LI3 22/23

Relatório 1ª Fase

Projeto desenvolvido por:

- Carlos Ferreira, a89509
- Daniel Du, a97763
- Henrique Pereira, a97205

Introdução

Na unidade curricular de Laboratórios de Informática III foi proposto desenvolver uma aplicação capaz de ler, interpretar e processar um input de dados de grande escala, e implementar soluções capazes de responder a queries sobre os mesmos dados.

Estratégia utilizada

Estruturas de dados utilizadas para representar utilizadores, condutores e viajens

As estruturas de dados atuais foram concebidas de forma a responder às queries propostas de forma mais cómoda e eficiente possível tanto ao nível de performance como de uso de memória.

Onde fizer sentido são apresentados os prós e os contras das decisões tomadas. Considerou-se sempre que os prós superam os contras.

Utilizador

```
struct user {
    char *username;                 // Alcunha
    char *name;                     // Nome
    char gender;                    // Género
    uint8_t age;                    // Idade
    unsigned short account_age;     // Idade da Conta
    U_Status account_status;        // Estado da Conta
    unsigned short sum_score;       // Sumatório de Avaliações
    double total_spent;             // Total Gasto
    unsigned short total_distance;  // Distancia Total
    unsigned short n_trips;         // Número de Viagens
    unsigned short last_ride_date;  // Data da Última Viagem
};
```

- **Alcunha**: É naturalmente representado como uma string.
- **Nome**: É naturalmente representado como uma string.

- **Género**: Foi considerado representar o género como um enum, porém optamos por representar como um char.

  Prós:

  - O próprio género é representado por um caracter tanto nos ficheiros de input como nas respotas à queries, pelo que guardá-lo como um char evita os processos redundantes de conversão entre char e enum.
  - Um enum ocupa o espaço de um int, 4 bytes, pelo que guardar como char, 1 byte, poupa 3 bytes por utilizador.
  - Os caractéres 'M' e 'F' são suficientemente representativos dos géneros que representam para se justificar prescindir da clareza que utilizar um enum poderia ter fornecido.

  Contras:

  - Representação no código não tão bom, mas pode ser coisado com macros.

- **Idade da Conta**:
- **Idade**: Representado por um inteiro positivo de 1 byte [0, 255].
- **Estado da Conta**: Apesar de, tal como no género, o estado da conta ser um dado binário, optámos por utilizar um enum que, apesar de ocupar espaço desnecessário, fornece uma maior clareza no código.
- **Sumatório de Avaliações**: Consideramos que um unsigned short que representa valores até 65536 é suficiente para representar o sumatório de todas as avaliações de um utilizador. Mesmo que o utilizador seja sempre avaliado com 5 estrelas, isto permite representar até 13107 viagens, a duas viagens por dia, 18 anos. Este campo poderá passar a ser representado por um int caso consideremos que 2 bytes não é suficiente.
- **Total Gasto**: Representado como double como indicado pelos docentes da UC.
- **Distancia Total**: Distancia total percorrida pelo utilizador. Consideramos que um unsigned short é suficiente para representar este campo por razões semelhantes ao sumatório de avaliações.
- **Número de Viagens**: Representa o total número de viagens realizadas pelo utilizador. Este campo será necessáiro para calcular as médias requeridas nas queries.
- **Data da Última Viagem**: Ver data em Viagem

Considerações adicionais: - O sumatório de avaliações, dinheiro gasto e distancia percorrida não são guardados por serem dados diretamente úteis para responder às queries, mas para serem divididos pelo número de viagens realizadas pelo utilizador de forma a obter a média de cada dado. Decidimos seguir esta opção pois: - à exceção do total gasto, evita representar estes dados como um

double poupando 6 bytes por campo, 12 bytes por utilizador. - evita guardar cada elemento numa lista e posteriormente somar e dividir - evitar fazer consecutivas multiplicações e divisões de forma a ter sempre a média e ao evitar fazê-lo desta forma também se evita perder precisão nos dados.

