Universidade Federal de São Carlos
Bacharelado em Ciência da Computação
Introdução à Programação
Prof. Tiago A. Almeida <talmeida@ufscar.br>
Profa. Tiemi C. Sakata <tiemi@ufscar.br>

LISTA 11
EXERCÍCIOS – ARQUIVO BINÁRIO

- **Prazo para entrega: 06/08/2018 – 23:55:00**
- **Atenção:**

1. **Arquivos:** o nome do arquivo referente ao código-fonte deverá respeitar o seguinte padrão: <número do RA>_**L**<número da lista>**EX**<número do exercício>.c. Exemplo: `123456_L11EX01.c`;
2. **Pastas:** cada arquivo-fonte e seus arquivos auxiliares deverão ser salvos dentro de uma pasta individual, cujo nome deverá seguir o padrão **L**<número da lista>**EX**<número do exercício>. Exemplo: `L11EX01`;
3. **Submissão:** submeta no Moodle apenas um arquivo zip contendo todas as pastas, arquivos-fonte (.c) e arquivos auxiliares. O nome do arquivo compactado deverá respeitar o seguinte padrão: <número do RA>_**L**<número da lista>.zip. Exemplo: `123456_L11.zip`;
4. **Identificadores de variáveis:** escolha nomes apropriados;
5. **Documentação:** inclua cabeçalho, comentários e indentação no programa;
6. **Observação:** assuma que os arquivos de entrada e saída estão ou deverão ser criados e mantidos na mesma pasta do programa.

- **Exercícios:**

1) Crie um programa para gerenciar múltiplos campeonatos de futebol. Seu programa deve oferecer 7 opções para manipulação dos arquivos binários, explicadas abaixo. Para todas as opções, a primeira ação realizada será a solicitação do nome do campeonato.

   1. `Criar um campeonato`: o programa deve receber dois inteiros, **T** e **J**, representando a quantidade de times no campeonato e a quantidade de jogos realizados no campeonato, respectivamente. Em seguida, deve receber **J** linhas - cada uma representando um jogo do campeonato - no formato "**t1 r t2**", em que **t1** e **t2** são inteiros de valor entre 1 e **T**, que representam os times participantes no jogo, e **r** é um caractere, que representa o resultado do jogo. Se o **r** de um jogo é igual 'x', então o time **t1** ganhou do time **t2**. Caso o **r** de um jogo seja '-', então houve um empate entre **t1** e **t2**. As entradas devem ser armazenadas no arquivo "`nome_campeonato` - Campeonato.dat", sendo que os valores inteiros devem ser armazenados no arquivo binário com tipo `int`, e os caracteres devem ser armazenados com tipo char.

2. `Calcular os resultados de um campeonato`: o programa deve verificar se já existe um arquivo para o campeonato fornecido. Se não existir, uma mensagem de `ARQUIVO_N_ENCONTRADO` deve ser exibida na tela, voltando para o menu. Caso encontre o arquivo, o programa deve gerar outro arquivo, chamado "`nome_campeonato - Resultados.dat`", que possuirá três inteiros para cada time participante, informando, respectivamente, a quantidade de vitórias, derrotas e empates do time.
3. `Calcular os pontos de um campeonato`: o programa deve verificar se já existe um arquivo de resultados para o campeonato fornecido. Se não existir, uma mensagem de `ARQUIVO_N_ENCONTRADO` deve ser exibida na tela, voltando para o menu. Caso encontre o arquivo, o programa deve gerar outro arquivo, chamado "`nome_campeonato - Pontos.dat`", que possuirá um inteiro para cada time participante (onde o primeiro inteiro representa o time 1 e assim por diante), representando o total de pontos do time, sendo que vitórias concedem 3 pontos, empates concedem 1 e derrotas não concedem pontos.
4. `Salvar os gols dos jogadores de um campeonato`: o programa deve verificar se já existe o arquivo do campeonato fornecido. Se não existir, uma mensagem de `ARQUIVO_N_ENCONTRADO` deve ser exibida na tela, voltando para o menu. Caso encontre o arquivo, o programa deve gerar outro arquivo, chamado "`nome_campeonato - Gols por jogador.dat`" e armazenar nele os dados fornecidos pelo usuário. O usuário fornecerá, para cada time no campeonato, onze inteiros, representando a quantidade de gols realizados por cada um dos onze jogadores do time. Os valores inteiros devem ser armazenados no arquivo binário com tipo `int`.
5. `Gerar os gols dos times de um campeonato`: o programa deve verificar se já existe um arquivo de gols por jogador para o campeonato fornecido. Se não existir, uma mensagem de `ARQUIVO_N_ENCONTRADO` deve ser exibida na tela, voltando para o menu. Caso encontre o arquivo, o programa deve gerar outro arquivo, chamado "`nome_campeonato - Gols por time.dat`", que possuirá um inteiro para cada time participante (onde o primeiro inteiro representa o time 1 e assim por diante), representando o total de gols do time.
6. `Gerar os artilheiros de um campeonato`: o programa deve verificar se já existe um arquivo de gols po jogador para o campeonato fornecido. Se não existir, uma mensagem de `ARQUIVO_N_ENCONTRADO` deve ser exibida na tela, voltando para o menu. Caso encontre o arquivo, o programa deve gerar outro arquivo, chamado "`nome_campeonato - Artilheiros.dat`", que possuirá um inteiro para cada time participante (onde o primeiro inteiro representa o time 1 e assim por diante), representando o número do jogador que mais fez gol para o time, sabendo que o número do jogador é a posição dele no arquivo de gols por jogador (variando de 1 a 11). Em caso de empate, o jogador de menor número deve ser escrito.
7. `Gerar a tabela geral de um campeonato`: o seu programa deve verificar se já existe os arquivos de campeonato, resultados, pontos, gols por time e artilheiros, nesta ordem. Se algum dos arquivos não existir, deve ser exibida mensagem de `ARQUIVO_N_ENCONTRADO`, retornando ao menu. Caso todos arquivos existam, o programa deve gerar

