Relatório de Projeto: Jogo da Velha com IA (Minimax) – Unidade 2

1. Introdução

O projeto desenvolvido durante a disciplina de Introdução a Técnicas de Programação consiste na implementação do clássico "Jogo da Velha", com o diferencial de permitir uma partida contra o computador (IA). O problema central que o software resolve é a criação de um oponente capaz de calcular os melhores movimentos possíveis.

Nesta etapa (Unidade 2), os objetivos foram expandidos para consolidar o entendimento sobre gerenciamento de memória e manipulação de dados complexos. Além da lógica base do algoritmo *Minimax*, o projeto recebeu atualizações críticas: a correção de falhas lógicas relacionadas a ponteiros, a implementação de um **histórico de jogadas** e a resolução de vazamentos de memória (*memory leaks*). Ademais, houve uma preocupação com a experiência do usuário, implementando uma **exibição gamificada**, onde as mensagens surgem letra por letra.

2. Metodologia

Para o desenvolvimento, foram utilizadas as seguintes ferramentas:

- **Compilador:** GCC (versão 6.3.0).
- **IDE/Editor:** VSCode.

A abordagem seguiu o princípio de "Dividir para Conquistar". A organização do código priorizou a legibilidade e a estrutura modular, o que foi essencial para identificar os erros da versão anterior e implementar as novas funcionalidades sem comprometer a lógica principal, outro grande marco que facilitou os avanços da segunda unidade foi o ajuste das funções que foram separadas logicamente, o que facilitou a identificação de erros da versão anterior e permite que futuras atualizações sejam implementadas sem comprometer a estabilidade do sistema.

3. Análise do Código

Nesta unidade, o foco recaiu sobre a aplicação dos novos conceitos estudados:

Strings

- **Como foram utilizadas:** As strings foram fundamentais para a implementação da interface gamificada (efeito de digitação) e para a estruturação dos textos do jogo.
- **Funções e Manipulação:** Não foi necessário o uso de funções complexas de cópia ou formatação (strcpy ou sprintf). A manipulação se baseou primariamente na função strlen.
- **Exemplos de uso:** Na função de impressão gamificada, o strlen é utilizado para determinar o tamanho exato da mensagem. Com esse valor, um laço de repetição percorre a string imprimindo caractere por caractere com um pequeno *delay*, sem a necessidade de buscar manualmente o terminador \0.

Estruturas de Repetição Aninhadas

- **Aplicação Principal (Minimax):** O uso mais complexo e robusto de estruturas aninhadas ocorre no algoritmo **Minimax**.
- **Complexidade e Funcionamento:** O algoritmo combina **recursão** com **laços de repetição**. A função chama a si mesma recursivamente (para aprofundar na árvore de possibilidades do jogo), mas dentro de cada chamada recursiva, existe um laço de repetição (for ou while) que percorre as posições disponíveis no tabuleiro.
- **Casos de uso:** Essa estrutura aninhada (loop dentro de recursão) é o que permite à IA simular todas as jogadas futuras possíveis antes de tomar uma decisão.

Matrizes

- **Implementação:** Enquanto o tabuleiro do jogo manteve uma lógica de vetor linear para facilitar certas operações, o conceito de **Matriz** (array bidimensional) foi explicitamente aplicado na criação do **Histórico de Jogadas**.
- **Operações:** A matriz serve como um banco de dados temporário, onde cada linha armazena uma string correspondente a uma jogada realizada.
- **Estratégia:** Isso permite acessar e exibir o registro de qualquer turno anterior através de índices de linha (número da jogada) e coluna (conteúdo do texto), facilitando a organização dos dados da partida.
- **Implementação:** Optou-se por um **mapeamento linear**. Ao invés de uma matriz 3x3, utilizou-se um vetor de tamanho 9.
- **Estratégia:** Logicamente, o jogo opera como matriz, mas tecnicamente é um vetor contíguo. Isso simplifica a passagem de parâmetros para funções e a cópia de memória.

Ponteiros e Alocação Dinâmica

- **Utilização Generalizada:** O uso de ponteiros foi intenso nesta etapa.
- **Praticamente todas as funções** do projeto usaram a passagem por referência devido a manipulação do conteúdo, e foi de suma importância o ajuste de ponteiros da versão anterior, tendo em vista que o problema de lógica da unidade passada estava sendo causada por conta de ponteiros indevidos.
- **Passagem por Referência e Deep Copy:** A passagem por referência foi adotada para evitar a duplicação desnecessária de dados na memória e permitir a alteração direta de variáveis. No entanto, isso trouxe a necessidade crítica do **Deep Copy** (cópia profunda). Ao passar o tabuleiro por referência para a IA simular jogadas, era vital não alterar o jogo real. Portanto, implementou-se o Deep Copy para criar clones independentes dos dados apontados, garantindo que a referência manipulada pela IA não afetasse o endereço de memória do tabuleiro original, sendo assim essencial o uso do malloc.

- **Gerenciamento de Memória:** A alocação dinâmica (malloc) é gerenciada com rigor. Para cada alocação (especialmente nas cópias da IA e no histórico), foi implementada a respectiva liberação (free), sanando os vazamentos de memória identificados anteriormente.

4. Dificuldades e Soluções

Principais Desafios Técnicos

O maior desafio foi o controle do fluxo de execução dentro da recursão do Minimax e o gerenciamento correto dos endereços de memória. Inicialmente, a passagem por referência causava efeitos colaterais indesejados, onde a simulação da IA alterava o jogo em andamento.

Superação e Aprendizados

- **Solução via Deep Copy:** A compreensão de que passar um ponteiro dá acesso ao dado original exigiu a implementação manual da cópia profunda dos dados antes da simulação recursiva.
- **Memória:** O aprendizado prático sobre alocação e liberação foi essencial para estabilizar o programa, garantindo que o histórico (matriz) e as simulações não esgotarem os recursos.

5. Conclusão

A evolução para a Unidade 2 transformou o projeto em uma aplicação robusta. A transição para o uso massivo de ponteiros e passagem por referência otimizou o código, enquanto a implementação de matrizes para o histórico e o uso inteligente de strings na interface gamificada enriqueceram a funcionalidade.

O programa agora é estável e visualmente dinâmico. Para atualizações, os passos incluem aprimorar a IA com níveis de dificuldade e refinar a interface com o usuário.