Trabalho Prático II - TEP - 2021/1

Acervo de Livros

Prof. Vinícius Mota

DI/UFES

Data de entrega: 03/10/2021 Locais de entrega: - submetido pelo Repl.it; (convidar o @AbrraaoAbe para o projeto) - E também pelo AVA.

1. Objetivos

Este segundo trabalho visa o desenvolvimento das habilidades de manipulação de funções genéricas, ponteiro para funções e estruturas encadeadas. Além disso, a organização e documentação correta de código, e gerenciamento de memória.

Para isto, o objetivo deste trabalho é o desenvolvimento de um aplicativo para controle de estoque de livros para uma livraria.

Considere um sistema de livraria simples no terminal, capaz de realizar cadastro de livros, carregar um arquivo de livros em memória, buscar livros por (título do livro, nome do autor, identificador - ISBN) e exclusão de um livro da lista. As funcionalidades e especificações do sistema (quanto a formato de arquivos, como compilar, etc.) seguem nesse arquivo, mas estão sujeitas a mudanças:

2. Descrição do sistema

O objetivo deste trabalho é implementar um aplicativo de estoque de livros. Para isto, o usuário do programa deve cadastrar livros, pesquisar os livros por autor, título, nota ou isbn, excluir o livro da lista.

Será fornecido 1 arquivo `.csv` contendo informações sobre os livros. O programa deve ler deste arquivo e armazenar as informações em uma lista encadeada.

O usuário deve ser capaz de visualizar a lista de livros gerada por uma busca especifica e selecionar (ou não) o livro para exclusão. Deve ser feito o tratamento da entrada do usuário, para caso o usuário digite alguma opção inválida o programa saiba lidar com isso.

IMPORTANTE:

- O código deve ser devidamente separado em bibliotecas com os devidos cabeçalhos e implementações, sempre levando em consideração as boas práticas.
- É responsabilidade do desenvolvedor garantir que o programa faça uso eficiente da memória.
- É seu dever como desenvolvedor filtrar erros de entrada dos usuários. Mesmo que não estejam listados aqui. Sempre comente essas verificações.
- A compilação do executável deve ser por meio de um Makefile.
- Código deve estar bem comentado.
- É opcional "limpar o terminal" entre as passagens do menu de opções.

2.1 Funcionalidades

O programa funcionará com um menu do sistema. Isso deve ser feito no terminal a partir da leitura de números que representam decisões do usuário.

Ao executar o programa, deve ser carregado na memória a lista de livros disponíveis. O Formato do arquivo é discutido na próxima seção.

2.1.1 Menu

A primeira "tela" que o usuário deve ver é uma tela de menu com as opções:

```
1- Cadastrar livro
2- Buscar livro
0- Sair
```

Opção 1 - Cadastrar Livro

Permite cadastrar livros no sistema. É necessário inserir informações como título(string), Id único(inteiro), quantidade de autores, seguida de n autores, nota(double,float), editora(string), quantidade de exemplares(inteiro), ano(inteiro). Como no exemplo:

```
ID: <entrada><Int>
Título: <entrada><Str>
Autores: <qtd autores><entrada><Int> <entrada><Str> <entrada><Str> ... <entrada><Str>
Nota: <entrada><double>
Valor: <entrada><double>
Data de publicação: <entrada><Data>
Editora: <entrada><Str>
Quantidade de exemplares: <entrada><Int>
```

OBS: quantidade de autores só sera aceito a leitura pelo terminal para facilitar a entrada de dados, leitura do arquivo livros.csv não tera a leitura desse inteiro, e os autores serão separados por '/'.

Se o livro já existir, será somado a quantidade de exemplares do livro na lista. Apesar de existirem vários exemplares de livros com mesmo título, o ID de cada livro é único, não pode existir dois livros com o mesmo ID.

