U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Quartetto

Relatório Final
Programação em Lógica

Mestrado Integrado em Engenharia Informática
e Computação

(18 de novembro de 2018)

Francisco Teixeira Ferreira _ up201605660@fe.up.pt
Nuno Tiago Tavares Lopes _ up201605337@fe.up.pt

Índice

1 Introdução

Este trabalho foi desenvolvido no âmbito da unidade curricular de Programação em Lógica, do 3º ano do Mestrado Integrado em Engenharia Informática e Computação, cujo objetivo é aprofundar os conhecimentos previamente adquiridos tanto nas aulas teóricas como nas aulas práticas, juntamente com a abordagem de problemas práticos recorrendo à linguagem PROLOG.

Sendo assim, o grupo propôs-se a implementar o jogo de tabuleiro Quartetto com os seguintes modos de jogo: *Player vs Player*, *Player vs Computer*, *Computer vs Computer* (com três níveis de dificuldade).

O relatório está dividido nas seguintes secções:

- **O Jogo : Quartetto**: Descrição do jogo e das suas regras.
- **Lógica do Jogo**: Descrição da implementação da lógica do jogo, tendo a seguinte estrutura:
  - **Representação do Estado do Jogo**: Exemplificação de estados iniciais e finais do jogo.
  - **Visualização do Tabuleiro**: Descrição do predicado de visualização.
  - **Lista de Jogadas Válidas**: Descrição dos predicados utilizados para a validação das jogadas.
  - **Execução de Jogadas**: Explicação de como é feita a validação e execução de uma jogada no tabuleiro, obtendo um novo estado de jogo.
  - **Final do Jogo**: Descrição dos predicados que verificam o fim do jogo.
  - **Avaliação do Tabuleiro**: Descrição dos predicados que avaliam o estado do jogo.
  - **Jogada do Computador**: Descrição dos predicados que geram os movimentos do computador
- **Interface com o Utilizador**: Demonstração e explicação do módulo de interface com o utilizador.

2 O Jogo : Quartetto

Quartetto é jogado no comum tabuleiro de 8x8.

Tabuleiro:

Cada jogador possui quatro peças, sendo que começam alinhadas nas quatro posições centrais das linhas com índices 1 e 8 (considerando que os índices do tabuleiro tradicional vão de 1 a 8).

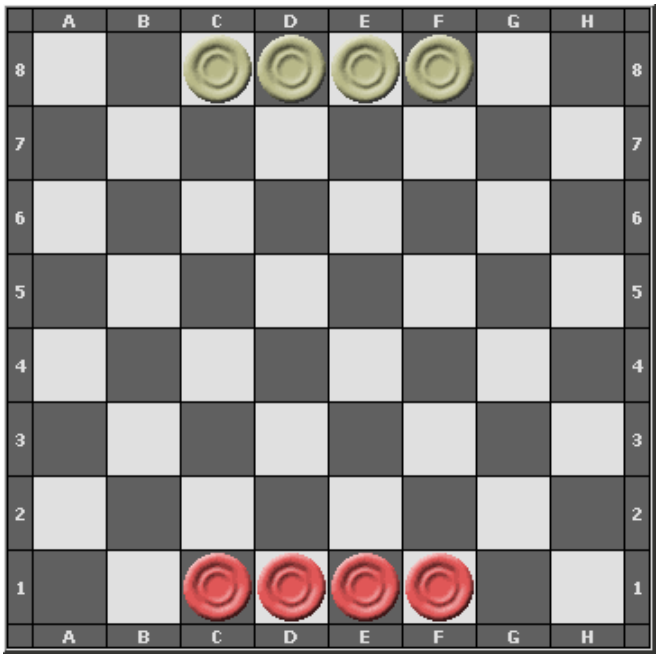

Fig. 1 - Estado inicial do tabuleiro

Objetivo:

O objetivo do jogo passa por colocar todas as peças de uma determinada cor nas seguintes condições:

1. Os pontos centrais das células devem ser vértices de um quadrado que tenha sido alvo de uma rotação.
2. O quadrado tem como dimensões mínimas 5x5.