outro arquivo, chamado "`nome_campeonato - Tabela geral.dat`", que possuirá seis inteiros para cada time participante, representando, respectivamente, as vitórias, derrotas, empates, pontos, gols do time e número do artilheiro do time.

Complete o arquivo `L11EX01.c`

Detalhes

a) Os valores inteiros serão do tipo `int` (4 bytes de tamanho) e os caracteres serão do tipo `char` (1 byte de tamanho).

2) Crie um programa para manter um cadastro de Alunos (vetor de registros). O seu programa deverá oferecer as seguintes funcionalidades (menu):

```
1. Cadastrar
2. Alterar
3. Remover
4. Buscar
5. Listar
6. Sair
```

Todos os dados deverão ser salvos no **arquivo-binário** `Alunos.dat` com os registros ordenados em ordem crescente de RA. Caso o arquivo já exista, o seu programa deverá ler os dados já existentes. Mantenha os dados dos alunos em uma estrutura `Aluno` com os seguintes campos: RA (`int`), nome (*string* - 100 posições), ano de ingresso (`int`) e quantidade de créditos cursados (`int`).

Cada opção deverá executar o seguinte procedimento:

- `Cadastrar`: solicita todos os dados do aluno (1 por linha, na ordem: RA, ano de ingresso, quantidade de créditos cursados e nome). Caso o RA informado já exista, imprimir na tela a frase "`Aluno já está cadastrado.`" e retornar ao menu. Caso contrário, cadastra o aluno.
- `Alterar`: solicita o RA do aluno. Caso ele seja encontrado, solicitar novamente os campos: ano de ingresso, quantidade de créditos cursados e nome (1 por linha), alterando os dados do aluno. Caso contrário, emitir mensagem "`Aluno não cadastrado.`" e retornar ao menu.
- `Remover`: solicitar o RA do aluno. Caso ele seja encontrado, remover o registro. Caso contrário, emitir a mensagem "`Aluno não cadastrado.`" e retornar ao menu.
- `Buscar`: solicitar o RA do aluno. Caso ele seja encontrado, exibir todos os campos do registro (utilize a função de exibição já disponibilizada no código-fonte). Caso contrário, emitir mensagem "`Aluno não cadastrado.`" e retornar ao menu.
- `Listar`: imprimir na tela todos os campos de todos os registros existentes em ordem crescente de RA (utilize a função de exibição já disponibilizada no código-fonte).

– `Sair`: sobrescrever o arquivo com dados atualizados, liberar memória e fechar o programa.

O seu programa deverá carregar/criar o arquivo na inicialização, manipular os dados em memória (sempre mantendo os registros em ordem crescente de RA) e sobrescrever o arquivo na finalização.

A escrita e a leitura do arquivo devem ser feitas de maneira sequencial, sem separadores de registros. Recomenda-se utilizar as funções `fread` e `fwrite` com a `struct aluno`.

Complete o arquivo `L11EX02.c`

Detalhes

a) Todas as entradas serão informadas corretamente.

b) Serão disponibilizados dois casos de teste abertos e um arquivo explicativo sobre os mesmos. Analisem estes casos de teste para entender a entrada do exercício e saída/arquivo resultante.

Casos de teste

– O exercício possui dez casos de teste.

– Para os cinco primeiros, não haverá arquivo existente.

1. Cadastrar e Listar. Sem erros
2. Cadastrar, Listar e Alterar. Sem erros
3. Cadastrar, Listar, Alterar e Remover. Sem erros
4. Cadastrar, Listar, Alterar, Remover e Buscar. Sem erros
5. Cadastrar, Listar, Alterar, Remover e Buscar. Com erros

– Erros são: inserção de RA já cadastrado, alteração, remoção ou busca de RA não cadastrado.

– Os outros cinco casos possuem o mesmo formato dos cinco primeiros, com o adicional de já existir um arquivo na inicialização do programa.

- **Cuidados:**

1. **Erros de compilação:** nota **zero** no exercício.
2. **Tentativa de fraude:** nota **zero na média** para todos os envolvidos.