Relatório Técnico: Jogo da Velha com IA (Minimax)

1. Introdução e Contexto

Nome do Projeto e Objetivo

O projeto implementa uma versão completa do jogo **Jogo da Velha**, onde o jogador compete contra um agente inteligente (o computador). O objetivo principal é desenvolver um software funcional em C que demonstre o uso de **estruturas de controle** e o **algoritmo Minimax** para garantir que a IA jogue de forma ótima (maximizando as chances de vitória ou buscando o empate).

Problema Resolvido

O projeto resolve o desafio de simular um jogo de estratégia simples contra uma IA, aplicando conceitos de **ciência da computação** e **Inteligência Artificial (IA)** no contexto da programação em C.

Justificativa da Escolha

O Jogo da Velha com IA (Minimax) foi escolhido por ser um projeto de **complexidade média ou alta** que obriga o uso de **todos os conceitos da Unidade 1**. Exige manipulação de *arrays* (vetores), *loops* de repetição, lógica de estado e, principalmente, a implementação de **múltiplas funções recursivas** para o algoritmo Minimax.

---

2. Análise Técnica

Metodologia e Ferramentas Utilizadas

- **Linguagem de Implementação:** Linguagem C.
- **Compilador:** GCC (GNU Compiler Collection).
- **Editor:** VS Code.
- **Algoritmo Central:** Minimax (para a tomada de decisão da IA).

Aplicação dos Conceitos da U1

Conceito da U1	Aplicação no Código

	Variáveis e Tipos	Uso de	char para o tabuleiro (X, O, VAZIO), int para coordenadas (linha, coluna) e bool (stdbool.h) para controle de estado (jogada_valida, maximizando_jogador).	
	Vetores (Arrays)	O tabuleiro é representado por um	**vetor bidimensional** char tabuleiro[3][3]. O	*array* coluna[N] (embora seja 3, o princípio se aplica) armazena o estado do jogo.
	Comandos Condicionais	Usados em todas as funções: if (tabuleiro[i][j] == VAZIO) para verificar movimentos; if (resultado == 10) para definir o vencedor final; if (pontuação == 10) para determinar o fim da recursão Minimax.		

	Comandos de Repetição	Vários *loops* for são usados para: 1. **Percorrer** o tabuleiro (funções inicializar_tabuleiro, tem_movimentos, checar_vitoria). 2. **Iterar** sobre todas as células disponíveis no Minimax. Um *loop* while é usado para **validação da entrada** (while (!jogada_valida)) e no **loop principal do jogo** (while (tem_movimentos() && checar_vitoria() == 0)).
	Funções	O código é modularizado em diversas funções com responsabilidades claras (mais de 3 além da `main`): `inicializar_tabuleiro()`, `exibir_tabuleiro()`, `tem_movimentos()`, `checar_vitoria()`, `minimax()`, e `encontrar_melhor_movimento()`.

Organização e Estruturas de Dados

O projeto utiliza a seguinte estrutura de dados principal:

- **char tabuleiro[3][3]:** É a estrutura de dados central, representando o estado do jogo em cada passo.

- **struct Movimento:** Uma estrutura simples usada para retornar de encontrar_melhor_movimento() as coordenadas (linha e coluna) da melhor jogada calculada pela IA.

---

3. Implementação e Reflexão

Dificuldades Encontradas e Soluções Implementadas

- **Desafio Principal (Recursão Minimax):** A maior dificuldade foi implementar corretamente a função **minimax()**, pois ela é recursiva e precisa alternar entre o **Maximizador** (IA) e o **Minimizador** (Jogador Humano).
    - **Solução:** A solução foi garantir o **Backtracking** (desfazer a jogada) ao final de cada chamada recursiva (tabuleiro[i][j] = VAZIO;). Isso garante que a função volte ao estado original para explorar o próximo ramo da árvore de decisões.
- **Tratamento de Entrada:** Garantir que o jogador digite números inteiros válidos para as coordenadas e lidar com caracteres não numéricos (limpando o buffer while (getchar() != '\n');).

Organização do Código

O código está organizado em blocos lógicos claros, com comentários de seção:

1. **Definições Globais:** (Constantes JOGADOR, IA, VAZIO).
2. **Inicialização e Exibição:** Funções de interface.
3. **Lógica do Jogo:** Funções de verificação de estado (tem_movimentos, checar_vitoria).
4. **Algoritmo Minimax:** Funções de IA (minimax, encontrar_melhor_movimento).
5. **main():** Controla o fluxo principal do jogo e a interação com o usuário.

Essa separação em funções com responsabilidades claras facilita a manutenção e a legibilidade, especialmente na depuração do algoritmo Minimax.

Conclusão e Aprendizados

O projeto demonstrou com sucesso a aplicação de

estruturas de repetição e **condicionais** para criar um programa interativo, além de utilizar a recursão para implementar um algoritmo de IA eficiente. O principal aprendizado foi aprofundar o entendimento sobre o

backtracking e como manipular variáveis globais (o tabuleiro) dentro de funções recursivas.