

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

LINGUAGENS DE PROGRAMAÇÃO

Jogo da Velha $N \times N$

Brenda Leite e Lima
Daniel Reis

Professor - Camilo Leles

João Monlevade
5 de julho de 2017

Sumário

1	Introdução	1
2	Testes de execução	2
3	Código em Prolog	4

1 Introdução

Este trabalho consiste no desenvolvimento de um jogo utilizando o paradigma lógico, nesse caso, a linguagem Prolog. Nesse âmbito, desenvolveu-se o tema sugerido: Jogo da Velha N x N, em que é preciso passar o N (Tamanho da matriz). Depois disso, o jogo foi feito de modo que dois usuários possam jogar, sem a opção usuário x computador. Vale ressaltar que todas tentativas de jogadas são validadas. Também, é verificado se o indivíduo ganhou com base em alguns quesitos (ter alguns desses preenchidos com 1 ou 2 - X ou O respectivamente): uma linha, coluna, diagonal principal ou diagonal secundária. O que pode ocorrer também é dar velha, ou seja, um empate - não se consegue atingir nenhum dos 4 citados anteriormente. É de suma importância mencionar que nesse jogo, utiliza-se uma matriz que é armazenada sob a forma de lista de listas, em que guarda-se várias linhas, como no exemplo abaixo:

[[0 , 0 , 0] , [0 , 0 , 0] , [0 , 0 , 0]]										[[1 , 2 , 0] , [1 , 0 , 2] , [2 , 0 , 1]]									
			1		2		3						1		2		3		
1									1	1		X		O				1	
2									2	2		X				O		2	
3									3	3		O				X		3	
			1		2		3						1		2		3		

Como mostrado acima, a forma de representação utilizada foi:

0 -> Vazio
1 -> Jogada do Jogador 1 (X)
2 -> Jogada do Jogador 2 (O)

2 Testes de execução

```
      1  2  3
1 | X | X | X | 1
2 | 0 | 0 |   | 2
3 |   |   |   | 3
      1  2  3

Parabens brenda voce ganhou!
false.
?- 
```

Figura 1: Ganhando pela Linha.

```
      1  2  3
1 | X | 0 |   | 1
2 | X | 0 |   | 2
3 | X |   |   | 3
      1  2  3

Parabens brenda voce ganhou!
false.
?- 
```

Figura 2: Ganhando pela Coluna.

```
      1  2  3
1 | X | 0 | X | 1
2 | 0 | X | 0 | 2
3 |   |   | X | 3
      1  2  3

Parabens brenda voce ganhou!
false.
?- 
```

Figura 3: Ganhando pela Diagonal Principal.

	1	2	3	
1	X	0	X	1
2	0	X	0	2
3	X			3

Parabens brenda voce ganhou!
false.

?- ☐

Figura 4: Ganhando pela Diagonal Secundária.

	1	2	3	
1	X	X	0	1
2	0	0	X	2
3	X	0	X	3

Empate
false.

?- ☐

Figura 5: Empate.

3 Código em Prolog

Nesta seção está disponibilizado o código completo do jogo.

Programa 1: Jogo da Velha N x N usando Prolog

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
%%%%%%%%%%%%%% JOGO DA VELHA NxN – BRENDA LEITE LIMA, DANIEL REIS %%%%%%%%%%%%%%%  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% ESTADO INICIAL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
  
insere(X,Lista,[X|Lista]). %Insere um valor no inicio da lista  
  
%Pra gerar uma linha, e preciso inserir um valor(0) N vezes dentro de  
um vetor  
gera_linha(N,[]):-N=0,!.  
  
%  
Quando n for 0, o vetor esta vazio  
gera_linha(N,Vetor):-N>0, N1 is N-1, insere(0, Vetor1, Vetor),  
gera_linha(N1,Vetor1). %Decrementa o n e vai insere um 0 no vetor  
ate preencher todo com zeros  
  
%Gera varias linhas, ou seja, gera o estado inicial. Entra com um N, N  
e retorna a matriz preenchida  
gera_matriz(N,Linha,[]):-N>0, Linha=0, !. %Cria e insere a primeira  
linha na matriz  
gera_matriz(N,Linha,Matriz):-N>0, Linha>0, Linha1 is Linha-1,  
gera_linha(N,Vetor), insere(Vetor, Mat, Matriz), gera_matriz(N,  
Linha1, Mat).  
  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MENU %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
  
inicio:-leitura(N), inicia_jogo(N), !. %Solicita leitura do tamanho e  
nome, pra depois iniciar o jogo  
  
leitura(N):-repeat, dados, read(N), N>0, !. %Repete a leitura enquanto  
o valor do N nao for maior que zero  
  
limpa_tela:-write('\e[H\e[2J'). %Apaga os dados da tela  
  
dados:-limpa_tela,  
write('Jogo da Velha NxN - Brenda Leite Lima e Daniel Reis'), nl,  
nl,  
write('Insira o valor de N(Dimensao do jogo), em que N>0:'), nl.  
  
