Trabalho prático 1 – implementação de biblioteca de listas ligadas circulares

1) Informação geral

O trabalho prático 1 consiste na implementação de alterações e funções adicionais a incorporar na biblioteca de funções para manipulação de listas circulares.

Este trabalho deverá ser feito de forma autónoma por cada grupo 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 **31 de Março às 21:00**.

2) Conceito

Pretende-se que implemente um sequenciador de música. Um sequenciador é uma espécie de instrumento musical em que notas ou sons são tocados consoante uma sequência temporal. A cada instante corresponde uma célula numa grelha. Esta grelha funciona em *loop*, i.e., em ciclo, voltando ao início depois de executar a última célula. Com o desenvolvimento da biblioteca de listas circulares, espera-se que gira este sequenciador.

3) Implementação do trabalho

O arquivo comprimido PROG2_1920_T1.zip contém os ficheiros necessários para a realização deste trabalho, nomeadamente:

- `musica.h`: declarações das funções da biblioteca de lista circular e dos elementos instante
- `musica.c`: ficheiro onde deverão ser implementadas as funções pedidas, relativas à biblioteca `musica.h`
- `musica-teste.c`: inclui o programa principal que invoca e realiza testes básicos às funções implementadas
- `loop_A.txt`: ficheiro de texto com informação sobre a canção e notas a serem importadas para o sequenciador
- `variante_B.txt`: ficheiro de texto com informação sobre as alterações a fazer no sequenciador

Notas importantes:

1) **Apenas deverá ser alterado o ficheiro musica.c que será o único a ser considerado na submissão dos trabalhos.**

2) **Detalhes adicionais sobre as funções (a implementar) poderão ser encontradas junto ao protótipo respetivo em musica.h**

a) biblioteca `musica`

Os registos de dados `instante` e `musica` são a base da biblioteca e têm a seguinte declaração:

```
typedef struct
{
  /** extensão de tempo em que as notas deste "instante" tocam:
valores de 1-16*/
  int duracao;

  /** notas a tocar na guitarra: letras de A-G, notas das
oitavas 2 e 3 nas posições 0 e 1, respetivamente, no máximo uma
por oitava. E.g.: se C2 E3 array teria "CE", se só E3 teria "-
E". */
  char notasGuitarra[2];

  /** notas a tocar no piano: letras de A-G, notas das oitavas 4
e 5 nas posições 0 e 1, respetivamente, no máximo uma por
oitava. E.g.: se C4 E5 array teria "CE", se só E5 teria "-E". */
  char notasPiano[2];

  /** indicação individual de cada instrumento da bateria sobre
se toca ou não (valor logico por elemento do array): exemplo {1,
0, 1, 0, 1, 1, 0, 0} */
  int bateria[8];

  /** ficheiro de som extra a tocar neste "instante" */
  char *instrumentoExtra;

  /** apontador para o "instante" seguinte */
  struct instante *_proximo;

} instante;
```

Neste registo *instante* são guardados: 1) a duração (`duracao`) das notas tocadas neste *instante*; 2) as notas baixa e alta do instrumento guitarra (`notasGuitarra[0]` e `notasGuitarra[1]`, respetivamente); 3) as notas baixa e alta do instrumento piano (`notasPiano[0]` e `notasPiano[1]`, respetivamente); 4) um *array* lógico que indica se o instrumento de percussão respetivo a cada posição vai tocar; 5) uma *string* com o nome do ficheiro extra a reproduzir; 6) um apontador para o *instante* seguinte.

O registo de dados `musica` aponta para o primeiro elemento da lista ligada circular cujo elemento base é do tipo `instante`. Naturalmente, terá de ter em conta que o último elemento adicionado à música deverá apontar para o primeiro de forma ao ciclo de reprodução funcionar como pretendido.

```
typedef struct
{
  /** apontador para o primeiro instante de todos armazenados */
  instante *inicio;

  /** apontador para o instante atualmente em reprodução ou para
uso genérico se não a tocar */
  instante *cursor;
} musica;
```

As funções a implementar neste trabalho encontram-se no ficheiro `musica.c` e são:

1. **instante *instante_novo**(int duracao, char *notas, int bateria[], char *instrumentoExtra);
   Cria uma nova instância do registo instante, copiando cada um dos argumentos para o respetivo elemento. Notas, se vazias, devem ser inicializadas a "--" ou, se só parcialmente vazias, a "X-" ou "-X". Deve retornar um apontador para um registo do tipo instante contendo os valores recebidos. Em caso de erro retornar NULL.
2. **musica *musica_nova**();
   Cria uma nova instância vazia do registo musica. Deve retornar um apontador para o registo. Em caso de erro deverá retornar NULL.
3. **int instante_insere**(musica *m, instante *t, int pos);
   Insere o instante t na lista circular na posição pos. Caso o valor de pos seja -1 deverá inserir t na última posição da musica (append). Caso pos seja maior que tamanho, menor que -1, ou outro caso de erro deverá retornar o código de erro -1. Em caso de sucesso deverá retornar 0.
4. **int musica_tamanho**(musica *m);
   Retorna o número de instantes armazenados no registo musica. Deve retornar um inteiro.
5. **instante *musica_pesquisa_nota**(musica *m, char nota, int instrumento);
   Retorna o apontador para o primeiro instante que apresentar determinada nota (A-G) no instrumento indicado, seja na oitava maior ou menor. Se o instrumento for 0 refere-se a guitarra, se 1 refere-se a piano. Deverá retornar o apontador para o instante encontrado. Em caso de insucesso deverá retornar NULL.
6. **instante *instante_remove**(musica *m, int pos);
   Remove o instante presente na posição pos da musica e reconecta a lista no ponto retirado. Deverá retornar o instante removido. Todos os apontadores que anteriormente apontavam para o instante removido deverão passar a contemplar o seguinte. Em caso de insucesso deverá retornar NULL.
7. **int instante_apaga**(instante *t);
   Elimina as posições de memória alocadas em t. Em caso de sucesso deverá retornar o valor zero ou -1 se insucesso.
8. **int musica_toca**(musica *m, int duracao);
   Avança o cursor de reprodução para o instante `duracao` *tempos à frente. Se ficar a meio de um instante o cursor deverá apontar para o instante seguinte. Recorde o objetivo desta biblioteca. Retorna 0 se bem-sucedido ou -1 em contrário.*
9. **int musica_apaga** (musica *m);
   Altera o conteúdo de m de forma a remover e eliminar todos os instantes nele guardados. A função deverá retornar 0 se bem-sucedida e -1 em contrário.
10. **int musica_corrige**(musica *m, char *ficheiro);
    Altera o conteúdo de m consoante as operações presentes no ficheiro apontado pelo argumento "ficheiro". Use o ficheiro disponibilizado variante_b.txt para testar a sua implementação. O ficheiro poderá ter operações não conformes, portanto, a função deverá retornar a quantidade de operações bem executadas. Não se deverão executar as operações com um formato não previsto. Pode encontrar instruções mais detalhadas no ficheiro da biblioteca.

Nota: Os ficheiros de entrada e de teste em que serão avaliados os programas submetidos poderão apresentar conteúdo diferente e incluir casos limite: argumentos de funções com gamas não previstas. Como tal, é sua responsabilidade garantir que os argumentos são devidamente testados de forma a só aceitar quando dentro da gama prevista.

3) Teste da biblioteca de funções

A biblioteca pode ser testada executando o programa `musica-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.

Estando as funções corretamente implementadas, o programa `musica-teste` quando executado deverá apresentar o seguinte resultado:

```
INICIO DOS TESTES: Boa sorte

...verifica_instante_novo: instante novo nao e' NULL (ok)
...verifica_instante_novo: bateria coincide com o esperado (= 10101010) (ok)
...verifica_instante_novo: Instrumento extra coincide com o esperado (= cv.mp3) (ok)
...verifica_instante_novo: duracao coincide com o esperado (= 4) (ok)
...verifica_instante_novo: notas da Guitarra coincide com o esperado (= AB) (ok)
...verifica_instante_novo: notas da Piano coincide com o esperado (= DG) (ok)
OK: verifica_instante_novo passou

(...)


Musica importada com sucesso: 47 instantes
...verifica_musica_toca (Teste de fim de instante): cursor coincide com o esperado
(ok)
...verifica_musica_toca (Teste de meio de instante): cursor coincide com o esperado
(ok)
OK: verifica_musica_toca passou

...verifica_musica_corrige: nu'mero de correcoes coincide com o esperado (= 10) (ok)
OK: verifica_musica_corrige passou

FIM DOS TESTES: Todos os testes passaram
```

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. A classificação final do trabalho (T1) é dada por:

T1 = 0.8 Implementação + 0.2 Memória

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%.

A gestão de memória também será avaliada, sendo considerados 3 patamares: 100% nenhum memory leak, 50% alguns memory leaks mas pouco significativos, 0% muitos memory leaks.

7) Teste em servidor

Em breve será disponibilizado um servidor para que possam testar o vosso código durante o desenvolvimento. O código submetido neste servidor **NÃO SERÁ AVALIADO**. Apenas a submissão via moodle é válida para efeitos de avaliação.

8) 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 musica.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`