**Laboratórios de Informática III – 2022/2023**

**Grupo 24**

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

1. **Estratégias:**
   1. **– Main – “main.c”**

No nosso projeto, começando pela função main, criamos um módulo (ficheiro .c) dedicado a ela.

A função principal do programa recebe 2 argumentos: o número de argumentos que o programa recebe (int argv), e os próprios argumentos do mesmo, colocados em forma de string (char \*argv). Dadas as informações dos docentes, o programa irá receber 2 argumentos: o caminho para a pasta onde estão os ficheiros de entrada e o caminho para um ficheiro de texto que contém uma lista de comandos (queries) a serem executados.

O grupo começou por fazer um “if” que confirma que o argc é igual a 3 (o próprio programa + os dois argumentos que o programa recebe), sendo que se não o for, imprime no ecrã “Número de argumentos inválido”, seguido do número de argumentos dado entre parênteses, retornando -1 logo de seguida. Caso o argv seja igual a 3, o programa corre nas normalidades.

A próxima parte da main é responsável por abrir os ficheiros que são dados como argumento: drivers.csv; users.csv; rides.csv e o ficheiro de inputs. O nosso método para essa etapa consiste em, para cada um dos ficheiros CSV, criar uma string temporária (cuja memória será libertada no final de abrir o ficheiro), copiando (com o comando strcpy) o argv[1] (diretório onde se encontram os ficheiros CSV) para a string. Seguidamente acrescentamos-lhe a string que corresponde ao nome do ficheiro (por exemplo: drivers.csv). Tendo já o caminho do ficheiro que pretendemos abrir, é só necessário abri-lo em modo “r” (read), para que nada o possa alterar. Acrescentamos também que no final da função main, os ficheiros são todos fechados com o uso da função “fclose”.

Existe também um comando “mkdir” que cria um diretório na pasta “trabalho-pratico” chamado “Resultados” para que seja possível criar lá dentro os ficheiros de output.

Para concluir, chamamos a função “insere\_dados”, dando como input todos os ficheiros (4). Completamos também que tivemos uma certa dificuldade no que toca aos ficheiros, pois necessitamos deles em várias funções ao longo do programa, sem os poder definir em qualquer ficheiro .h (sob pena de obtermos o erro de “multiple definitions”).

* 1. **– Catálogo – “catalog.c” & “catalog.h”**

Para armazenar cada elemento (driver, user e ride), decidimos utilizar Structs que permitem guardar todas as informações dos mesmos, como: data de nascimento, ID, username, entre outros.

Usamos também uma hash table que, com o uso da função g\_hash\_table\_lookup, dada uma key (username), devolve o índice desse user na array do catálogo dos users.

A função insere\_dados começa por criar uma string “temp” que irá armazenar cada linha dos ficheiros CSV nos loops.

Para criar os catálogos, decidimos (por agora) utilizar 3 arrays de structs que memorizam os dados dos 3 tipos de elementos. Para esse efeito, decidimos, para cada tipo de elemento, alocar memória usando a função “malloc” com espaço para as respetivas quantidades + 1 (10001, 100001 e 1000001), pois o índice 0 das arrays não serão utilizados (por pura conveniência). Posteriormente é feito um ciclo while fgets (para cada tipo de elemento), em que cada linha de cada ficheiro é armazenado na string temporária mencionada acima. Dentro desse ciclo será criada uma struct (também temporária), que depois será copiada para a array. Achamos adequado utilizar a função “sscanf”, com vários argumentos “%[^;];” (excepto no último que tem de ser %[^\n]; pois acaba a linha), para rechear a struct temporária. É também preenchida a hash table dentro desse for.

No final dessa função é chamada a “handle\_inputs”, passando como argumentos os 3 arrays (catálogos) já preenchidos, a hash table e o ficheiro inputs.

* 1. **– Inputs – “inputs.c” & “inputs.h”**

A função “handle\_inputs” irá criar, da mesma forma da função insere\_dados, um ciclo while fgets que irá percorrer o ficheiros de inputs, armazenando cada linha numa string temporária “input”. Irá também incluir um contador “int i” para o programa “saber” em que linha está (para posteriormente utilizarmos essa informação na criação dos ficheiros de outputs.

Caso o primeiro char da mesma for 1, a função “handle\_inputs” chamará a função “query1” com os argumentos: contador, catálogos, e input+2 (pois irá ignorar o char ‘1’ e o char espaço). Caso o primeiro char for 2, chamará a função query2, com a mesma ordem de ideias.

