Programação em Lógica  
IRIS 4

short line

Leonor Martins de Sousa 201705377

Sílvia Jorge Moreira da Rocha 201704684

MIEIC 3 - Turma 1

Prof. Nuno Fonseca

Prof. Daniel Castro Silva

Prof. Rui Camacho

**Índice**

[Introdução 2](#_Toc24835829)

[O Jogo 3](#_Toc24835830)

[História 3](#_Toc24835831)

[Material 3](#_Toc24835832)

[Regras 3](#_Toc24835833)

[Pontuação e Vencedor 4](#_Toc24835834)

[A Lógica 5](#_Toc24835835)

[Representação do Estado do Jogo 5](#_Toc24835836)

[Visualização do Tabuleiro 7](#_Toc24835837)

[Lista de Jogadas Válidas 8](#_Toc24835838)

[Execução de Jogadas 8](#_Toc24835839)

[Final do Jogo 8](#_Toc24835840)

[Avaliação do Tabuleiro 8](#_Toc24835841)

[Jogada do Computador 8](#_Toc24835842)

[Conclusões 9](#_Toc24835843)

[Bibliografia 10](#_Toc24835844)

# 

# 

# Introdução

# O Jogo

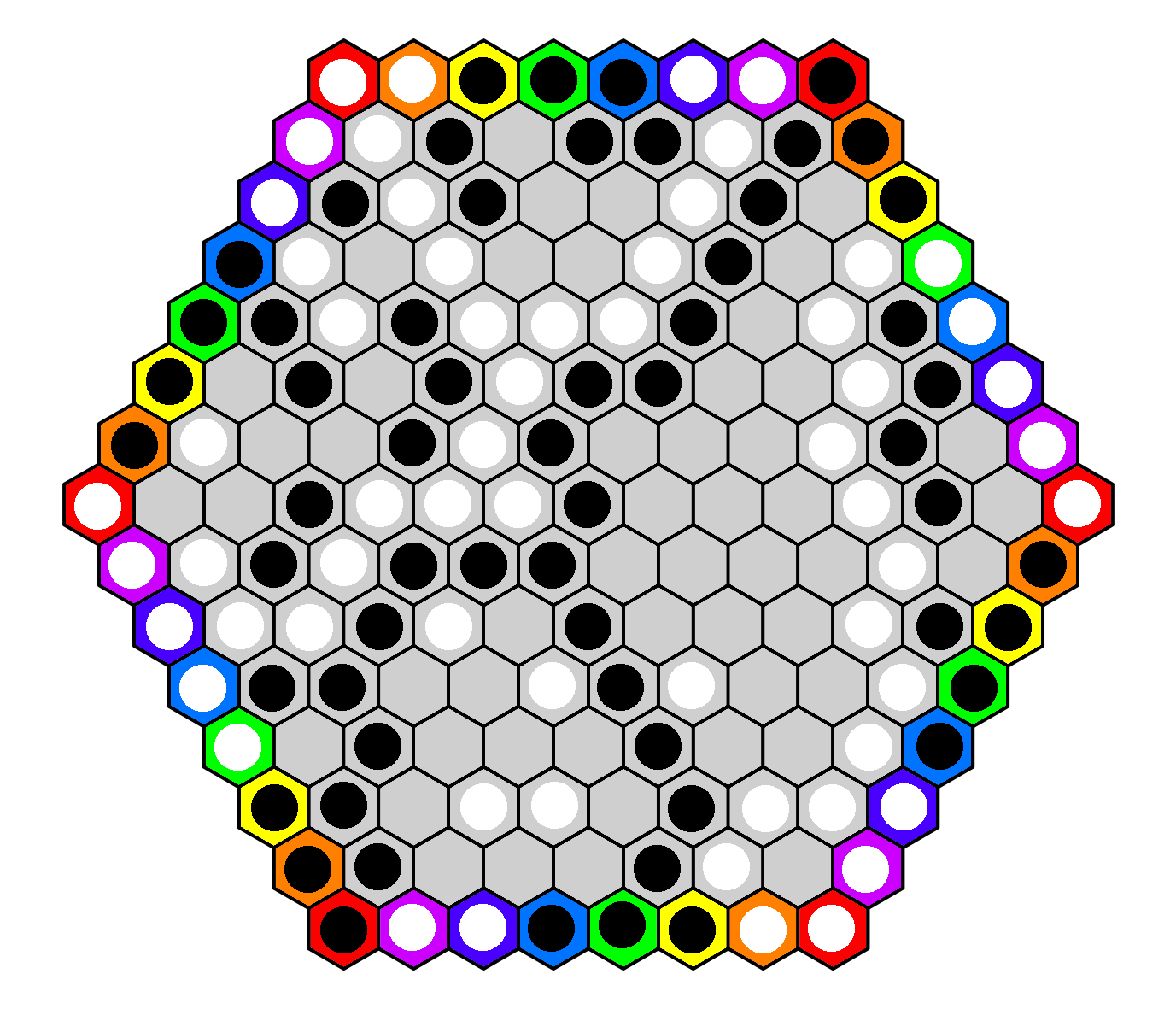
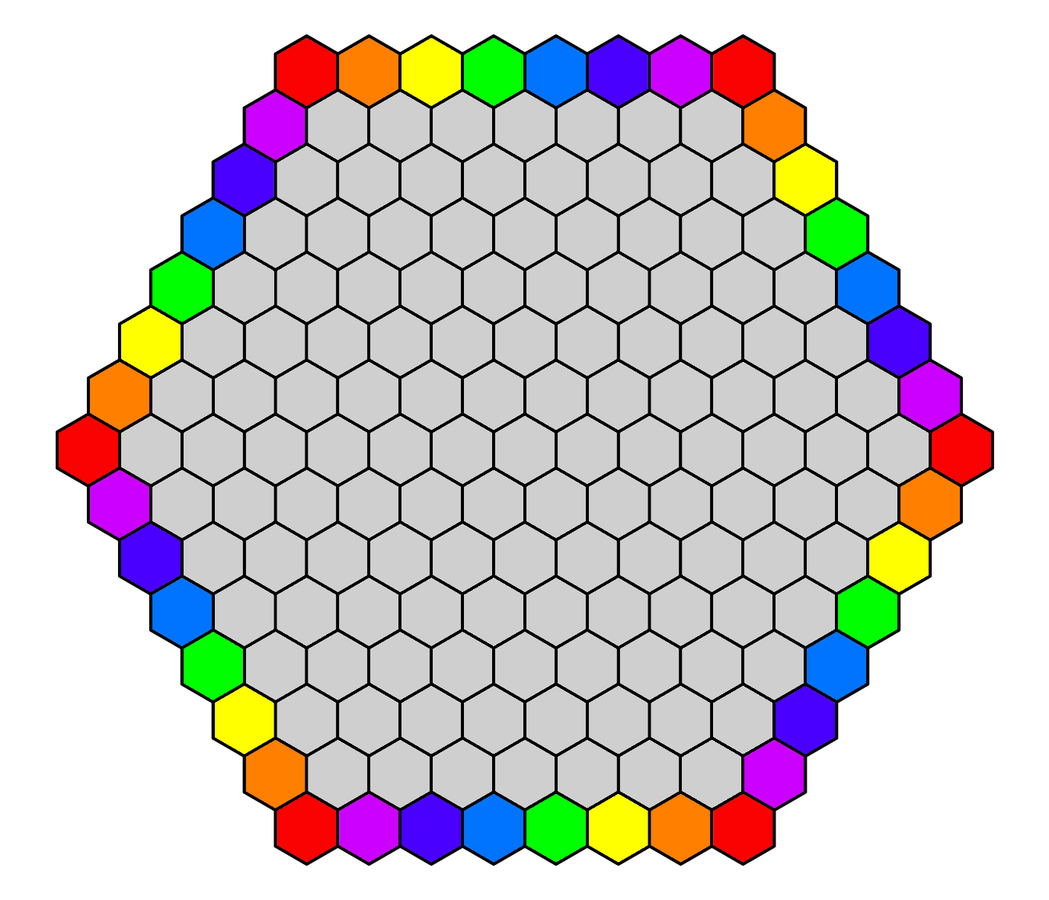
## História

Iris é um jogo que existe desde 2019 e que foi criado por Craig Duncan. Este jogo de tabuleiro enquadra-se na categoria “Estratégia Abstrata” segue um mecanismo de “Construção de Padrões” e é da família “Combinatória”. Deste modo, é um jogo sem tema/enredo, cujo resultado não é dependente da sorte / do acaso. Para além disso, é um jogo desenvolvido de forma a ter 2 jogadores, em que os jogadores alternam os turnos.

## Material

Para jogar este jogo, é necessário um tabuleiro “hexhex”, ou seja, um tabuleiro hexagonal com células hexagonais. As células que compõem o perímetro do tabuleiro são coloridas (formando as cores do arco-íris) e as restantes células são de cor cinzenta.

Para além do tabuleiro, são ainda necessárias peças de tamanho inferior ou igual às células do tabuleiro, sendo que algumas tem que ser pretas e outras brancas (pretas para o jogador 1 e brancas para o jogador 2).