Opção 2 - Buscar livro

Um novo menu aparece possibilitando o usuário do sistema pesquisar os livros pelo Título, Autor, Nota do livro ou ID. Use a opção 4 para voltar ao menu anterior.

```
 1- Pesquisar Por Autor
2- Pesquisar Por Título
3- Pesquisar Por nota
4- Pesquisar Por ID
9- Voltar
0- Sair
```

`REGRAS IMPORTANTES`

1. **Uso de função genérica**: Implemente uma função para cada pesquisa de forma que todas tenham a mesma assinatura. Deste modo, deve-se criar uma função de busca genérica que receba como parâmetro a função para busca desejada (autor, nome, nota e ID).
2. **Tabela de dispersão**: Cada função de busca será adicionada em uma tabela de dispersão. Dessa forma, o programa principal usará apenas a função genérica com a chave para a função específica na **tabela de dispersão**.

Por exemplo, supondo que a função de pesquisa tenha a assinatura abaixo, onde valor é a busca e TipoBusca, um caractere que indique a chave na tabela dispersão com os ponteiros de funções específicos.

```
Livro *  Pesquisa(char *valor, char TipoBusca )
```

Supondo ainda que você inseriu cada função de Busca em uma tabela com chaves 'A', 'T', 'I', etc. A chamada para busca por Autor poderia então ser:

```
Pesquisa("Rowling", 'A');
```

3. **Ordenação**: Caso a função pesquisa retorne mais de um resultado, esta lista deve ser ordenada. Você deve usar a função `qsort` da `stdlib.h`, implementando a função de comparação.

Extra (5%): Implementar funções de comparação (utilizadas pelo qsort) para cada um dos atributos possíveis de busca. Adicionar essas funções em uma outra tabela de dispersão. Em seguida prover uma função pesquisa que receba também como argumento, a chave para qual função de ordenação será utilizada. Isto é, utilizando o exemplo acima, suponha que queira todos os livros de Rowling, ordenados pelo título do livro, a chamada então seria:

```
PesquisaOrdenada("Rowling", 'A', 'T');
```

Opção 2.1 - Pesquisar Por Autor

Permite pesquisar livros a partir de seu autor. O sistema deverá mostrar no terminal as informações dos livros cujo autor será o mencionado na busca. O argumento de busca pode ser uma substring dos autores de livros no acervo. Por exemplo, o usuário pode buscar por *Rowling* e a busca retornará todos os livros cujo autores tenham *Rowling* em seus nomes:

```
Autor do livro: J. K. Rowling
```

Informações a serem mostradas:

```
 Título A: harry potter e o calice de fogo
ID A: <saida>
Editora A: <saida>
Nota A: <saida>
Exemplares A: <saida>
Ano A: <saida>

Título B: harry potter e o prisioneiro de askabam
ID B: <saida>
Editora B: <saida>
Nota B: <saida>
Exemplares B: <saida>
Ano B: <saida>

.
.
.

Quantidade de livros encontrados foram: <saida>

1- Exclusão
9- Voltar
0- Sair
```

Caso o autor não exista deverá ser exibida a seguinte mensagem:

```
Autor não cadastrado.

9- Voltar
0- Sair
```

Opção 2.2 - Pesquisar Título

Permite pesquisar um livro a partir de seu título. O sistema deverá mostrar no terminal as informações do livro. O número de exemplares deve ser considerado neste cálculo de quantidade.

OBS: CASO SEJA DADO COMO ENTRADA DE BUSCA "MAÇA" DEVE-SE SER IMPRESSO TODOS OS LIVROS CUJO TITULO CONTENHA A PALAVRA "MAÇA" (Lembre-se de testar letras maiúsculas e minúsculas). Como no exemplo:

```
Título do Livro: maca
```

Informações a serem mostradas:

```
Título A: a maca arrependida
ID A: <saida>
Autores A: <saida>, <saida>, ..., <saida>
Editora A: <saida>
Nota A: <saida>
Exemplares A: <saida>
Ano A: <saida>

Título B: comi uma maca e olha no que deu
ID B: <saida>
Autores B: <saida>, <saida>, ..., <saida>
Editora B: <saida>
Nota B: <saida>
Exemplares B: <saida>
Ano B: <saida>

.
.
.


Quantidade de livros encontrados foram: <saida>

1- Exclusão
9- Voltar
0- Sair
```

