1. **Introdução e Contexto:**

O projeto que optei por trabalhar durante a disciplina de Introdução a técnicas de programação foi o Jogo da velha com IA, o problema que o projeto resolve é a possibilidade de jogar uma partida de jogo da velha sozinho, onde uma IA responde com os melhores movimentos, a minha justificativa para a escolha desse projeto foi devido a experiencia de conhecer algoritmos que envolvessem IA e tivesse um certo grau de dificuldade, e que dentro dessa dificuldade houvesse aprendizado na área.

2. **Análise Técnica:**

2.1. Metodologia:

Sobre as Ferramentas utlizadas:

- Compilador: gcc - versão: 6.3.0;
- Editor: VScode;
- Bibliotecas: stdio.h, stdlib.h, stdbool.h e math.h;

Sobre as bibliotecas usadas:

A biblioteca stdio, foi usada para as tarefas basícas, como scanf (para receber as entradas)

A biblioteca stdlib foi usada para a utilização da alocação dinâmica por meio da função malloc, essa função foi essencial na parte da criação de um tabuleiro base, e além dessa utilização também foi primordial nadDeep Copy (Copia profunda) que é amplamente utilizado na função principal.

A biblioteca stdbool a princípio foi usada para criar vetores binários, porém durante as atualizações do projeto, percebi que os vetores binários não poderiam ser usados, pois não conseguimos mapear as 3 opções (Vazio, “x” e “o”), então após isso preferi definir um enum, correspondente aos possíveis valores do tabuleiro, além dessa primeira utilização, ela também foi usada para criação de uma função que retorna um vetor booleano o qual retorna os espaços vazios dentro do vetor que é usado de referência.

A biblioteca math foi usada para fazer o cálculo, de um número binário para um número decimal, vamos discutir no 2.2.

2.2. Aplicação dos Conceitos da U1

Antes de falar sobre a criação de uma estrutura de dados que foi necessária para representar o tabuleiro de forma legível, é correspondente ao typedef enum, que cria os possíveis valores, sendo os valores de “x”, “o” e o valor nulo (não chamei de vazio para não confundir com tipo null) que representa um espaço vazio, esse tipo é nomeado de valores.

Sobre as funções, existem 11 funções, e para apresentá-las vamos dividir em 3 categorias de acordo com a complexidades, sendo elas: baixa complexidade, média complexidade, alta complexidade;

- Baixa Complexidade:
    1. criarTabuleiroBase:

Ela retorna o endereço do tabuleiro de tipo valores de acordo com o tamanho passado como argumento (vale ressaltar que esse tamanho é do tipo size_t devido a manipulação de um vetor) e inicializa as posições como valores nulos ou vazio;

2. getTabuleiro:

Ela é do tipo void e imprime no terminal o tabuleiro que é passada como argumento;

3. setPosicao:

É do tipo void e ela apenas modifica o tabuleiro com os argumentos passados, que são o tabuleiro a ser modificado, a posição que queremos alterar e o valor da jogada;

4. valoresParaDecimal:

Como usamos apenas um vetor para ser correspondente ao tabuleiro, e ele é de tipo valor, que são apenas 0 para jogador “o”, 1 para “x” e –1 para valor nulo, podemos fazer a transformação para um valor decimal, esse valor é dependente de que ponto de vista estamos olhando, basicamente fazemos a transformação de binario para decimal quando a posição do tabuleiro for igual a o valor que estamos analisando o ponto de vista.

5. verificaGanho:

Usa a função valoresParaVecimal e de acordo com o resultado verifica a posição bit a bit para ver se há uma correspondência com as posições de vitória, retornando 1 para se há um vencedor

6. temEspacos:

Percorre o tabuleiro e ao encontrar um espaço, retorna true se houver um espaço, e false se não achar.

7. copiarVetor:

Usando o conceito de deep copy (cópia profunda), basicamente criamos um vetor idêntico ao original, ao invés de fazer uma shallow copy (cópia superficial) que apenas os ponteiros, ao invés dos dados.

