# O jogo de Xadrez

Alexandra Bernardo & Ricardo Fernandes

09 de Janeiro de 2003

# Conteúdo

**1 Objectivo** **3**

**2 Motivação** **5**

**3 Funcionalidades** **7**

**4 Estrutura do Programa** **9**

4.1 Base de conhecimento . . . . . 10

4.2 Inicialização das peças no tabuleiro . . . . . 11

4.3 Inicialização do jogo . . . . . 12

4.4 Movimentos dos Jogadores . . . . . 12

4.5 Jogada do jogador . . . . . 12

4.6 Pedido dos nomes dos jogadores . . . . . 12

4.7 Menu do jogo . . . . . 12

4.8 Desenho do tabuleiro . . . . . 13

4.8.1 Aspecto inicial do tabuleiro . . . . . 13

4.9 Regras das peças . . . . . 13

4.9.1 Torre . . . . . 13

4.9.2 Cavalo . . . . . 14

4.9.3 Bispo . . . . . 14

4.9.4 Raínha . . . . . 15

4.9.5 Rei . . . . . 15

4.9.6 Peão . . . . . 16

4.10 Movimento das peças . . . . . 16

4.11 Heurísticas . . . . . 17

4.11.1 Valor inicial das peças . . . . . 17
4.11.2 Distância do centro . . . . . 17
4.11.3 Distância do rei . . . . . 17
4.11.4 Posição diferente da inicial . . . . . 18
4.11.5 Peças defendidas . . . . . 19
4.12 Algoritmo Min Max com cortes Alfa beta . . . . . 19

**5 Esquemas de Representação** **21**

**6 Implementação dos Esquemas de Representação** **23**

**7 Complexidade** **25**

**8 Ambiente de desenvolvimento** **27**

**9 Conclusão** **29**

**10 Melhoramentos** **31**

**A Manual do Utilizador** **35**

**B Exemplo de uma execução** **37**
B.1 1-Humano vs Humano . . . . . 37
B.1.1 Início do jogo . . . . . 37
B.1.2 Movimento inválido . . . . . 39
B.1.3 Fim do jogo . . . . . 40
B.2 2-Humano vs Computador (nível 1) 3- Humano vs Computador (nível 2) . . . . . 40

**C Listagem do código** **41**

# Capítulo 1

# Motivação

"*O xadrez já não pode ser entendido como um jogo, senão que apesar de seu carácter de entretenimento é uma prova de força e testemunho da mente humana. Se reflecte em nossa história cultural e enxadrística, tanto como em nossos comportamentos considerados clássicos, como ocorre na história política. A natureza humana se mostra, aqui como lá, em grande profundidade ideológica, no auxílio paciente, no temperamento, na vigorosa agressividade, na preparação táctica e na cegueira repentina.*"

"*Sabendo que o xadrez é um reflexo da condição humana, sua prática suporta formas peculiares de raciocinar cuja origem tem a ver com a ver com a base ideológica de cada um. O jogo/ciência, unem-se ao ideológico por meio do carácter, da experiência e da teoria. Assim, o jogador (do xadrez) planeja, decifra, constrói e determina a partida segundo sua maneira de pensar e sua ideologia. Em certa medida, o êxito ou o fracasso dependem da solidez e da profundidade desta.*"

Com os conhecimentos teóricos adquiridos na disciplina MIA, a motivação de aplicar estes conhecimentos num jogo com o nível de complexidade do xadrez , foi bastante elevada.

# Capítulo 2

# Funcionalidades

No início da execução do programa é disponibilizado um menu que permite ao utilizador fazer a escolha de entre três possíveis jogos:

1. Jogador 1 contra Jogador 2;
2. Jogador 1 contra computador (nível 1);
3. Jogador 1 contra computador (nível 2).

No primeiro caso, Jogador 1 contra Jogador 2, é pedida uma identificação aos utilizadores. Essa identificação não pode conter espaços logo o uso de um *nick name* será o mais apropriado. A partir desse momento o programa informa os jogadores sobre a cor das peças com que cada um irá jogar e é apresentado o tabuleiro que tem o seguinte aspecto:

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | TB | HB | BB | KB | QB | BB | HB | TB |
| 2 | PB | PB | PB | PB | PB | PB | PB | PB |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |
| 7 | PW | PW | PW | PW | PW | PW | PW | PW |
| 8 | TW | HW | BW | KW | QW | BW | HW | TW |

As peças do tabuleiro estão representados em inglês. Este facto deve-se à possível confusão da representação do Rei e da Rainha, pois ambos começam pela mesma letra. Assim, adoptou-se

a língua inglesa e tem-se KB (King Black) para representar o Rei Preto e QB (Queen Black) para representar a Raínha Preta:

- TB representa Torre Preta;
- HB representa Cavalo Preto;
- BB representa Bispo Preto;
- KB representa Rei Preto;
- QB representa Raínha Preta;
- PB representa Peão Preto;
- TW representa Torre Branca;
- HW representa Cavalo Branco;
- BW representa Bispo Branco;
- KW representa Rei Branco;
- QW representa Raínha Branca;
- PW representa Torre Branca;

Em seguida, é pedido ao Jogador 1 que faça a sua jogada e se essa for permitida passa a vez ao Jogador 2. Caso algum dos jogadores pretenda avançar um movimento ilegal, terá oportunidade de refazer o pedido.

No segundo caso, existe apenas a necessidade de existência de um jogador a quem se pede a identificação e se informa sobre as peças com que irá jogar. A partir daí o jogo inicia e é pedido ao jogador que faça o seu movimento e de seguida o computador fará a sua jogada.

O terceiro caso apenas se distingue do segundo pelo nível de profundidade do algoritmo usado. É suposto que nesta funcionalidade o computador seja mais "inteligente" que no segundo caso, daí o nível dois. Para o efeito, e no primeiro caso, usou-se o algoritmo Min-Max com profundidade um (melhor primeira jogada). No segundo caso o algoritmo Min-Max com cortes Alpha-Beta com nível de profundidade dois.