**Fig. 1** - Aspeto do tabuleiro de Iris, sem e com peças.

## Regras

Na primeira jogada, o jogador 1 deverá colocar uma única peça preta numa célula cinzenta à escolha. A partir daí, começando o jogador 2, cada jogador deverá colocar 2 peças em cada jogada, seguindo as duas seguintes regras:

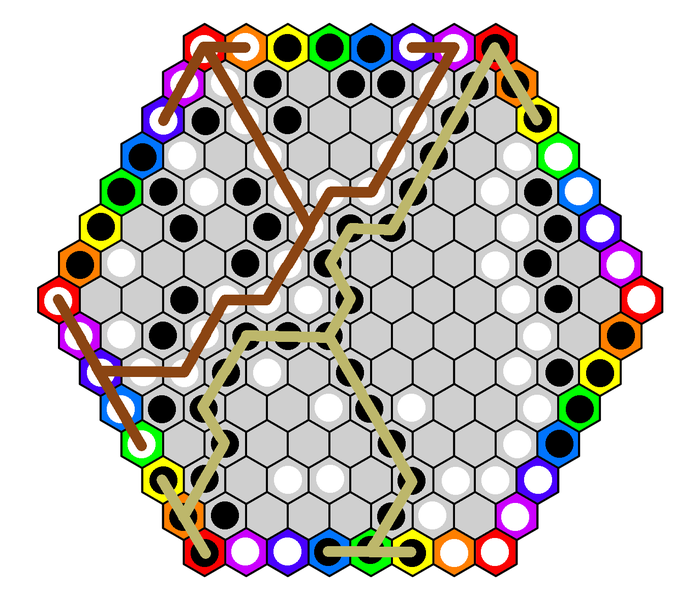
1. Se um jogador coloca a primeira peça numa célula colorida, a segunda peça deverá ser colocada na célula com a mesma cor que se encontra no lado oposto do tabuleiro;
2. Se a primeira peça é colocada numa célula cinzenta, então a segunda peça deverá ser colocada numa célula cinzenta não adjacente à primeira célula. Se já todas as células cinzentas não adjacentes estiverem ocupadas, a segunda peça deverá ser ignorada.

Quando todas as células estão preenchidas ou ambos os jogadores “passam” a sua vez, o jogo termina.

## Pontuação e Vencedor

Cada jogador deverá dividir as suas peças em grupos, sendo que cada grupo é constituído por peças adjacentes. Ganha o jogador que tiver o grupo com maior pontuação. Se os grupos de maior pontuação dos dois jogadores possuírem a mesma pontuação, então comparam-se os grupos com 2ª maior pontuação, e assim consecutivamente, até ocorrer o desempate.

A pontuação de cada grupo é equivalente ao número de células coloridas que ocupa.



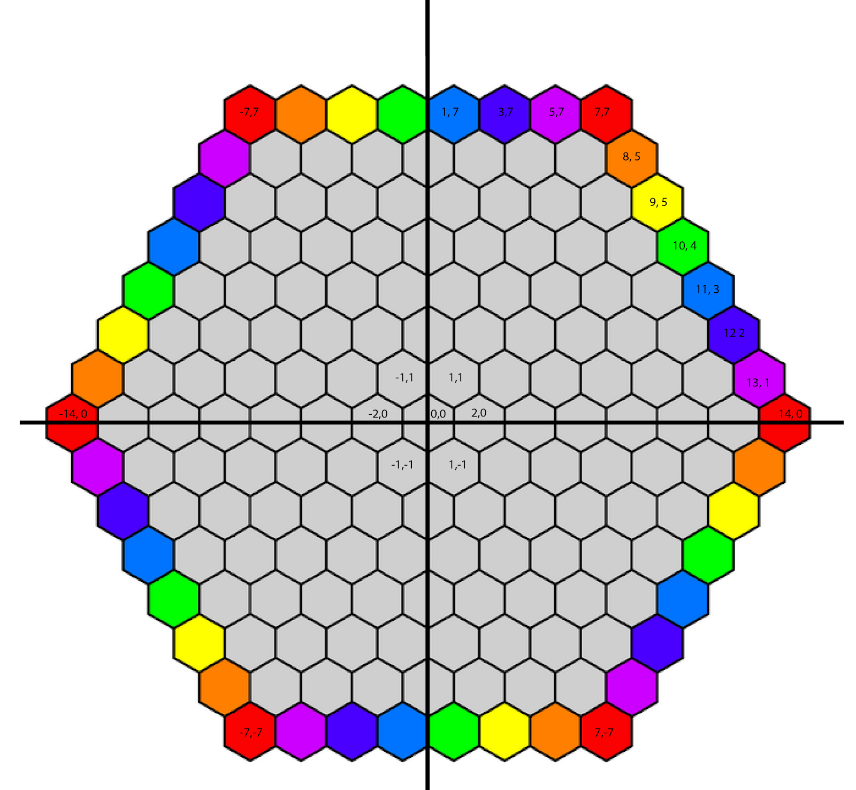
**Fig. 2** - Exemplo de grupos com maior pontuação no final de um jogo.