- Os tipos de `sum_score`, `total_distance` e `n_trips` podem não ser grande suficientes para inputs de maiores dimensões pelo que estes tipos poderão ser alterados na segunda fase do projeto.
- Como não é útil para responder a qualquer das queries, o método de pagamento não é incluído na estrutura de um utilizador.

Condutor

```
struct driver {
    long id;                        // ID
    char *name;                     // Nome
    char gender;                    // Género
    uint8_t age;                    // Idade
    Car_Class car_class;            // Classe de Carro
    unsigned short account_age;     // Idade da Conta
    D_Status account_status;        // Estado da Conta
    unsigned short sum_score;       // Sumatório de Avaliações
    double total_earned;            // Total auferido
    unsigned short n_trips;         // Número de Viagens
    unsigned short last_ride_date;  // Data da Última Viagem
};
```

- **ID**: Representado por um inteiro de 4 bytes. O ID pode ter até 12 digitos pelo que é necessários 4 bytes para representar todos os valores possíveis. Utilizar uma string iria ocupar até 12 bytes.
- **Classe de Carro**: Enum que se justifica pela melhor legibilidade de código.
- **Distancia Total**: unsigned short serve é suficientemente grande para armazenar este parâmetro
- Todos os outros elementos são idênticos ou análogos aos presentes no Utilizador.

Considerações adicionais: - Como não é útil para responder a qualquer das queries, a matrícula não é incluída na estrutura de um condutor.

Viagem

```c
struct ride {
    long id;                // ID
    unsigned short date;    // Data
    long driver;            // ID do Condutor
    char *user;             // Alcunha do Utilizador
    char *city;             // Cidade
    uint8_t distance;       // Distância
    uint8_t score_user;     // Avaliação do User
    uint8_t score_driver;   // Avaliação do Condutor
    double cost;            // Custo da Viagem
    double tip;             // Valor da Gorjeta
};
```

- **ID**: Idêntico a id do Condutor.
- **Data**: A data da última viagem é guardada como o número de dias desde 1 de janeiro de 1970 até à data da viagem. Este dado é guardado para motivos de ordenação requeridos pela queries X e Y.

  Prós:

  - Um `unsigned short` é capaz de representar 65536 dias, 180 anos, pelo que é mais do que suficiente para representar as datas até 2150.
  - Tal representção da data permite uma maior facilidade em comparar datas.

  Contras:

  - Implica um processo de conversão para armazenar e representar de folta ao utilizador.
- **ID do Condutor**: Ver id do Condutor.
- **Alcunha do Utilizador**: Naturalmente representado como uma string.
- **Cidade**: Naturalmente representado como uma string.
- **Distância**: Inteiro de 1 byte ([0, 255]). Consideramos 255 como valor máximo de kilometros viajados numa viagem um valor suficiente tendo em conta o tipo de serviço e os valores presentes no dataset exemplo. Porém, se tal não for suficiente, será alterado para um inteiro de 2 bytes ([0, 65536]).
- **Avaliação do Utilizador**: Inteiro de 1 byte.
- **Avaliação do Condutor**: Ineitro de 1 byte
- **Custo**: Double.
- **Gorjeta**: Inteiro de 2 bytes.

Considerações adicionais: - Como não é útil para responder a qualquer das queries, o comentário não é incluído na estrutura de uma viagem.

Estratégia de modularização e reutilização de código

Módulos criados

- catalog: agregação dos vários módulos e estruturas
- db: leitura dos inputs para estruturas
- user: representação de um utilizador
- users: coleção de utilizadores
- driver: representação de um condutor
- drivers: coleção de condutores
- ride: representação de uma viagem
- rides: coleção de viagens
- query2: solução de query2
- query3: solução de query3

Estratégia de encapsulamento e abstração

- Todos os módulos escondem como são implementados pelo que qualquer operação sobre os seus dados pode e deve ser feita com as funções que estes desponibilizam.
- O controller recebe sempre cópias dos dados do model.

