Trabalho prático 1 - pesquisa e ordenação de dados

1) Informação geral

O trabalho prático 1 consiste na implementação de funções adicionais a incorporar numa biblioteca de funções para manipulação de vetores em C que guardam informação sobre cidades. A biblioteca fornecida contém um conjunto de funções já implementadas (adaptadas da biblioteca usada nas aulas práticas).

Este trabalho deverá ser feito de forma autónoma por cada grupo na aula prática 4 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 12 de março às 21:00.

2) Implementação do trabalho

O arquivo comprimido PROG2_1718_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
- `cidades.h`: declarações das funções a implementar
- `cidades.c`: ficheiro onde deverão ser implementadas as funções pedidas
- `cidades-teste.c`: inclui o programa principal que invoca e realiza testes básicos às funções implementadas
- `cidades.bin`: ficheiro binário com informação sobre cidades, estando ordenado crescentemente por ordem alfabética do nome da cidade

Note-se que apenas o ficheiro cidades.c deve ser alterado, todos os restantes devem manter-se inalterados.

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 */
  cidade* 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 `cidade`.

```
typedef struct
{
  char nome[50];
  char pais[50];
  int populacao;
  int area;
} cidade;
```

As funções a implementar (no ficheiro `cidades.c`) são[1]:

1. **vetor *cidades_load** (const char *nomef)**;**
   lê o conteúdo do ficheiro binário de nome nomef para um vetor
2. **int cidades_save** (const vetor *vec, const char *nomef);
   coloca num ficheiro binario de nome nomef a informação contida no vetor
3. **int cidades_resort** (vetor *vec, char criterio);
   ordena o vetor crescentemente, de acordo com criterio especificado
4. **int cidades_peek** (const char *nomef, const char *nomecidade, cidade *resultado);
   pesquisa o elemento (cidade) com nome nomecidade diretamente no ficheiro
5. **int cidades_poke** (const char *nomef, const char *nomecidade, cidade nova);
   altera o valor do elemento (cidade) com nome nomecidade diretamente no ficheiro
6. **char **cidades_similar** (vetor *vec, const char *nomecidade, int deltapop, int *nsimilares);
   procura as cidades similares em população a uma cidade especificada

O ficheiro `cidades.h` contém informação adicional sobre cada uma das funções a implementar (funcionalidade, parâmetros e valor de retorno). Sempre que necessário, podem ser criadas funções auxiliares (apenas no ficheiro `cidades.c`). Na implementação de cada função devem ser testados casos de erro, como parâmetros inválidos ou NULL.

3) Teste da biblioteca de funções

A biblioteca pode ser testada executando o programa *cidades-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 e não testam, por exemplo, o uso de parâmetros inválidos ou fugas de memória. Por isso, os testes devem ser considerados apenas como um indicador de uma aparente correta implementação das funcionalidades esperadas.

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

```
cidades_load():
      erro na leitura do ficheiro './cidades.bin'
FOI ENCONTRADO UM TOTAL DE 1 ERROS.
```

[1] Nota histórica: os nomes peek e poke fazem referência a comandos muito utilizados noutras linguagens de programação, como o BASIC. Eram especialmente famosas entre os jogadores de ZX Spectrum nos anos 80: https://en.wikipedia.org/wiki/PEEK_and_POKE

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

```
cidades_load(): OK
cidades_save(): OK
cidades_resort():
       Tempo de execucao ordenacao por pais (s): 0.001795
       Tempo de execucao ordenacao por area (s): 0.001526
       OK
cidades_peek():
       Tempo de execucao (s): 0.000157
       OK
cidades_poke():
       Tempo de execucao (s): 0.000282
       OK
cidades_similar(): OK
FIM DE TODOS OS TESTES.
```

(*os tempos de execução podem variar)

4) 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.

5) 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.

6) 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 cidades.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`