# A Lógica

## Representação do Estado do Jogo

A representação interna do jogo consiste numa lista de listas de listas. Deste modo o Tabuleiro é representado por uma lista de Linhas. Cada linha é uma lista, cujo *header* é o número da linha e cujos restantes elementos representam as várias células que compõem a linha. Cada célula é representada por uma lista composta por dois elementos: o número da coluna da célula e o estado atual da célula. O estado atual da célula poderá ter 3 valores: ‘B’ se a célula estiver vazia, 1 se a célula estiver preenchida por uma peça do jogador 1 (preta) ou 2 se a célula estiver preenchida por uma peça do jogador 2 (branca).

O sistema de coordenadas utilizado para um tabuleiro 15x15 é o apresentado abaixo.



**Fig. 3** – Sistema de Coordenadas do Jogo.

**Estado Inicial**

[[7, [-7, 'B'], [-5, 'B'], [-3, 'B'], [-1, 'B'], [1, 'B'], [3, 'B'], [5, 'B'], [7, 'B']],

[6, [-8, 'B'], [-6, 'B'], [-4, 'B'], [-2, 'B'], [0, 'B'], [2, 'B'], [4, 'B'], [6, 'B'], [8, 'B']],

[5, [-9, 'B'], [-7, 'B'], [-5, 'B'], [-3, 'B'], [1, 'B'], [1, 'B'], [3, 'B'], [5, 'B'], [7, 'B'], [9, 'B']],

[4, [-10, 'B'], [-8, 'B'], [-6, 'B'], [-4, 'B'], [-2, 'B'], [0, 'B'], [2, 'B'], [4, 'B'], [6, 'B'], [8, 'B'], [10, 'B']],

[3, [-11, 'B'], [-9, 'B'], [-7, 'B'], [-5, 'B'], [-3, 'B'], [-1, 'B'], [1, 'B'], [3, 'B'], [5, 'B'], [7, 'B'], [9, 'B'], [11, 'B']],

[2, [-12, 'B'], [-10, 'B'], [-8, 'B'], [-6, 'B'], [-4, 'B'], [-2, 'B'], [0, 'B'], [2, 'B'], [4, 'B'], [6, 'B'], [8, 'B'], [10, 'B'], [12, 'B']],

[1, [-13, 'B'], [-11, 'B'], [-9, 'B'], [-7, 'B'], [-5, 'B'], [-3, 'B'], [-1, 'B'], [1, 'B'], [3, 'B'], [5, 'B'], [7, 'B'], [9, 'B'], [11, 'B'], [13, 'B']],

[0, [-14, 'B'], [-12, 'B'], [-10, 'B'], [-8, 'B'], [-6, 'B'], [-4, 'B'], [-2, 'B'], [0, 'B'], [2, 'B'], [4, 'B'], [6, 'B'], [8, 'B'], [10, 'B'], [12, 'B'], [14, 'B']],

[-1, [-13, 'B'], [-11, 'B'], [-9, 'B'], [-7, 'B'], [-5, 'B'], [-3, 'B'], [-1, 'B'], [1, 'B'], [3, 'B'], [5, 'B'], [7, 'B'], [9, 'B'], [11, 'B'], [13, 'B']],

[-2, [-12, 'B'], [-10, 'B'], [-8, 'B'], [-6, 'B'], [-4, 'B'], [-2, 'B'], [0, 'B'], [2, 'B'], [4, 'B'], [6, 'B'], [8, 'B'], [10, 'B'], [12, 'B']],