* 1. **– Query 1 – “query1.c” & “query1.h”**

Para saber se o resto da string do input (sem o 1 e o espaço) é um ID ou um username, a função “query1” utiliza uma função auxiliar (definida em randomfuncs.c & randomfuncs.h) “isDriver”, que retorna um int: 1 se a string dada for um Driver (se todos os elementos dessa string forem dígitos), e, caso contário 0 se a string for um username.

Caso seja um Driver, buscar-se-á ao catálogo a struct correspondente ao mesmo, através do índice (que é igual ao ID). Posteriormente, irá ser feito um loop for que percorre o catálogo das rides, e em cada Ride verifica se o ID do Driver é igual ao ID do Driver da Ride. Caso seja, irá proceder a calcular: total de avaliações; número de viagens e total auferido. No final do loop, irá se dividir o total de avaliações pelo número de viagens de modo a obter a média das avaliações. Se o número de viagens for igual a 0, a avaliação média irá ser preenchida automaticamente com o valor 0.

Todas as informações serão juntadas numa única string “output”, sendo que a idade será calculada com ajuda da função “age”, (que calcula a idade através da data de nascimento).

Caso seja User, a estratégia utilizada foi a mesma, porém tivemos de obter o índice do user a partir da hash table, de modo a obter todas as informações do mesmo acedendo a esse índice no catálogo.

Posteriormente irá ser utilizada outra função auxiliar (definida em outputs.c & outputs.h) “handle\_outputs”, que recebe o contador referido no tópico passado e a string que foi dada como output, para que seja escrita no respetivo ficheiro. Essa função cria uma string temporária “filename” em que, utilizando a função sprintf e o contador, é preenchida com o caminho correto para o novo ficheiro ("Resultados/command%i\_output.txt", counter). Para criar esse tal ficheiro é criado um novo FILE temporário, utilizando depois o “fopen” para o fazer, com o modo write.

Finalmente, é apenas necessário escrever o output no ficheiro com a ajuda do “fprintf”.

* 1. **– Query 2 – “query2.c” & “query2.h”**

A query 2 recebe como argumento o counter, os catálogos e o N em forma de string.

Com ajuda do malloc/calloc iremos alocar memória para as variáveis necessárias a esta query, sendo essas libertadas no final.

A array de recente\_rides (que guarda a viagem mais recente de cada user) é inicializada toda com “00/00/0000”, para que qualquer data seja mais recente do que esta.

Posteriormente é usado um loop for que percorre todo o catálogo das rides, em que para cada uma delas: verifica se o driver dessa ride está ativo; incrementa o total de avaliações desse driver com a avaliação dessa ride; incrementa em 1 o número de viagens desse driver, e se a data dessa ride for mais recente do que a data que está colocada na array recente\_rides, substitui a mesma na array.

Nesta etapa, a array das avaliações médias será preenchida com a divisão do total de avaliações de cada driver com o número de viagens do mesmo.

Com a ajuda de outro for loop, irão ser calculados as N maiores avaliações médias, e o índice dos respetivos drivers serão guardados na array id\_maiores (sendo que id\_maiores[0] guarda o ID do driver com a maior avaliação média).

De seguida, é feito o desempate diretamente na array id\_maiores.

Finalizando, apenas é necessário chamar a função handle\_outputs para criar o ficheiro com o output pretendido.

**1.6 – Query 3 – “query3.c” & “query3.h”**

Relativamente à Q3, a estrutura da mesma é bastante similar à da Q2, sendo que as diferenças são que em vez de avaliação média, temos a distância total, e sendo que a Q3 engloba também os users, usamos a hash table para dar o índice do mesmos. A partir dessa etapa, a estratégia é a mesma do que para os drivers.

1. **Limitações**

Encontramos algumas limitações na realização do projeto, tais como:

- Na abertura inicial dos ficheiros, tentamos colocar a definição dos mesmos num ficheiro “main.h”, de modo a que conseguíssemos acessá-los em qualquer parte do programa, incluindo o main.h no mesmo. Porém, infelizmente deparámo-nos com o erro de “multiple definitions”, e depois de muitas tentativas de resolução sem sucesso, encontrámos a brilhante ideia de passar os catálogos como argumentos para o resto das funções.