# Capítulo 3

# Estrutura do Programa

O programa está dividido nas seguintes partes:

1. Base de conhecimento;
2. Inicialização das peças no tabuleiro;
3. Inicialização do jogo;
4. Movimentos dos Jogadores;
5. Jogada do jogador;
6. Pedido dos nomes dos jogadores;
7. Menu do jogo;
8. Desenho do tabuleiro;
9. Movimento das peças;
10. Regras das peças;
11. Verificação da existência ou não existência de peças intermédias;
12. Funções Geradoras de movimentos;
13. Funções Auxiliares;
14. Função Avaliadora do tabuleiro;
15. Função Mover Peça para o computador;

16. Algoritmo Min-Max , com cortes alpha-beta;

17. Função avaliadora;

## 3.1 Base de conhecimento

A base de conhecimento é composta pelas posições iniciais das peças no jogo e pelo seu valor (valor(Peca, Valor)). É também aqui que se indica o número de linhas e colunas total necessárias para o desenho posterior do tabuleiro. Existe ainda uma base de conhecimento, diga-se, virtual composta pelas posições actuais das peças em qualquer momento do jogo bem como um cemitério de peças que indica ao programa quais as peças que saíram de jogo.

```
:-dynamic cemiterio/2.


posicao_inicial('TB','A',1).
posicao_inicial('HB','B',1).
posicao_inicial('BB','C',1).
posicao_inicial('KB','D',1).
posicao_inicial('QB','E',1).
posicao_inicial('BB','F',1).
posicao_inicial('HB','G',1).
posicao_inicial('TB','H',1).
posicao_inicial('TW','A',8).
posicao_inicial('HW','B',8).
posicao_inicial('BW','C',8).
posicao_inicial('KW','D',8).
posicao_inicial('QW','E',8).
posicao_inicial('BW','F',8).
posicao_inicial('HW','G',8).
posicao_inicial('TW','H',8).
posicao_inicial('PB','A',2).
posicao_inicial('PB','B',2).
posicao_inicial('PB','C',2).
posicao_inicial('PB','D',2).
```

```
posicao_inicial('PB','E',2).
posicao_inicial('PB','F',2).
posicao_inicial('PB','G',2).
posicao_inicial('PB','H',2).
posicao_inicial('PW','A',7).
posicao_inicial('PW','B',7).
posicao_inicial('PW','C',7).
posicao_inicial('PW','D',7).
posicao_inicial('PW','E',7).
posicao_inicial('PW','F',7).
posicao_inicial('PW','G',7).
posicao_inicial('PW','H',7).

linhas(19).
colunas(37).
valor(80,100).
valor(66,325).
valor(72,325).
valor(84,500).
valor(81,1000).
valor(75,20000).
```

Note-se que o 80, 66, 72, 84, 81, 75 são o código ASCII das letras P, B, H, T, Q, K que dizem respeito às iniciais das peças intervenientes no jogo.

## 3.2 Inicialização das peças no tabuleiro

É nesta secção do código que se inicializa o tabuleiro e se cria a tal base de conhecimento virtual que diz respeito às posições actuais das peças no tabuleiro. No início, a posição actual toma o valor da posição inicial apresentada na secção anterior e vai sendo alterada à medida que o jogo avança:

$$posicao\_actual(Peca, X, Y, nivelDeProfundidade)$$

onde *nivelDeProfundidade* indica o nível de profundidade da peça *Peca* que se encontra na coordenada (X,Y).

## 3.3 Inicialização do jogo

Neste momento, o programa está pronto a ser inicializado. Faz-se uma limpeza da base de conhecimento virtual e dependendo do número de jogadores envolvidos começa-se o jogo.

## 3.4 Movimentos dos Jogadores

Ao iniciar a execução do programa é sabido qual o número de jogadores envolvido por causa da escolha do menu. Sendo assim, há a necessidade de fazer com que cada um dos jogadores tenha oportunidade de jogar e saiba quando fazê-lo. Uma vez que existem três modalidades diferentes de jogo é aqui que se faz a alternância de jogadores quer sejam ambos humanos ou não.

## 3.5 Jogada do jogador

Quando um jogador introduz a sua jogada, há a necessidade de converter a mesma para valores que o programa possa processar. Neste caso concreto, a conversão num dos eixos é feita para código ASCII, para facilitar a introdução das jogadas por parte do jogador. Esta medida foi tomada como necessária embora se pudesse jogar com um tabuleiro que apresentasse a numeração de 1 a 8 na horizontal e vertical.

## 3.6 Pedido dos nomes dos jogadores

Uma vez que existe a opção do jogo ser jogado com dois jogadores, nesta secção faz-se o controlo e guarda-se a informação que se acha necessária em relação aos jogadores, neste caso o nome, para que a interface se torne mais amigável.

## 3.7 Menu do jogo

Antes de começar a jogar, existe um menu que permite ao jogador escolher a modalidade que pretende jogar. O menu é constituído por:

1. Jogador 1 contra Jogador 2

2. Jogador 1 contra computador (nível 1)

3. Jogador 1 contra computador (nível 2)

4. Sair

## 3.8 Desenho do tabuleiro

O tabuleiro é construído dinamicamente, isto é, consegue construir-se um tabuleiro de dimensão $n \times n$.

### 3.8.1 Aspecto inicial do tabuleiro

Este é o aspecto inicial do tabuleiro:

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | TB | HB | BB | KB | QB | BB | HB | TB |
| 2 | PB | PB | PB | PB | PB | PB | PB | PB |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |
| 7 | PW | PW | PW | PW | PW | PW | PW | PW |
| 8 | TW | HW | BW | KW | QW | BW | HW | TW |

## 3.9 Regras das peças

### 3.9.1 Torre

É permitido à torre mover-se na horizontal ou na vertical quantas casas pretender.

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | | | | ↑ | | | | |
| 2 | | | | ↑ | | | | |
| 3 | | | | ↑ | | | | |
| 4 | ← | ← | ← | T | → | → | → | → |
| 5 | | | | ↓ | | | | |
| 6 | | | | ↓ | | | | |
| 7 | | | | ↓ | | | | |
| 8 | | | | ↓ | | | | |

### 3.9.2 Cavalo

É permitido ao cavalo mover-se em L.

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

### 3.9.3 Bispo

É permitido ao Bispo mover-se na na diagonal quantas casas pretender.

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | ↖ | | | | | | ↗ | |
| 2 | | ↖ | | | | ↗ | | |
| 3 | | | ↖ | | ↗ | | | |
| 4 | | | | B | | | | |
| 5 | | | ↙ | | ↘ | | | |
| 6 | | ↙ | | | | ↘ | | |
| 7 | ↙ | | | | | | ↘ | |
| 8 | | | | | | | | ↘ |

### 3.9.4 Raínha

É permitido à Raínha mover-se na diagonal, horizontal ou vertical quantas casas pretender.

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | ↖ | | | ↑ | | | ↗ | |
| 2 | | ↖ | | ↑ | | ↗ | | |
| 3 | | | ↖ | ↑ | ↗ | | | |
| 4 | ← | ← | ← | Q | → | → | → | → |
| 5 | | | ↙ | ↓ | ↘ | | | |
| 6 | | ↙ | | ↓ | | ↘ | | |
| 7 | ↙ | | | ↓ | | | ↘ | |
| 8 | | | | ↓ | | | | ↘ |

### 3.9.5 Rei

É permitido ao Rei mover-se na diagonal, horizontal ou vertical mas apenas uma casa em cada direcção.

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

### 3.9.6 Peão

É permitido ao Peão mover-se na vertical apenas uma casa (⊗) e só numa direcção. No entanto, é permitido que o Peão se mova duas casas na primeira jogada.