insira_nome_jogador_1(Nome):- write('Insira o nome do jogador 1: '), nl  
, read(Nome).  
insira_nome_jogador_2(Nome):- write('Insira o nome do jogador 2: '), nl  
, read(Nome).  
  
%Gera a matriz preenchida com 0(espacos em branco)  
inicia_jogo(N):-  
limpa_tela, insira_nome_jogador_1(NomeJogador1), %Solicita o
```

```

    nome do jogador 1
limpa_tela, insira_nome_jogador_2(NomeJogador2),      %Solicita o
    nome do jogador 2
limpa_tela, gera_matriz(N, N, Matriz),                %Gera a matriz
    vazia com o tamanho N por N, ou seja, toda preenchida com 0
limpa_tela, visualiza_estado(N, Matriz, 1, NomeJogador1,
    NomeJogador2), jogo(N, Matriz, NomeJogador1, NomeJogador2).
    %Mostra o estado inicial do jogo e inicia

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  VISUALIZA O ESTADO DO JOGO  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Visualiza o estado atual da matriz
visualiza_estado(N, Matriz, Jogador, NomeJogador1, NomeJogador2):-
    limpa_tela,
    nl, write(' '), gera_sequencia(N, 1, Sequencia), mostra_sequencia(
        Sequencia), nl, %Mostra uma uma sequencia (1,2,3...,N)
    mostra_linhas(1,Matriz),
                                                                    %
        Imprime a matriz linha por linha
    write(' '), gera_sequencia(N, 1, Sequencia), mostra_sequencia(
        Sequencia), nl, %Mostra uma uma sequencia (1,2,3...,N)
    imprime_jogador(Jogador, NomeJogador1, NomeJogador2), !.
                                                                    %Imprime quem devera jogar no
        momento

imprime_jogador(Jogador, _, _):-Jogador=0, !. %Usado pra mostrar o
    estado final do jogo, em que ninguem mais deve jogar, pois o jogo
    acabou
imprime_jogador(Jogador, NomeJogador1, _):-Jogador=1, write('Jogada de
'), write(NomeJogador1), !. %Mostra que quem deve jogar agora e o
    player1
imprime_jogador(Jogador, _, NomeJogador2):-Jogador=2, write('Jogada de
'), write(NomeJogador2). %Mostra que quem deve jogar agora e o
    player2

%Gera uma sequencia (1,2,3...,N)
gera_sequencia(N,_,[]):-N=0,!.
gera_sequencia(N,Inicio,Vetor):-N>0, N1 is N-1, Inicio1 is Inicio+1,
    insere(Inicio, Vetor1, Vetor), gera_sequencia(N1,Inicio1,Vetor1).

%Imprime a sequencia gerada
mostra_sequencia([]).
mostra_sequencia([Linha|Resto]):-imprime_valor(Linha), mostra_sequencia
    (Resto).

imprime_valor(V):-V<10, write(' '), write(V).
imprime_valor(V):-V=10, write(' '), write(V).
imprime_valor(V):-V>10, write(' '), write(V).

%Imprime todas as linhas
mostra_linhas(_,[]). %Pra qualquer N(numero da linha),
    se a lista tiver vazia pare
mostra_linhas(N,[Linha|Resto]):-
    imprimeN(N), write(' | '), %Escreve a referencia da linha atual
    mostra_linha(Linha), %Escreve os dados da linha

```

```

        write(' '), write(N), nl,           %Da um espaco, escreve a referencia
        da linha atual e pula uma linha
    N1 is N+1,                               %Avanca o contador pra proxima linha
    mostra_linhas(N1, Resto).                %Recurso: manda imprimir o restante,
        ou seja, avanca pra proxima linha

imprimeN(N):-N<10, write(' '), write(N).
imprimeN(N):-N=10, write(N).
imprimeN(N):-N>10, write(N).

%Imprime cada elemento da linha lado a lado, um a um
mostra_linha([]).