Exemplos de Jogos Finalizados com Sucesso:

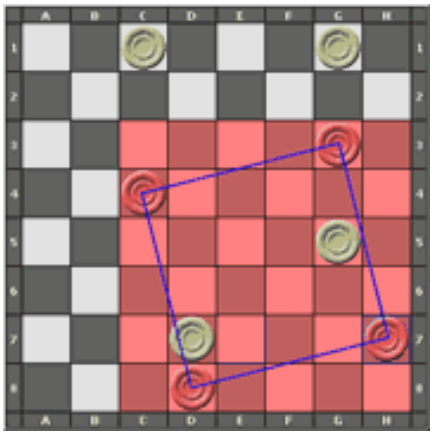

Fig. 2 - Vitória das peças pretas (6x6)

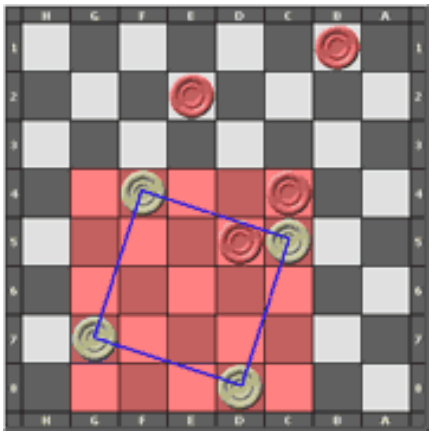

Fig. 3 - Vitória das peças brancas (5x5)

Exemplos de Jogadas que não finalizam um jogo:

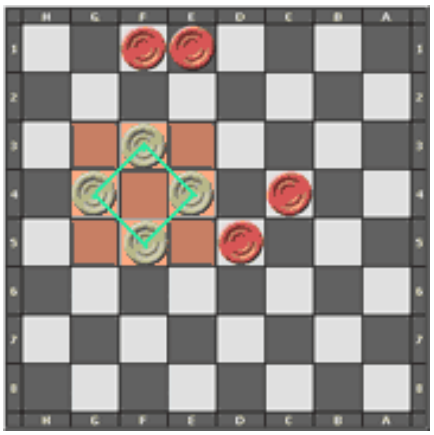

Fig. 4 - Peças brancas não ganham porque o quadrado é apenas de 3x3

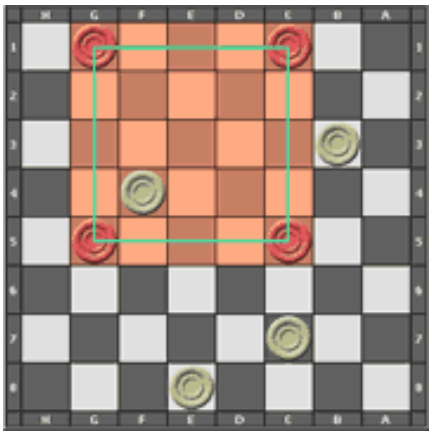

Fig. 5 - Peças pretas não ganham porque o quadrado formado não se encontra inclinado

Gameplay:

As movimentações do jogo são iniciadas pelo jogador que tenha as peças pretas, alternando depois sempre que houver a movimentação de uma peça.

As peças podem ser movidas o número de casas que for desejado pelo jogador, tanto horizontal como verticalmente, com as únicas condições de que não podem ficar numa casa que já esteja ocupada nem passar por cima dessa mesma casa (movimentação semelhante à torre no jogo de Xadrez).

3 Lógica do Jogo

3.1 Representação do estado do jogo

Situação inicial:

```
initialBoard([
    [empty, empty, white, white, white, white, empty, empty],
    [empty, empty, empty, empty, empty, empty, empty, empty],
    [empty, empty, empty, empty, empty, empty, empty, empty],
    [empty, empty, empty, empty, empty, empty, empty, empty],
    [empty, empty, empty, empty, empty, empty, empty, empty],
    [empty, empty, empty, empty, empty, empty, empty, empty],
    [empty, empty, empty, empty, empty, empty, empty, empty],
    [empty, empty, black, black, black, black, empty, empty]
]).
```

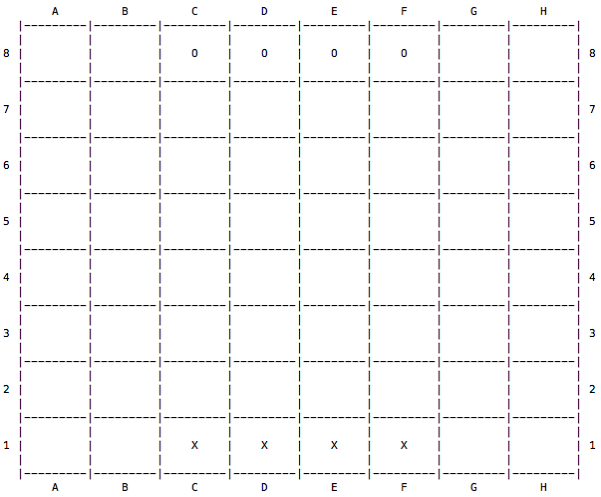

Fig. 6 - Situação inicial vista na consola

Exemplo de uma situação final:

```
finalBoard([
    [empty, empty, white, empty, empty, empty, white, empty],
    [empty, empty, empty, empty, empty, empty, empty, empty],
    [empty, empty, empty, empty, empty, empty, black, empty],
    [empty, empty, black, empty, empty, empty, empty, empty],
    [empty, empty, empty, empty, empty, empty, white, empty],
    [empty, empty, empty, empty, empty, empty, empty, empty],
    [empty, empty, empty, white, empty, empty, empty, black],
    [empty, empty, empty, black, empty, empty, empty, empty]
]).
```