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | |
| 2 | PB | | | | | | | |
| 3 | ⊗ | | | | | | | |
| 4 | ⊕ | | PB | | | | | |
| 5 | | | ⊗ | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |
| 8 | | | | | | | | |

## 3.10 Movimento das peças

Depois da jogada ser validada pelas regras das peças, o movimento desejado é realizado. Para isso existem dois tipos de acções:

- comer - retira-se a peça comida do tabuleiro e altera-se o estado da peça a ser movida;
- mover - alteração do estado da peça.

## 3.11 Heurísticas

Para que o computador tome a decisão do movimento mais correcto, é necessário dar algumas informações sobre o jogo - Heurísticas. Quanto melhor a heurística, melhor a resposta do computador.

### 3.11.1 Valor inicial das peças

No início do jogo o valor das peças é o seguinte:

- O Peão vale 100 pontos.
- A Torre vale 325 pontos.
- O Cavalo vale 325 pontos.
- O Bispo vale 500 pontos.
- A Raínha vale 1000 pontos.

### 3.11.2 Distância do centro

O Cavalo e o Peão receberão um bonús equivalente ao apresentado no quadro a seguir por aproximação do centro do tabuleiro;

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 5 | 10 | 15 | 15 | 10 | 5 | 0 |
| 2 | 5 | 10 | 15 | 20 | 20 | 15 | 10 | 5 |
| 3 | 10 | 15 | 20 | 25 | 25 | 20 | 15 | 10 |
| 4 | 15 | 20 | 25 | 30 | 30 | 25 | 20 | 15 |
| 4 | 15 | 20 | 25 | 30 | 30 | 25 | 20 | 15 |
| 6 | 10 | 15 | 20 | 25 | 25 | 20 | 15 | 10 |
| 7 | 5 | 10 | 15 | 20 | 20 | 15 | 10 | 5 |
| 8 | 0 | 5 | 10 | 15 | 15 | 10 | 5 | 0 |

### 3.11.3 Distância do rei

A Raínha, a Torre e o Cavalo são compensados se se encontrarem perto do Rei. Nos quadros seguintes, suponha-se que a peça correspondente que ganhará pontos se encontra na posição H8.

**Caso da Rainha**

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | KC | 130 | 120 | 110 | 100 | 90 | 80 | 70 |
| 2 | 130 | 120 | 110 | 100 | 90 | 80 | 70 | 60 |
| 3 | 120 | 110 | 100 | 90 | 80 | 70 | 60 | 50 |
| 4 | 110 | 100 | 90 | 80 | 70 | 60 | 50 | 40 |
| 4 | 100 | 90 | 80 | 70 | 60 | 50 | 40 | 30 |
| 6 | 90 | 80 | 70 | 60 | 50 | 40 | 30 | 20 |
| 7 | 80 | 70 | 60 | 50 | 40 | 30 | 20 | 10 |
| 8 | 70 | 60 | 50 | 40 | 30 | 20 | 10 | 0 |

**Caso da Torre ou Cavalo**

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | KC | 65 | 60 | 55 | 50 | 45 | 40 | 35 |
| 2 | 65 | 60 | 55 | 50 | 45 | 40 | 35 | 30 |
| 3 | 60 | 55 | 50 | 45 | 40 | 35 | 30 | 25 |
| 4 | 55 | 50 | 45 | 40 | 35 | 30 | 25 | 20 |
| 4 | 50 | 45 | 40 | 35 | 30 | 25 | 20 | 15 |
| 6 | 45 | 40 | 35 | 30 | 25 | 20 | 15 | 10 |
| 7 | 40 | 35 | 30 | 25 | 20 | 15 | 10 | 5 |
| 8 | 35 | 30 | 25 | 20 | 15 | 10 | 5 | 0 |

### 3.11.4 Posição diferente da inicial

O peão recebe um bónus de 10 vezes o número de casas em que se encontra e que é diferente da posição inicial. No tabuleiro seguinte encontra-se a pontuação recebida peça/valor.

**Peão**

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | PW/60 |
| 2 | PB/0 | | | | | | | PW/50 |
| 3 | PB/10 | | | | | | | PW/40 |
| 4 | PB/20 | | | | | | | PW/30 |
| 5 | PB/30 | | | | | | | PW/20 |
| 6 | PB/40 | | | | | | | PW/10 |
| 7 | PB/50 | | | | | | | PW/0 |
| 8 | PB/60 | | | | | | | |

### 3.11.5 Peças defendidas

#### por uma peça igual

O peão recebe um bónus de 10 pontos por estar defendido por um peão. O bispo e a torre recebem um bónus de 25 pontos por serem defendidos por um bispo e uma torre respectivamente.

#### Por uma peça diferente

O bispo e a torre recebem um bónus de 25 pontos por serem defendidos pela rainha.

## 3.12 Algoritmo Min Max com cortes Alfa beta

A codificação do algoritmo foi a última parte da estrutura do programa a ser desenvolvida.

# Capítulo 4

# Esquemas de Representação

O jogo de Xadrez tem vários tipos de conhecimentos que têm que serem representados numa forma simples. Esses conhecimentos são:

- *dados dos jogadores*- o número de jogador e o seu respectivo nome;
- *vários tipos de jogos*- Humano vs Humano, Humano vs Computador, etc..;
- *tabuleiro inicial*- conjunto de posições de todas as peças do tabuleiro;
- *tabuleiro actual*- conjunto de posições das peças que ainda estão em jogo;
- *tabuleiro "futuro"*- conjunto de posições das peças que formam um tabuleiro simulado se efectuar uma acção virtual (Algoritmo MinMax com cortes Alpha Beta);
- *regras do jogo*- conjunto de regras válidas do jogo, tais como: os movimentos, fim de jogo;
- *peças comidas*- conjunto de peças que foram mortas (ou comidas) pelos jogadores;

# Capítulo 5

# Implementação dos Esquemas de Representação

Os esquemas de representação de conhecimento foram implementados usando factos. Por exemplo, para representar os tabuleiros:

$$posicao\_inicial(Peca, X, Y)$$

$$posicao\_actual(Peca, X, Y, Nivel\_De\_Profundidade)$$

O nível de profundidade, no facto *posicao_actual*, tem como finalidade poder representar o tabuleiro actual e o tabuleiro final. Para as peças do tabuleiro actual o valor do nível de profundidade deve ser igual a zero.

Não foram usadas, nestas representações, as listas do Prolog, pois os factos são mais intuitivos e fáceis de serem utilizados.

No Apêndice C encontra-se a listagem do código onde se poderá consultar as outras representações do conhecimento.