Queries

Pré-processamento

Antes de uma query poder ser chamada, os dados são processados de forma a poder executar qualquer query o mais eficientement possível:

1. É criada uma hash table para os utilizadores com o utilizador como valor e os seu *nickname* como chave.
2. É criada uma hash table para os condutores com o condutor como valor e os seu *id* como chave.
3. É criado um array para as viagens.
4. O array de viagens é percorrido e os dados em falta em cada viagem e nos utilizadores e condutores correspondentes à viagem são preenchidos.
5. É criado um array para os condutores, ordenado de acordo com especificações da query 2.
6. É criado um array para os utilizadores, ordenado de acordo com especificações da query 3.

Querie 1

Descrição:

> Q1: Listar o resumo de um perfil registado no serviço através do seu identificador, representado por <ID>. Note que o identificador poderá corresponder a um utilizador (i.e., username no ficheiro

users.csv) ou a um condutor (i.e., id no ficheiro drivers.csv). Em cada caso, o output será diferente, mais especificamente: - Utilizador

nome;genero;idade;avaliacao_media;numero_viagens;total_gasto - Condutor

nome;genero;idade;avaliacao_media;numero_viagens;total_auferido

Caso o utilizador/condutor não tenha nenhumas viagens associadas, considerar a avaliacao_media, numero_viagens, e total_gasto como sendo 0.000, 0, e 0.000, respetivamente.

```
Comando
1 <ID>
Output
nome;genero;idade;avaliacao_media;numero_viagens;total_gasto
```

Implementação:

Pesquisar na hash table de condutores se o identificador conter apenas digitos.

Pesquisar na hash table de utilizadores em caso contrário.

Em ambos os casos é devolvida uma cópia do valor encontrado ou `NULL` se o utilizador/condutor não existir.

Análise de Complexidade:

$O(n)$

Querie 2

Descrição:

Q2: Listar os N condutores com maior avaliação média. Em caso de empate, o resultado deverá ser ordenado de forma a que os condutores com a viagem mais recente surjam primeiro. Caso haja novo empate, deverá ser usado o id do condutor para desempatar (por ordem crescente). O parâmetro N é representado por <N>.

```
Comando
2 <N>
Output
id;nome;avaliacao_media
id;nome;avaliacao_media
...
```

Overview:

Para responder à query 2, no pré-processamento dos dados é criado um array de pointers para os valores da hash table que contém os condutores, ordenado de

acordo com as especificações da query. Quando a query é invocada são devolvidos os primeiros n elementos deste array.

Implementação:

Foi criado o módulo *query2* que expõe os seguintes tipos e funções:

```
typedef GPtrArray *Query2;

Query2 query2_new(Drivers drivers);

void query2_free(Query2 q2);

Query2 query2_top_n_drivers(Query2 q2, int N);
```

A query 2 utiliza um `GPtrArray`, um array de pointers, que armazena os pointers dos condutores da hash table de condutores. A utilização de um array permite iterar mais eficientemente sobre os condutores e permite ordená-los o que não é possível numa hash table.

O array é inicializado com o tamanho do número de elementos da hash table pelo que não será necessário expandir o array nem é ocupado mais espaço do que o necessário.

Apenas são armazenados os pointers de forma a não duplicar dados desnecessariamente, poupando memória e evitando o impacto de performance de copiar os dados de cada condutor. Para conseguir obter estes pointers são utilizadas as funções da glib para aceder à hash table pois as funções que o módulo *drivers* desponibiliza apenas devolvem cópias dos condutores.

Depois de criar o array este é ordenado com a função da glib `g_ptr_array_sort` com a nossa função de comparação `_driver_comparator` para comparar condutores de acordo com as especificações da query.

Quando a query é chamada para N condutores, devolve um novo `GPtrArray` de tamanho N com cópias dos primeiros N elementos no `GPtrArray` original.

Quanto à função de ordenação, a documentação da glib afirma que:

> All the sort functions are internally calling a quick-sort (or similar) function with an average cost of O(n log(n)) and a worst case cost of O(n^2).