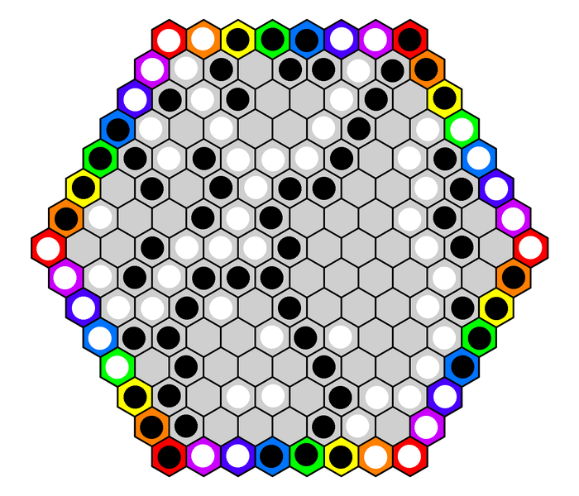
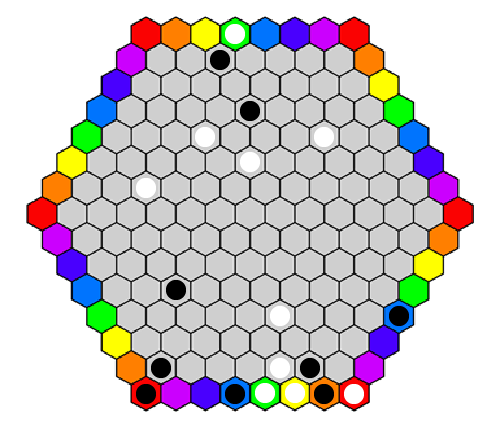
[-3, [-11, 'B'], [-9, 'B'], [-7, 'B'], [-5, 'B'], [-3, 'B'], [-1, 'B'], [1, 'B'], [3, 'B'], [5, 'B'], [7, 'B'], [9, 'B'], [11, 'B']],

[-4, [-10, 'B'], [-8, 'B'], [-6, 'B'], [-4, 'B'], [-2, 'B'], [0, 'B'], [2, 'B'], [4, 'B'], [6, 'B'], [8, 'B'], [10, 'B']],

[-5, [-9, 'B'], [-7, 'B'], [-5, 'B'], [-3, 'B'], [-1, 'B'], [1, 'B'], [3, 'B'], [5, 'B'], [7, 'B'], [9, 'B']],

[-6, [-8, 'B'], [-6, 'B'], [-4, 'B'], [-2, 'B'], [0, 'B'], [2, 'B'], [4, 'B'], [6, 'B'], [8, 'B']],

[-7, [-7, 'B'], [-5, 'B'], [-3, 'B'], [-1, 'B'], [1, 'B'], [3, 'B'], [5, 'B'], [7, 'B']]]



**Fig. 4** – Estado Intermédio do Jogo

**Fig. 5** – Estado Final do Jogo.

**Estado Intermédio**

[[7, [-7, 'B'], [-5, 'B'], [-3, 'B'], [-1, '2'], [1, 'B'], [3, 'B'], [5, 'B'], [7, 'B']],

[6, [-8, 'B'], [-6, 'B'], [-4, 'B'], [-2, '1'], [0, 'B'], [2, 'B'], [4, 'B'], [6, 'B'], [8, 'B']],

[5, [-9, 'B'], [-7, 'B'], [-5, 'B'], [-3, 'B'], [1, 'B'], [1, 'B'], [3, 'B'], [5, 'B'], [7, 'B'], [9, 'B']],

[4, [-10, 'B'], [-8, 'B'], [-6, 'B'], [-4, 'B'], [-2, 'B'], [0, '1'], [2, 'B'], [4, 'B'], [6, 'B'], [8, 'B'], [10, 'B']],

[3, [-11, 'B'], [-9, 'B'], [-7, 'B'], [-5, 'B'], [-3, '2'], [-1, 'B'], [1, 'B'], [3, 'B'], [5, '2'], [7, 'B'], [9, 'B'], [11, 'B']],

[2, [-12, 'B'], [-10, 'B'], [-8, 'B'], [-6, 'B'], [-4, 'B'], [-2, 'B'], [0, '2'], [2, 'B'], [4, 'B'], [6, 'B'], [8, 'B'], [10, 'B'], [12, 'B']],

[1, [-13, 'B'], [-11, 'B'], [-9, 'B'], [-7, '2'], [-5, 'B'], [-3, 'B'], [-1, 'B'], [1, 'B'], [3, 'B'], [5, 'B'], [7, 'B'], [9, 'B'], [11, 'B'], [13, 'B']],

[0, [-14, 'B'], [-12, 'B'], [-10, 'B'], [-8, 'B'], [-6, 'B'], [-4, 'B'], [-2, 'B'], [0, 'B'], [2, 'B'], [4, 'B'], [6, 'B'], [8, 'B'], [10, 'B'], [12, 'B'], [14, 'B']],

[-1, [-13, 'B'], [-11, 'B'], [-9, 'B'], [-7, 'B'], [-5, 'B'], [-3, 'B'], [-1, 'B'], [1, 'B'], [3, 'B'], [5, 'B'], [7, 'B'], [9, 'B'], [11, 'B'], [13, 'B']],