# Capítulo 6

# Complexidade

O espaço de estados do jogo de Xadrez é bastante grande. A partir de um tabuleiro em qualquer fase do jogo, podem realizar-se, em média, trinta e cinco jogadas e como a duração de um jogo envolve **aproximadamente** cinquenta movimentos, para gerar a árvore de todos os movimentos possíveis (ou tabuleiros) seriam necessários cerca de $35^{100}$ tabuleiros. Para diminuir o número de tabuleiros e aumentar a performance do jogo a nível de tempo, usou-se o algoritmo Min-Max com cortes Alfa- Beta com nível dois de profundidade.

# Capítulo 7

# Ambiente de desenvolvimento

O trabalho foi realizado num computador HP Pentium IV a 1.4, com 392MB de memória RAM cujo sistema operativo é o 2000 Professional. Para além deste, o programa também foi testado HP, OmiBook Xe3L, com Processador Intel Celeron a 930 MHZ com 397 MB de memória RAM cujo sistema operativo é o XP Professional.

Em ambos os casos foi usado o programa SWI-Prolog version 5.0.10 by Jan Wielemaker. O programa foi escrito em PROLOG e não foi usado qualquer tipo de pacotes de ‘software” extra.

# Capítulo 8

# Conclusão

Como é sabido, o jogo do xadrez é um jogo de elevado nível de complexidade. O número de peças e regras envolvidas são a causa dessa complexidade.

A passagem da filosofia do xadrez para o programa foi um passo difícil mas o resultado obtido é bastante satisfatório uma vez que o objectivo de implementar um jogo de xadrez com o algoritmo Min-Max com cortes Alfa-Beta foi alcançado. O algoritmo atrás mencionado diminuiu significativamente o tempo de resposta por parte do computador. Apesar destes tempos não serem os inicialmente pretendidos, o trabalho desenvolvido na criação das heurísticas, função avaliadora e a função geradora eram objectivos primários. A diminuição do tempo poderia ser conseguida com a diminuição de heurísticas usadas.

# Capítulo 9

# Melhoramentos

Uma vez que o tempo disponível para realizar o trabalho de MIA não foi o desejado pois os compromissos profissionais nem sempre permitem essa flexibilidade a nível de tempo, a verdade é que a interface gráfica podia ser melhorada. No que diz respeito a esse aspecto, o uso de PROLOG para a concretização da tarefa. Apesar do PROLOG-SWI conter uma componente gráfica que se poderia ter experimentado mas a verdade é que o nível de dificuldade do trabalho e mais uma vez o tempo disponível, não permitiram o aprofundamento nessa àrea.

Outro aspecto que talvez se pudesse melhorar com o uso de outro tipo de programação seria o aspecto que se relaciona com os tempos conseguidos, principalmente no nível dois nota-se que a procura da melhor solução é um pouco lenta e mais ainda quando existem muitas peças no tabuleiro.

As heurísticas também poderiam ser melhoradas, principalmente no que diz respeito ao fim do jogo, mas caso houvesse oportunidade ter-se-ia implementado, com certeza o Check-Mate, o Rock, e a troca de peça com peão depois deste ter atravessado o tabuleiro são algumas regras fundamentais do xadrez que não foram implementadas devido à falta de tempo.

# Bibliografia

[1] Prolog Programming for artificial intelligence, Ivan Bratko;

[2] Artificial Intelligence. A modern approach, Stuart Russell and Peter Norvig;

[3] http://www.clubedexadrez.com.br, 30 de Dezembro de 2002;

[4] Xadrez, 2000 anos de história, Editora Anaya, 1989;

# Apêndice A

# Manual do Utilizador

Para utilizar este programa tem que ter-se acesso ao SWI-PROLOG.

1. Após a compilação, digita-se "**jogar.**".
2. É apresentado um menu com as seguintes quatro opções.

   1 Humano vs Humano

   2 Humano vs Computador (nível 1)

   3 Humano vs Computador (nível 2)

   4 Não quero jogar nada!
3. Escolhe-se a opção pretendida (de 1 a 4);
4. É pedida a identificação dos jogadores. Insira o nome que pretende para ser identificado e seguidamente será informado das peças com que vai jogar;
5. É apresentado o tabuleiro inicial e é pedida a jogada que pretende fazer. As entradas das jogadas são do tipo a2a4, em que as primeira e terceira coordenadas são valores de a a h ou de A a H e as segunda e quarta coordenadas de 1 a 8. O primeiro par de coordenadas refere-se à posição em que a peça se encontra e o segundo par para onde se pretende que a peça se mova.
6. Repetir o passo 5 até ao fim do jogo. Se pretender abandonar o jogo digite "**s**".

# Apêndice B

# Exemplo de uma execução

## B.1 1-Humano vs Humano

### B.1.1 Início do jogo

#### Apresentação dos jogadores

**Primeira Jogada**

### B.1.2 Movimento inválido

### B.1.3 Fim do jogo

## B.2 2-Humano vs Computador (nível 1) 3- Humano vs Computador (nível 2)

Nestes dois casos a apresentação do programa é semelhante. A única diferença é que existe apenas um jogador.

# Apêndice C

# Listagem do código

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% PROGRAMA - XADREZ-PROLOG
%
% Ricardo Fernandes e Alexandra Bernardo
%
% Metodologias de Inteligência Artificial
%
% MIACC 2002/2003
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% BASE DE CONHECIMENTO
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


:-dynamic cemiterio/2.
```

```
posicao_inicial('TB','A',1).
posicao_inicial('HB','B',1).
posicao_inicial('BB','C',1).
posicao_inicial('KB','D',1).
posicao_inicial('QB','E',1).
posicao_inicial('BB','F',1).
posicao_inicial('HB','G',1).
posicao_inicial('TB','H',1).
posicao_inicial('TW','A',8).
posicao_inicial('HW','B',8).
posicao_inicial('BW','C',8).
posicao_inicial('KW','D',8).
posicao_inicial('QW','E',8).
posicao_inicial('BW','F',8).
posicao_inicial('HW','G',8).
posicao_inicial('TW','H',8).
posicao_inicial('PB','A',2).
posicao_inicial('PB','B',2).
posicao_inicial('PB','C',2).
posicao_inicial('PB','D',2).
posicao_inicial('PB','E',2).
posicao_inicial('PB','F',2).
posicao_inicial('PB','G',2).
posicao_inicial('PB','H',2).
posicao_inicial('PW','A',7).
posicao_inicial('PW','B',7).
posicao_inicial('PW','C',7).
posicao_inicial('PW','D',7).
posicao_inicial('PW','E',7).
posicao_inicial('PW','F',7).
posicao_inicial('PW','G',7).
posicao_inicial('PW','H',7).
```

```
linhas(19).
colunas(37).
valor(80,100).
valor(66,325).
valor(72,325).
valor(84,500).
valor(81,1000).
valor(75,20000).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Inicialização das peças no tabuleiro
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


:-dynamic posicao_actual/4,jogadaN/1,nivel_prof/1.