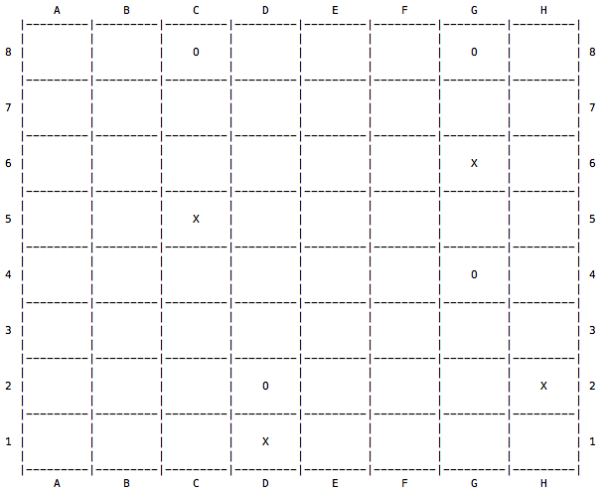

Fig. 7 - Situação final vista na consola

3.2 Visualização do tabuleiro

Os predicados responsáveis pela visualização do tabuleiro na linha de comandos encontram-se no ficheiro *board.pl*.

Para imprimir o tabuleiro basta chamar o predicado ***display_game(+Board, +Player)***, onde ***Board*** é um tabuleiro de um estado de jogo e o ***Player*** é o jogador atual. Consoante o jogador, o predicado chama outros dois predicados, ***printBoard(+Board)*** e ***printBoardReverse(+Board)***, para inverter o tabuleiro, simulando assim um jogo de tabuleiro na vida real. Por sua vez, cada um destes predicados faz uso de outros predicados com funções cada vez mais específicas para imprimir o tabuleiro de uma forma simples, formatada e concisa sempre que necessário.

As figuras 6 e 7, apresentadas anteriormente, representam os outputs na consola produzidos pelo código acima apresentado.

3.3 Lista de Jogadas Válidas

Uma jogada é válida se a peça for jogada nas direções horizontal e vertical, o número de casas que for desejado pelo jogador, sem que a posição final seja uma casa já ocupada por uma peça ou que tenha passado por cima de uma peça.

A obtenção de uma lista de todas as jogadas válidas é feita através da função ***valid_moves(+Player, +Board, -ListOfMoves),*** que dado o respetivo jogador calcula todas as suas movimentações válidas, tanto na horizontal como na vertical, que são retornadas no formato de uma lista através de *ListOfMoves*.

Para se obter a lista com as jogadas válidas são realizados os seguintes passos: são calculadas as posições das quatro peças desse jogador, que é feito pelo predicado ***getPiecesList(+Player, +Board, -ListOfPieces)***. Este predicado retorna no elemento *ListOfPieces* uma lista com a posição de todas a peças do *Player*. Depois é chamado o predicado ***getMovesList(+Player, +Board, +OldRow, +OldColumn, +Row, +Col, +List, -Ret)*** que obtém em *Ret* as jogadas válidas para a peça com as posições *OldRow* e *OldColumn*. Por fim este predicado é executado para cada uma das peças do jogador e as listas com as jogadas válidas de cada peça são inseridas na mesma lista, *ListOfMoves*.

3.4 Execução de Jogadas

Em cada jogada é pedido ao utilizador que tem a vez de jogar que indique as coordenadas da peça no tabuleiro que pretende mover. Assim que essas coordenadas são introduzidas é feita a verificação se de facto existe uma peça desse mesmo *Player* pronta a ser movida e se esta peça tem jogadas possíveis. Esta verificação é feita utilizando o predicado ***checkPieceOnPosition(+OldRow, +OldColumn, +Player, +Board, -Valid)***, que retorna através de *Valid* se existe uma peça com as coordenadas *OldRow* e *OldColumn*. Caso contrário, o programa

retrocede e o *Player* recebe uma mensagem que o avisa que naquela posição não tem nenhuma peça ou que a peça selecionada não tem jogadas possíveis, pedindo para introduzir novas coordenadas para outra peça.

