Tecnologia em Análise e Desenvolvimento de Sistemas
Lógica de Programação
Professor Vinícius Fritzen Machado

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande

TRABALHO VETORES

ATENÇÃO

1. SOMENTE RECURSOS UTILIZADOS EM SALA DE AULA DEVEM SER UTILIZADOS NA SOLUÇÃO DESTE TRABALHO.

2. O TRABALHO DEVE SER APRESENTADO PARA O PROFESSOR NO DIA 16/12/2024 NO HORÁRIO DE AULA.

3. O MODELO DE TRABALHO SERÁ DE ACORDO COM A INICIAL DO SEU NOME

Modelo A - Iniciais de A até G	(Adriel - Gabriel)
Modelo B - Iniciais de J até J	(Jaaziel - Jorge)
Modelo C - Iniciais de L até R	(Lineker - Roberto)

Novas soluções são encorajadas, no entanto, é necessário que os alunos demonstrem domínio sobre as técnicas apresentadas. Os códigos fontes serão avaliados quanto a funcionalidade, legibilidade, estrutura e organização. Enviar os códigos fontes para o email

vinicius.machado+logica2024@riogrande.ifrs.edu.br

No Assunto, incluir seu Nome + sobrenome.
Compacte os arquivos .java em .zip e renomeie o arquivo com seu nome.

MODELO A

Sistema de Gerenciamento de Turmas e Alunos

Objetivo do Exercício:

Desenvolver um sistema para gerenciar as turmas e alunos de uma escola, incluindo funcionalidades para adicionar, buscar, listar e transferir alunos entre turmas. O sistema deve ser capaz de manipular dados complexos utilizando vetores e permitir operações comuns de gerenciamento escolar.

Estrutura de Dados:

1. **Vetores Principais:**
   - `turmas[]`: Armazena o nome das turmas (por exemplo, "Turma A", "Turma B").
   - `professores[]`: Armazena o nome dos professores responsáveis por cada turma, correspondendo ao índice de `turmas`.
   - `alunos[][]`: Um vetor bidimensional onde cada linha representa uma turma e cada coluna contém uma lista de alunos associados a essa turma.
2. **Vetores Auxiliares (se necessário):**
   - `idsTurmas[]`: Armazena IDs únicos para cada turma.
   - `disciplinas[][]`: Um vetor bidimensional onde cada linha representa uma turma e cada coluna contém uma lista de disciplinas ensinadas na turma.

Funcionalidades a Implementar:

1. **Adicionar Nova Turma:**
   - O sistema deve permitir que o usuário adicione uma nova turma, especificando o nome da turma e o professor responsável.
   - A nova turma deve ser adicionada aos vetores `turmas[]` e `professores[]`, com um novo vetor vazio criado em `alunos[][]` para armazenar os alunos dessa turma.
2. **Adicionar Alunos a uma Turma:**
   - O usuário deve poder adicionar alunos a uma turma existente, especificando o nome do aluno e a turma a qual ele pertence.
   - O nome do aluno deve ser adicionado ao vetor correspondente em `alunos[][]`.
3. **Buscar Turmas por Nome do Professor:**
   - Implementar uma função que permite ao usuário buscar uma turma específica pelo nome do professor.
   - A função deve retornar o nome da turma e a lista de alunos associados.
4. **Listar Alunos de uma Turma:**
   - Permitir que o usuário visualize todos os alunos de uma turma específica, utilizando o vetor `alunos[][]`.
   - O sistema deve permitir a escolha da turma com base no nome da turma ou do professor.
5. **Transferir Aluno entre Turmas:**
   - O sistema deve permitir a transferência de alunos de uma turma para outra.
   - Isso envolve remover o aluno do vetor correspondente na turma de origem e adicioná-lo ao vetor da turma de destino.