%Quando a
    lista estiver vazia pare
mostra_linha([Elemento|Resto]):-escreve(Elemento), mostra_linha(Resto).
    %Escreve o elemento e passa pro proxima da mesma linha

escreve(0):-write('  |').    %Imprime vazio, pois 0 na matriz que
    representa vazio
escreve(1):-write(' X |').    %Impriem X, pois 1 na matriz que
    representa a jogada do player 1
escreve(2):-write(' O |').    %Imprime O, pois 2 na matriz representa a
    jogada do player 2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% JOGO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Execucao do jogo
jogo(N, Matriz, NomeJogador1, NomeJogador2):-
    leitura_posicao(N, Matriz, Linha1, Coluna1),           %
        Solicita a posicao em que o player1 quer jogar
    realiza_jogada(Matriz, Linha1, Coluna1, 1, Matriz1),    %
        Atualiza a matriz - Coloca o identificador do player1 na
        posicao escolhida
    testa_fim_de_jogo(N, Matriz1, NomeJogador1, NomeJogador2), %
        Testa se o jogo deve terminar ou nao com base na matriz
        atualizada
    visualiza_estado(N, Matriz1, 2, NomeJogador1, NomeJogador2), %
        Imprime o novo estado do jogo, indicando que a proxima jogada e
        do player2

    leitura_posicao(N, Matriz1, Linha2, Coluna2),           %
        Solicita a posicao em que o player2 quer jogar
    realiza_jogada(Matriz1, Linha2, Coluna2, 2, Matriz2),    %
        Atualiza a matriz - Coloca o identificador do player2 na
        posicao escolhida
    testa_fim_de_jogo(N, Matriz2, NomeJogador1, NomeJogador2), %
        Testa se o jogo deve terminar ou nao com base na matriz
        atualizada
    visualiza_estado(N, Matriz2, 1, NomeJogador1, NomeJogador2), %
        Imprime o novo estado do jogo, indicando que a proxima jogada e
        do player1

    jogo(N, Matriz2, NomeJogador1, NomeJogador2).           %
        Recursao - Chama o jogo novamente

```



```

%Testa se o jogo deve continuar ou nao
testa_fim_de_jogo(N, Matriz, NomeJogador1, NomeJogador2):-
    not(verifica_fim_de_jogo(N, Matriz));    %Verifica fim de jogo
        sempre retorna true se o jogo deve continuar. Caso retorne
        false, cai pra linha de baixo
    mostra_ganhador(N, Matriz, NomeJogador1, NomeJogador2), !, fail.
        %Imprime o estado final do jogo e quem ganhou, alem de parar
        o jogo

%Imprime o estado final do jogo e quem ganhou
mostra_ganhador(N, Matriz, NomeJogador1, NomeJogador2):-
    verifica_ganhador(N, Matriz, JogadorGanhador),                                %
        Descobre quem e o ganhador
    visualiza_estado(N, Matriz, 0, NomeJogador1, NomeJogador2),                    %
        Imprime o estado final do jogo
    mostra_jogador(JogadorGanhador, NomeJogador1, NomeJogador2).                  %
        Imprime quem ganhou o jogo

%Imprime quem ganhou
mostra_jogador(Id, _, _):-
    Id=0, nl, write('Empate'), nl.
mostra_jogador(Id, NomeJogador1, _):-
    Id=1, nl, write('Parabens '), write(NomeJogador1), write(' voce
        ganhou!'), nl.
mostra_jogador(Id, _, NomeJogador2):-
    Id=2, nl, write('Parabens '), write(NomeJogador2), write(' voce
        ganhou!'), nl.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% SOLICITA JOGADA %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Leitura das posicoes - Repete enquanto nao for valida
leitura_posicao(N, Matriz, Linha, Coluna):-
    repeat,
    nl, write('Digite uma posicao - Linha: '), nl, read(Linha),                    %
        Solicita a linha e le o valor
    nl, write('Digite uma posicao - Coluna: '), nl, read(Coluna),                    %
        Solicita a coluna e le o valor
    (Linha>0, Linha<N; Linha=N), (Coluna>0, Coluna<N; Coluna=N),                    %
        Linha e coluna escolhida deve estar dentro o intervalo das
        dimensoes da matriz
    get_posicao(Matriz, Linha, Coluna, Elemento), Elemento=0, !.                    %
        Posicao correspondente a linha e coluna escolhida deve estar
        vazia

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% CAPTURA O VALOR DE UMA POSICAO DA MATRIZ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Pega uma linha da matriz
get_linha(Cont, Indice, [H|_], H):- Indice=Cont, !.
get_linha(Cont, Indice, [_|T], R):- Cont1 is Cont+1, get_linha(Cont1,
    Indice, T, R).

