# Meerkats

Relatório Trabalho Prático 1


Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

**Grupo Meerkats_2:**

Daniel Silva Reis – up201308586
Guilherme Vieira Pinto – up201305803

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

# Resumo

No âmbito da unidade curricular de Programação Lógica, foi-nos proposto o desenvolvimento de um jogo de tabuleiro, em linguagem ProLog.

O jogo apresentado pelo nosso grupo é o Meerkats. Focamo-nos na criação de uma aplicação de simples interação com o utilizador, com as intruções necessárias para que qualquer um entenda facilmente as regras e o objetivo do jogo.

Podemos afirmar que não alcançámos os objetivos pretendidos, criando uma versão parcialmente funcional do Meerkats para Prolog, com uma interface legível e uma interação intuitiva, mas com falhas na estruturação lógica do código, as quais teremos de futuramente afinar.

# Índice

# Introdução

O objetivo deste projeto será a criação de um jogo simples e de fácil adaptação, disponibilizando na consola a informação indispensável para que o utilizador entende que ações lhe estão disponíveis. Para além de uma aplicação simples, pretendemos apresentar um programa robusto que seja capaz de cobrir qualquer jogada e condição que surja.

Durante este relatório começaremos por introduzir as principais regras do jogo, seguido de uma explicação detalhada da lógica implementada no código entregue e, por fim, a demonstração, com casos ilustrativos, da interface disponibilizada ao utilizador.

# O jogo Meerkats

### História

Meerkats, traduzido a "suricatas", corresponde a um jogo de tabuleiro com regras inspiradas no comportamento desta espécie de mamífero. Sendo que um grupo de suricatas é mais forte dependendo do seu tamanho e desempenho dos seus membros, também o objetivo do jogo consiste na criação do maior grupo de pedras de uma determinada cor.

### Objetivo

O objetivo do jogo é que cada jogador consiga agrupar o maior número possível de peças da sua cor. No final, cada participante revela a cor que sorteou ao início e vence o jogador que tiver o maior aglomerado de pedras da sua cor, em campo. Em caso de empate, verificam-se as dimensões dos segundos maiores aglomerados respetivos às cores empatadas.

### Equipamento

É utilizado um tabuleiro hexagonal, com cinco células de cada lado, juntamente com 16 peças de cada cor (azul, vermelho, verde e amarelo).

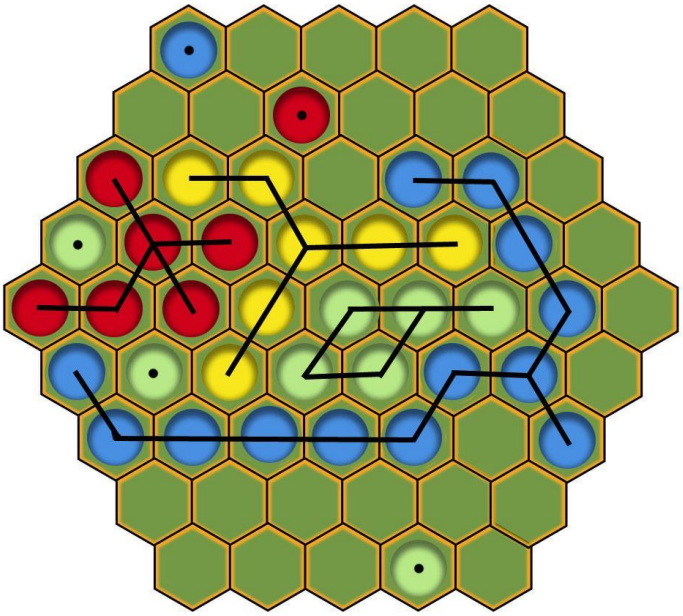
**Preparação**

O jogo começa com o tabuleiro inicialmente vazio, e cada jogador deve retirar, aleatoriamente, uma entre quatro peças de cores diferentes, sem reposição. A cor da pedra sorteada por esse jogador definirá a cor pela qual ele deverá jogar. A cor da peça que cada jogador retira deverá manter-se secreta até ao final do jogo.

Após o sorteio das cores, encontram-se disponíveis sessenta pedras (quinze de cada cor) para serem jogadas.

**Grupos**

O termo ‘grupo’, neste jogo, é definido como um conjunto de peças da mesma cor que se encontram interligadas (uma única peça pode ser considerada um grupo).