iniciar:-(posicao_inicial(P,X,Y),not(posicao_actual(P,X,Y,0)),
    assert(posicao_actual(P,X,Y,0)),iniciar);!.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Inicialização do JOGO
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

jogar(Tipo_jogo):-
    retractall(posicao_actual(_,_,_,_)),
    retractall(jogador(_,_)),
    retractall(cemiterio(_,_)),
```

```
    retractall(jogadaN(_)),
    retractall(nivel_prof(_)),
    assert(jogadaN(0)),
    assert(nivel_prof(2)),
    iniciar,
    nome(Tipo_jogo),!,
    l(Tipo_jogo,1).

l(Tipo_jogo,Jogador):-
    (not(movimento(Tipo_jogo,Jogador)),
    ((Jogador=1,JogadorN=2,jogadaN(Num),retractall(jogadaN(_)),NumN is Num+1,
    assert(jogadaN(NumN)));(JogadorN=1)),
    l(Tipo_jogo,JogadorN));jogar.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Movimentos dos Jogadores
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


movimento(1,Jogador):-
    desenhar,
    jogador(Jogador,[Nome]),
    write(Nome),
    write(', qual a sua jogada?'),nl,
    readln(Jogada),nl,
    (((Jogada=['S'];Jogada=['s']),!,jogar);
     (((jogada(Jogada,Jogador),((Jogador=1,((validar_fim_Jogo(66,Nome),jogar);movimento(1,2)
        (Jogador=2,((validar_fim_Jogo(87,Nome),jogar);movimento(1,1)))));
     (nl,write(Nome),write(', efectuou um movimento inválido!!'),nl,movimento(1,Jogador))))
```

```
movimento(2,Jogador):-
    jogador(Jogador,[Nome]),
    ((Jogador=1,desenhar,!,write(Nome),write(', qual a sua jogada?'),nl,readln(Jogada),nl,
    (((Jogada=['S'];Jogada=['s']),retractall(cemiterio(_,_)),!,jogar);
    ((jogada(Jogada,Jogador),
    ((validar_fim_Jogo(66,Nome),retractall(cemiterio(_,_)),jogar);(!,fail)));
    (nl,write(Nome),write(', efectuou um movimento inválido!!'),nl,movimento(2,Jogador)))))
    (
            Jogador=2,
            movimento_maximo_valor(66,1,Xantes,Yantes,Xdepois,Ydepois,Accao),
            mover_PecaNivel(Xantes,Yantes,Xdepois,Ydepois,0,Accao),!,
            (validar_fim_Jogo(87,Nome);fail)
        )
     ).

movimento(3,Jogador):-
    jogador(Jogador,[Nome]),
    ((Jogador=1,desenhar,!,write(Nome),write(', qual a sua jogada?'),nl,readln(Jogada),nl,
    (((Jogada=['S'];Jogada=['s']),retractall(cemiterio(_,_)),!,jogar);
    ((jogada(Jogada,Jogador),
    ((validar_fim_Jogo(66,Nome),retractall(cemiterio(_,_)),jogar);(!,fail)));
    (nl,write(Nome),write(', efectuou um movimento inválido!!'),nl,movimento(3,Jogador)))))
    (
            Jogador=2,
            alphabeta([], -100000, 100000, [Xantes,Yantes,Xdepois,Ydepois,Accao,_,_],_),
            apagar_tab_aux(1),
            mover_PecaNivel(Xantes,Yantes,Xdepois,Ydepois,0,Accao),!,
            (validar_fim_Jogo(87,Nome);fail)
        )
     ).

validar_fim_Jogo(Cor,Nome):-
```

```
    fim_Jogo(Cor,0,Valor),
    (
        (
            (
                Valor=(-1),write('Fim de Jogo. Parabéns '),write(Nome),write('.')
            );
            (
                Valor=0,write('Fim de Jogo. Empate.')
            )
        ),nl,!
    );fail.



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Jogada do jogador
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


jogada([Jogada|R],Jogador):-R=[],
    name(Jogada,JogadaArray),
    converter(JogadaArray,[Xantes,Yantes,Xdepois,Ydepois]),
    ASCIIa is 64+Xantes,
    char_code(LetraA,ASCIIa),
    posicao_actual(Peca,LetraA,Yantes,_),
    name(Peca,[_,Cor]),
    ((Jogador=1,Cor=87);(Jogador=2,Cor=66)),
    mover_Peca(Xantes,Yantes,Xdepois,Ydepois).

converter([],[]).
```

```
converter([LetraA,NumA,LetraD,NumD],[LetraAn,NumAn,LetraDn,NumDn]):-
    ((LetraA>=65,LetraA=<72,LetraAn is LetraA-64);
    (LetraA>=97,LetraA=<104,LetraAn is LetraA-96)),
    ((LetraD>=65,LetraD=<72,LetraDn is LetraD-64);
    (LetraD>=97,LetraD=<104,LetraDn is LetraD-96)),
    (NumA>=49,NumA=<56,NumAn is NumA-48),
    (NumD>=49,NumD=<56,NumDn is NumD-48).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Pedido dos Nomes dos JOGADORES
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


nome(1):-
    write('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'),nl,nl,
    write('Quais os nomes dos jogadores?'),nl,nl,
    write('Jogador 1:'),
    readln(Nome1),
    assert(jogador(1,Nome1)),
    nl,
    write('Jogador 2:'),
    readln(Nome2),
    assert(jogador(2,Nome2)),
    nl,nl,
    jogador(1,[N1]),
    jogador(2,[N2]),
    write(N1),write(', vai jogar com as peças brancas.'),
    nl,nl,
    write(N2),write(', vai jogar com as peças pretas.'),nl,nl,
    write('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'),
```

```
    nl,nl,write('Boa Sorte!!'),nl,nl.

nome(X):-
    (X=2;X=3),
    write('Qual o nome do jogador que se atreve a jogar contra nós?'),nl,nl,
    write('Jogador:'),
    readln(Nome1),
    assert(jogador(1,Nome1)),
    nl,
    Nome='Xana&tbs',
    assert(jogador(2,[Nome])).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Menu do JOGO
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


jogar:-
    write('Jogo do Xadrez (Xana&Tbs Lda).'),
    nl,
    nl,
    write('     [1]Humano vs Humano'),
    nl,
    write('     [2]Humano vs Computador (nível 1)'),
    nl,
    write('     [3]Humano vs Computador (nível 2)'),
    nl,
    write('     [4]Não quero jogar nada!'),
    nl,
    readln(X),(validar(X);jogar).
```

```
validar([X]):-
    ((X=1;X=2;X=3),!,jogar(X));
    (X=4,nl,write('Adeus. Volte sempre.'));
    (write('O jogo '),
    write(X),
    write(' não está disponível!! Tente novamente!'),nl,nl,jogar).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Desenho do TABULEIRO
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

desenhar:-desenharTabuleiro(1).