%Pega um elemento da linha da matriz
get_elemento_da_linha(Cont, Indice, [H|_], H):- Indice=Cont, !.
get_elemento_da_linha(Cont, Indice, [_|T], R):- Cont1 is Cont+1,

```

```

get_elemento_da_linha(Cont1, Indice, T, R).

%Pega um elemento de uma posicao da matriz
get_posicao(Matriz, LinhaProcurada, ColunaProcurada, ElementoEncontrado) :-
    get_linha(1, LinhaProcurada, Matriz, R), get_elemento_da_linha(1,
        ColunaProcurada, R, ElementoEncontrado).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% ATUALIZA O VALOR DE UMA POSICAO DA MATRIZ – JOGADA %%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Seta na matriz aquela posicao(linha,coluna) o valor do player (1 ou 2
- X ou O respectivamente)
set_posicao([_|T], 1, Jogador, [Jogador|T]) :- !.
set_posicao([H|T], Indice, Jogador, [H|R]) :- Indice > 0, Indice1 is
    Indice - 1, set_posicao(T, Indice1, Jogador, R).

%Realiza jogada
realiza_jogada(Matriz, Linha, Coluna, Jogador, NovaMatriz) :-
    get_linha(1, Linha, Matriz, R), %Pega a
        linha em que se esta realizando a jogada
    set_posicao(R, Coluna, Jogador, NovaLinha), %Atualiza a
        lista correspondente a aquela linha
    set_posicao(Matriz, Linha, NovaLinha, NovaMatriz), !. %Atualiza a
        lista de listas – Coloca a linha atualizada

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% VERIFICA SE JA GANHOU %%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Verifica o fim de jogo. Nao e necessario saber quem ganhou
verifica_fim_de_jogo(N, Matriz) :-
    verifica_ganhador(N, Matriz, _), !.

%Verifica o fim de jogo pelas condicoes abaixo e retorna o ganhador
verifica_ganhador(N, Matriz, JogadorGanhador) :-
    verifica_fim_de_jogo_pela_linha(N, Matriz, JogadorGanhador), !;
        %Testa se existe alguma linha toda preenchida
        por 1 ou 2
    verifica_fim_de_jogo_pela_coluna(N, Matriz, JogadorGanhador), !;
        %Testa se existe alguma coluna toda preenchida
        por 1 ou 2
    verifica_fim_de_jogo_pela_diagonal_principal(N, Matriz,
        JogadorGanhador), !; %Testa se a diagonal principal esta
        toda preenchida por 1 ou 2
    verifica_fim_de_jogo_pela_diagonal_secundaria(N, Matriz,
        JogadorGanhador), !; %Testa se a diagonal secundaria esta
        toda preenchida por 1 ou 2
    verifica_fim_de_jogo_deu_velha(N, Matriz, JogadorGanhador), !.
        %Testa se a matriz esta toda preenchida – Nao
        existe nenhum valor 0 (Lugar vazio)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% TESTA AS CONDICAOES PRA GANHAR – VALORES IGUAIS %%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Verifica se um elemento e um membro de uma lista
membro(X, [X|_]) :- !.

```

```
membro(X,[_|T]):-membro(X,T).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%Testa se tem alguma linha com todos valores 1 ou 2 (X ou O  
respectivamente)
```

```
verifica_fim_de_jogo_pela_linha(N, Matriz, 1):-testa_1_linha(N, 1,  
    Matriz), !. %Testa se existe alguma linha em que todos elementos  
    sao 1, a partir da linha 1
```

```
verifica_fim_de_jogo_pela_linha(N, Matriz, 2):-testa_2_linha(N, 1,  
    Matriz), !. %Testa se existe alguma linha em que todos elementos  
    sao 2, a partir da linha 1
```

```
%Testa se existe alguma linha em que todos elementos sao 1
```

```
testa_1_linha(_, Linha, Matriz):-
```

```
    get_linha(1, Linha, Matriz, R), %Pega a  
    linha
```

```
    membro(1, R), not(membro(0, R)), not(membro(2, R)), !. %Testa se a  
    linha so tem 1 - Nao tem 0 nem 2
```