8. possiveisJogadas:

Retorna em um vetor de booleanos o qual onde a posição no tabuleiro é um espaço vazio, o seu correspondente no vetor de booleanos é 1;

- Média Complexidade:
  9. jogada:

Essa função ela é usada na função principal (minimax), ela é considerada média complexidade devido o conceito abstrato de simular jogadas que não ocorreram, sem modificar o tabuleiro original, ela cria tabuleiros por meio

copiarVetor e altera os valores por meio de setTabuleiro aplicando nesse novo vetor que o copiarVetor criou;

10. melhorJogada:

Parecido com o minimax em relação a criação de um vetor resultado, que é aplicação de jogada no tabuleiro, ele aplica minimax no resultado buscando maximizar os ganhos, e se for o maior dos atuais, ele atualiza os maior e a melhor posição recebe a posição que gerou esse resultado, após disso retorna a melhor posição;

- Alta Complexidade:

11. minimax:

A função minimax é a principal, pois usamos uma lógica, onde buscamos um caminho o qual procuramos a melhor posição para maximar os resultados, buscando sempre a vitoria e escapando da derrota, a minimax usa ela recursivamente, então ela precisa das condições de parada, sendo elas a vitoria do jogador que estamos buscando (no caso no nosso jogo é a IA), nesse caso vai retornar 1, outro caso de parada é quando há uma vitoria e não é a IA, nesse caso será retornado –1, se não houver mais espaços retornará 0.

Após essas condições de paradas criamos um vetor de booleanos que vai receber o valor das possíveis jogadas do tabuleiro em questão, e temos 2 casos, sendo eles a jogada da vez sendo a IA e não sendo, se caso for verdadeiro buscamos maximizar os ganhos, criamos um tabuleiro de resultados onde recebe a aplicação de jogada no tabuleiro atual, que no caso serão todas as jogadas, então criamos 2 variaveis inteiras que corresponderão a maior posição e o valor que recebe a aplicação de minimax no tabuleiro resultados, ai se caso esse valor for maior que o atual maior, apenas atualizamos o maior, o mesmo ocorre quando a jogada atual não é nossa, porém nesse caso buscamos minimizar, então ao invés de maior usamos menor e fazemos o mesmo procedimento;

No main, apenas criamos um tabuleiro e pedimos que o usuário escolha quem ele quer ser no caso “x” ou “o”, após isso entramos em um while que não saímos dele enquanto não tenha nenhum ganhador, e ainda tenha espaços, e pedimos que o usuário escolha se quer ver o tabuleiro ou escolher uma jogada, após ele escolher uma jogada, escolhemos a melhor jogada para IA de acordo com o tabuleiro, após a saída do while, imprimimos o tabuleiro.

3. Implementação e Reflexão:

3.1. Dificuldades Encontradas e Soluções:

Uma das principais dificuldades foi encontrar uma forma de fazer a resposta por meio de IA, a princípio já sabia que seria uma certa dificuldade em fazer todo esse pensamento, pois bem, ao procurar soluções no caso encontrei o algoritmo minimax e entrei de cabeça, aprendendo como fazer essa implementação, ao tentar implementar encontrei outras dificuldades, sendo elas: como manipular o tabuleiro de forma simples, sem usar uma matriz a solução para esse caso veio de uma conversa sobre

outro caso, onde me falaram sobre o Bitwise e seus operadores, pois bem, a resposta desse caso foi a criação de valores e a função que retorna os decimais, outro grande problema foi em relação a cópia de um vetor, pois a princípio não estava conseguindo criar um novo vetor, e ao buscar como fazer isso encontre os casos de deep copy e shallow copy, a implementação está na função de copiar vetor, a minha atual dificuldade está em aprimorar a inteligência artificial, onde em alguns momentos ela não está respondendo corretamente.

3.2. Organização:

A estrutura do código em relação a ordem das funções, foi escrita antes as funções primordiais e logo após elas foram suas dependentes, e a estrutura foi de suma importância a existência das funções, devido a manutenção do código e usando o princípio de dividir para conquistar, onde dividimos o problema em casos menores e resolvemos eles para resolvermos o problema como um todo.

4. Conclusão

Os conceitos da unidade foram utilizados em diversos momentos no programa, como a criação das variáveis em todos os instantes, as estruturas condicionais na verificação de quem ganhou, os laços de repetição no main, para começar um jogo e mantê-lo, os vetores sendo usados para representar o tabuleiro e as funções sendo as principais em todos os momentos.

O programa tem sua primeira versão, e nessa primeira versão me deu contato com novos algoritmos como o minimax, e me trouxe conceitos importantíssimos como o deep copy, apesar dos meus contatos com a linguagem C, nunca tive o contato com a criação de uma Inteligência Artificial e vejo que nesse caso ainda tem muitas oportunidades de melhora, pois, o programa ainda tem seus problemas de jogadas erradas que podem ser resolvidas, ainda existem boas melhorias que busco implementar, sendo elas:

- Acrescentar a opção de dificuldades;
- Aprimorar as funções;
- Otimizar o Código, pois ele ainda está muito bruto;

Essas alterações estão na lista de implementação da segunda versão desde projeto!