Caso o livro não exista deverá ser exibida a seguinte mensagem:

```
Livro inexistente.

9- Voltar
0- Sair
```

Opção 2.3 - Pesquisar Nota

Permite pesquisar livros a partir da nota dada. O sistema deverá mostrar no terminal as informações do livro que tenha determinada nota. Como no exemplo:

```
Nota do livro: 10
```

Informações a serem mostradas:

```
Título A: Ana Maria Braga receitas

Autores A: <saida>, <saida>, ..., <saida>
ID A: <saida>
Editora A: <saida>
Exemplares A: <saida>
Ano A: <saida>

Título B: comi uma maca e olha no que deu
ID B: <saida>
Autores B: <saida>, <saida>, ..., <saida>
Editora B: <saida>
Exemplares B: <saida>
Ano B: <saida>

.
.
.


Quantidade de livros encontrados foram: <saida>

1- Exclusão
9- Voltar
0- Sair
```

Opção 2.4 - Pesquisar Por ID

Permite pesquisar um específico a partir do ID. O sistema deverá mostrar no terminal as informações do livro. Como no exemplo:

```
ID do livro: 407
```

Informações a serem mostradas:

```
Título: Ana Maria Braga receitas
Autores: <saida>, <saida>, ..., <saida>
Editora: <saida>
Nota: <saida>
Exemplares: <saida>
Ano: <saida>

1- Exclusão
9- Voltar
0- Sair
```

Opção EXCLUSÃO

A opção de exclusão em todos os casos deve ser utilizada para deletar uma certa quantidade de exemplares ou o próprio livro do sistema. Utilizando como entrada o ID do livro.

```
Informe o ID a ser excluído: 407
```

Informações a serem mostradas:

```
Título: Ana Maria Braga receitas
Autores: <saida>, <saida>, ..., <saida>
Editora: <saida>
Nota: <saida>
Exemplares: <saida>
Ano: <saida>

1- Deletar o livro
2- Atualizar numero de exemplares
9- Voltar
0- Sair
```

Opção EXCLUSÃO 1 Deletar o livro

```
Tem certeza de que deseja deletar o livro de ID: 407?

1- SIM
2- NÃO
```

A escolha do não irá retornar ao menu inicial

```
O livro não foi deletado
```

A escolha do sim irá deletar o livro e retornar ao menu inicial.

```
Livro deletado com sucesso
```

Opção EXCLUSÃO 2 Atualizar Exemplares

```
Informe a quantidade de exemplares a ser excluidas: 20
```

Observe que deve ser feita uma verificação para checar se a quantidade a ser excluída é igual a quantidade de exemplares disponíveis.

Informações a serem mostradas caso o número de exemplares disponíveis seja igual ao solicitado a exclusão:

```
O livro foi deletado, pois atingiu 0 Exemplares.
```

E o livro deve ser deletado da lista

Informações a serem mostradas caso o numero de exemplares disponiveis NÃO seja igual ao solicitado a exclusão:

```
Foram excluidos 20 exemplares.
```

OBS: caso o número seja maior que a quantidade de exemplares disponíveis, deverá ser informado um erro ao usuário e retornar ao menu inicial.

Opção 0 - Sair

Sempre que o usuário sair do programa pelo menu, deve-se garantir que quaisquer dados que precisam ser persistidos já foram salvos em arquivos, ou seja, se for cadastrado um novo livro deve ser registrado ao final do arquivo que foi dado como entrada.

Além disso, é dever do desenvolvedor garantir que toda a memória foi liberada. Vazamentos de memória serão penalizados.

3. Arquivos de entrada

O sistema realiza a leitura de um arquivo `.csv`: o arquivo de livros. Suas informações são separadas por vírgula simples.

Para o arquivo de livros, são dispostas as informações

```
id, titulo, autor1/autor2/...autorn, nota, valor, data de publicação, editora, qtd de exemplares
```

em que titulo, autores e editora são strings, qtd. de autores e ano são inteiros e o valor é um float/double.

OBS: VEJA QUE AO CADASTRAR UM LIVRO É REQUISITADO UM INT PARA A QUANTIDADE DE AUTORES PARA FACILITAR O TRATAMENTO, MAS NA LEITURA DO ARQUIVO CSV OS AUTORES SÃO APENAS SEPARADOS POR '/'.

