Aula prática - Semana 12 - ERE

Nesta lista, as questões 1 e 2 constituem as atividades avaliativas que serão utilizadas para contabilizar notas de atividades práticas. As demais questões são desafios que não contabilizam nota, mas cuja realização é desejável para exercitar lógica. As questões 1 e 2 possuem exemplos de execução associado. Note que são apenas exemplos, uma vez que vocês podem propor suas próprias formas de entrada e saída de dados. No entanto, é desejado que se assemelhe ao padrão apresentado no exemplo.
Os seguintes pontos serão considerados na avaliação:

- Atender os requisitos do enunciado.
- Indentação e organização de código.
- Utilização dos tipos corretos de variáveis.
- Utilização correta de constantes nomeadas quando necessário.
- Observação das boas práticas de programação vistas em aula.
- Realização de validação quando a questão requisitar.

ATENÇÃO:

- Nas atividades dessa semana, além de me enviarem os arquivos de **código fonte** dos programas, me enviem também (no mesmo arquivo compactado) os **arquivos** "funcionários.dat" e "atualizados.dat" resultantes da execução dos programas que vocês criaram.
- O desafio dessa semana envolve uma estratégia que vocês podem utilizar para implementar o salvamento e carregamento do estado do jogo no trabalho final da disciplina. Não é necessário que me entreguem. Vocês podem implementar diretamente no trabalho, efetuando as devidas modificações da ideia. Mas fiquem à vontade para me enviar a atividade conforme o enunciado.

1. Faça um programa para manipular informações de funcionários de uma empresa. O programa deve:
    a. Definir uma estrutura FUNCIONARIO que armazena o nome (de até 80 caracteres) e o salário de um funcionário.
    b. Definir uma função que recebe uma string que representa o nome do arquivo em que deseja-se salvar a informação e um FUNCIONARIO como parâmetros e que salva as informações do funcionário no fim do arquivo associado à stream. A função retorna 1 caso a operação de escrita tenha sido um sucesso e 0 caso tenha ocorrido uma falha. Essa função deve encapsular toda a lógica de abrir o arquivo da forma correta, salvar a informação e fechar o arquivo.

    Este programa assume que todas as informações serão armazenadas no arquivo "funcionários.dat". O programa deve continuamente perguntar para o usuário se ele quer inserir um novo funcionário (opção 1) ou sair (opção 2). Caso o usuário queira incluir um novo funcionário, o programa deve requisitar que ele informe o nome e o salário e deve incluir o registro desse funcionário no fim do arquivo aberto. Faça consistência do nome (não pode ser string vazia) e do salário (deve ser maior que zero). Note que, caso eu execute novamente o programa, ele deve permitir que eu

seja capaz de salvar informações de novos funcionários no fim do arquivo existente, sem perder as informações já salvas no arquivo. Ou seja, é necessário decidir o modo de abertura do arquivo com atenção. **OPCIONAL**: Caso você queira fazer uma versão mais completa deste programa, inclua uma opção adicional (além das que já existem) para o programa listar todos os funcionários do arquivo. Esta opção é útil para verificar se todos os registros incluídos no arquivo estão corretos.

Exemplo de execução:

```
Incluir novo funcionário (1) ou sair (2)? 1
Informe o nome: Carol Danvers
Informe o salário: R$ 7000.00
Incluir novo funcionário (1) ou sair (2)? 1
Informe o nome:
Nome inválido!
Informe o nome: Bruce Banner
Informe o salário: R$ 5000.00
Incluir novo funcionário (1) ou sair (2)? 1
Informe o nome: Natasha Romanoff
Informe o salário: R$ 0
Salário inválido!
Informe o salário: R$ 5000.00
Incluir novo funcionário (1) ou sair (2)? 2
Sistema finalizado.
```

2. Faça um programa que:
    a. Leia um valor que define o percentual do aumento do salário dos funcionários.
    b. Leia do arquivo "funcionários.dat", criado pelo programa anterior, cada funcionário, atualize o salário dele em função do percentual de aumento definido pelo usuário e salve a nova informação em um novo arquivo "atualizados.dat".
    c. O programa deve usar uma função que recebe por referência uma estrutura FUNCIONARIO e o percentual de aumento salarial como parâmetros e atualiza o salário do funcionário (incrementando o salário em função do percentual). Note que essa função deve apenas atualizar o salário do funcionário e nada além disso. Ou seja, ela não faz a gravação dessa informação em arquivo e nem exibe informações na tela, por exemplo. A função tem o seguinte cabeçalho:
    ```
    void atualiza(FUNCIONARIO *func, float percentual)
    ```
    d. O programa também deve exibir na tela o nome de cada funcionário lido do arquivo "funcionários.dat", bem como seu salário antes e depois do aumento.
    e. Note que você pode reutilizar a função para gravar dados de funcionário definida no programa anterior.
    f. Note que o programa deve funcionar para qualquer número de funcionários presentes no arquivo "funcionários.dat", sem supor algum número máximo.

Exemplo de execução:
Informe o percentual de aumento salarial: 10
Funcionários:

_____
Nome: Carol Danvers
Salário anterior: R$ 7000.00
Salário atualizado: R$ 7700.00

_____
Nome: Bruce Banner
Salário anterior: R$ 5000.00
Salário atualizado: R$ 5500.00

_____
Nome: Natasha Romanoff
Salário anterior: R$ 5000.00
Salário atualizado: R$ 5500.00

3. (Desafio) Ideia para utilizarem no trabalho final. Crie uma estrutura chamada ESTADO_JOGO, cujos campos constituem todas as informações que representam o estado atual do jogo desenvolvido no trabalho final. Esses campos podem ser: arranjo que representa as informações dos ninjas, posição do jogador, nome do mapa em que o jogador está, número de vidas do jogador, número de pontos do jogador, etc. Pense a respeito de quais são as informações que são necessárias que sejam salvas para que, caso sejam carregadas em um outro momento, permita que o jogador continue jogando o jogo a partir do estado em que ele foi salvo. Faça um programa que preencha uma variável estado1 do tipo ESTADO_JOGO com valores para todos os campos necessários. Esse preenchimento pode ser realizado na inicialização ou através de leitura de informações do usuário. O programa deve salvar essa informação em um arquivo “save.sav”. A seguir, o programa deve carregar a informação do arquivo “save.save” para outra variável estado2 do tipo ESTADO_JOGO, e exibir as informações de todos os campos que constituem o estado. O objetivo é verificar se as informações carregadas são exatamente iguais às informações salvas. Você pode testar a ideia com uma versão simplificada da estrutura ESTADO_JOGO e depois aplicar a ideia ao trabalho final, incluindo todos os campos necessários para garantir o requisito do jogo.