Seguidamente é pedido ao utilizador que introduza a nova posição da peça que deseja movimentar e caso a posição seja válida dá-se o movimento, caso contrário o programa retrocede e pede novamente por novas coordenadas. É através do predicado ***checkValidNewPosition(+OldRow, +OldColumn, +NewRow, +NewColumn, +OldBoard, -Valid)*** que verifica se o movimento é válido.

Após estas verificações o movimento é realizado e são verificadas as condições de vitória, se tal não acontecer o jogo procede normalmente e é dada a vez ao outro *Player*.

Estas verificações são realizadas unicamente quando é o Jogador é um *Player* e não um *Computer*, através do predicado ***move(+OldBoard, +NewBoard, +Player)***. No caso de uma jogada do *Computer* já é outro predicado utilizado, ***moveComputer(+OldBoard, +NewBoard, +Player, +Level)*** que vai ser descrito mais à frente.

3.5 Final do Jogo

As verificações de finalização de jogo são realizadas no predicado ***game_over(+Player, +Board, -Result)*** que é chamado no final de cada jogada de cada *Player*. Para saber se um dos jogadores é o vencedor, este predicado realiza três verificações:

- ***checkIfSquare(+ListOfPieces, -Square)***:
  Verifica se as quatro peças do Player formam um quadrado. Para tal utilizamos um algoritmo encontrado no link que se encontra na Bibliografia.

- ***checkIfRotate(+ListOfPieces, +Flag, -Rotate)***:
  Verifica se o quadrado tem rotação.

- ***checkIfMinSize(+ListOfPieces, +Flag, -Result)***:
  Verifica se o quadrado tem as dimensões mínimas de 5x5.

Após a verificação destas condições, o predicado ***game_over*** retorna em *Result* o valor 2 caso haja vitória e 1 caso não haja, prosseguindo com o jogo normalmente.

3.6 Avaliação do Tabuleiro

A avaliação do tabuleiro só é para o jogador *Computer*, visto que este precisa de realizar jogadas inteligentes consoante o nível de dificuldade selecionado. Os predicados utilizados que avaliam o tabuleiro são os seguintes:

- moveToVictory(+Player, +Board, +Moves, -Record, -BestMove, -BestValue)
Verifica, utilizando o predicado ***valid_moves***, se existe uma jogada que se for realizada leva o *Player* à vitória, guardando-a em *BestMove*.

- ***moveToAvoidLoss(+Player, +Board, +OtherPlayerMoves, -BestMove, -Value)***
Verifica, utilizando o predicado ***valid_moves***, se existe uma jogada que evite a vitória do adversário quando este está a uma jogada de ganhar, guardando-a em *BestMove*.

- ***twoMovesToVictory(+Player, +Board, +Moves, -Record, -BestMove, -BestValue)***
Verifica, utilizando o predicado ***valid_moves*** duas vezes, se o *Player* está a duas jogadas da vitória, guardando a primeira jogada em *BestMove*.

- ***twoMovesToAvoidLoss(+Player, +Board, +OtherPlayerMoves, -BestMove, -Value)***
Verifica, utilizando o predicado ***valid_moves*** duas vezes, se o jogador adversário está a duas jogadas da vitória e se é possível impedir que tal aconteça, guardando em *BestMove* a primeira posição para onde o adversário irá.

3.7 Jogada do Computador

Para a jogada do computador existem 3 níveis de dificuldade: *Easy*, *Normal* e *Hard*. O nível de dificuldade pode ser escolhido pelo utilizador antes de começar o jogo. O predicado responsável pela jogada do computador é ***choose_move(+Player, +OldBoard, +Level, -OldRow, -OldColumn, -NewRow, -NewColumn)***.

No caso da dificuldade *Easy (Level = 1)*, as jogadas do computador são puramente aleatórias, sem que haja qualquer avaliação do tabuleiro.

No caso dos níveis de dificuldade *Normal (Level = 2)* e *Hard (Level = 3)*, a jogada é realizada utilizando os predicados descritos no ponto **Avaliação do Tabuleiro**, sendo que o algoritmo por nós utilizado para cada movimento tem como base um algoritmo Minimax. O algoritmo utilizado por nós tem como predicado ***evaluate_and_choose_normal(+Player, +Board, -OldRow, -OldColumn, -NewRow, -NewColumn)*** para a dificuldade *Normal* enquanto que para a dificuldade *Hard* é ***evaluate_and_choose_hard(+Player, +Board, -OldRow, -OldColumn, -NewRow, -NewColumn)***.