Uma análise da fonte do código da função mostra que esta realiza um merge-sort de forma a assegurar uma ordenação estável:

- garray.c:

```
void
g_ptr_array_sort (GPtrArray    *array,
                  GCompareFunc  compare_func)
{
```

```
  g_return_if_fail (array != NULL);

  /* Don't use qsort as we want a guaranteed stable sort */
  if (array->len > 0)
    g_qsort_with_data (array->pdata,
                       array->len,
                       sizeof (gpointer),
                       (GCompareDataFunc)compare_func,
                       NULL);
}
```

- gqsort.c:

```
void
g_qsort_with_data (gconstpointer    pbase,
                   gint             total_elems,
                   gsize            size,
                   GCompareDataFunc compare_func,
                   gpointer         user_data)
{
  msort_r ((gpointer)pbase, total_elems, size, compare_func, user_data);
}
```

Análise de Complexidade:

Complexidade no momento de pré-processamento, n corresponde ao número de condutores no input:

- Criação do array: $O(n)$
- Ordenação do array (merge-sort): $O(n\ log(n))$
- Total: $O(n + n\ log(n)) = O(n\ log(n))$

Complexidade no momento de resposta, n corresponde ao número de condutores passado como argumento na query:

- Criar array com cópias dos primeiros n elementos: $O(n)$

Considerações adicionais:

Ao obter os pointers dos utilizadores na hash table `drivers` através do uso de funções da glib, estámos a violar o encapsulamento do módulo *drivers*. Porém, consideramos esta violação justificável pois, para além das vantagens em performance já mencionadas, ao contrário de outros módulos, o módulo *query2* tem apenas o objetivo de responder a um problema do nosso projeto pelo que conhece o projeto, incluindo o módulo *drivers*, e não deve ser utilizado em qualquer outro contexto. Para além disso, o próprio módulo devolve apenas cópias dos condutores pelo o encapsulamento é mantido para quem usa o módulo *query2*, apenas não dentro do módulo *query2*.

Apesar da glib fornecer a função g_hash_table_get_values que devolve os valores de uma hash table numa lista duplamente ligada, optou-se por usar um

array pois o comprimento deste é conhecido e usa menos memória.

Querie 3

Descrição:

> Q3: Listar os N utilizadores com maior distância viajada. Em caso de empate, o resultado deverá ser ordenado de forma a que os utilizadores com a viagem mais recente surjam primeiro. Caso haja novo empate, deverá ser usado o username do utilizador para desempatar (por ordem crescente). O parâmetro N é representado por <N>.
>
> ```
> Comando
> 3 <N>
> Output
> username;nome;distancia_total
> username;nome;distancia_total
> ...
> ```

Overview:

A query 3 é funcionalmente igual à query 2, mas para utilizadores em vez de condutores.

Análise de correção

Uma análise mais formal da correção está ainda em desenvolvimento, mas os outputs das queries testadas coincidem com os outputs fornecidos pelos docentes, assim como todas estas queries passaram nos testes da plataforma online dos docentes.

Análise de desempenho

Uma análise mais formal do desempenho está ainda em desenvolvimento, mas todas as queries são realizadas em menos de 10 segundos (incluíndo o tempo de processamento dos dados) o que é considerado o máximo de tempo útil para uma query.

