional, sentindo-nos mais aptos para a realização de projetos em C após este trabalho.