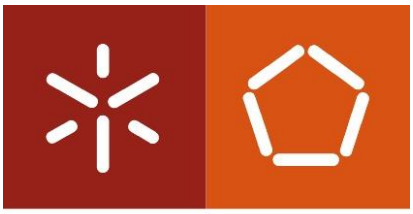

Universidade do Minho

Licenciatura em Engenharia Informática

Laboratórios de Informática - Projeto Prático

Grupo 82

João Ferreira (A79495)

Ano Letivo 2022/2023

Conteúdo

Capítulo 1

Resumo

Neste relatório encontra-se descrita a resolução da primeira fase do Enunciado do Projeto de 2022/2023 para a UC de Laboratórios de Informática, e descritas as estruturas e estratégias usadas por este grupo de forma a cumprir as exigências propostas.

1.1 Introdução

O foco deste trabalho é consolidação de conhecimentos essenciais da linguagem C e de Engenharia de Software,nomeadamente, modularidade e encapsulamento, estruturas dinâmicas de dados, validação funcional e medição de desempenho (computacional, consumo de memória, etc), bem como,a consolidação do uso de ferramentas essenciais ao desenvolvimento de projetos em C, nomeadamente, compilação, linkagem, definição de objetivos de projeto com base nas suas dependências e depuração de erros, e de gestão de repositórios colaborativos.

Capítulo 2

Análise do Problema

2.1 Primeira abordagem

De uma forma simplista e tendo como contexto os ficheiros "users.csv", "drivers.csv"e "rides.csv"fornecidos, o primeiro contacto com estes dados foi uma importante fonte de informação a que se deu a devida análise de forma a criar as primeiras estruturas compatíveis com os dados fornecidos e também com vista á resolução das queries propostas. Da análise dos dados, recorrendo á plataforma Knime, foram retiradas as seguintes informações:

- Dados users.csv - 5.1

  Username - String
  Name - String
  Gender - String (Convertível para M/F) - Enum
  Birth_date - String (Convertível para Data) - Struct tm
  Account_creation - String (Convertível para Data) - Struct tm
  Pay_method - String (Convertível para cash/credit_card/debit_card) - Enum
  Account_status - String (Convertível para active/inactive) - Enum

- Dados drivers.csv - 5.2

  Id - Number - Int
  Name - String
  Brith_day - String (Convertível para Data) - Struct tm
  Gender - String (Convertível para M/F) - Enum
  Car_class - String (Convertível para Basic/Green/Premium) - Enum
  License_plate - String
  City - String (Convertível para Lisboa/Porto/Faro/Braga/Setúbal) - Enum
  Account_creation - String (Convertível para Data) - Struct tm
  Account_status - String (Convertível para active/inactive) - Enum

- Dados rides.csv - 5.3

  Id - Number - Int
  Date - String (Convertível para Data) - Struct tm
  Id_driver - Number - Int
  Username - String
  City - (Convertível para Lisboa/Porto/Faro/Braga/Setúbal) - Enum
  Distance - Number - Int
  Score_user - Number - Int
  Score_driver - Number - Int
  Tip - Number - Double
  Comment - String

2.2 Estruturação

Durante o processo de abordagem ao problema e tendo consciência de todas as exigências das queries, foram encontrados dados que seriam redundantes, mas que no entanto, não foram removidos das bases de dados por decisão de uma não perda de informação.
Foram, no entanto, adicionados campos ás estruturas de forma a incluir dados pertinentes sobre a entidade (User/Driver) em questão, não se tendo optado pela organização sugerida de forma a simplificar o código gerado e otimizar os gastos de espaço e tempo na máquina de todo o processo.
O resultado deste processo permitiu chegar ás estruturas finais em anexo:

- Users 5.4
- Drivers 5.5
- Rides 5.6

2.3 Fluxo do Processo

- Entrada na main com dois argumentos
- Verificação de Path's input (argumentos)
- Criação de catálogos para os dados lidos
- Leitura dos ficheiros de input
- Carregamento das leituras para os catálogos
- Processamento das Queries
- Escrita de Resultados
- Libertação de memória alocada
- Encerramento de Ficheiros
- Fim de Processo main