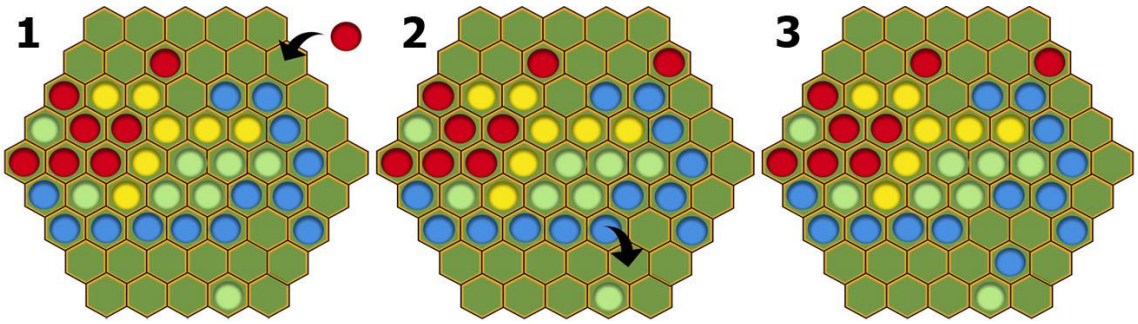
A imagem representativa do tabuleiro, acima apresentada, demonstra uma possível situação de jogo na qual se podem distingui diversos grupos de pedras das quatro cores diferentes. Dado que basta duas peças serem adjacentes para formarem um grupo, conclui-se que a cor azul possui a liderança.

**Rondas**

Na primeira ronda, o primeiro jogador deverá escolher uma peça e coloca-la em campo. Depois desta primeira jogada, cada jogador, no seu turno, deverá não só continuar a colocar uma nova peça, sob as mesmas condições, mas deverá também mover uma pedra presente no tabuleiro para um espaço vazio.

**Jogada**

O jogador deverá depositar uma pedra de qualquer cor, retirada aleatoriamente do saco, num espaço livre do tabuleiro e de seguida deslocar uma outra peça, já em campo, para outro espaço livre.




**Restrições**

Uma pedra pode ser movida em linha reta, em qualquer direção, até que uma outra pedra se depare no seu caminho ou os limites do tabuleiro lhe bloqueiem o movimento.

É, também, estritamente proibido que os jogadores comuniquem entre si durante o jogo.

# Logica de Jogo

## Representação da Logica de Jogo

A informação do jogo será armazenada numa estrutura composta por uma lista de listas. A lista inicial representará o tabuleiro, a secundária armazenará as posições de cada fila do campo e as respetivas sublistas apresentarão o estado de cada posição.

Tendo em atenção que o campo de jogo apresenta um formato hexagonal, com 5 unidades de cada lado, as listas secundárias serão constituídas por um número variável de elementos.

Nas figuras seguintes apresentamos os diferentes estados da lista ao longo do desenrolar do jogo:

```
1- Lista inicial
logicalBoard([
                    [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, empty, empty, empty, empty],
                    [empty, empty, empty, empty, empty]
    ]).
```

```
2- Lista de jogo a decorrer
logicalBoard([
                    [empty, red, green, blue, yellow],
                [empty, green, green, red, blue, empty],
            [green, empty, empty, empty, yellow, blue, yellow],
        [empty, blue, blue, blue, yellow, blue, yellow, yellow],
    [green, empty, green, empty, red, empty, yellow, yellow, yellow],
        [green, empty, empty, blue, red, blue, blue, empty],
            [green, green, red, empty, empty, red, yellow],
                [red, empty, empty, empty, yellow, yellow],
                    [red, red, red, red, blue]
    ]).
```

```
3- Final de jogo 1 (15 peças da mesma cor a formar um grupo)
logicalBoard([
                [red, red, green, blue, yellow],
              [green, green, green, red, blue, yellow],
            [green, green, yellow, blue, yellow, blue, yellow],
          [green, blue, blue, blue, yellow, blue, yellow, yellow],
        [green, green, green, yellow, red, blue, yellow, yellow, yellow],
          [green, green, green, blue, red, blue, blue, empty],
            [green, green, red, red, blue, red, yellow],
              [red, red, red, blue, yellow, yellow],
                [red, red, red, red, blue]
        ]).


4-  Final de jogo 2 (Todas as peças do tabuleiro jogadas)
    logicalBoard([
                [red, red, green, yellow, red],
              [green, green, green, red, blue, yellow],
            [green, green, yellow, blue, yellow, blue, yellow],
          [green, blue, blue, blue, yellow, blue, yellow, yellow],
        [green, green, green, yellow, yellow, blue, red, yellow, yellow],
          [green, green, green, blue, red, blue, blue, red],
            [green, green, red, blue, blue, red, yellow],
              [red, red, red, blue, yellow, yellow],
                [red, empty, red, red, blue]
        ]).
```

Uma posição vazia na lista de posições do tabuleiro será representada por ‘empty’. Por sua vez, a posição ocupada por cada pedra guardará a sua respetiva cor, podendo esta ser azul(‘blue’) vermelha(‘red’), verde(‘green’) ou amarela(‘yellow’).

