Trabalho prático 1 - algoritmos genéricos de ordenação em vetores

1) Informação geral

O trabalho prático 1 consiste na implementação de funções adicionais a incorporar na biblioteca de funções para manipulação de vetores em C (anteriormente fornecida) e desenvolver uma nova biblioteca para manipulação de vetores genéricos.

Este trabalho deverá ser feito de forma autónoma por cada grupo na aula prática 4/5 e completado fora das aulas até à data limite estabelecida. A consulta de informação nas diversas fontes disponíveis é aceitável. No entanto, o código submetido deverá ser apenas da autoria dos elementos do grupo e quaisquer cópias detetadas serão devidamente penalizadas. A incapacidade de explicar o código submetido por parte de algum elemento do grupo implicará também uma penalização.

O prazo-limite para submissão (através do Moodle) é o dia 25 de Março às 21:00.

2) Implementação do trabalho

O arquivo comprimido PROG2_1819_T1.zip contém os ficheiros necessários para a realização deste trabalho, nomeadamente:

- `vetor.h`: declarações das funções da biblioteca de vetores
- `vetor.c`: implementação das funções da biblioteca de vetores
- `jogos.h`: declarações das funções a implementar
- `jogos.c`: ficheiro onde deverão ser implementadas as funções pedidas
- `jogos-teste.c`: inclui o programa principal que invoca e realiza testes básicos às funções implementadas
- `stats.txt`: ficheiro de texto com informação sobre os jogos da liga inglesa de futebol das últimas três temporadas
- `errata.txt`: ficheiro de texto com informação sobre alguns jogos que no ficheiro original não têm informação correta

Nota importante: apenas deverá ser alterado o ficheiro jogos.c que será o único a ser considerado na submissão dos trabalhos.

a) biblioteca `vetor`

A estrutura de dados `vetor` é a base da biblioteca e tem a seguinte declaração:

```
typedef struct
{
  /** numero de elementos do vetor */
  int tamanho;

  /** capacidade do vetor */
  int capacidade;

  /** array de elementos armazenados */
  jogo* elementos;

} vetor;
```

Nesta estrutura são guardados: 1) número de elementos do vetor (`tamanho`); 2) capacidade do vetor (`capacidade`); e 3) o apontador para o *array* de elementos armazenados (`elementos`). A estrutura de dados `vetor` utiliza um *array* de elementos armazenados do tipo `jogo`.

```
typedef struct
{
  /** época a que o jogo se refere */
  char epoca[6];

  /** equipa que jogou em casa */
  char nome_casa[30];

  /** equipa que jogou fora */
  char nome_fora[30];

  /** golos marcados pela equipa da casa */
  int golos_casa;

  /** golos marcados pela equipa de fora */
  int golos_fora;

  /** cartões vermelhos da equipa da casa */
  int vermelhos_casa;

  /** cartões vermelhos da equipa de fora */
  int vermelhos_fora;
} jogo;
```

Para o cálculo de algumas estatísticas por equipa facilita ter uma nova estrutura de dados `vetor_equipas` que tem a seguinte declaração:

```
typedef struct
{
  /** número de elementos do vetor */
  int tamanho;

  /** capacidade do vetor */
  int capacidade;

  /** array de elementos armazenados */
  equipa* elementos;

} vetor_equipas;
```

Nesta estrutura são guardados: 1) número de elementos do vetor (`tamanho`); 2) capacidade do vetor (`capacidade`); e 3) o apontador para o *array* de elementos armazenados (`elementos`). A estrutura de dados `vetor_equipas` utiliza um *array* de elementos armazenados do tipo `equipa`.

```
typedef struct
{
  /** nome da equipa */
  char nome_equipa[30];
```

```
    /** total de diferença de golos (marcados – sofridos) nas três
    épocas */
    int diff_golos;

    /** média de cartões vermelhos por época */
    float vermelhos[3];

} equipa;
```

As funções a implementar (no ficheiro `jogos.c`) são:

1. **vetor *jogos_load**(const char *nomef);
   lê o conteúdo do ficheiro de texto de nome nomef que inclui informação de um conjunto de jogos para um vetor. Deve retornar apontador para o vetor criado. Em caso de erro retornar NULL.
   Exemplo do conteúdo do ficheiro: época (que pode ser 15/16, 16/17 ou 17/18), nome da equipa da casa, nome da equipa de fora, golos marcados pela equipa da casa, golos marcados pela equipa de fora, cartões vermelhos da equipa da casa, e cartões vermelhos da equipa de fora.

   ```
   15/16 Bournemouth Aston_Villa 0 1 0 0
   15/16 Chelsea Swansea 2 2 1 0
   15/16 Everton Watford 2 2 0 0
   ```

2. **int jogos_save**(vetor *vec, const char *nomef);
   cria/substitui um ficheiro de texto de nome nomef onde deve guardar a informação contida no vetor vec de acordo com a formatação descrita anteriormente. Deve retornar -1 se ocorrer algum erro ou o número de elementos guardados no ficheiro.