desenharTabuleiro(Linha):-
        (linhas(Z),Linha=<Z,
        ((Y is Linha mod 2,Y=1,colunas(X),escrever_Linha(X),nl);
        (Y is Linha mod 2,Y=0,escrever_Coluna(Linha,1),nl)),
        LinhaN is Linha+1,desenharTabuleiro(LinhaN));!.

escrever_Linha(X):-(X>0,write('_'),escrever_Linha(X-1));!.


escrever_Coluna(2,X):-
        (colunas(C),X=<C,
        ((ver_traco(X,0),write('|'));
        (ver_espaco(X,0),write(' '));
        ((X=3,write(' '));(ver_letra(X,0),
        ASCII is 63+(X+1)/4,char_code(Letra,ASCII),write(Letra)))),
```

```
        Y is X+1,escrever_Coluna(2,Y));!.

escrever_Coluna(Y,X):-
        (colunas(C),X=<C,
        ((ver_traco(X,0),write('|'),Z is X+1);
        ((X=2;ver_espacoPeca(X,0)),write(' '),Z is X+1);
        (X=3,W is (Y/2)-1,write(W),Z is X+1);
        (ver_Peca(Y,X),Z is X+2);
        (write(' '),Z is X+1)),
        escrever_Coluna(Y,Z));!.

ver_traco(Pos,Max):-
        Max=<10,((Y is Pos-4*Max,Y=1,!);(MaxN is Max+1,ver_traco(Pos,MaxN))).


ver_espaco(Pos,Max):-
        Max=<10,(((Y is Pos-4*Max,Y=2,!);
        (Y is Pos-4*Max,Y=4,!));(MaxN is Max+1,ver_espaco(Pos,MaxN))).


ver_letra(Pos,Max):-
        Max=<10,((Y is Pos-4*Max,Y=3,!);(MaxN is Max+1,ver_letra(Pos,MaxN))).

ver_espacoPeca(Pos,Max):-
        Max=<10,(((Y is Pos-4*Max,Y=4,!));
        (MaxN is Max+1,ver_espacoPeca(Pos,MaxN))).

ver_Peca(Y,X):-
        Z is (Y/2)-1,
        ASCII is 63+(X+2)/4,
        posicao_actual(Peca,Letra,Z,_),
        char_code(Letra,ASCII_Letra),
```

```
        ASCII_Letra=ASCII,write(Peca).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Movimento das PEÇAS
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mover_PecaNivel(Xantes,Yantes,Xdepois,Ydepois,MaxNivel,Accao):-
    (not(Ydepois=Yantes);
    not(Xdepois=Xantes)),
    ASCIId is 64+Xdepois,
    char_code(LetraD,ASCIId),
    ASCIIa is 64+Xantes,
    char_code(LetraA,ASCIIa),
    posicao_actual(Peca,LetraA,Yantes,MaxNivel),
    ((Accao=0,
    retract(posicao_actual(Z,LetraA,Yantes,MaxNivel)),
    assert(posicao_actual(Z,LetraD,Ydepois,MaxNivel)));
    (Accao=1,
    retract(posicao_actual(PecaMorta,LetraD,Ydepois,MaxNivel)),upss,
    assert(cemiterio(PecaMorta,MaxNivel)),
    retract(posicao_actual(Peca,LetraA,Yantes,MaxNivel)),
    assert(posicao_actual(Peca,LetraD,Ydepois,MaxNivel)))).


mover_Peca(Xantes,Yantes,Xdepois,Ydepois):-
    (not(Ydepois=Yantes);
    not(Xdepois=Xantes)),
    ASCIId is 64+Xdepois,
    char_code(LetraD,ASCIId),
    ASCIIa is 64+Xantes,
```

```
    char_code(LetraA,ASCIIa),
    posicao_actual(Peca,LetraA,Yantes,0),
    name(Peca,PecaArray),
    regra(PecaArray,[Xantes,Yantes],[Xdepois,Ydepois],Accao,0),
    ((Accao=0,
    retract(posicao_actual(Z,LetraA,Yantes,0)),
    assert(posicao_actual(Z,LetraD,Ydepois,0)));
    (Accao=1,
    retract(posicao_actual(PecaMorta,LetraD,Ydepois,0)),
    assert(cemiterio(PecaMorta,0)),
    retract(posicao_actual(Peca,LetraA,Yantes,0)),
    assert(posicao_actual(Peca,LetraD,Ydepois,0)))).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% REGRAS DAS PEÇAS
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Peao Branco
regra([80,87],[Xantes,Yantes],[Xdepois,Ydepois],Accao,MaxNivel):-
    Xantes=Xdepois,
    ((Z is Yantes-1,
    Ydepois=Z,
    ASCII is 64+Xdepois,
    char_code(Letra,ASCII),
    not(posicao_actual(_,Letra,Ydepois,MaxNivel)));
    (Ydepois=5,Yantes=7,
    ASCII is 64+Xdepois,
    char_code(Letra,ASCII),
    not(posicao_actual(_,Letra,Ydepois,MaxNivel)),
```

```
    W is Ydepois+1,!, not(posicao_actual(_,Letra,W,MaxNivel))),
    Accao is 0.

regra([80,87],[Xantes,Yantes],[Xdepois,Ydepois],Accao,MaxNivel):-
    ((Z is Xantes+1,Z=Xdepois);(Z is Xantes-1,Z=Xdepois)),
    W is Yantes-1,
    W=Ydepois,
    ASCII is 64+Xdepois,
    char_code(Letra,ASCII),
    posicao_actual(Peca,Letra,Ydepois,MaxNivel),
    name(Peca,[_,66]),
    Accao is 1.


%Peao Preto
regra([80,66],[Xantes,Yantes],[Xdepois,Ydepois],Accao,MaxNivel):-
    Xantes=Xdepois,
    ((Z is Yantes+1,
    Ydepois=Z,
    ASCII is 64+Xdepois,
    char_code(Letra,ASCII),
    not(posicao_actual(_,Letra,Ydepois,MaxNivel)));
    (Ydepois=4,Yantes=2,
    ASCII is 64+Xdepois,
    char_code(Letra,ASCII),
    not(posicao_actual(_,Letra,Ydepois,MaxNivel)),
    W is Ydepois-1,!,not(posicao_actual(_,Letra,W,MaxNivel)))),
    Accao is 0.

regra([80,66],[Xantes,Yantes],[Xdepois,Ydepois],Accao,MaxNivel):-
    ((Z is Xantes+1,Z=Xdepois);(Z is Xantes-1,Z=Xdepois)),
    W is Yantes+1,
```

```
    W=Ydepois,
    ASCII is 64+Xdepois,
    char_code(Letra,ASCII),
    posicao_actual(Peca,Letra,Ydepois,MaxNivel),
    name(Peca,[_,87]),
    Accao is 1.

