📄 Relatório Técnico – Jogo da Velha com IA (Minimax + Alpha-Beta)

main2

1. Introdução e Contexto

Nome do Projeto e Objetivo

O projeto implementa um **Jogo da Velha dinâmico em C**, onde o jogador humano compete contra uma **IA inteligente** que utiliza o algoritmo **Minimax com Poda Alpha-Beta**.
O diferencial do código é permitir jogar em tabuleiros de **3x3 até 5x5**, ajustando automaticamente a profundidade de busca do algoritmo para evitar travamentos.

Problema Resolvido

O jogo simula uma partida estratégica em tempo real entre humano e computador. A IA analisa todas as melhores possibilidades usando Minimax e sempre tenta **maximizar suas chances de vitória ou empatar a partida**.

Justificativa da Escolha

Este projeto foi escolhido porque exige a aplicação de **vários conceitos da Unidade 1**, como:

- **Vetores bidimensionais**,
- **Estruturas condicionais e laços de repetição**,
- **Modularização por funções**,
- **Entrada e validação de dados**,
- **Recursão e implementação de IA (Minimax)**.

Trata-se de um projeto **desafiador e completo**, ideal para consolidar os conteúdos iniciais da disciplina.

---

2. Análise Técnica

Metodologia e Ferramentas Utilizadas

Componente	Utilização
Linguagem	C
Compilador	GCC
Editor	VS Code
Algoritmo Principal	Minimax com Alpha-Beta
Estruturas de Dados	Vetor 2D e struct Movimento

Aplicação dos Conceitos da Unidade 1

Conceito da U1	Aplicação no Código
Variáveis e Tipos	Uso de `char`, `int`, `bool` e `struct`.
Vetores (Arrays)	`char tabuleiro[MAX_TAM][MAX_TAM]` representa o jogo.
Comandos Condicionais	Validação de jogada e verificação de vitória.
Repetição (`for` e `while`)	Utilizados para percorrer o tabuleiro e validar input.
Funções	Código organizado em funções específicas como `inicializar_tabuleiro()`, `checar_vitoria()`, `minimax()` e outras.

Organização e Estruturas de Dados

- `char tabuleiro[MAX_TAM][MAX_TAM]` → estrutura de dados principal.
- `struct Movimento { int linha, coluna; };` → retorna a melhor jogada da IA.
- Constantes globais para controle do jogo (`JOGADOR`, `IA`, `VAZIO`).
  main2

---

3. Implementação e Reflexão

Dificuldades Encontradas

1. **Recursão do Minimax em tabuleiros grandes (4x4 ou 5x5)**
   → Para evitar travamento, o código limitou a profundidade de busca com `max_depth`.

2. **Validação da entrada do usuário**
   → Necessário limpar o buffer com `while (getchar() != '\n');` para evitar erros com caracteres inválidos.

Soluções Implementadas

- Uso de **poda Alpha-Beta** para acelerar o cálculo da IA.

Limite dinâmico de profundidade no Minimax:

```
int max_depth = (TAMANHO == 3) ? 9 : 6;
```

-
- Sistema de exibição dinâmica do tabuleiro com linhas e colunas adaptadas ao tamanho inserido pelo usuário.

Organização do Código

O código segue uma estrutura lógica clara:

1. **Definições Globais e Variáveis**

2. **Funções de Inicialização e Exibição**

3. **Funções de Lógica do Jogo**

4. **Algoritmo Minimax + Alpha-Beta**

5. **Função `main()` – interação com usuário e loop principal**

---

4. Conclusão e Aprendizados

O projeto **cumpriu com sucesso o objetivo de criar um jogo funcional com IA**, aplicando os conceitos da Unidade 1 em um cenário real. O principal aprendizado foi compreender como a **recursão, backtracking e poda Alpha-Beta** tornam a IA eficiente — mesmo em tabuleiros maiores.

Além disso, o trabalho proporcionou prática na **modularização de funções, manipulação de vetores, validação de entrada e uso de estruturas de dados**. Trata-se de um projeto completo e de ótimo nível para consolidação do conteúdo.