Departamento de Informática

Escola de Engenharia

Universidade do Minho

Relatório Fase 1

Laboratórios de Informática III

Licenciatura em Engenharia Informática

2023/2024

Grupo 47

Ana Cerqueira, a104188

Nuno Silva, a104089

Eduardo Faria, a104353

18/11/2023

Índice

Introdução

No âmbito da disciplina "Laboratórios de Informática III", fomos incumbidos do desenvolvimento de um sistema para organizar dados relacionados a viagens. Este sistema envolve a manipulação de informações sobre voos e respetivos passageiros, utilizadores e reservas cujas informações são inseridas em *Hash Tables* de modo a ficarem disponíveis de forma organizada e otimizada para uso durante o programa.

Escolhemos estas estruturas de dados devido à sua organização por *keys* que nos permite uma procura rápida e resultados eficientes. Estas *Hash Tables* são criadas de acordo com *structs* definidas por nós que se adequam aos diferentes dados que serão necessários manipular no decorrer do programa.

Estrutura de Dados

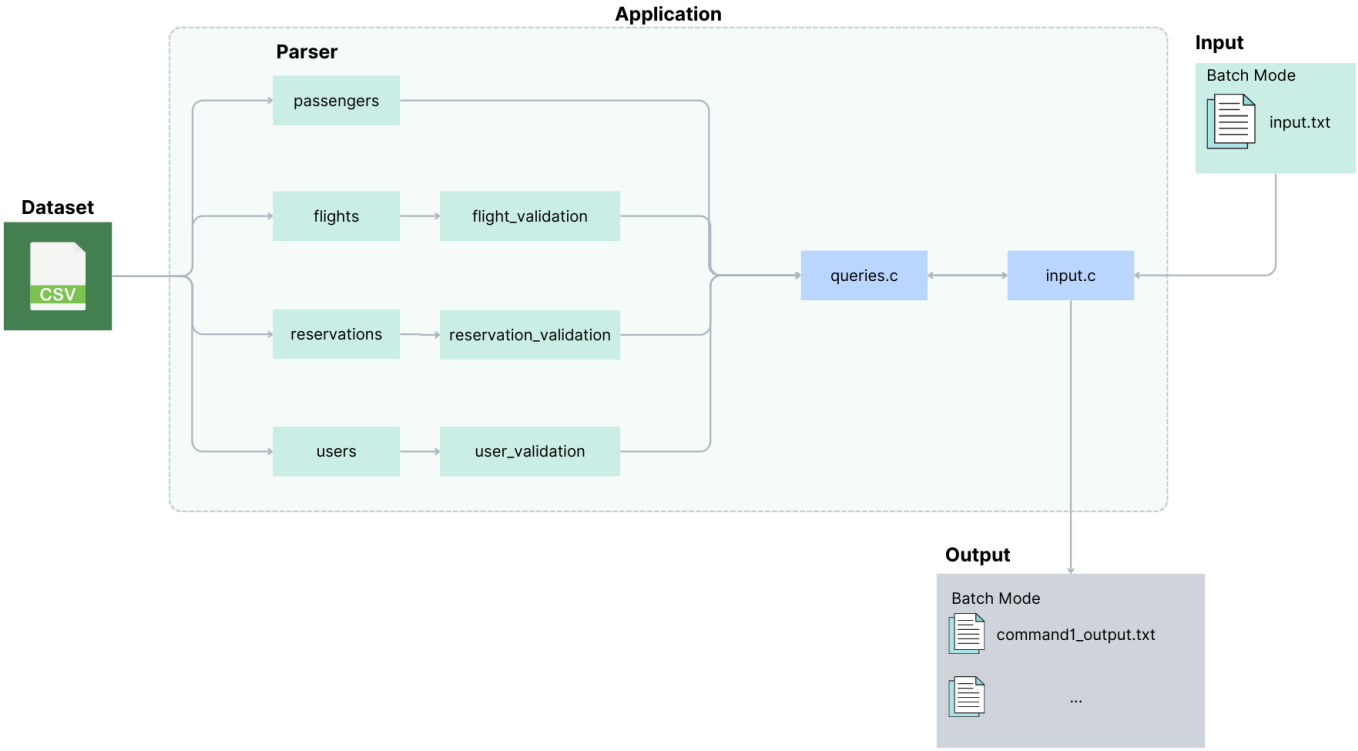
Com o intuito de guardar e posteriormente manipular os dados provenientes dos ficheiros .csv decidimos adotar uma estrutura de dados que nos permite uma procura eficiente, considerando a grande quantidade de dados que temos que gerir. Com isto, a solução passou pelo uso de *Hash Tables*.