Capítulo 3

Estratégia de resolução de Queries e Controlo de Memória

3.1 Querie 1

- Coleta de inputs
- Busca de identificador de entidade (User / Driver) de contas activas na HashTable respectiva
- Geração de ficheiro output

3.1.1 Análise de Complexidade/Custo

Constante

3.2 Querie 2

- Coleta de inputs
- Criação de Lista com n User's ordenada pela classificação removendo contas não ativas
- Caso Avaliação de User n seja igual a n+1, comparar datas de ultima ride e id. Caso n menor que n+1, swap de elementos.
- Geração de ficheiro output

3.2.1 Análise de Complexidade/Custo

Quadrática

3.3 Querie 3

- Coleta de inputs
- Criação de Lista com n Drivers's ordenada pela classificação removendo contas não ativas
- Caso Avaliação de Driver n seja igual a n+1, comparar datas de ultima ride e id. Caso n menor que n+1, swap de elementos.
- Geração de ficheiro output

3.3.1 Análise de Complexidade/Custo

Quadrática

3.4 Querie 4

- Coleta de inputs
- Iterar sobre a HashTable Rides procurando equivalências de cidade
- Caso haja equivalência, guardar informação sobre custo e incrementar contador de viagem.
- Calcular resultado
- Geração de ficheiro output

3.4.1 Análise de Complexidade/Custo

Linear

3.5 Querie 5

- Coleta de inputs
- Iterar sobre HashTable rides procurando rides entre as datas input gerando uma list
- Usando a list gerada calcular valores para média de valores gerando um resultado total.
- Geração de ficheiro output com o resultado total

3.5.1 Análise de Complexidade/Custo

Linear

3.6 Querie 6

- Coleta de inputs
- Iterar sobre HashTable rides procurando rides entre as datas input e a cidade input gerando uma lista
- Usando a lista gerada calcular valores para média de distância gerando um resultado total.
- Geração de ficheiro output com o resultado total

3.6.1 Análise de Complexidade/Custo

Linear

3.7 Querie 7

- Coleta de inputs
- Iterar sobre HashTable rides procurando condutores ativos numa cidade input gerando uma lista
- Gerar uma HashTable de Drivers nova de modo a iterar pela lista gerada anteriormente para adicionar elementos com stats inicializadas.
- Iterar sobre a HashTable gerada anteriormente para gerar uma lista.
- Sort da Lista por score, em caso de empate por id.
- Geração de ficheiro output com a lista ordenada

3.7.1 Análise de Complexidade/Custo

Quadrática

3.8 Querie 8

- Coleta de inputs
- Iterar sobre a HashTable de Rides procurando por viagens em que o user e driver têm o mesmo gender, são active e o nº de anos de perfil é maior a x input
- Criação de uma estrutura auxiliar com os dados recolhidos sobre as viagens anteriores que cumprem os requisitos e inserir a estrutura numa lista
- Fazer o sort da lista anteriormente gerada
- Geração de ficheiro output com a lista ordenada

3.8.1 Análise de Complexidade/Custo

Quadrática

3.9 Querie 9

- Coleta de inputs
- Iterar sobre a HashTable rides comparando datas de input e tip da viagem ¿ 0, gerando uma lista.
- Sort da lista gerada
- Geração de ficheiro output com a lista ordenada

3.9.1 Análise de Complexidade/Custo

Linear

3.10 Controlo de memória

Pelo anexo 5.7, verificamos que existem blocos de memória que estão disponíveis á saída do programa, no entanto, são blocos referentes e controlados pela Glib e não do projeto desenvolvido. Verificamos também que foi utilizado um total de 433 Mbts de memória, com um total de 14,012,982 alocações de memória, referente ao anexo 5.8

Capítulo 4

Conclusão

A realização deste projeto permitiu consolidar a matéria lecionada não só sobre C, mas também sobre encapsulamento e modularidade. Possibilitou, especificamente a consolidação de validação funcional e medição de desempenho (computacional, consumo de memória), bem como a consolidação do uso de ferramentas essenciais ao desenvolvimento de projetos em C.