MODELO B

Jogo "Cara a Cara" com Vetores

Objetivo do Jogo: O objetivo é criar um jogo "Cara a Cara" em que cada pessoa é representada por um índice em vetores paralelos. Os jogadores devem adivinhar a pessoa selecionada pelo oponente fazendo perguntas sobre suas características. O jogo deve contar com 20 pessoas

Descrição do Projeto:

1. **Vetores de Dados:**
    - Crie vetores paralelos onde cada índice representa uma pessoa diferente. Os vetores podem incluir:
        - `nomes[]`: Contém os nomes das pessoas.
        - `generos[]`: Contém o gênero das pessoas (por exemplo, "M", "F").
        - `cabelos[]`: Descreve a cor do cabelo (por exemplo, "loiro", "castanho", "preto").
        - `olhos[]`: Descreve a cor dos olhos (por exemplo, "azul", "verde", "castanho").
        - `alturas[]`: Representa a altura das pessoas (por exemplo, "alto", "médio", "baixo").
        - `acessorios[]`: Contém informações sobre acessórios (por exemplo, "óculos", "chapéu", "nenhum").

    Para cada vetor, você deve sortear aleatoriamente valores para que cada partida seja única.

2. **Seleção de Personagem:**
    - Usando o random, uma posição é sorteada indicando qual o nome escolhido.
3. **Sistema de Perguntas:**
    - O jogador faz perguntas que podem ser respondidas com "Sim" ou "Não". As perguntas são baseadas nas características armazenadas nos vetores. Por exemplo:
        - "A pessoa escolhida tem cabelo loiro?"
        - "A pessoa usa óculos?"
4. **Eliminação de Personagens:**
    - Com base nas respostas, o jogador elimina opções. Por exemplo, se a resposta para "A pessoa tem cabelo loiro?" for "Não", o jogador 2 elimina todas as pessoas com cabelo loiro.
5. **Adivinhação Final:**
    - O jogador tenta adivinhar a pessoa escolhida com base nas características restantes.

MODELO C

Sistema de Cadastro e Consulta de Produtos (Sem Vendas e Clientes)

Objetivo do Exercício:

Desenvolver um sistema completo de gerenciamento de produtos para uma loja. O sistema deve permitir o cadastro, busca, atualização e exclusão de produtos, além de funcionalidades para calcular impostos, gerar relatórios de estoque e gerenciar o inventário. Os alunos deverão usar vetores paralelos para armazenar informações sobre os produtos e aplicar lógica de programação para manipular esses dados.

Estrutura de Dados:

1. **Vetores Principais:**
    - `nomes[]`: Armazena os nomes dos produtos.
    - `precos[]`: Armazena os preços dos produtos.
    - `quantidades[]`: Armazena a quantidade em estoque de cada produto.
    - `categorias[]`: Armazena a categoria do produto (por exemplo, "Eletrônicos", "Alimentos").
    - `ids[]`: Armazena um identificador único para cada produto.

Funcionalidades a Implementar:

1. **Adicionar Novo Produto:**
    - Permitir que o usuário adicione um novo produto ao sistema, fornecendo nome, preço, quantidade inicial em estoque e categoria. O produto deve receber um ID único gerado automaticamente.
2. **Buscar Produtos:**
    - Implementar funções para buscar produtos por:
        - Nome: Retorna informações detalhadas do produto.
        - Categoria: Lista todos os produtos dentro de uma determinada categoria.
        - Intervalo de Preço: Lista todos os produtos dentro de um intervalo de preço especificado.
3. **Atualizar Estoque:**
    - Implementar uma função que permita ao usuário atualizar a quantidade em estoque de um produto, seja para ajustar a quantidade por causa de um erro anterior ou para registrar a reposição de estoque.
4. **Calcular Impostos:**
    - Adicionar uma função para calcular impostos sobre os produtos. Os impostos podem variar de acordo com a categoria do produto. Por exemplo, produtos eletrônicos podem ter uma alíquota de imposto mais alta que alimentos.
5. **Gerar Relatórios:**
    - Implementar relatórios que incluam:
        - Produtos com estoque baixo (menos de uma determinada quantidade).
        - Relatório de produtos por categoria, listando todos os produtos de uma categoria específica.
        - Relatório de produtos acima ou abaixo de um determinado preço.