Instituto Superior de Engenharia de Lisboa

LEIRT– LEETC

Programação II

2021/22 – 2.º semestre letivo

Segunda Série de Exercícios

Esta série de exercícios consiste na implementação de um programa para consulta interativa de dados, relativos a faixas de áudio, com os campos: *Title*; *Artist*; *Album*; *Year*; *Comment*; *Track*; *Genre*. O programa permitirá pesquisa de títulos e listagens ordenadas com diferentes critérios aplicados a alguns dos seus campos de informação.

Os dados estão disponíveis num ficheiro de texto com a tabela em formato CSV. Foram obtidos por leitura da informação de *tag* de um conjunto de ficheiros de áudio com formato MP3. Cada linha da tabela contém a *tag* de uma faixa de áudio, com os campos separados por ponto-e-vírgula.

Para melhor compreensão da informação de *tag* dos ficheiros MP3, recomenda-se a consulta das normas *MPEG Audio Tag* ID3v1 e ID3v1.1 (http://mpgedit.org/mpgedit/mpeg_format/mpeghdr.htm).

O programa recebe, como argumento de linha de comando, o nome de um ficheiro de texto contendo a tabela de dados. Na sua atividade há duas fases distintas – preparação e comandos. Na fase de preparação, lê a tabela e preenche a estrutura de dados. Na fase de comandos, apresenta, ciclicamente, um *prompt* e espera, de *standard input*, um dos comandos seguintes.

- `a` – (*artists*) apresenta a lista de faixas áudio ordenada por *Artist*. Se houver várias faixas do mesmo intérprete, aplica, sucessivamente, a ordem por *Album* e *Title*;
- `t` – (*titles*) apresenta a lista de faixas áudio ordenada por *Title*. Se houver várias faixas com título idêntico, aplica, sucessivamente, a ordem por *Artist* e *Album*;
- `s` *`title`* – (*search*) apresenta a faixa áudio com o título indicado pelo parâmetro. Se o título não existir, apresenta uma mensagem de erro;
- `q` – (*quit*) termina.

As listas produzidas pelos comandos `a` e `t`, em *standard output*, são por ordem alfabética crescente. Cada linha de texto em *standard output*, bem como a linha de resposta ao comando `s`, exibe a informação de uma faixa áudio, contendo os campos da *tag* separados por ponto-e-vírgula.

1. Organize a estrutura de dados, considerando os três tipos de estrutura `MP3Tag_t`, `TagArr_t` e `TagRef_t` seguintes:

   O tipo `MP3Tag_t` representa a informação de uma *tag*; os campos destinados a *strings* têm dimensão fixa, de acordo com o formato das *tags* nos ficheiros MP3. As respetivas macros de dimensão devem ser definidas pelos alunos, com o espaço adequado para os conteúdos da *tag*, conforme a especificação referida.

```
typedef struct{
        char title[MAX_TIT + 1];
        char artist[MAX_ART + 1];
        char album[MAX_ALB + 1];
        short year;
        char comment[MAX_COM + 1];
        char track;
        char genre;
} MP3Tag_t;
```

O tipo `TagArr_t` representa o descritor de um *array* de *tags*; é a estrutura principal destinada a armazenar as *tags* dos ficheiros MP3. Por simplificação, assume-se que a quantidade de *tags* a armazenar tem um máximo conhecido. Os alunos devem definir a macro `MAX_TAGS` de acordo com a amostra de dados disponível para ensaiar o programa.

```
typedef struct{
        int count; // Quantidade de elementos preenchidos no campo tags.
        MP3Tag_t tags[MAX_TAGS]; // Array de tags.
} TagArr_t;
```

O tipo `TagRef_t` representa o descritor de um *array* de ponteiros para *tag*; é uma estrutura auxiliar que contém referências para as *tags* armazenadas no *array* principal. Destina-se a facilitar o acesso às mesmas a partir deum critério de ordenação diferente.

```
typedef struct{
        int count; // Quantidade de elementos preenchidos no campo refs.
        MP3Tag_t *refs[MAX_TAGS]; // Array de ponteiros para tag.
} TagRef_t;
```

Para responder eficientemente aos comandos, é necessário dispor de acessos ordenados aos dados das *tags* armazenados no *array* principal, usando dois critérios: um para o comando `a`; outro para os comandos `t` e `s`. Seria possível ordenar o *array* principal cada vez que um comando fosse executado, aplicando o respetivo critério. No entanto, isso não é recomendável porque implicaria a espera por tempo de processamento e o desperdício de trabalho já realizado, por exemplo se executar alternadamente os dois comandos `a` e `t`.

