11ª Lista de Exercícios de Introdução à Programação II
Arquivos

1ª Questão: Faça um programa para **criar** um arquivo binário para leitura e gravação. O programa deve solicitar ao usuário que forneça o nome do arquivo a ser criado. Caso já exista um arquivo com o nome fornecido pelo usuário, o programa deverá informar que as informações do arquivo existente serão perdidas e pedir a confirmação da operação. Caso o usuário confirme que deseja criar o arquivo com o nome dado, efetue a operação. Caso, contrário, encerre a execução do programa. Faça todas as checagens necessárias para garantir o sucesso das operações realizadas. Lembre-se de que, no final do programa, é necessário fechar todos os arquivos que, por ventura, tenham sido abertos.

2ª Questão: Faça um programa para **excluir** (remover fisicamente) um arquivo binário. O programa deve solicitar ao usuário que forneça o nome do arquivo a ser excluído. Caso não exista um arquivo com o nome fornecido pelo usuário, o programa deve emitir uma mensagem informando o ocorrido. Caso exista um arquivo com o nome fornecido pelo usuário, o programa deve informar que as informações do arquivo existente serão perdidas e pedir a confirmação da operação. Caso o usuário confirme que deseja excluir o arquivo com o nome dado, efetue a operação. Caso contrário, encerre a execução do programa. Faça todas as checagens necessárias para garantir o sucesso das operações realizadas. Lembre-se de que, no final do programa, é necessário fechar todos os arquivos que, por ventura, tenham sido abertos.

3ªQuestão: Faça um programa para criar e preencher um arquivo chamado **"turmaProgI.dat"** com os dados (matrícula, nome e média) dos alunos de uma turma de Introdução à Programação I. Os dados serão fornecidos pelo usuário.

Estrutura dos registros armazenados no arquivo **"turmaProgI.dat"**:

```
struct aluno {
      char matr [10];
      char nome [30];
      float media;
};
```

4ªQuestão: Considere o arquivo **"turmaProgI.dat"** da questão anterior. Assuma ainda que desejamos criar um novo arquivo chamado **"novoProgI.dat"** que será composto de registros do tipo **novoAluno** (descrito abaixo).

```
struct novoAluno {
      char matr [10];
      char nome [30];
      float media;
      int faltas;
};
```

O arquivo **"novoProgI.dat"** deve ser preenchido com os dados constantes do arquivo **"turmaProgI.dat"** colocando-se 0 (zero) no campo relativo à quantidade de faltas do aluno.

5ª Questão: Faça um para criar dois arquivos: "aprovados.dat" e "reprovados.dat". Esses dois arquivos devem ser preenchidos com os dados constantes no arquivo **"turmaProgI.dat"**, da 3ª questão, de tal forma que no arquivo "aprovados.dat" fiquem as informações dos alunos com média maior ou igual a sete e no arquivo "reprovados.dat" fiquem as informações dos alunos com média menor que sete. Ao final, o programa deve exibir o conteúdo dos dois arquivos criados.

6ªQuestão: Considere o arquivo **"turmaProgI.dat"** da 3ª questão. Faça um programa para, dado um nome de aluno, procurá-lo no arquivo e, caso encontre, alterar sua média. Caso não encontre, informar o fato ao usuário através de mensagem.

7ªQuestão: João possui uma papelaria. Ele necessita automatizar o cadastro do estoque de suas mercadorias. Você foi contratado para realizar este trabalho. Faça um programa para criar e preencher o cadastro de estoque de mercadorias da papelaria. O cadastro deverá conter as seguintes informações sobre cada produto: código do produto, descrição do produto, quantidade em estoque e preço. O programa deverá ter as seguintes funcionalidades:

(1) Procedimento para **cadastrar** um novo produto. Recebe como parâmetro o endereço do arquivo e o código do produto a ser cadastrado. **Não podem existir dois produtos com o mesmo código**. Caso o código não exista no arquivo, procede com o cadastramento do produto; caso contrário, informa que o cadastramento não pode ser efetuado, pois o código já existe;
(2) Procedimento para **alterar o preço** de um produto. Recebe como parâmetro o endereço do arquivo e o código do produto. Caso seja um código válido, procede com a operação de alteração do preço; caso contrário, informa que a alteração não pode ser efetuada, pois o código não existe no cadastro;
(3) Procedimento para **exibir** os dados (descrição, quantidade disponível no estoque e preço) de um produto. Recebe como parâmetro o endereço do arquivo e o código do produto a ser pesquisado. Caso seja um código válido, procede com a exibição dos dados do produto; caso contrário, informa que o código fornecido não é válido, pois não existe no cadastro;
(4) Função **consultar** produto que recebe como parâmetro de entrada o endereço do arquivo e o código do produto e retorna como saída a posição do arquivo onde o registro do produto se encontra. Caso não encontre o código procurado, deve retornar -1. **Deverá ser utilizada pelos procedimentos de cadastramento, alteração e exibição.**
(5) Procedimento para **listar** todos os produtos cadastrados com o código, a descrição, a quantidade disponível no estoque e o preço.