O programa não tem *memory leaks*, porém existem 18.804 bytes "still reachable" que surgem do uso da glib e que não podem ser resolvidos pelo grupo. Foi usado o programa *valgrind* para chegar a estes resultados, que se seguem:

```
==779848== Memcheck, a memory error detector
==779848== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==779848== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==779848== Command: ././programa-principal ./datasets/dataset_fase1 ./datasets/exemplos_quer
==779848== Parent PID: 779622
```

```
==779848==
==779848==
==779848== HEAP SUMMARY:
==779848==     in use at exit: 18,804 bytes in 9 blocks
==779848==   total heap usage: 80,920,985 allocs, 80,920,976 frees, 2,232,040,493 bytes allo
==779848==
==779848== 4 bytes in 1 blocks are still reachable in loss record 1 of 9
==779848==    at 0x4841888: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==779848==    by 0x492F353: g_private_impl_new (gthread-posix.c:1029)
==779848==    by 0x492F403: UnknownInlinedFun (gthread-posix.c:1057)
==779848==    by 0x492F403: g_private_get (gthread-posix.c:1084)
==779848==    by 0x48FFEF4: UnknownInlinedFun (gslice.c:554)
==779848==    by 0x48FFEF4: g_slice_alloc (gslice.c:1052)
==779848==    by 0x48CCEF2: g_hash_table_new_full (ghash.c:1073)
==779848==    by 0x48E8263: g_quark_init (gquark.c:63)
==779848==    by 0x48A70D8: UnknownInlinedFun (glib-init.c:341)
==779848==    by 0x48A70D8: UnknownInlinedFun (glib-init.c:330)
==779848==    by 0x48A70D8: glib_init_ctor (glib-init.c:455)
==779848==    by 0x4004CED: call_init (dl-init.c:70)
==779848==    by 0x4004CED: call_init (dl-init.c:26)
==779848==    by 0x4004DDB: _dl_init (dl-init.c:117)
==779848==    by 0x401B75F: ??? (in /usr/lib/ld-linux-x86-64.so.2)
==779848==    by 0x2: ???
==779848==    by 0x1FFEFFFEBE: ???
==779848==
==779848== 32 bytes in 1 blocks are still reachable in loss record 2 of 9
==779848==    at 0x4846A73: calloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==779848==    by 0x48DD681: g_malloc0 (gmem.c:163)
==779848==    by 0x48C9AE0: g_hash_table_setup_storage (ghash.c:593)
==779848==    by 0x48CCF2B: g_hash_table_new_full (ghash.c:1085)
==779848==    by 0x48E8263: g_quark_init (gquark.c:63)
==779848==    by 0x48A70D8: UnknownInlinedFun (glib-init.c:341)
==779848==    by 0x48A70D8: UnknownInlinedFun (glib-init.c:330)
==779848==    by 0x48A70D8: glib_init_ctor (glib-init.c:455)
==779848==    by 0x4004CED: call_init (dl-init.c:70)
==779848==    by 0x4004CED: call_init (dl-init.c:26)
==779848==    by 0x4004DDB: _dl_init (dl-init.c:117)
==779848==    by 0x401B75F: ??? (in /usr/lib/ld-linux-x86-64.so.2)
==779848==    by 0x2: ???
==779848==    by 0x1FFEFFFEBE: ???
==779848==    by 0x1FFEFFFED5: ???
==779848==
==779848== 32 bytes in 1 blocks are still reachable in loss record 3 of 9
==779848==    at 0x4846A73: calloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==779848==    by 0x48DD681: g_malloc0 (gmem.c:163)
==779848==    by 0x48C9AE0: g_hash_table_setup_storage (ghash.c:593)
```

```
==779848==    by 0x48CCF2B: g_hash_table_new_full (ghash.c:1085)
==779848==    by 0x48A70E2: UnknownInlinedFun (gerror.c:525)
==779848==    by 0x48A70E2: UnknownInlinedFun (glib-init.c:342)
==779848==    by 0x48A70E2: UnknownInlinedFun (glib-init.c:330)
==779848==    by 0x48A70E2: glib_init_ctor (glib-init.c:455)
==779848==    by 0x4004CED: call_init (dl-init.c:70)
==779848==    by 0x4004CED: call_init (dl-init.c:26)