O algoritmo do nível *Normal* utiliza apenas os predicados ***moveToVictory*** e ***moveToAvoidLoss***, ou seja, apenas verifica uma jogada à frente antes de a realizar enquanto que o algoritmo do nível Hard utiliza os predicados, ***moveToVictory***, ***moveToAvoidLoss***, ***twoMovesToVictory*** e ***twoMovesToAvoidLoss***, verificando assim duas jogadas à frente antes de realizar a sua jogada.

Quando nenhum dos predicados da avaliação do tabuleiro apresenta sucesso a jogada realizada pelo computador vai ser aleatória como no caso do nível de dificuldade *Easy*. Porém mesmo assim o computador é bastante eficiente e rápido, conseguindo finalizar o jogo com poucas jogadas no caso do modo *Hard*.

4 Interface com o Utilizador

Os predicados responsáveis pela interface com o utilizador estão maioritariamente no ficheiro *menus.pl*.

A interface foi realizada com o intuito de ser simples e fácil para o utilizador comum, para tal, todos os menus estão devidamente identificados e para navegar entre eles basta selecionar o respetivo identificador, colocar um *Ponto* (.) e pressionar *Enter*.

Para iniciar o jogo basta executar o predicado ***play*** dentro do SICStus Prolog.

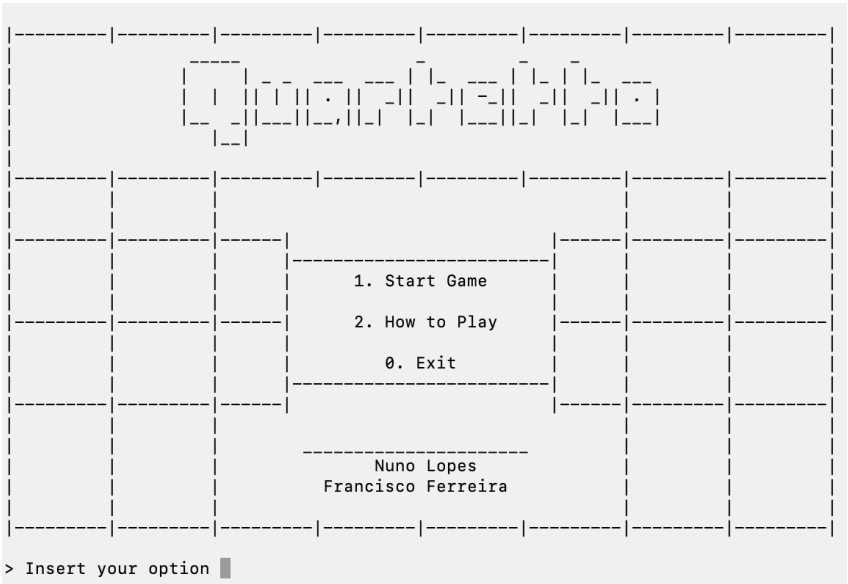

Fig. 8 - Menu principal.

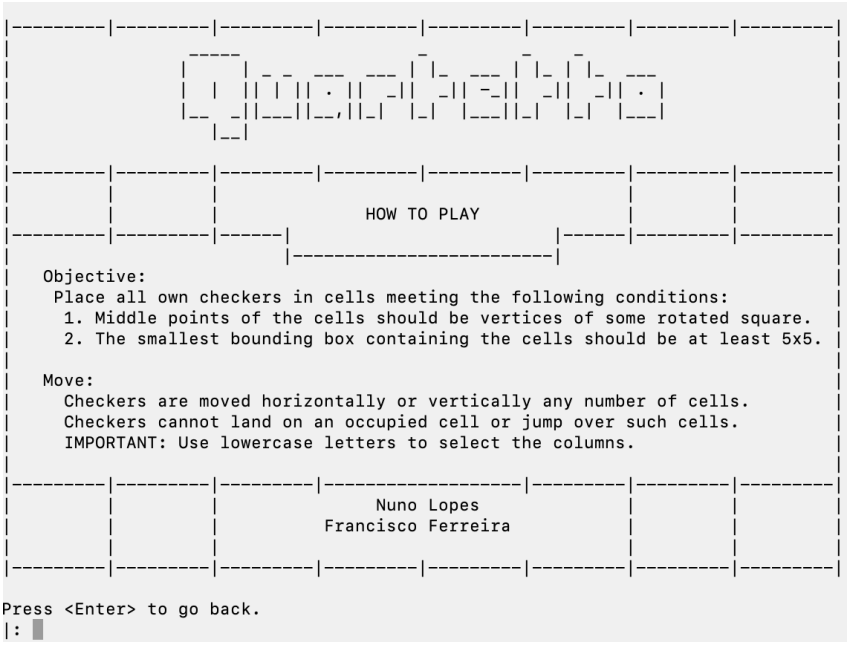

Fig. 9 - Instruções do jogo.