4. Compilação e execução

O arquivo `Makefile` compila e gera o executável `tp1`. Na compilação ele procura arquivos cabeçalhos na pasta `include`, códigos fonte em `src` e o arquivo que contém a função main em `client`. Ao executar `make`, modifique a variável `CLI` para mudar o arquivo cliente. Por exemplo,

```
make CLI=client/main2.c
```

utilizaria a função main do arquivo `client/main2.c`. Por padrão, o makefile usa o arquivo `client/main.c`.

Para rodar o trabalho, execute o comando

```
./tp2
```

5. Regras Gerais

O trabalho pode ser feito **em dupla** e pelos próprios alunos, isto é, os alunos deverão necessariamente conhecer e dominar todos os trechos de código criados.

Cada dupla deverá trabalhar independente das outras, não sendo permitido a cópia ou compartilhamento de código. O professor irá fazer verificação automática de plágio. Trabalhos identificados como iguais, em termos de programação, serão penalizados com a nota zero. Isso também inclui a dupla que forneceu o trabalho, sendo, portanto, de sua obrigação a proteção de seu trabalho contra cópias ilícitas.

6. Entrega do Trabalho

O trabalho deverá ser submetido pelo Repl.it e também pelo AVA até as 23:59 do dia 03/10/2021.

O arquivo enviado pelo AVA deve seguir o padrão: TEP-2021_MATRICULA1_MATRICULA2.zip, substituindo a matrícula pelos respectivos números de matrícula da dupla.
O arquivo zip deve conter: Todos os arquivos necessários para compilação (cabeçalhos e códigos fontes) `Makefile` que compile o programa corretamente em um ambiente Linux. `README.md` contendo nome da dupla e com considerações de projeto que julgue necessário, incluindo as decisões de projeto tomadas. Por exemplo, explicação breve das bibliotecas, etc.

O código fonte deverá estar bem comentado, isto é, todas os structs e funções devem ter descrições claras do que ela faz, o que representa cada parâmetro e o que ela retorna (se houver). Pode-se adicionar também comentários que ajude a entender a lógica da questão.

7. Avaliação

O trabalho será corrigido em dois turnos: correção do professor e entrevista. A correção do professor (CP) gerará uma nota entre 0 e 10 e as entrevistas (E) serão realizadas posteriormente à data de entrega do trabalho e será atribuída uma nota entre 0 e 1. Alunos que fizeram o trabalho em dupla farão a entrevista juntos, porém terão seus desempenhos na entrevista avaliados de forma individual. Pequenas e pontuais correções de código serão permitidas no momento da entrevista, o que poderá acarretar uma CP maior.

A nota final (NF) do trabalho será dada pela seguinte fórmula: NF = E * CP

O trabalho será pontuado de acordo com sua funcionalidade e outros aspectos de implementação, conforme as listas abaixo.

Aspectos funcionais

1. Inicialização. Capacidade de carregar os arquivos usando estruturas de Listas.
2. Cadastrar e persistir livros. Isto é, ao cadastrar um livro e sair do programa, este livro é salvo no mesmo arquivo de entrada.
3. Pesquisar livros utilizando ponteiro para função (por título, por autor, por nota e por ID).
4. Exclusão de Livros.

Aspectos de desenvolvimento:

1. (10%) Uso correto de TADs para abstração dos elementos envolvidos no trabalho. Menos *main* mais TADs!
2. (20%) Uso correto de estruturas encadeadas (lista) para uso eficiente da memória. Observe que é obrigatório o uso de estrutura encadeada.
3. (10%) Executar o programa com o valgrind, sair corretamente e não ter vazamento de memória.
4. (50%) Pesquisar livros utilizando tabelas de dispersão com ponteiros de funções. Sendo 10% para busca de cada atributo de busca e 10% do uso correto da tabela de dispersão.
5. Cadastro de novos livros (5%)
6. Exclusão de livros (5%)
7. Pesquisa ordenada por atributo (5% extra).