[-2, [-12, 'B'], [-10, 'B'], [-8, 'B'], [-6, 'B'], [-4, 'B'], [-2, 'B'], [0, 'B'], [2, 'B'], [4, 'B'], [6, 'B'], [8, 'B'], [10, 'B'], [12, 'B']],

[-3, [-11, 'B'], [-9, 'B'], [-7, 'B'], [-5, '1'], [-3, 'B'], [-1, 'B'], [1, 'B'], [3, 'B'], [5, 'B'], [7, 'B'], [9, 'B'], [11, 'B']],

[-4, [-10, 'B'], [-8, 'B'], [-6, 'B'], [-4, 'B'], [-2, 'B'], [0, 'B'], [2, '2'], [4, 'B'], [6, 'B'], [8, 'B'], [10, '1']],

[-5, [-9, 'B'], [-7, 'B'], [-5, 'B'], [-3, 'B'], [-1, 'B'], [1, 'B'], [3, 'B'], [5, 'B'], [7, 'B'], [9, 'B']],

[-6, [-8, 'B'], [-6, '1'], [-4, 'B'], [-2, 'B'], [0, 'B'], [2, '2'], [4, '1'], [6, 'B'], [8, 'B']],

[-7, [-7, '1'], [-5, 'B'], [-3, 'B'], [-1, '1'], [1, '2'], [3, '2'], [5, '1'], [7, '2']]]

**Estado Final**

[[7, [-7, '2'], [-5, '2'], [-3, '1'], [-1, '1'], [1, '1'], [3, '2'], [5, '2'], [7, '1']],

[6, [-8, '2'], [-6, '2'], [-4, '1'], [-2, 'B'], [0, '1'], [2, '1'], [4, '2'], [6, '1'], [8, '1']],

[5, [-9, '2'], [-7, '1'], [-5, '2'], [-3, '1'], [1, 'B'], [1, 'B'], [3, '2'], [5, '1'], [7, 'B'], [9, '1']],

[4, [-10, '1'], [-8, '2'], [-6, 'B'], [-4, '2'], [-2, 'B'], [0, 'B'], [2, '2'], [4, '1'], [6, 'B'], [8, '2'], [10, '2']],

[3, [-11, '1'], [-9, '1'], [-7, '2'], [-5, '1'], [-3, '2’], [-1, '2'], [1, '2'], [3, '1'], [5, 'B'], [7, '2'], [9, '1'], [11, '2']],

[2, [-12, '1'], [-10, 'B'], [-8, '1'], [-6, 'B'], [-4, '1'], [-2, '2'], [0, '1'], [2, '1'], [4, 'B'], [6, 'B'], [8, '2'], [10, '1'], [12, '2']],

[1, [-13, '1'], [-11, '2'], [-9, 'B'], [-7, 'B'], [-5, '1'], [-3, '2'], [-1, '1'], [1, 'B'], [3, 'B'], [5, 'B'], [7, '2'], [9, '1'], [11, 'B'], [13, '2']],

[0, [-14, '2'], [-12, 'B'], [-10, 'B'], [-8, '1'], [-6, '2'], [-4, '2'], [-2, '2'], [0, '1'], [2, 'B'], [4, 'B'], [6, 'B'], [8, '2'], [10, '1'], [12, 'B'], [14, '2']],

[-1, [-13, '2'], [-11, '2'], [-9, '1'], [-7, '2'], [-5, '1'], [-3, '1'], [-1, '1'], [1, 'B'], [3, 'B'], [5, 'B'], [7, 'B'], [9, '2'], [11, 'B'], [13, '1']],

[-2, [-12, '2'], [-10, '2'], [-8, '2'], [-6, '1'], [-4, '2'], [-2, 'B'], [0, '1'], [2, 'B'], [4, 'B'], [6, 'B'], [8, '2'], [10, '1'], [12, '1']],

[-3, [-11, '2'], [-9, '1'], [-7, '1'], [-5, 'B'], [-3, 'B'], [-1, '2'], [1, '1'], [3, '2'], [5, 'B'], [7, 'B'], [9, '2'], [11, '1']],

[-4, [-10, '2'], [-8, 'B'], [-6, '1'], [-4, 'B'], [-2, 'B'], [0, 'B'], [2, '1'], [4, 'B'], [6, 'B'], [8, '2'], [10, '1']],