%Torre de cor "Cor"
regra([84,Cor],[Xantes,Yantes],[Xdepois,Ydepois],Accao,MaxNivel):-
    ((var(Xdepois),
    (!,retornar_num(Z),(
    (Xdepois is Xantes+Z,Ydepois is Yantes);
    (((Z=<Xantes,Xdepois is Xantes-Z);
    (Z>Xantes,Xdepois is Z-Xantes)),Ydepois is Yantes);
    (Ydepois is Yantes+Z,Xdepois is Xantes);
    (((Z=<Yantes,Ydepois is Yantes-Z);
    (Z>Yantes,Ydepois is Z-Yantes)),Xdepois is Xantes)
    )),Xdepois>0,Xdepois=<8,Ydepois>0,Ydepois=<8
    );
    (not(var(Xdepois)),
    ((Xantes=Xdepois);(Yantes=Ydepois)))),
    ASCII is 64+Xdepois,
    char_code(Letra,ASCII),
    ((not(posicao_actual(_,Letra,Ydepois,MaxNivel)),
    ver_Peca_Intermedia([Xantes,Yantes],[Xdepois,Ydepois],MaxNivel),
    Accao is 0);
    (posicao_actual(Peca,Letra,Ydepois,MaxNivel),
    not(name(Peca,[_,Cor])),
    ver_Peca_Intermedia([Xantes,Yantes],[Xdepois,Ydepois],MaxNivel),
    Accao is 1)).

%Bispo de cor "Cor"
```

```
regra([66,Cor],[Xantes,Yantes],[Xdepois,Ydepois],Accao,MaxNivel):-
    ((var(Xdepois),
    (!,retornar_num(Z),(
    (Xdepois is Xantes+Z,(Ydepois is Yantes+Z;Ydepois is Yantes-Z));
    (Xdepois is Xantes-Z,(Ydepois is Yantes+Z;Ydepois is Yantes-Z))
    )),Xdepois>0,Xdepois=<8,Ydepois>0,Ydepois=<8
    );
    (not(var(Xdepois)),
    (Dif1 is abs(Xantes-Xdepois),
    Dif2 is abs(Yantes-Ydepois),
    Dif1 = Dif2))),
    ASCII is 64+Xdepois,
    char_code(Letra,ASCII),
    ((not(posicao_actual(_,Letra,Ydepois,MaxNivel)),
    ver_Peca_Intermedia([Xantes,Yantes],[Xdepois,Ydepois],MaxNivel),
    Accao is 0);
    (posicao_actual(Peca,Letra,Ydepois,MaxNivel),
    not(name(Peca,[_,Cor])),
    ver_Peca_Intermedia([Xantes,Yantes],[Xdepois,Ydepois],MaxNivel),
    Accao is 1)).


%Rei de cor "cor"
regra([75,Cor],[Xantes,Yantes],[Xdepois,Ydepois],Accao,MaxNivel):-
    ((var(Xdepois),
    (
    (Xdepois is Xantes+1,(Ydepois is Yantes+1;Ydepois is Yantes-1));
    (Xdepois is Xantes-1,(Ydepois is Yantes+1;Ydepois is Yantes-1));
    (Xdepois is Xantes+1,Ydepois is Yantes);
    (Xdepois is Xantes-1,Ydepois is Yantes);
    (Ydepois is Yantes+1,Xdepois is Xantes);
    (Ydepois is Yantes-1,Xdepois is Xantes)
```

```
    ),Xdepois>0,Xdepois=<8,Ydepois>0,Ydepois=<8
    );
    (not(var(Xdepois)),
    ((Dif1 is abs(Xantes-Xdepois),
    Dif2 is abs(Yantes-Ydepois),
    Dif1 = Dif2,
    Dif1 = 1);
    (Xantes=Xdepois,
        ((Z is Yantes+1,Ydepois=Z);
        (Z is Yantes-1,Ydepois=Z)));
    (Yantes=Ydepois,
        ((Z is Xantes+1,Xdepois=Z);
        (Z is Xantes-1,Xdepois=Z))))),
    ASCII is 64+Xdepois,
    char_code(Letra,ASCII),
    ((not(posicao_actual(_,Letra,Ydepois,MaxNivel)),
    Accao is 0);
    (posicao_actual(Peca,Letra,Ydepois,MaxNivel),
    not(name(Peca,[_,Cor])),
    Accao is 1)).


%Rainha de cor "cor"
regra([81,Cor],[Xantes,Yantes],[Xdepois,Ydepois],Accao,MaxNivel):-
    ((var(Xdepois),
    (!,retornar_num(Z),(
    (Xdepois is Xantes+Z,(Ydepois is Yantes+Z;Ydepois is Yantes-Z));
    (Xdepois is Xantes-Z,(Ydepois is Yantes+Z;Ydepois is Yantes-Z));
    (Xdepois is Xantes+Z,Ydepois is Yantes);
    (((Z=<Xantes,Xdepois is Xantes-Z);
    (Z>Xantes,Xdepois is Z-Xantes)),Ydepois is Yantes);
    (Ydepois is Yantes+Z,Xdepois is Xantes);
```

```
    (((Z=<Yantes,Ydepois is Yantes-Z);
    (Z>Yantes,Ydepois is Z-Yantes)),Xdepois is Xantes)
    )),Xdepois>0,Xdepois=<8,Ydepois>0,Ydepois=<8
    );
    (not(var(Xdepois)),
    ((Dif1 is abs(Xantes-Xdepois),
    Dif2 is abs(Yantes-Ydepois),
    Dif1 = Dif2);
    (Xantes=Xdepois);
    (Yantes=Ydepois)))),
    ASCII is 64+Xdepois,
    char_code(Letra,ASCII),
    ((not(posicao_actual(_,Letra,Ydepois,MaxNivel)),
    ver_Peca_Intermedia([Xantes,Yantes],[Xdepois,Ydepois],MaxNivel),
    Accao is 0);
    (posicao_actual(Peca,Letra,Ydepois,MaxNivel),
    not(name(Peca,[_,Cor])),
    ver_Peca_Intermedia([Xantes,Yantes],[Xdepois,Ydepois],MaxNivel),
    Accao is 1)).