==779848==    by 0x4004DDB: _dl_init (dl-init.c:117)
==779848==    by 0x401B75F: ??? (in /usr/lib/ld-linux-x86-64.so.2)
==779848==    by 0x2: ???
==779848==    by 0x1FFEFFFEBE: ???
==779848==    by 0x1FFEFFFED5: ???
==779848==    by 0x1FFEFFFEEE: ???
==779848==
==779848== 64 bytes in 1 blocks are still reachable in loss record 4 of 9
==779848==    at 0x4841798: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==779848==    by 0x48DE810: g_realloc (gmem.c:201)
==779848==    by 0x48C9ACA: UnknownInlinedFun (ghash.c:382)
==779848==    by 0x48C9ACA: g_hash_table_setup_storage (ghash.c:591)
==779848==    by 0x48CCF2B: g_hash_table_new_full (ghash.c:1085)
==779848==    by 0x48E8263: g_quark_init (gquark.c:63)
==779848==    by 0x48A70D8: UnknownInlinedFun (glib-init.c:341)
==779848==    by 0x48A70D8: UnknownInlinedFun (glib-init.c:330)
==779848==    by 0x48A70D8: glib_init_ctor (glib-init.c:455)
==779848==    by 0x4004CED: call_init (dl-init.c:70)
==779848==    by 0x4004CED: call_init (dl-init.c:26)
==779848==    by 0x4004DDB: _dl_init (dl-init.c:117)
==779848==    by 0x401B75F: ??? (in /usr/lib/ld-linux-x86-64.so.2)
==779848==    by 0x2: ???
==779848==    by 0x1FFEFFFEBE: ???
==779848==    by 0x1FFEFFFED5: ???
==779848==
==779848== 64 bytes in 1 blocks are still reachable in loss record 5 of 9
==779848==    at 0x4841798: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==779848==    by 0x48DE810: g_realloc (gmem.c:201)
==779848==    by 0x48C9ACA: UnknownInlinedFun (ghash.c:382)
==779848==    by 0x48C9ACA: g_hash_table_setup_storage (ghash.c:591)
==779848==    by 0x48CCF2B: g_hash_table_new_full (ghash.c:1085)
==779848==    by 0x48A70E2: UnknownInlinedFun (gerror.c:525)
==779848==    by 0x48A70E2: UnknownInlinedFun (glib-init.c:342)
==779848==    by 0x48A70E2: UnknownInlinedFun (glib-init.c:330)
==779848==    by 0x48A70E2: glib_init_ctor (glib-init.c:455)
==779848==    by 0x4004CED: call_init (dl-init.c:70)
==779848==    by 0x4004CED: call_init (dl-init.c:26)
==779848==    by 0x4004DDB: _dl_init (dl-init.c:117)
==779848==    by 0x401B75F: ??? (in /usr/lib/ld-linux-x86-64.so.2)
```

```
==779848==    by 0x2: ???
==779848==    by 0x1FFEFFFEBE: ???
==779848==    by 0x1FFEFFFED5: ???
==779848==    by 0x1FFEFFFEEE: ???
==779848==
==779848== 96 bytes in 1 blocks are still reachable in loss record 6 of 9
==779848==    at 0x4841888: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==779848==    by 0x48DD329: g_malloc (gmem.c:130)
==779848==    by 0x48FFF17: g_slice_alloc (gslice.c:1074)
==779848==    by 0x48CCEF2: g_hash_table_new_full (ghash.c:1073)
==779848==    by 0x48E8263: g_quark_init (gquark.c:63)
==779848==    by 0x48A70D8: UnknownInlinedFun (glib-init.c:341)
==779848==    by 0x48A70D8: UnknownInlinedFun (glib-init.c:330)
==779848==    by 0x48A70D8: glib_init_ctor (glib-init.c:455)
==779848==    by 0x4004CED: call_init (dl-init.c:70)
==779848==    by 0x4004CED: call_init (dl-init.c:26)
==779848==    by 0x4004DDB: _dl_init (dl-init.c:117)
==779848==    by 0x401B75F: ??? (in /usr/lib/ld-linux-x86-64.so.2)
==779848==    by 0x2: ???
==779848==    by 0x1FFEFFFEBE: ???
==779848==    by 0x1FFEFFFED5: ???
==779848==
==779848== 96 bytes in 1 blocks are still reachable in loss record 7 of 9
==779848==    at 0x4841888: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==779848==    by 0x48DD329: g_malloc (gmem.c:130)
==779848==    by 0x48FFF17: g_slice_alloc (gslice.c:1074)