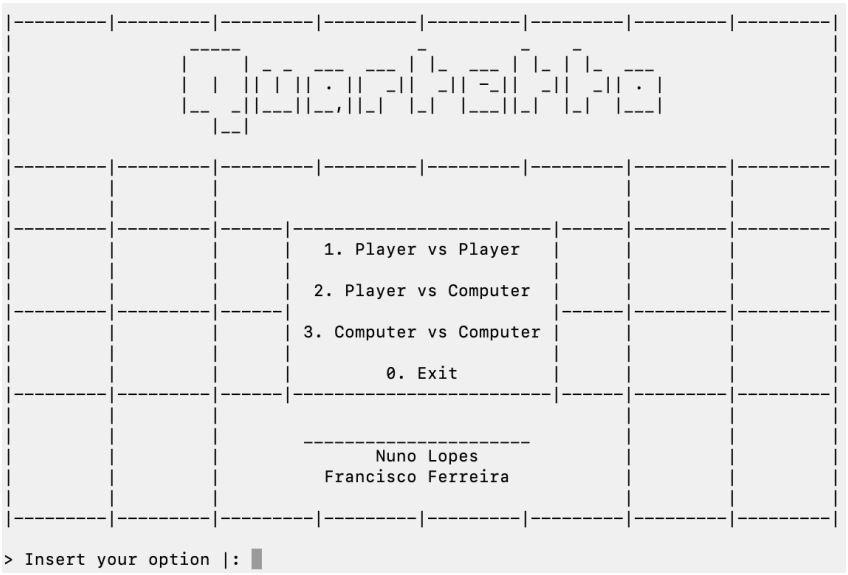

Fig. 10 - Menu de escolha do modo de jogo.