[-5, [-9, '1'], [-7, '1’], [-5, 'B'], [-3, '2'], [-1, '2'], [1, 'B'], [3, '1'], [5, '2'], [7, '2'], [9, '2']],

[-6, [-8, '1'], [-6, '1'], [-4, 'B'], [-2, 'B'], [0, 'B'], [2, '1'], [4, '2'], [6, 'B'], [8, '2']],

[-7, [-7, '1'], [-5, '2'], [-3, '2'], [-1, '1'], [1, '1'], [3, '1'], [5, '2'], [7, '2']]]

## Visualização do Tabuleiro

A vizualização do tabuleiro *hexhex* de dimensões 15x15 é feita com recurso ao código abaixo apresentado:

drawSpace(0).

drawSpace(N) :- write(' '), N1 is N-1, drawSpace(N1).

displayCell([\_|[P]]):- write(P), write(' ').

displayLineCells([]).

displayLineCells([H|T]) :- displayCell(H), displayLineCells(T).

displayLine([H|T]) :- (H<0 -> write(H); write(' '), write(H)), (H>0 -> N1 is H+1, drawSpace(N1); N1 is -H+1, drawSpace(N1)), (H<0 -> write('\\ '); (H>0 -> write('/ '); write('| '))), displayLineCells(T), (H<0 -> write('/ \n'); (H>0 -> write('\\ \n'); write('| \n'))).

displayBoard([]).

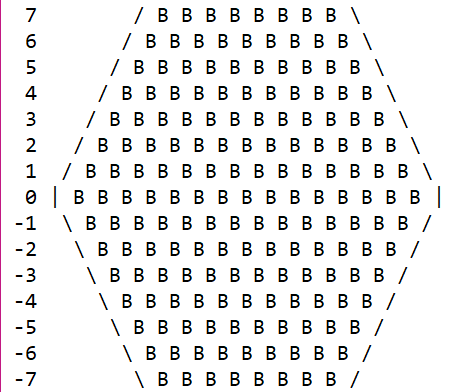
displayBoard([H|T]) :- displayLine(H), displayBoard(T).

display\_game(Board, \_) :- displayBoard(Board).

Desta forma, para se poder visualizar o tabuleiro deve-se ser usado o predicado display\_game(+Board,+Player).

As peças do jogador 1 (peças brancas) são representadas pelo símbolo ‘1’, as peças do jogador 2 (peças pretas) são representadas pelo símbolo ‘2’ e as células vazias (sem nenhuma peça) são representadas pelo símbolo ‘B’.

Usando a lista de listas correspondente ao tabuleiro no estado inicial, no predicado display\_game, já mencionado acima, o resultado obtido é o visível na imagem:



**Fig. 6** – Visualização do Tabuleiro em Modo de Texto

## Lista de Jogadas Válidas

Para obter uma lista com todas as jogadas possíveis para um jogador, dado um determinado estado do jogo, deve ser usado o predicado valid\_moves(+Board, +Player, -ListOfMoves). Este predicado é implementado através do seguinte código:

generateMovesFromLine(\_, [], \_, \_, \_, \_, []).

generateMovesFromLine(Board, [[CellBoard | \_] | T],[Cell | Value], LineBoard, Line, Player, ValidMoves) :- generateMovesFromLine(Board, T, [Cell, Value], LineBoard, Line, Player, ValidMovesAux), (verifyMove(Board, Line, Cell, LineBoard, CellBoard), (Line =:= LineBoard, Cell =:=CellBoard -> fail;!) -> append(ValidMovesAux, [ [Line,Cell, LineBoard, CellBoard] ], ValidMoves); ValidMoves = ValidMovesAux, !).

generateValidMovesCell(\_, [], \_, \_, \_, []).

generateValidMovesCell(Board, Board, [Cell, \_], Line, \_, ValidMoves):- cellColor(Line, Cell), cellEmpty(Board, Line, Cell), Line2 is -Line, Column2 is -Cell, ValidMoves = [[Line, Cell, Line2, Column2]].

generateValidMovesCell(Board, [[LineBoard | Cells ] | T], [Cell, Value], Line, Player, ValidMoves):- generateValidMovesCell(Board, T, [Cell, Value], Line, Player, ValidMovesAux), generateMovesFromLine(Board, Cells, [Cell, Value], LineBoard, Line, Player, ValidFromLine), append(ValidMovesAux, ValidFromLine, ValidMoves).

generateValidMovesLine(\_, \_, [], \_, []).

generateValidMovesLine(Board, Line, [ Cell | T], Player, ValidMoves):- generateValidMovesLine(Board, Line, T, Player, ValidMovesAux), generateValidMovesCell(Board, Board, Cell, Line, Player, ValidInCell), append(ValidMovesAux, ValidInCell, ValidMoves).

generateValidMoves(\_, [], \_, []).

generateValidMoves(Board, [[Line| Cells] | T ], Player, ValidMoves):- generateValidMoves(Board, T, Player, ValidMovesAux), generateValidMovesLine(Board, Line, Cells, Player, ValidInLine), append(ValidMovesAux, ValidInLine, ValidMoves).

valid\_moves(Board, Player, ListOfMoves) :- generateValidMoves(Board, Board, Player, ListOfMoves).