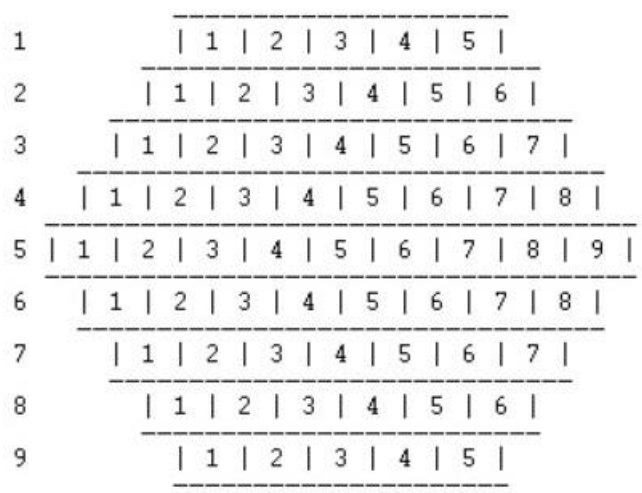
# Visualização do Tabuleiro

O tabuleiro é representado através da interface textual do SWI-Prolog. Para o efeito, foram criadas várias funções que permitem visualizar o tabuleiro.

*displayBoard(+Board), logicalBoard(+LogicalBoard),drawBoard(Board,LogicalBoard).*

O predicado *displayBoard(+Board)* é responsável por inicializar as posições do tabuleiro. Através de *logicalBoard(+LogicalBoard),* é criado o tabuleiro que irá guardar as pedras já jogadas no tabuleiro.

Chamando a função acima descrita é possível obter a visualização do tabuleiro:




# Jogadas Válidas

Durante o jogo, será pedido ao jogador para que introduza as coordenadas do tabuleiro para onde deseja mover a peça:

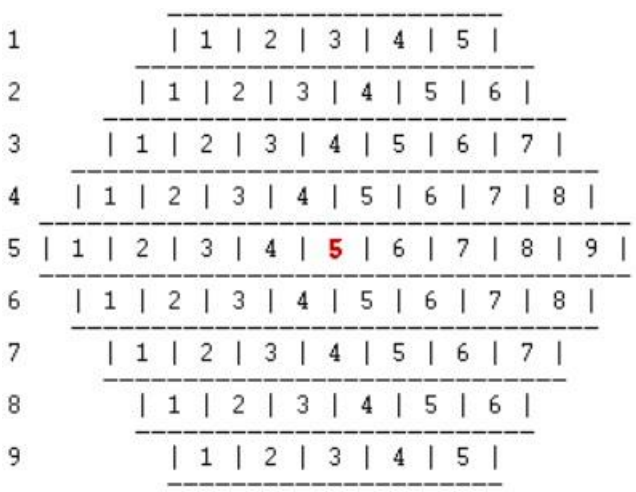
Através do predicado:

*dropStone (LogicalBoard, Stones, RemainingStones1, RowIdentifier, RowPos, ResultBoard1).*

Este predicado pergunta ao utilizador para onde deseja mover as peça e qual a peça que pretende colocar no tabuleiro (pedindo em caso de coordenadas erradas novas coordenadas e em caso de pedra inválida outra pedra) e coloca a peça no tabuleiro.

Depois de chamado, retorna as pedras que ainda faltam jogar (*RemainingStones1)*, a posição onde introduziu a peça (*RowIdentifier*, *RowPos*) e o tabuleiro resultante da jogada(*ResultBoard1*).

A seguinte imagem representa o tabuleiro resultante da movimentação da peça através do predicado para a posição de coordenadas (5,5):



# Execução de Jogadas

Em cada jogada será pedido ao jogador para que efetue 2 operações.

Uma delas será para que introduza uma peça numa coordenada válida do tabuleiro e a outra para que mova uma peça sem ser a que acabou de jogar numa direção válida à escolha do mesmo.

Para isto recorre a 2 predicados:

*dropStone (LogicalBoard, Stones, RemainingStones1, RowIdentifier, RowPos, ResultBoard1).*

*dragStone(FinalBoard,RowIdentifier,RowPos,FinalBoard1).*

# Avaliação do Tabuleiro

A avaliação do tabuleiro é feita através dos predicados:

*getWinningOnDrop(Board,RemainingStones1,FinalBoard,Tail,Players,ResultBoard,RemainingStones,2,Winner,RowIdentifier,RowPos)*

*getWinningOnDrag(Board,RemainingStones1,FinalBoard1,Tail,Players,ResultBoard,RemainingStones,2,Winner,RowIdentifier,RowPos)*

A primeira faz a verificação para determinar se existe um vencedor após a colocação de uma peça no tabuleiro pelo jogador. A segunda faz a verificação para determinar se existe um vencedor após  a movimentação de uma peça no tabuleiro pelo jogador.