A estratégia proposta, para realizar as duas ordenações apenas uma vez e manter os seus resultados, consiste em ordenar o *array* principal (`TagArr_t`) segundo um dos critérios e mantê-lo nessa forma, resolvendo o outro critério através de um segundo *array*, este de ponteiros (`TagRef_t`), designado por auxiliar.

Inicialmente, ordena-se o *array* principal; em seguida, prepara-se o *array* auxiliar a partir do principal, referenciando os mesmos elementos; finalmente, ordena-se o *array* de ponteiros auxiliar com o segundo critério; a partir daí, o acesso através de cada um dos *arrays* obtém os dados pela ordem respetiva.

A preparação do *array* auxiliar e as ordenações devem ser realizadas na função `setupEnd` especificada adiante.

2. Planeie a organização do programa através de grupos de funções, caraterizando a respetiva funcionalidade e definindo as interações entre eles. Propõe-se que considere, pelo menos, os grupos seguintes.

2.1. Gestão dos dados; deve conter, pelo menos, as funções:

```
void tagArrInit( TagArr_t *data );
```

Esta função inicia o descritor de um *array* de *tags* no estado inicial, vazio, pelo que regista o valor 0 na quantidade de elementos preenchidos.

```
int tagArrAdd( TagArr_t *data, MP3Tag_t *tag );
```

Esta função adiciona ao descritor, indicado por `data`, uma *tag*, indicada por `tag`.

Retorna um código de resultado: 0, em caso de sucesso; -1, se falhar, nomeadamente devido a espaço insuficiente no *array*.

```
void setupEnd( TagArr_t *data, TagRef_t *ref );
```

Esta função marca como concluída a fase de inserção das *tags*. Prepara os descritores dos dois *arrays* (de *tags* e de referências), para dar as respostas pretendidas na fase de comandos.

As ordenações devem ser realizadas com a função `qsort` da biblioteca normalizada.

Para uniformização das soluções, pretende-se que o *array* principal seja ordenado pelo critério do comando `a` e o *array* auxiliar pelo critério do comando `t`.

```
void command( TagArr_t *data, TagRef_t *ref, char *cmdLine );
```

Esta função apresenta, em *standard output*, os resultados do comando inserido pelo utilizador. A linha de comando é passada à função através do parâmetro `cmdLine`.

Para os comados `a` e `t` deve utilizar, respetivamente, o *array* principal e o *array* auxiliar de referências. Para o comando `s` deve utilizar a função `bsearch`, da biblioteca normalizada, sobre o *array* auxiliar de referências.

2.2. Processamento da lista de ficheiros MP3; disponibiliza, pelo menos, a função:

```
int tableRead( char *tableName, TagArr_t *data );
```

Esta função destina-se preencher o *array* de *tags* indicado por `data`. Deve percorrer o ficheiro de texto com nome indicado por `tableName`, ler os dados de cada *tag*, e adicioná-los ao *array* de *tags*, usando a função `tagArrAdd`.

No caso de haver carateres de espaço nas extremidades dos campos, estes espaços devem ser eliminados. Se houver *tags* em que um dos campos *Title* e *Artist* seja *string* vazia, a respetiva linha da tabela deve ser ignorada, não ficando registada para qualquer processamento.

Esta função retorna 0, em caso de sucesso, ou -1, em caso de falha na leitura da tabela.

2.3. Aplicação, responsável por gerir a obtenção e armazenamento dos dados, bem como a interação com o utilizador; invoca as funções dos módulos anteriores para:

- Iniciar o funcionamento da gestão dos dados;
- Percorrer a tabela recebida por parâmetro de linha de comando, obter os dados de cada *tag* e armazená-los no *array* de dados principal;
- Acionar a passagem da fase de inserção à fase de comandos, criando os acessos ordenados;
- Aceitar os comandos do utilizador e promover as respostas correspondentes.

3. Desenvolva o programa especificado.

Recomenda-se o desenvolvimento faseado e o teste exaustivo cada função desenvolvida.

Nesta série não é exigida a programação em módulos separados. No entanto, os alunos que já tiverem domínio desta metodologia poderão utilizá-la; neste caso, devem definir os *header files* (.h) adequados para partilhar os tipos de dados e as assinaturas das funções públicas.

Em anexo é disponibilizada uma amostra de tabela, gerada a partir das *tags* de um conjunto de ficheiros MP3, para utilizar no desenvolvimento e demonstração.

ISEL, 2022.04.27

(rev. 2022.05.04)