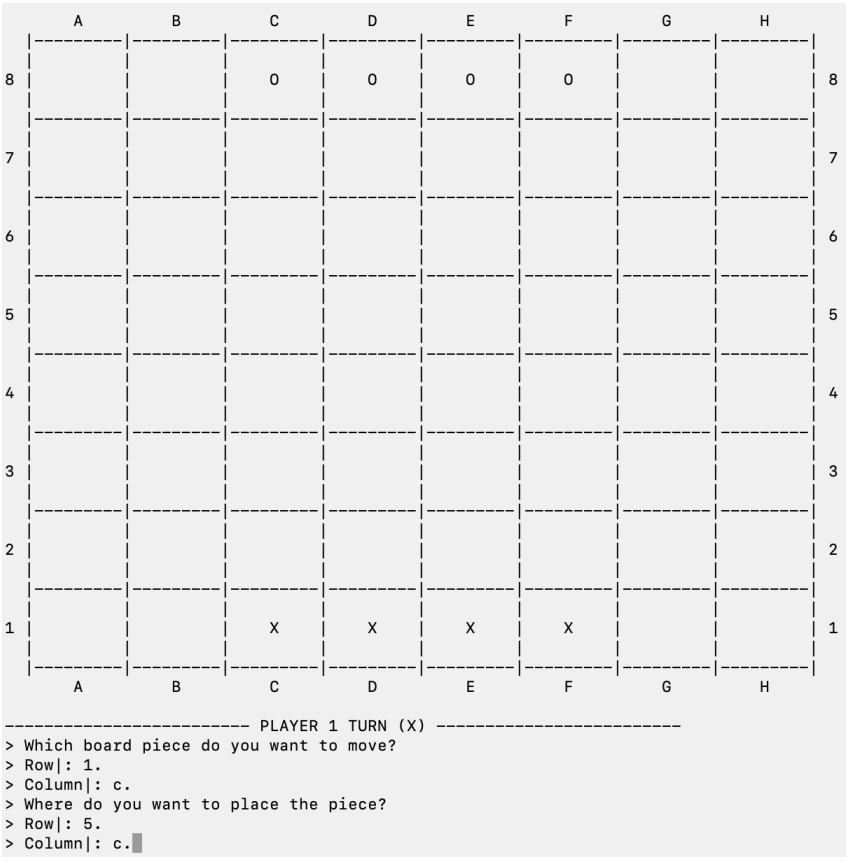

Fig. 11 - Estado inicial de um jogo.

```
------------------------- PLAYER 1 MOVE (X) -------------------------
FROM:
 > Row: 1
 > Column: C
TO:
 > Row: 5
 > Column: C

      H        G        F        E        D        C        B        A
  |---------|---------|---------|---------|---------|---------|---------|---------|
  |         |         |         |         |         |         |         |         |
1 |         |         |    X    |    X    |    X    |         |         |         | 1
  |         |         |         |         |         |         |         |         |
  |---------|---------|---------|---------|---------|---------|---------|---------|
  |         |         |         |         |         |         |         |         |
2 |         |         |         |         |         |         |         |         | 2
  |         |         |         |         |         |         |         |         |
  |---------|---------|---------|---------|---------|---------|---------|---------|
  |         |         |         |         |         |         |         |         |
3 |         |         |         |         |         |         |         |         | 3
  |         |         |         |         |         |         |         |         |
  |---------|---------|---------|---------|---------|---------|---------|---------|
  |         |         |         |         |         |         |         |         |
4 |         |         |         |         |         |         |         |         | 4
  |         |         |         |         |         |         |         |         |
  |---------|---------|---------|---------|---------|---------|---------|---------|
  |         |         |         |         |         |         |         |         |
5 |         |         |         |         |         |    X    |         |         | 5
  |         |         |         |         |         |         |         |         |
  |---------|---------|---------|---------|---------|---------|---------|---------|
  |         |         |         |         |         |         |         |         |
6 |         |         |         |         |         |         |         |         | 6
  |         |         |         |         |         |         |         |         |
  |---------|---------|---------|---------|---------|---------|---------|---------|
  |         |         |         |         |         |         |         |         |
7 |         |         |         |         |         |         |         |         | 7
  |         |         |         |         |         |         |         |         |
  |---------|---------|---------|---------|---------|---------|---------|---------|
  |         |         |         |         |         |         |         |         |
8 |         |         |    O    |    O    |    O    |    O    |         |         | 8
  |         |         |         |         |         |         |         |         |
  |---------|---------|---------|---------|---------|---------|---------|---------|
      H        G        F        E        D        C        B        A

------------------------- PLAYER 2 TURN (O) -------------------------
> Which board piece do you want to move?
> Row|: 8.
> Column|: e.
> Where do you want to place the piece?
> Row|: 7.
> Column|: c.
> Invalid position to place the piece, please chose another one
> Row|:
```

Fig. 12 - Exemplo da seleção de uma posição inválida para colocar a peça.

```
|---------|---------|---------|---------|---------|---------|---------|---------|
|                  _____                  _       _    _                        |
|                 |     | _ _  ___  ___ | |_  ___ | |_ | |_  ___                |
|                 |  |  || | || . ||  _||  _|| -_||  _||  _|| . |               |
|                 |__  _||___||__,||_|  |_|  |___||_|  |_|  |___|               |
|                    |__|                                                       |
|                                                                               |
|---------|---------|---------|---------|---------|---------|---------|---------|
|         |         |                                       |         |         |
|         |         |                                       |         |         |
|---------|---------|------|--------------------------|------|---------|---------|
|         |         |      |         1. Easy          |      |         |         |
|         |         |      |                          |      |         |         |
|         |         |      |         2. Normal        |      |         |         |
|---------|---------|------|                          |------|---------|---------|
|         |         |      |         3. Hard          |      |         |         |
|         |         |      |                          |      |         |         |
|         |         |      |         0. Exit          |      |         |         |
|---------|---------|------|--------------------------|------|---------|---------|
|         |         |                                       |         |         |
|         |         |        ----------------------         |         |         |
|         |         |               Nuno Lopes              |         |         |
|         |         |           Francisco Ferreira          |         |         |
|         |         |                                       |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|

> Insert your option |:
```

Fig. 13 - Menu de escolha da dificuldade do jogo.

```
------------------------- COMPUTER 2 MOVE (O) -------------------------
FROM:
 > Row: 8
 > Column: D
TO:
 > Row: 6
 > Column: D

      A         B         C         D         E         F         G         H
   |---------|---------|---------|---------|---------|---------|---------|---------|
   |         |         |         |         |         |         |         |         |
8  |         |         |    O    |         |    O    |    O    |         |         |  8
   |         |         |         |         |         |         |         |         |
   |---------|---------|---------|---------|---------|---------|---------|---------|
   |         |         |         |         |         |         |         |         |
7  |         |         |         |         |         |         |         |         |  7
   |         |         |         |         |         |         |         |         |
   |---------|---------|---------|---------|---------|---------|---------|---------|
   |         |         |         |         |         |         |         |         |
6  |         |         |         |    O    |         |         |         |         |  6
   |         |         |         |         |         |         |         |         |
   |---------|---------|---------|---------|---------|---------|---------|---------|
   |         |         |         |         |         |         |         |         |
5  |         |         |    X    |         |         |         |         |         |  5
   |         |         |         |         |         |         |         |         |
   |---------|---------|---------|---------|---------|---------|---------|---------|
   |         |         |         |         |         |         |         |         |
4  |         |         |         |         |         |         |         |         |  4
   |         |         |         |         |         |         |         |         |
   |---------|---------|---------|---------|---------|---------|---------|---------|
   |         |         |         |         |         |         |         |         |
3  |         |         |         |         |         |         |         |         |  3
   |         |         |         |         |         |         |         |         |
   |---------|---------|---------|---------|---------|---------|---------|---------|
   |         |         |         |         |         |         |         |         |
2  |         |         |         |         |         |         |         |         |  2
   |         |         |         |         |         |         |         |         |
   |---------|---------|---------|---------|---------|---------|---------|---------|
   |         |         |         |         |         |         |         |         |
1  |         |         |         |    X    |    X    |    X    |         |         |  1
   |         |         |         |         |         |         |         |         |
   |---------|---------|---------|---------|---------|---------|---------|---------|
      A         B         C         D         E         F         G         H

------------------------ PLAYER 1 TURN (X) -------------------------
> Which board piece do you want to move?
> Row|:
```