## Critério de Vitória

Um jogador vence o jogo se conseguir obter um grupo de 15 peças ou então após todas as peças terem sido jogadas e ele tem o maior grupo.

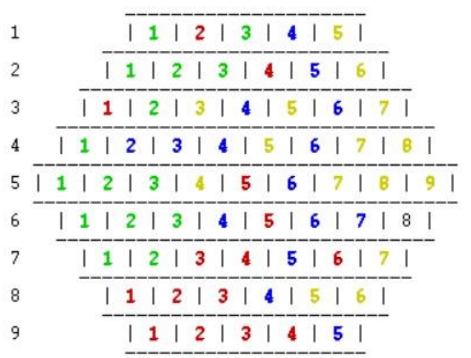



*Figura 1- Final de jogo após todas as peças jogadas*

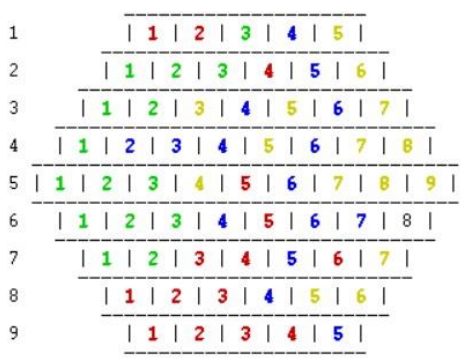



*Figura 2- Final de jogo após maior grupo de 15 peças*

## Final de Jogo

Sempre que o ciclo de jogo é executado, o seguinte predicado é chamado:

*playRound(Board,Stones,LogicalBoard,Players,Players,ResultBoard,RemainingStones, Round,WinningPlayer).*

Se em algum momento o *WinningPlayer* contiver algum valor válido, o ciclo de jogo é interrompido e é apresentado o jogador que ganha o jogo retornando novamente ao menu inicial de selecção.

## Jogada do Computador

O computador apenas tem um modo que é um modo fácil. Neste modo o computador joga aleatoriamente, ou seja, coloca uma peça num sítio aleatório do tabuleiro e move outra diferente para um sítio igualmente aleatório.

O predicado que faz a sua movimentação é o seguinte:

*playBot(Board,Stones,LogicalBoard,RemainingPlayers,Players,ResultBoard,Remaining Stones,N,Winner).*

# Interface com o jogador

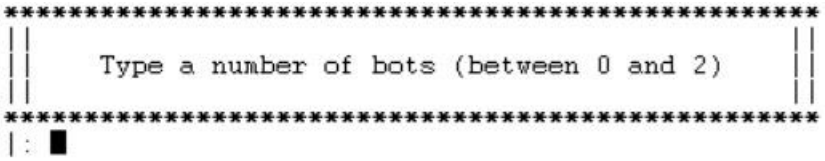
Quando um jogador inicial o jogo, é apresentado um menu que lhe permite escolher entre jogar, saber como se joga ou então sair do programa.

```
**********************************
||            MEERKATS          ||
||                              ||
||  1 - Play                    ||
||  2 - How to                  ||
||  3 - Exit Game               ||
||                              ||
**********************************
Choose an option:
|:
```

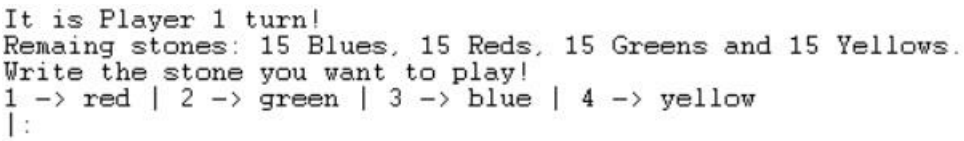
Caso se escolha jogar, é apresentado um menu de seleção de jogadores.

```
**************************************************
||                                              ||
||   Type a number of players (between 2 or 4)  ||
||                                              ||
**************************************************
|:
```

É depois selecionado um menu que pergunta quantos bots se querem no jogo.




Quando o jogo inicia, é pedido coordenadas de input para a colocação da peça no tabuleiro.




# Conclusões

A elaboração deste jogo exigiu bastante esforça e tempo para a sua implementação.

Lamentamos a incompletude do projeto, que se deveu à pobre gestão do tempo que deveríamos ter aplicado na sua elaboração. Devemos, no entanto, comprometer-nos a completar os objetivos pretendidos e implementar o as funcionalidades que não foram devidamente concluídas, nomeadamente o bot inteligente a funcionar completamente.

# Anexos

O código do projeto é enviado juntamente com o projeto em anexo.