

Trabalho Prático 2 – Sistema de Gerenciamento de Jogos Online

Observações:

- O trabalho é individual.
- Cópias de trabalho receberão nota **ZERO**.
- O programa deve ser desenvolvido na linguagem de programação C.
- A finalização do trabalho será avaliada em 3 pontos.

Tema: Jogo de Adivinhação com Níveis e Pontuação

Conteúdos abordados

Decisão, repetição, vetor, matriz, função, procedimento, recursividade, ponteiro, string, arquivos e struct.

Objetivo

Desenvolver um sistema que simule a gestão de um servidor de jogos online, utilizando conceitos fundamentais da programação estruturada. O sistema deve evoluir em etapas, de funcionalidades básicas (como menu e repetição) até recursos mais complexos (uso de ponteiros, arquivos e structs).

Descrição Geral

Os alunos deverão implementar um programa que simula a administração de um servidor de jogos online. O programa permitirá que usuários registrem partidas, calculuem pontuações, gerenciem rankings e tomem decisões estratégicas em situações críticas.

As funcionalidades serão construídas etapa por etapa, e a última etapa obrigatoriamente será feita em sala de aula, com avaliação prática.

Etapas do Trabalho

Etapa 1 – Menu Principal

Objetivo:

Criar o menu de opções que será usado ao longo de todo o projeto.

O que deve ser feito:

- Usar `do...while` para manter o menu em execução até a escolha de “Sair”.
- Usar `switch case` para selecionar opções.
- Criar uma função para exibir o menu.

Exemplo de menu:

===== SISTEMA DE JOGOS ONLINE =====

1. Ranking Recursivo
2. Registrar Pontuação
3. Estatísticas de Partidas
4. Simulação de Decisão Tática
5. Manipulação de Dados com Ponteiros
6. Registrar Jogadores em Arquivo
7. Migrar Sistema para Struct
8. Sair

Etapa 2 – Ranking Recursivo

Objetivo:

Implementar uma função recursiva que simula o cálculo da pontuação acumulada de um jogador ao longo de n fases.

O que deve ser feito:

- Criar uma função recursiva que receba como parâmetro o número de fases concluídas (n) e retorne a pontuação total.
- O jogador começa com **100 pontos iniciais**, independente do total de fases concluídas.
- A cada fase concluída, o jogador ganha 100 pontos. Além dos pontos por fase, o jogador ganha um bônus de 50 pontos adicionais a cada 3 fases concluídas.
 - Exemplos:
 - Jogador concluiu 5 fases: 100 (pontos iniciais) + (5 x 100 (pontos por fase)) + 50 (1 bônus, pois concluiu 5 fases) = 650 pontos
 - Jogador concluiu 9 fases: 100 (pontos iniciais) + (9 x 100 (pontos por fase)) + (3 x 50 (3 bônus, pois concluiu 9 fases)) = 1150 pontos

- Exibir o total de pontos ao final.
 - Exibir também uma mensagem interpretativa de desempenho, como:
 - “Está só começando...” (até 300 pontos)
 - “Bom desempenho!” (301–600 pontos)
 - “Incrível, você dominou as fases!” (acima de 600 pontos).
-


Etapa 3 – Registro de Pontuações (com Vetores)

Objetivo:

Permitir que vários jogadores registrem suas pontuações usando a função recursiva da Etapa 2.

O que deve ser feito:

- Criar um vetor de inteiros para armazenar até **10 pontuações**.
- Para registrar uma nova pontuação:
 - Perguntar ao usuário quantas fases o jogador concluiu.
 - Chamar a função recursiva da **Etapa 2** para calcular a pontuação.
 - Armazenar o valor retornado no vetor.
- Criar função para exibir todas as pontuações armazenadas (ranking simples).

 Assim, a pontuação de cada jogador só pode ser registrada se passar pelo cálculo recursivo da etapa anterior.

Etapa 4 – Estatísticas com Matriz

Objetivo:

Gerar estatísticas a partir das pontuações registradas, combinando os vetores anteriores em uma **matriz de resultados**.

O que deve ser feito:

- Criar uma matriz **3x10**, onde:
 - Cada **linha** representa um modo de jogo: Casual (0), Competitivo (1), Hardcore (2).
 - Cada **coluna** representa partidas disputadas (até 10 por modo).
- Cada elemento da matriz deve ser preenchido a partir de pontuações já registradas no vetor da **Etapa 3**.
- Criar função que percorre a matriz e calcula a **média de pontos por modo de jogo**.

👉 A matriz é construída **reutilizando dados do vetor de pontuações**, consolidando a progressão natural do sistema.

🔥 Etapa 5 – Manipulação com Ponteiros

🎯 Objetivo:

Aplicar modificações em pontuações já armazenadas, usando **ponteiros**.

📌 O que deve ser feito:

- Criar uma função que receba o **endereço da pontuação de um jogador** e aplique um bônus de +50 pontos.
- Exibir a pontuação antes e depois da alteração.
- Escolher uma posição do vetor ou da matriz para aplicar esse bônus.

🔥 Etapa 6 – Jogadores, Strings e Arquivos

🎯 Objetivo:

Registrar jogadores de forma mais completa, salvando nomes e pontuações em um arquivo.

📌 O que deve ser feito:

- Criar um vetor de `char` para armazenar os nomes de jogadores.
- Associar cada nome à sua pontuação calculada (Etapa 2) e armazenada (Etapa 3).
- Gravar os pares (nome + pontuação) no arquivo `jogadores.txt`.
- Criar função para **ler o arquivo** e exibir todos os jogadores registrados.

🔥 Etapa 7 – Estrutura com Struct

🎯 Objetivo:

Unificar todas as informações em uma **struct**, substituindo vetores separados.

📌 O que deve ser feito:

- Criar uma `struct Jogador` com:

```
struct Jogador {  
  
    char nome[50];  
    int pontuacao[3][10];  
    int partidas[3];  
};
```

- Reimplementar as funções das etapas anteriores para trabalhar com **um vetor de structs**, em vez de vetores separados.
- Exibir o **ranking final** baseado no campo `pontuacao`.

👉 Essa é a última evolução: todas as funcionalidades anteriores passam a trabalhar com uma única estrutura organizada, que representa cada jogador de forma completa.