Capítulo 5

Anexos

Table "default" - Rows: 100000 | Spec - Columns: 7 | Properties | Flow Variables

Columns: 7	Column Type	Co...	C...	...	Shape Handler	Filter Handler	Low...	Upper Bo...	Value 0	Value 1	Value 2
username	String	0					?	?	?	?	?
name	String	1					?	?	?	?	?
gender	String	2					?	?	M	F	?
birth_date	String	3					?	?	?	?	?
account_creation	String	4					?	?	?	?	?
pay_method	String	5					?	?	cash	credit_card	debit_card
account_status	String	6					?	?	active	inactive	?

Figura 5.1: Análise a dados users.csv

Table "default" - Rows: 10000 | Spec - Columns: 9 | Properties | Flow Variables

Columns: 9	Column Type	Column In...	Color Han...	Size Hand...	Shape Ha...	Filter Han...	Lower Bo...	Upper Bo...	Value 0	Value 1	Value 2	Value 3	Value 4
id	Number (integer)	0					1	10000	?	?	?	?	?
name	String	1					?	?	?	?	?	?	?
birth_day	String	2					?	?	?	?	?	?	?
gender	String	3					?	?	F	M	?	?	?
car_class	String	4					?	?	green	premium	basic	?	?
license_plate	String	5					?	?	?	?	?	?	?
city	String	6					?	?	Setúbal	Braga	Lisboa	Porto	Faro
account_creation	String	7					?	?	?	?	?	?	?
account_status	String	8					?	?	active	inactive	?	?	?

Figura 5.2: Análise a dados drivers.csv

Table "default" - Rows: 1000000 | Spec - Columns: 10 | Properties | Flow Variables

Columns: 10	Column Type	Co...	...	...	...	...	Lower Bo...	Upper Bo...	Value 0	Value 1	Value 2	Value 3	Value 4
id	Number (integer)	0					1	1,000,000	?	?	?	?	?
date	String	1					?	?	?	?	?	?	?
driver	Number (integer)	2					1	10,000	?	?	?	?	?
user	String	3					?	?	?	?	?	?	?
city	String	4					?	?	Setúbal	Faro	Porto	Braga	Lisboa
distance	Number (integer)	5					1	14	?	?	?	?	?
score_user	Number (integer)	6					1	5	?	?	?	?	?
score_driver	Number (integer)	7					1	5	?	?	?	?	?
tip	Number (double)	8					0.5	5	?	?	?	?	?
comment	String	9					?	?	?	?	?	?	?

Figura 5.3: Análise a dados rides.csv

```
typedef struct data_user{

    char *username;
    char *name;
    enum gender gender;
    struct tm birth_date;
    struct tm account_creation;
    enum pay_method pay_method;
    enum account_status account_status;
    int idade;
    int num_viagens;
    int distancia_viajada;
    int total_avaliacao;
    double total_gasto;
    struct tm data_ultima_ride_user;

}*DATA_USER;
```

Figura 5.4: Struct Users

```c
typedef struct data_driver{

    int id;
    char *name;
    struct tm birth_day;
    enum gender gender;
    enum car_calss car_class;
    char *license_plate;
    enum city city;
    struct tm account_creation;
    enum account_status account_status;
    int age;
    int num_viagens;
    int avaliacao_total;
    double total_auferido_driver;
    struct tm data_ultima_ride_driver;


}*DATA_DRIVER;
```

Figura 5.5: Struct Drivers

```c
typedef struct data_rides{

    int id;
    struct tm date;
    int id_driver;
    char *username;
    enum city city;
    int distance;
    int score_user;
    int score_driver;
    double tip;
    char *comment;


}*DATA_RIDES;
```

Figura 5.6: Struct Rides

```
==162936== LEAK SUMMARY:
==162936==    definitely lost: 0 bytes in 0 blocks
==162936==    indirectly lost: 0 bytes in 0 blocks
==162936==      possibly lost: 0 bytes in 0 blocks
==162936==    still reachable: 18,804 bytes in 9 blocks
==162936==         suppressed: 0 bytes in 0 blocks
```

Figura 5.7: Struct Rides

```
==162936== HEAP SUMMARY:
==162936==     in use at exit: 18,804 bytes in 9 blocks
==162936==   total heap usage: 14,012,982 allocs, 14,012,973 frees, 454,226,354 bytes allocated
```

Figura 5.8: Struct Rides