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 6 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 30 de Março às 21:00.

2) Implementação do trabalho

O arquivo comprimido PROG2_1617_T1.zip contém os ficheiros necessários para a realização deste trabalho, nomeadamente:

- `vetor.h`: inclui as declarações das funções a implementar - **não deve ser alterado**
- `vetor.c`: ficheiro onde deverão ser implementadas as funções da biblioteca
- `vetorg.h`: inclui as declarações das funções a implementar - **não deve ser alterado**
- `vetorg.c`: ficheiro onde deverão ser implementadas as funções da biblioteca
- `vetor-teste.c`: inclui os testes feitos à biblioteca - **não deve ser alterado**

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 */
  v_elemento* elementos;

} vetor;
```

Nesta estrutura são guardados: 1) numero 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 `v_elemento` que contém uma `string`.

O ficheiro `vetor.h` contém informação adicional sobre cada uma das funções a implementar (funcionalidade, parâmetros e valor de retorno).

As funções a implementar (no ficheiro `vetor.c`) e que estão associadas à estrutura de dados `vetor` são:

1. **int vetor_ordena_sel**(vetor *vtr);
   ordena um vetor por ordem alfabética recorrendo ao algoritmo selection sort
2. **int vetor_ordena_qsort**(vetor *vtr);
   ordena um vetor por ordem alfabética recorrendo ao algoritmo quicksort
3. **int vetor_ordena**(vetor *vtr);
   ordena um vetor por ordem alfabética recorrendo a um algoritmo quicksort "melhorado"

De modo a testar estas funções terá de começar por implementar (no ficheiro `vetor.c`) a função `le_ficheiro`:

4. **vetor *le_ficheiro**(char *nome)**;**
   lê o conteúdo do ficheiro de texto com nome para um vetor

b) biblioteca `vetorg`

A estrutura de dados `vetorg` utiliza um *array* de elementos armazenados do tipo `v_elemento`, que permite armazenar um `int`, um `float`, um `int` e uma `string` como apresentado abaixo.

```
typedef struct
{
  /** numero de votos */
  int votos;

  /** classificação do filme */
  float classifica;

  /** ano do filme */
  int ano;

  /** nome do filme */
  char* titulo;

} vg_elemento;
```

Nesta estrutura é guardada informação relativa a um filme: 1) o número de votos que o filme recebeu (`int`), 2) a classificação atual do filme (`float`), 3) o ano (`int`) e 4) o título do filme (`string`).

As funções a implementar (no ficheiro `vetorg.c`) e que estão associadas à estrutura de dados `vetorg` são:

5. int **vetorg_insere**(vetorg* vec, int votos, float classifica, int ano, const char* titulo, int pos);
   coloca informação sobre relativa a um filme num elemento do tipo v_elemento e insere num vetorg
6. **int vetorg_ordena**(vetorg *vtr, comp ordem);
   ordena um vetor por ordem alfabética recorrendo ao algoritmo quicksort "melhorado" utilizando o critério de ordem especificado

O critério de ordem a utilizar pela função de ordenação é uma função de comparação específica para os diferentes campos do registo `vg_elemento`, e é passado à função de comparação usando um **apontador para função**.

```
/**
* tipo "comp" representa um apontador para uma função de ordem.
* uma função que implemente esta assinatura deverá retornar:
*  -1 se a < b
*  0 se a == b
*  1 se a > b
*/

typedef int (*comp)(const vg_elemento a, const vg_elemento b);
```

A função de comparação que se segue exemplifica a comparação de `vg_elemento` pelo campo `ano` de forma ascendente:

```
int comp_ano_asc(const vg_elemento a, const vg_elemento b){

    if(b.ano > a.ano)
        return -1;
    if(a.ano > b.ano)
        return 1;
    return 0;
}
```

As restantes funções de comparação a implementar devem respeitar os seguintes protótipos:

```
/* comparação do campo votos de forma ascendente */
    int comp_votos_asc(const vg_elemento a, const vg_elemento b);

/* comparação do campo classifica de forma ascendente */
    int comp_classifica_asc(const vg_elemento a, const vg_elemento b);

/* comparação do campo título de forma ascendente */
    int comp_titulo_asc(const vg_elemento a, const vg_elemento b);
```

De modo a testar estas funções terá de começar por implementar (no ficheiro `vetorg.c`) a função `le_ficheiro_g`

7. **vetorg *le_ficheiro_g**(char *nome)**;**
   *lê o conteúdo do ficheiro de texto com **nome** para um vetor*

O ficheiro `vetorg.h` contém informação adicional sobre cada uma das funções a implementar (funcionalidade, parâmetros e valor de retorno).

3) Teste da biblioteca de funções

A biblioteca pode ser testada executando o programa *vetor-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 *vetor-teste* quando executado apresentará o seguinte resultado:

```
TESTES VETOR.
le_ficheiro():
	erro na leitura do ficheiro './plantas.txt'
FIM DE TESTES VETOR.

TESTES VETOR GENERICO.
vetorg_insere():
	erro na inserção no vetor genérico

le_ficheiro_g():
	erro na leitura do ficheiro './IMDB.txt'
FIM DE TESTES VETOR GENERICO.
FOI ENCONTRADO UM TOTAL DE 3 ERROS.
```

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

```
TESTES VETOR.
le_ficheiro(): OK
vetor_ordena_sel():
	Tempo de execucao (s): 78.458733
OK
vetor_ordena_qsort():
	Tempo de execucao (s): 0.025678
OK
vetor_ordena():
	Tempo de execucao (s): 0.022172
OK
FIM DE TESTES VETOR.

TESTES VETOR GENERICO.
vetorg_insere(): OK
le_ficheiro_g(): OK
vetorg_ordena() (por título):
	Tempo de execucao (s): 0.029842
OK
FIM DE TESTES VETOR GENERICO.
FIM DE TODOS OS TESTES.
```

(*os tempos de execução podem variar)

5) Ferramenta de desenvolvimento

A utilização de um IDE, por exemplo o Eclipse, é aconselhável no desenvolvimento deste trabalho. Para além gerir o processo de compilação, o IDE permite fazer depuração de uma forma mais eficaz. Poderá encontrar informações sobre a utilização do Eclipse 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:

- os ficheiros vetor.c e vetorg.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`