```
%Recursao - Passa pra proxima linha
```

```
testa_1_linha(N, Linha, Matriz):-
```

```
    (Linha<N; Linha=N),
```

```
    Linha1 is Linha+1,
```

```
    testa_1_linha(N, Linha1, Matriz), !.
```

```
%Testa se existe alguma linha em que todos elementos sao 2
```

```
testa_2_linha(_, Linha, Matriz):-
```

```
    get_linha(1, Linha, Matriz, R), %Pega a  
    linha
```

```
    membro(2, R), not(membro(0, R)), not(membro(1, R)), !. %Testa se a  
    linha so tem 2 - Nao tem 0 nem 1
```

```
%Recursao - Passa pra proxima linha
```

```
testa_2_linha(N, Linha, Matriz):-
```

```
    (Linha<N; Linha=N),
```

```
    Linha2 is Linha+1,
```

```
    testa_2_linha(N, Linha2, Matriz), !.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%Testa se tem alguma coluna com todos valores 1 ou 2 (X ou O  
respectivamente)
```

```
verifica_fim_de_jogo_pela_coluna(N, Matriz, 1):-testa_1_colunas(N,  
    Matriz, 1, 1), !. %Testa se existe alguma coluna em que todos  
    elementos sao 1
```

```
verifica_fim_de_jogo_pela_coluna(N, Matriz, 2):-testa_2_colunas(N,  
    Matriz, 1, 1), !. %Testa se existe alguma coluna em que todos  
    elementos sao 2
```

```
%Testa todas colunas buscando algum elemento diferente de 1
```

```
testa_1_colunas(N, Matriz, Linha, Coluna):-
```

```
    (Coluna<N; Coluna=N),
```

```
    Coluna1 is Coluna+1,
```

```
    (not(testa_1_coluna(N, Matriz, Linha, Coluna)); testa_1_colunas(N,  
    Matriz, Linha, Coluna1)), !. %Se nao encontrou, passa pra  
    proxima coluna
```

```
%Testa cada coluna buscando algum elemento diferente de 1
```

```

testa_1_coluna(_, Matriz, Linha, Coluna):-
    get_posicao(Matriz, Linha, Coluna, ElementoEncontrado),      %Pega o
        elemento da matriz correspondente aquela posicao (linha,
        coluna)
    ElementoEncontrado\=1, !.                                     %Testa
        se existe algum elemento diferente de 1
%Recurcao - Passa pra proxima linha
testa_1_coluna(N, Matriz, Linha, Coluna):-
    (Linha<N; Linha=N),
    Linha1 is Linha+1,
    testa_1_coluna(N, Matriz, Linha1, Coluna).

%Testa todas colunas buscando algum elemento diferente de 2
testa_2_colunas(N, Matriz, Linha, Coluna):-
    (Coluna<N; Coluna=N),
    Coluna2 is Coluna+1,
    (not(testa_2_coluna(N, Matriz, Linha, Coluna)); testa_2_colunas(N,
        Matriz, Linha, Coluna2)), !. %Se nao encontrou, passa pra
        proxima coluna

%Testa cada coluna buscando algum elemento diferente de 2
testa_2_coluna(_, Matriz, Linha, Coluna):-
    get_posicao(Matriz, Linha, Coluna, ElementoEncontrado),      %Pega o
        elemento da matriz correspondente aquela posicao (linha,
        coluna)
    ElementoEncontrado\=2, !.                                     %Testa
        se existe algum elemento diferente de 2
%Recurcao - Passa pra proxima linha
testa_2_coluna(N, Matriz, Linha, Coluna):-
    (Linha<N; Linha=N),
    Linha2 is Linha+1,
    testa_2_coluna(N, Matriz, Linha2, Coluna).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Testa se tem alguma diagonal principal com todos valores 1 ou 2 (X ou
    O respectivamente)
verifica_fim_de_jogo_pela_diagonal_principal(N, Matriz, 1):-not(
    testa_1_diagonal_principal(N, Matriz, 1, 1)), !. %Testa se nao
    encontrou nenhum 0 ou 2 - So tem 1
verifica_fim_de_jogo_pela_diagonal_principal(N, Matriz, 2):-not(
    testa_2_diagonal_principal(N, Matriz, 1, 1)), !. %Testa se nao
    encontrou nenhum 0 ou 1 - So tem 2

testa_1_diagonal_principal(_, Matriz, Linha, Coluna):-
    get_posicao(Matriz, Linha, Coluna, ElementoEncontrado), %Pega o
        elemento da matriz correspondente aquela posicao (linha, coluna
        )
    (ElementoEncontrado=0; ElementoEncontrado=2), !.             %Tenta
        encontrar algum 0 ou 2. Se nao achar, e porque so tem 1
%Recurcao - Avanca uma linha e uma coluna, ja que se esta andando na
    diagonal principal
testa_1_diagonal_principal(N, Matriz, Linha, Coluna):-
    (Linha<N; Linha=N), (Coluna<N; Coluna=N),
    Linha1 is Linha+1, Coluna1 is Coluna+1,
    testa_1_diagonal_principal(N, Matriz, Linha1, Coluna1).

```