Sílvia explica isto por favor que eu não sei explicar.

## Execução de Jogadas

De forma a executar uma jogada, deve-se utilizar o predicado move(+Move, +Board, -NewBoard). O código seguinte implementa este mesmo predicado:

cellValue([[L|[[C, P]|T1]]|T], Line, Column, Value) :- (L=Line -> (C=Column -> Value=P; cellValue([[L|T1]|T], Line, Column, Value)); cellValue(T, Line, Column, Value)).

cellEmpty(Board, Line, Column) :- cellValue(Board, Line, Column, Value), Value='B'.

adjacentPieces(Line1, Column1, Line2, Column2) :- Line1 =:= Line2, Column1 =:= Column2 + 2.

adjacentPieces(Line1, Column1, Line2, Column2) :- Line1 =:= Line2, Column1 =:= Column2 - 2.

adjacentPieces(Line1, Column1, Line2, Column2) :- Line1 =:= Line2 + 1, Column1 =:= Column2 + 1.

adjacentPieces(Line1, Column1, Line2, Column2) :- Line1 =:= Line2 + 1, Column1 =:= Column2 - 1.

adjacentPieces(Line1, Column1, Line2, Column2) :- Line1 =:= Line2 - 1, Column1 =:= Column2 + 1.

adjacentPieces(Line1, Column1, Line2, Column2) :- Line1 =:= Line2 - 1, Column1 =:= Column2 - 1.

cellColor(Line, \_) :- abs(Line) =:= 7.

cellColor(Line, Column) :- abs(Line) + abs(Column) =:= 14.

verifyMove(Board, Line1, Column1, [], []) :- cellEmpty(Board, Line1, Column1), \+cellColor(Line1, Column1).

verifyMove(Board, Line1, Column1, Line2, Column2) :- cellEmpty(Board, Line1, Column1), cellEmpty(Board, Line2, Column2), cellColor(Line1, Column1), Line2 =:= -Line1, Column2 =:= -Column1.

verifyMove(Board, Line1, Column1, Line2, Column2) :- cellEmpty(Board, Line1, Column1), cellEmpty(Board, Line2, Column2), \+cellColor(Line1, Column1), \+cellColor(Line2, Column2), \+adjacentPieces(Line1, Column1, Line2, Column2).

changeCell(\_, \_, [], []).

changeCell(Player, Column, [[H|T1]|T], NewLine) :- changeCell(Player, Column, T, AuxLine), (H=Column -> append([[Column, Player]], AuxLine, NewLine); append([[H|T1]], AuxLine, NewLine)).

implement\_move(\_, \_, \_, [], []).

implement\_move(\_, [], [], Board, Board).

implement\_move(Player, Line, Column, [[H|T1]|T], NewBoard) :- implement\_move(Player, Line, Column, T, AuxBoard), (H=Line -> changeCell(Player, Column, T1, NewLine), append([[H|NewLine]], AuxBoard, NewBoard); append([[H|T1]], AuxBoard, NewBoard)).

implement\_moves([Player,Line1,Column1,Line2,Column2], Board, NewBoard) :- implement\_move(Player, Line1, Column1, Board, BoardAux), implement\_move(Player, Line2, Column2, BoardAux, NewBoard).

move([Player,Line1,Column1,Line2,Column2], Board, NewBoard) :- verifyMove(Board, Line1, Column1, Line2, Column2), implement\_moves([Player, Line1, Column1, Line2, Column2], Board, NewBoard).

O predicado cellValue permite descubrir qual o valor atual de uma determinada célula, dada a sua linha e coluna

## Final do Jogo

Verificação do fim do jogo, com identificação do vencedor. O predicado deve chamar-se game\_over(+Board, -Winner).

## Avaliação do Tabuleiro

Forma(s) de avaliação do estado do jogo. O predicado deve chamar-se value(+Board, +Player, -Value).

## Jogada do Computador

Escolha da jogada a efetuar pelo computador, dependendo do nível de dificuldade. O predicado deve chamar-se choose\_move(+Board, +Level, -Move).

# Conclusões

# Bibliografia

[Board Game Geek](https://www.boardgamegeek.com/boardgame/286792/iris)

[Wikipedia](https://pt.wikipedia.org/wiki/Jogo_de_estrat%C3%A9gia_abstrato)