3. **vetor_equipas *stats_equipa** (vetor *vec);
   lê o conteúdo do vetor vec, calcula as estatísticas referidas por equipa e guarda em vetor do tipo vetor_equipas. Deve retornar apontador para o vetor_equipas criado. Em caso de erro, deve retornar NULL.
   Nota: cada equipa joga um total de 38 jogos por época (19 em casa e 19 fora).

4. **int equipas_ordena**(vetor_equipas *v, int criterio);
   *ordena equipas de forma crescente, de acordo com o critério especificado. Se critério=0, ordenar equipas por ordem alfabética; se critério=1, ordenar equipas por diferença de golos. **A ordenação deve ser feita recorrendo ao algoritmo quicksort**. Deve retornar -1 se ocorrer algum erro ou 0 se for bem sucedido.*

5. **int corrige_jogo**(vetor *vec, const char *nomef);
   corrige a informação guardado no vetor vec (referente a resultado e número de cartões vermelhos), tendo em conta a nova informação fornecida pelo ficheiro de texto nomef. Deve retornar -1 se ocorrer algum erro ou 0 se for bem sucedido.
   Um famoso hacker português alterou remotamente a informação relativa a alguns dos jogos presentes no ficheiro "`stats.txt`". Um novo ficheiro com a informação correta referente em exclusivo aos jogos que foram alterados é dado - "`errata.txt`".
   Nota: de forma a saber qual o jogo a que a informação se refere há que comparar os três primeiros campos: época, equipa da casa e equipa de fora.

6. **int pontos_de_equipa**(vetor *vec, char *nome_equipa, char *epoca);
   calcula o número de pontos de uma determinada equipa numa determinada época - vitória vale 3 pontos, empate 1 ponto e derrota 0 pontos. Deve retornar -1 se ocorrer algum erro ou o número de pontos da equipa na época.

3) Teste da biblioteca de funções

A biblioteca pode ser testada executando o programa *jogos-teste*. Existe um teste por cada função a implementar e que determina se essa função tem o comportamento esperado. Note que os testes não são exaustivos. Por isso, os testes devem ser considerados apenas como um indicador de uma aparente correta implementação das funcionalidades esperadas.

Inicialmente o programa *jogos-teste* quando executado apresentará o seguinte resultado:

```
jogos_load():
       erro na leitura do ficheiro './stats.txt'
FOI ENCONTRADO UM TOTAL DE 1 ERROS.
```

Depois de todas as funções corretamente implementadas, o resultado do programa apresentará o seguinte resultado:

```
jogos_load(): Carregados 1140 jogos. OK
jogos_save(): OK
stats_equipa(): Foram calculadas estatísticas para 25 equipas. OK
equipas_ordena():
       Elemento inicial quando ordenado por ordem alfabética: Arsenal
       Elemento inicial quando ordenado por diferença de golos: Stoke, com
diferença de golos de -102
       Tempo de execucao ordenacao por ordem alfabética (s): 0.000008
       Tempo de execucao ordenacao por diferença de golos (s): 0.000005
       OK
corrige_jogo(): OK
pontos_de_equipa(): O Man_City conquistou 100 pontos na época 17/18. OK
FIM DE TODOS OS TESTES.
```

(*os tempos de execução podem variar)

5) Ferramenta de desenvolvimento

A utilização de um IDE ou do Visual Studio Code é aconselhável no desenvolvimento deste trabalho uma vez que permite fazer depuração de uma forma mais eficaz. Poderá encontrar informações sobre a utilização do Visual Studio Code num breve tutorial disponibilizado no Moodle.

6) Avaliação

A classificação do trabalho é dada pela avaliação feita à implementação submetida pelos estudantes, mas também pelo desempenho dos estudantes na aula dedicada a este trabalho. A classificação final do trabalho (T1) é dada por:

T1 = 0.8 Implementação + 0.2 Desempenho

A classificação da implementação é essencialmente determinada por testes automáticos adicionais (por exemplo, recorrendo a ficheiros de teste de maiores dimensões). No caso de a implementação submetida não compilar, esta componente será 0%.

O desempenho será avaliado durante a aula e está dependente da entrega do formulário "Preparação do trabalho" que se encontra disponível no Moodle. A classificação de desempenho poderá ser diferente para cada elemento do grupo.

7) Submissão da resolução

A submissão é apenas possível através do Moodle e até à data indicada no início do documento. Deverá ser submetido um ficheiro *zip* contendo:

- o ficheiro jogos.c com as funções implementadas
- um ficheiro autores.txt indicando o nome e número dos elementos do grupo

Nota importante: apenas as submissões com o seguinte nome serão aceites: `T1_G<numero_do_grupo>.zip`. Por exemplo, `T1_G999.zip`