Programação II

Bacharelado em Ciência da Computação | UFMT Araguaia

Semestre 2025/2

Miniprojeto: Jogo da Velha (POO e C++ Moderno)

1. Informações Gerais e Regras

- **Realização em grupo:** O trabalho deve ser realizado em grupos de, no máximo, 2 integrantes.
- **Pontuação:** O miniprojeto vale até um máximo de 10 pontos, divididos em Níveis de complexidade (Nível I vale 4 pontos, Nível II vale 7 pontos e Nível III vale 10 pontos).
- **Submissão:** Submeter no **AVA** um arquivo *.zip contendo:
  - Arquivos de código-fonte C++ do programa implementado.
  - Arquivo PDF com um pequeno relatório (cerca de 1 página A4) justificando as principais opções tomadas em termos de estruturas de dados e arquitetura Orientada a Objetos.
- **Defesa:** Defesa oral obrigatória por um dos integrantes (mediante sorteio). O não comparecimento implica na anulação (nota 0). A validação da nota obtida está sujeita à correta defesa das decisões tomadas no código, bem como às respostas corretas para as perguntas feitas sobre o mesmo.
- **Avaliação: A implementação deve ser estritamente orientada a objetos e usar boas práticas de C++ Moderno, caso contrário haverá uma forte penalização. A implementação das classes deve seguir a estratégia de separação entre a sua definição geral (.h) e sua implementação (.cpp), contendo um Makefile que faça as ligações e compilações corretamente. A impossibilidade de compilação por parte do professor através do comando `make` resultará em forte penalização.**

2. Descrição Sumária do Jogo

Pretende-se desenvolver um software para permitir a um usuário jogar o Jogo da Velha (*Tic-Tac-Toe*) contra o computador. O jogo ocorre em um tabuleiro de 3x3. Ganha o jogador que conseguir formar primeiro uma linha, coluna ou diagonal com 3 das suas marcas ('X' ou 'O').

O computador deve suportar vários modos de funcionamento (níveis de IA):

1. **Elementar:** O computador joga aleatoriamente.
2. **Básico:** O computador procura ganhar na jogada atual (estratégia *greedy*). Se não for possível, joga aleatoriamente.
3. **Médio:** O computador procura ganhar na jogada atual e, não sendo possível, bloqueia o usuário para evitar que este ganhe na jogada seguinte.

Durante o jogo, a tela deve mostrar: o tabuleiro, o número da jogada, quem está jogando, o modo de funcionamento atual e as estatísticas (vitórias, empates, derrotas).

3. Arquitetura Obrigatória (O que deve ser implementado)

Para obter a pontuação total, a implementação deve abandonar a abordagem procedimental e adotar o paradigma Orientado a Objetos, demonstrando o domínio de Encapsulamento, Herança, Classes Abstratas, Polimorfismo e **Gestão de Memória Moderna (Smart Pointers)**.

A. Encapsulamento: Classe Tabuleiro

O estado do jogo não deve estar solto no main.

- **Privado:** A matriz 3x3 (ou vetor de 9 posições) e o número de jogadas efetuadas.
- **Público:** Métodos para aplicarJogada(posicao, simbolo), verificarVencedor(), isCheio() e imprimirTabuleiro().

B. Classes Abstratas (Contratos): Classe Base Jogador

Não existe um "jogador genérico"; ou é um humano digitando no teclado, ou é um algoritmo.

- Deve-se criar uma **Classe Abstrata** Jogador.
- **Atributos protegidos:** nome (std::string) e simbolo ('X' ou 'O').
- **Método Virtual Puro:** virtual int escolherJogada(Tabuleiro& tab) = 0;
    - *Nota:* Este método obriga que qualquer tipo de jogador saiba analisar o tabuleiro e devolver um número de 1 a 9.
- **Destrutor Virtual:** virtual ~Jogador() = default; (Obrigatório para garantir a correta liberação de memória polimórfica).

C. Herança e Polimorfismo

A partir da classe Jogador, devem derivar classes concretas:

- JogadorHumano: A implementação de escolherJogada pede ao usuário (via cin) uma posição (1 a 9, sendo 0 para suspender) e valida se está livre.
- JogadorComputador: Classe base para os bots, que se desdobra nas estratégias: BotAleatorio (nível 1) e BotTatico (nível 2/3), cada um com o seu *override* de escolherJogada().

D. O Gestor do Jogo e Smart Pointers

A classe que controla o fluxo do jogo (ex: MotorJogo) não precisa saber se quem joga é humano ou máquina.

- Deve conter um vetor/array de **ponteiros inteligentes**
- O ciclo principal do jogo será simplesmente alternar turnos, chamando a função correspondente de forma polimórfica

4. Menu da Aplicação

O programa deve apresentar um menu com as seguintes opções:

1. **Iniciar/continuar jogo:** Escolher entre novo jogo ou continuar um jogo suspenso.
2. **Escolher quem inicia:** Definir se começa o Humano, o Computador, ou se é aleatório.
3. **Modo de funcionamento:** Escolher o nível de dificuldade do bot (cancela jogos em andamento).
4. **Suspender o jogo:** Grava o estado atual e termina o programa para ser retomado mais tarde *(Nível III)*.
5. **Mostrar o top 10:** Estatísticas dos melhores jogadores *(Nível II)*.
6. **Sair:** Pede confirmação.

5. Especificações e Níveis de Avaliação

Nível I (Vale 4 pontos)

- Implementação do menu (opções 1 básica, 2, 3 e 6).
- Implementação estrita da Arquitetura POO descrita (Tabuleiro, Jogador, JogadorHumano, e BotAleatorio para o nível elementar).
- O motor do jogo deve utilizar polimorfismo através de std::unique_ptr para gerenciar a partida.

Nível II (Vale 7 pontos)

- Tudo do Nível I + Implementação da opção 5 do menu (Top 10).
- **C++ Moderno rigoroso:** É estritamente proibido o uso de new e delete manuais (*raw pointers*). Os jogadores devem ser instanciados com std::make_unique e armazenados em std::unique_ptr. O destrutor virtual atuará automaticamente.
- Leitura em modo texto protegida contra má utilização (validação de *inputs* do usuário).
- Implementação dos modos Básico e Médio do computador através de novas classes derivadas.

Nível III (Vale 10 pontos)

- Tudo do Nível II + Implementação da opção 4 do menu (e complemento da opção 1).
- **Persistência:** O estado do jogo (tabuleiro, estatísticas e **tipos de jogadores**) é salvo de forma persistente em arquivo(s).
- *Desafio POO:* Ao carregar o jogo, o programa deve conseguir instanciar (via make_unique) as classes filhas corretas (Humano ou Computador) com base em um identificador lido do arquivo, restaurando a partida polimórfica perfeitamente.

6. Bibliografia Sugerida

- Jogo da Velha (Wikipedia)
- Tic-tac-toe (Wikipedia)
- Algoritmos Minimax (para alunos curiosos em criar um Bot Imbatível): GeeksforGeeks.