```

testa_2_diagonal_principal(_, Matriz, Linha, Coluna):-
    get_posicao(Matriz, Linha, Coluna, ElementoEncontrado), %Pega o
        elemento da matriz correspondente aquela posicao (linha, coluna
        )
    (ElementoEncontrado=0; ElementoEncontrado=1), !. %Tenta
        encontrar algum 0 ou 1. Se nao achar, e porque so tem 2
%Recurso - Avanca uma linha e uma coluna, ja que se esta andando na
diagonal principal
testa_2_diagonal_principal(N, Matriz, Linha, Coluna):-
    (Linha<N; Linha=N), (Coluna<N; Coluna=N),
    Linha2 is Linha+1, Coluna2 is Coluna+1,
    testa_2_diagonal_principal(N, Matriz, Linha2, Coluna2).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Testa se tem alguma diagonal secundaria com todos valores 1 ou 2 (X ou
O respectivamente)
verifica_fim_de_jogo_pela_diagonal_secundaria(N, Matriz, 1):-not(
    testa_1_diagonal_secundaria(N, Matriz, 1, N)), !. %Testa se nao
encontrou nenhum 0 ou 2 - So tem 1
verifica_fim_de_jogo_pela_diagonal_secundaria(N, Matriz, 2):-not(
    testa_2_diagonal_secundaria(N, Matriz, 1, N)), !. %Testa se nao
encontrou nenhum 0 ou 1 - So tem 2

testa_1_diagonal_secundaria(_, Matriz, Linha, Coluna):-
    get_posicao(Matriz, Linha, Coluna, ElementoEncontrado), %Pega o
        elemento da matriz correspondente aquela posicao (linha, coluna
        )
    (ElementoEncontrado=0; ElementoEncontrado=2), !. %Tenta
        encontrar algum 0 ou 2. Se nao achar, e porque so tem 1
%Recurso - Avanca uma linha e volta uma coluna, ja que se esta andando
na diagonal secundaria
testa_1_diagonal_secundaria(N, Matriz, Linha, Coluna):-
    (Linha<N; Linha=N), (Coluna<N; Coluna=N),
    Linha1 is Linha+1, Coluna1 is Coluna-1,
    testa_1_diagonal_secundaria(N, Matriz, Linha1, Coluna1).

testa_2_diagonal_secundaria(_, Matriz, Linha, Coluna):-
    get_posicao(Matriz, Linha, Coluna, ElementoEncontrado), %Pega o
        elemento da matriz correspondente aquela posicao (linha, coluna
        )
    (ElementoEncontrado=0; ElementoEncontrado=1), !. %Tenta
        encontrar algum 0 ou 1. Se nao achar, e porque so tem 2
%Recurso - Avanca uma linha e volta uma coluna, ja que se esta andando
na diagonal secundaria
testa_2_diagonal_secundaria(N, Matriz, Linha, Coluna):-
    (Linha<N; Linha=N), (Coluna<N; Coluna=N),
    Linha2 is Linha+1, Coluna2 is Coluna-1,
    testa_2_diagonal_secundaria(N, Matriz, Linha2, Coluna2).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Testa se deu velha
verifica_fim_de_jogo_deu_velha(N, Matriz, 0):-not(testa_0_linha(N, 1,
Matriz)), !. %Testa se nao encontrou nenhum 0 - Se nao existe
lugar vazio na matriz

```

```

testa_0_linha(_, Linha, Matriz):-
    get_linha(1, Linha, Matriz, R), %Pega uma linha
    membro(0, R), !. %Tenta encontrar algum 0 na linha.
    Se nao achar, e porque so tem 1 ou 2
%Recurcao - Avanca pra proxima linha
testa_0_linha(N, Linha, Matriz):-
    (Linha<N; Linha=N),
    Linha0 is Linha+1,
    testa_0_linha(N, Linha0, Matriz), !.

```