Fig. 14 - Exemplo de um estado de jogo entre o Computador e um Player

```
------------------------- COMPUTER 2 MOVE (O) -------------------------
FROM:
 > Row: 1
 > Column: C
TO:
 > Row: 6
 > Column: C

      A         B         C         D         E         F         G         H
   |---------|---------|---------|---------|---------|---------|---------|---------|
   |         |         |         |         |         |         |         |         |
8  |         |         |         |         |         |         |         |         |  8
   |         |         |         |         |         |         |         |         |
   |---------|---------|---------|---------|---------|---------|---------|---------|
   |         |         |         |         |         |         |         |         |
7  |         |         |         |         |         |         |         |         |  7
   |         |         |         |         |         |         |         |         |
   |---------|---------|---------|---------|---------|---------|---------|---------|
   |         |         |         |         |         |         |         |         |
6  |         |         |    O    |         |         |         |         |         |  6
   |         |         |         |         |         |         |         |         |
   |---------|---------|---------|---------|---------|---------|---------|---------|
   |         |         |         |         |         |         |         |         |
5  |         |    X    |         |         |         |         |    O    |         |  5
   |         |         |         |         |         |         |         |         |
   |---------|---------|---------|---------|---------|---------|---------|---------|
   |         |         |         |         |         |         |         |         |
4  |         |         |         |    X    |         |         |         |         |  4
   |         |         |         |         |         |         |         |         |
   |---------|---------|---------|---------|---------|---------|---------|---------|
   |         |         |         |         |         |         |         |         |
3  |         |         |         |         |         |         |         |         |  3
   |         |         |         |         |         |         |         |         |
   |---------|---------|---------|---------|---------|---------|---------|---------|
   |         |         |         |         |         |         |         |         |
2  |         |    O    |         |         |         |         |         |         |  2
   |         |         |         |         |         |         |         |         |
   |---------|---------|---------|---------|---------|---------|---------|---------|
   |         |         |         |         |         |         |         |         |
1  |    X    |         |         |         |         |    O    |         |    X    |  1
   |         |         |         |         |         |         |         |         |
   |---------|---------|---------|---------|---------|---------|---------|---------|
      A         B         C         D         E         F         G         H


------------------------------------------------------------------------
|                        COMPUTER 2 WINS (O)                           |
------------------------------------------------------------------------
```

Fig. 15 - Exemplo de uma vitória.

5 Conclusão

O presente projeto teve como principal objetivo aplicar o conhecimento adquirido nas aulas teóricas e práticas da unidade curricular de Programação em Lógica.

Podemos dizer que os nossos conhecimentos ao nível da programação em lógica foram vastamente aumentados, tendo sido alcançadas todas as metas a que nos propusemos na planificação do trabalho.

As maiores dificuldades foram encontradas na adaptação a um paradigma de programação ao qual não estávamos habituados e também na elaboração do predicado que faz a tomada de decisão sobre a melhor jogada a escolher num certo momento. No entanto, todos os problemas foram ultrapassados e estamos satisfeitos com as soluções encontradas.

Em suma, apesar das diferenças que PROLOG apresenta relativamente às linguagens a que estamos mais acomodados a trabalhar foi uma experiência bastante enriquecedora e que nos permitiu compreender melhor as vantagens e as desvantagens deste paradigma de programação.

6 Bibliografia

- http://www.iggamecenter.com/info/en/quartetto.html
- https://boardgamegeek.com/boardgame/35164/quartetto
- https://www.geeksforgeeks.org/check-given-four-points-form-square/
- Documentos fornecidos na página da unidade curricular presentes no Moodle.