==779848==    by 0x48CCEF2: g_hash_table_new_full (ghash.c:1073)
==779848==    by 0x48A70E2: UnknownInlinedFun (gerror.c:525)
==779848==    by 0x48A70E2: UnknownInlinedFun (glib-init.c:342)
==779848==    by 0x48A70E2: UnknownInlinedFun (glib-init.c:330)
==779848==    by 0x48A70E2: glib_init_ctor (glib-init.c:455)
==779848==    by 0x4004CED: call_init (dl-init.c:70)
==779848==    by 0x4004CED: call_init (dl-init.c:26)
==779848==    by 0x4004DDB: _dl_init (dl-init.c:117)
==779848==    by 0x401B75F: ??? (in /usr/lib/ld-linux-x86-64.so.2)
==779848==    by 0x2: ???
==779848==    by 0x1FFEFFFEBE: ???
==779848==    by 0x1FFEFFFED5: ???
==779848==    by 0x1FFEFFFEEE: ???
==779848==
==779848== 2,032 bytes in 1 blocks are still reachable in loss record 8 of 9
==779848==    at 0x4846A73: calloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==779848==    by 0x48DD681: g_malloc0 (gmem.c:163)
==779848==    by 0x490010E: UnknownInlinedFun (gthread.c:544)
==779848==    by 0x490010E: UnknownInlinedFun (gslice.c:566)
==779848==    by 0x490010E: UnknownInlinedFun (gslice.c:552)
```

```
==779848==    by 0x490010E: g_slice_alloc (gslice.c:1052)
==779848==    by 0x48CCEF2: g_hash_table_new_full (ghash.c:1073)
==779848==    by 0x48E8263: g_quark_init (gquark.c:63)
==779848==    by 0x48A70D8: UnknownInlinedFun (glib-init.c:341)
==779848==    by 0x48A70D8: UnknownInlinedFun (glib-init.c:330)
==779848==    by 0x48A70D8: glib_init_ctor (glib-init.c:455)
==779848==    by 0x4004CED: call_init (dl-init.c:70)
==779848==    by 0x4004CED: call_init (dl-init.c:26)
==779848==    by 0x4004DDB: _dl_init (dl-init.c:117)
==779848==    by 0x401B75F: ??? (in /usr/lib/ld-linux-x86-64.so.2)
==779848==    by 0x2: ???
==779848==    by 0x1FFEFFFEBE: ???
==779848==    by 0x1FFEFFFED5: ???
==779848==
==779848== 16,384 bytes in 1 blocks are still reachable in loss record 9 of 9
==779848==    at 0x4841888: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==779848==    by 0x48DD329: g_malloc (gmem.c:130)
==779848==    by 0x48E8275: g_quark_init (gquark.c:64)
==779848==    by 0x48A70D8: UnknownInlinedFun (glib-init.c:341)
==779848==    by 0x48A70D8: UnknownInlinedFun (glib-init.c:330)
==779848==    by 0x48A70D8: glib_init_ctor (glib-init.c:455)
==779848==    by 0x4004CED: call_init (dl-init.c:70)
==779848==    by 0x4004CED: call_init (dl-init.c:26)
==779848==    by 0x4004DDB: _dl_init (dl-init.c:117)
==779848==    by 0x401B75F: ??? (in /usr/lib/ld-linux-x86-64.so.2)
==779848==    by 0x2: ???
==779848==    by 0x1FFEFFFEBE: ???
==779848==    by 0x1FFEFFFED5: ???
==779848==    by 0x1FFEFFFEEE: ???
==779848==
==779848== LEAK SUMMARY:
==779848==    definitely lost: 0 bytes in 0 blocks
==779848==    indirectly lost: 0 bytes in 0 blocks
==779848==      possibly lost: 0 bytes in 0 blocks
==779848==    still reachable: 18,804 bytes in 9 blocks
==779848==         suppressed: 0 bytes in 0 blocks
==779848==
==779848== For lists of detected and suppressed errors, rerun with: -s
==779848== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Conclusão

De uma forma geral, o trabalho desenvolvido, neste último mês, tem satisfeito todos os requisitos presentes no enunciado, para esta primeira fase. Embora ainda seja necessário aprimorar alguns elementos formais do trabalho, a arquitetura deste deste programa foi concebida com o objetivo de ser o mais eficiente e rápida

possível, cumprindo as regras de encapsulamento e modularidade.