Utilizando *structs* específicas, recolhemos os dados provenientes dos diferentes arquivos .csv e efetuamos a organização de acordo com a sua origem. Deste modo, o ficheiro *users.csv,* por exemplo, vai ser lido e os seus dados serão guardados numa *struct User*. Um processo semelhante é feito para os restantes ficheiros.

Modularidade e encapsulamento

De forma a modularmos o código, separámos os ficheiros em ficheiros .c e .h.
Assim, a organização do nosso projeto segue o esquema:

```
|-trabalho-pratico
    |-include
          |----defines.h
          |----flights.h
          |----...
    |-Relatório
    |-Resultados
          |----command1_output.txt
          |---- ...
    |-src
          |----obj
                  |----flights.o
                  |---- ...
          |----flights.c
          |----main.c
          |---- ...
    |-Makefile
|-README.md
```



O cérebro do programa está definido na main.c. É a partir dela que se direciona o programa para as respetivas funções que estão distribuídas pelos restantes ficheiros .c. Cada módulo, exceto a main, acede apenas aos *includes* e livrarias que serão necessários ao seu funcionamento. Cada um tem definido um header file (.h) correspondente onde são declaradas as funções que serão utilizadas fora do ficheiro de origem.

Deste modo, assegurámos o encapsulamento dividindo todas as funções de acordo com a sua utilização ao longo do funcionamento do programa. Garantimos também que funções que não serão necessárias em determinado módulo permanecerão imutáveis visto que não serão chamadas. Isto garante também uma maior fiabilidade dos dados.

Execução

Nesta primeira etapa, apenas foi implementado o modo *batch.* Deste modo, após fazer *make* o utilizador de invocar através da linha de comandos dois parâmetros: o primeiro com o caminho para a pasta onde se encontram os ficheiros de entrada “flights.csv”, “passengers.csv”, etc.; e o segundo com o caminho para o ficheiro de input que contém a lista de comandos para chamar as diferentes queries.

No ficheiro de input, as linhas devem ter o formato ‘2 U000000001’ ou ‘2F U000000001’, sendo o primeiro parâmetro o número da querry a ser chamada.

O ‘F’ define apenas a forma de output que pode ser no formato (sem ‘F’):

```
F000000123;2023/10/06;flight
R000000456;2023/10/02;reservation
F000000121;2023/10/01;flight
```

Ou no formato (com ‘F’):

```
--- 1 ---
id: F000000123
date: 2023/10/06
type: flight
--- 2 ---
id: R000000456
date: 2023/10/02
type: reservation
```

Os dados que se seguem à definição do número da query e à sua forma de output vão depender da query selecionada podendo assumir várias formas como ‘JéssiTavares910 flights’, ‘HTL1001 2023/05/02 2023/05/03’, ‘J’, etc.

Após a leitura de cada linha é efetuada a execução correspondente e os resultados são guardados em ficheiros commandN_output.txt, dentro da pasta Resultados, onde N corresponde ao número da linha no ficheiro de input.

Queries

Nesta primeira fase, era pedida a resolução de 60% das queries. O nosso grupo selecionou para o projeto as queries 1, 2, 3, 4, 8 e 9. De seguida, apresentamos a explicação individual do raciocínio usado na resolução de cada query.

Query 1

Nesta pergunta é pedido para gerar resumos formatados com base num <ID> fornecido, abrangendo três entidades principais: utilizadores, voos e reservas. O <ID> determina o tipo de entidade e, consequentemente, o formato do resumo retornado.

Utilizador:
nome;sexo;idade;código_do_país;passaporte;número_voos;número_reservas;total_gasto

Voo:
companhia;avião;origem;destino;partida_est;chegada_est;número_passageiros;tempo_atraso

Reserva:
id_hotel;nome_hotel;estrelas_hotel;data_início;data_m;pequeno_almoço;número_de_noites;preço_total

No caso de uma reserva, identificada pelo prefixo "Book" no <ID>, a função verifica a existência da reserva, calcula informações como datas e preço total, e gera um resumo formatado com detalhes específicos do hotel da mesma.

Para voos, a função verifica se o identificador começa com um dígito, indicando que se refere a um voo. Em seguida, obtém informações sobre o voo, como datas de partida e chegada, número de passageiros e atrasos, gerando um resumo formatado correspondente.

Quando o identificador não é uma reserva nem um voo, presume-se que se refere a um utilizador. A função verifica a existência e a ativação da conta do utilizador, obtém dados sobre o número de voos, reservas e total gasto, gerando um resumo formatado de acordo.

Query 2

Nesta segunda pergunta era pedido para listar os voos e/ou reservas de um usuário.

Existem três opções de entrada possíveis. Uma para quando o ID fornecido é o ID de um voo, outra para o caso de ser o ID de uma reserva e ainda uma opção quando nenhum ID é fornecido. Vamos analisar o seu comportamento para os diferentes casos:

- Se for um voo, pesquisa pelo *id_user* na *Hash Table* de passageiros.

- Se for uma reserva, pesquisar pelo *id_user* na *Hash Table* de reservas.

- Se o *id_arg* estiver vazio, pesquisar pelo *id_user* na *Hash Table* de passageiros e na *Hash Table* de reservas.

Em todos os casos, as ocorrências são armazenadas em um array chamado Output2. Posteriormente, os resultados são ordenados de acordo com as datas, do mais recente ao mais

antigo, usando as funções ‘qsort’ e ‘sort’. Caso as datas sejam iguais, a ordenação é feita de acordo com o id, também de forma crescente.

Query 3

Nesta terceira questão era solicitado o resultado da classificação média de um hotel.

Esta função exibe a classificação média de um hotel de acordo com o identificador 'hotel-id'. Ela procura na *Hash Table* 'reservations_hash_table' pela 'reservation->rating' e usa a variável 'rating' para acumular a soma de todos esses valores. Utiliza também a 'nReservations' para contar quantas reservas foram adicionadas. No final, divide-as para obter a média. Deste modo, média = rating/nReservations.

Query 4

Este quarto problema solicita a lista de reservas de um hotel.

A função fornece uma lista de reservas de hotel de acordo com o ID fornecido. Ela procura na tabela 'reservetions_hash_table' pelas reservas do hotel desejado. Em seguida, cria uma lista com todas as reservas e, à medida que procura por mais reservas, insere as novas de forma ordenada no 'reservationsArray'. Este array é ordenado pelas datas (da mais recente para a mais antiga) e, se as datas forem iguais, ordena de acordo com o ID.

Query 8

A query 8 pede para calcular a receita total de um hotel num intervalo de datas específico, considerando apenas o preço por noite das reservas que ocorrem dentro desse intervalo. O <ID> do hotel é fornecido como parâmetro, juntamente com as datas de início e fim do intervalo desejado.

A função utiliza uma iteração sobre a *Hash Table* de reservas (reservations_hash_table), verificando se cada reserva pertence ao hotel identificado pelo hotel_id e se suas datas se enquadram no intervalo especificado. Em caso afirmativo, a função calcula a receita com base na diferença entre as datas de início e fim da reserva, multiplicada pelo preço por noite da mesma.

O resultado final é a soma total das receitas de todas as reservas que atendem aos critérios estabelecidos.

É importante observar que a função faz uso de outra função chamada func_compare_dates para comparar as datas e determinar como as reservas se encaixam no intervalo desejado. A lógica da comparação de datas visa cobrir diferentes cenários, como reservas que começam ou terminam dentro do intervalo, reservas que abrangem totalmente o intervalo, entre outros, retornando a receita total do hotel para o período especificado.

Query 9

A função func_answer_query_9 tem como objetivo listar todos os utilizadores cujo nome começa com um prefixo fornecido por argumento. A lista é ordenada de forma crescente pelo nome dos utilizadores. Em caso de nomes iguais, é utilizado o <ID> de utilizador como critério de desempate, também de forma crescente. Utilizadores inativos não são considerados na pesquisa.

A função utiliza uma *Hash Table* de utilizadores (users_hash_table) para iterar sobre os utilizadores armazenados. Durante a iteração, ela verifica se o nome do utilizador começa com o prefixo fornecido e se a conta está ativa. Caso ambas as condições sejam atendidas, o nome e o identificador do utilizador são inseridos numa lista ordenada por meio da função func_insert_name_in_order.

Ao final da iteração, a função aloca memória para a string de output (output9) e formata os resultados de acordo com o formato especificado pelo parâmetro input_prefix. O output é composto por uma lista de utilizadores formatados conforme o formato especificado no enunciado.

Conclusão e 2º fase

Nesta primeira fase deparámo-nos com algumas adversidades. O início foi algo confuso e atribulado devido às dúvidas sobre qual estrutura de dados adotar e, posteriormente, devido à nossa falta de conhecimento sobre as *Hash Tables*. Contudo, sentimos que conseguimos cumprir com os objetivos propostos. Esperámos, ainda assim, na segunda fase conseguir melhorar a gestão de memória do nosso projeto e se possível otimizar algumas funções.