%Cavalo de cor "cor"
regra([72,Cor],[Xantes,Yantes],[Xdepois,Ydepois],Accao,MaxNivel):-
    ((var(Xdepois),
    (
    (Z is Xantes+1,Xdepois=Z,(W is Yantes+2;W is Yantes-2),Ydepois=W);
    (Z is Xantes-1,Xdepois=Z,(W is Yantes+2;W is Yantes-2),Ydepois=W);
    (Z is Xantes+2,Xdepois=Z,(W is Yantes+1;W is Yantes-1),Ydepois=W);
    (Z is Xantes-2,Xdepois=Z,(W is Yantes+1;W is Yantes-1),Ydepois=W)
    ),Xdepois>0,Xdepois=<8,Ydepois>0,Ydepois=<8
    );
    (not(var(Xdepois)),Dif1 is abs(Xantes-Xdepois),
```

```
    Dif2 is abs(Yantes-Ydepois),
    ((Dif1 = 1,Dif2 = 2);
    (Dif1 = 2,Dif2 = 1))
    )),
    ASCII is 64+Xdepois,
    char_code(Letra,ASCII),
    ((not(posicao_actual(_,Letra,Ydepois,MaxNivel)),
    Accao is 0);
    (posicao_actual(Peca,Letra,Ydepois,MaxNivel),
    not(name(Peca,[_,Cor])),
    Accao is 1)).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Verificação da existência ou não existência de peças intermédias
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Movimento vertical
ver_Peca_Intermedia([Xantes,Yantes],[Xantes,Ydepois],MaxNivel):-
    not(Yantes=Ydepois),
    ASCII is 64+Xantes,
    char_code(Letra,ASCII),
    ((Ydepois>Yantes,Z is Yantes+1);(Ydepois<Yantes,Z is Yantes-1)),
    (Z=Ydepois;not(posicao_actual(_,Letra,Z,MaxNivel))),
    ver_Peca_Intermedia([Xantes,Z],[Xantes,Ydepois],MaxNivel).

%Movimento horizontal
ver_Peca_Intermedia([Xantes,Yantes],[Xdepois,Yantes],MaxNivel):-
    not(Xantes=Xdepois),
    ((Xdepois>Xantes,Z is Xantes+1);(Xdepois<Xantes,Z is Xantes-1)),
```

```
    ASCII is 64+Z,
    char_code(Letra,ASCII),
    (Z=Xdepois;not(posicao_actual(_,Letra,Yantes,MaxNivel))),
    ver_Peca_Intermedia([Z,Yantes],[Xdepois,Yantes],MaxNivel).

%Movimento Diagonal
ver_Peca_Intermedia([Xantes,Yantes],[Xdepois,Ydepois],MaxNivel):-
    Dif1 is abs(Xantes-Xdepois),
    Dif2 is abs(Yantes-Ydepois),
    Dif1 = Dif2,
    ((Xdepois>Xantes,Z is Xantes+1);(Xdepois<Xantes,Z is Xantes-1)),
    ((Ydepois>Yantes,W is Yantes+1);(Ydepois<Yantes,W is Yantes-1)),
    ASCII is 64+Z,
    char_code(Letra,ASCII),
    ((W=Ydepois,Z=Xdepois);not(posicao_actual(_,Letra,W,MaxNivel))),
    ver_Peca_Intermedia([Z,W],[Xdepois,Ydepois],MaxNivel).

%Movimento final
ver_Peca_Intermedia([Xdepois,Ydepois],[Xdepois,Ydepois],_).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Funções Geradoras de movimentos
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


funcao_geradora(Peca,X,Y,L,R,Nivel):-
    regra(Peca,[X,Y],[Xdepois,Ydepois],Accao,Nivel),
    Elemento=[Xdepois,Ydepois,Accao,Nivel],
    not(member(Elemento,L)),
    funcao_geradora(Peca,X,Y,[Elemento|L],R,Nivel).
```

```
funcao_geradora(_,_,_,R,R,_):-!.


funcao_geradoraM(Peca,X,Y,L,R,Nivel):-
    regra(Peca,[X,Y],[Xdepois,Ydepois],Accao,Nivel),
    Elemento=[X,Y,Xdepois,Ydepois,Accao],
    not(member(Elemento,L)),
    funcao_geradoraM(Peca,X,Y,[Elemento|L],R,Nivel).

funcao_geradoraM(_,_,_,R,R,_):-!.


funcao_geradoraAB(Peca,X,Y,L,R,Nivel,Cont):-
    regra(Peca,[X,Y],[Xdepois,Ydepois],Accao,Nivel),
    NivelN is Nivel+1,
    ElementoT=[X,Y,Xdepois,Ydepois,Accao,NivelN,_],
    not(member(ElementoT,L)),
    ContN is Cont+1,
    Elemento=[X,Y,Xdepois,Ydepois,Accao,NivelN,ContN],
    funcao_geradoraAB(Peca,X,Y,[Elemento|L],R,Nivel,ContN).

funcao_geradoraAB(_,_,_,R,R,_,_):-!.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Funções Auxiliares
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

retornar_num(X):-
```

```
    (X=1;
    X=2;
    X=3;
    X=4;
    X=5;
    X=6;
    X=7).

max_Tab_Nivel(MaxNivel):-
    constroi_Nivel(Lista,[0]),
    max(Lista,0,MaxNivel).


constroi_Nivel(Lista,L):-
    posicao_actual(_,_,_,Nivel),
    not(member(Nivel,L)),
    constroi_Nivel(Lista,[Nivel|L]).

constroi_Nivel(L,L).

max([],Max,Max).

max([X|R],Inic,Max):-
    (Inic<X,
    InicN is X,
    max(R,InicN,Max));
    max(R,Inic,Max).

soma([],Inic,Inic).

soma([X|R],Valor,Inic):-
    InicN is Inic+X,
```

```
    soma(R,Valor,InicN).

maxValor([],_,_,Z,W,Y,T,Z,W,Y,T,A,A):-!.

maxValor([[Xantes,Yantes,Xdepois,Ydepois,Accao,V,ValorAdv]|R],Inic,Max,Z,W,Y,T,Xa,Ya,Xd,Yd,A
    (Inic<(V-ValorAdv),
    InicN is (V-ValorAdv),
    maxValor(R,InicN,Max,Xantes,Yantes,Xdepois,Ydepois,Xa,Ya,Xd,Yd,Accao,AccaoN));
    maxValor(R,Inic,Max,Z,W,Y,T,Xa,Ya,Xd,Yd,AccaoInic,AccaoN).

nivela([],L,L).

nivela([[]|R],L,RF):-
    nivela(R,L,RF),!.

nivela([[X|R]],_,[X|R]):-!.

nivela([[X|R]|RL],LI,L):-
    nivela([[X|R]],[],LX),
    nivela(RL,[],LR),
    concatena(LX,LR,L2),
    concatena(L2,LI,L),!.

concatena([],L,L).

concatena([X|R],L,[X|C]):-
    concatena(R,L,C).

escreve([]).
escreve([X|L]):-
    writeln(X),
    escreve(L).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Função Avaliadora do tabuleiro
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


fim_Jogo(Cor,MaxNivel,Valor):-
    (name(Peca,[75,Cor]),
    cemiterio(Peca,MaxNivel),Valor is -1);
    (((Cor=66,CorN is 87);(Cor=87,CorN is 66)),
    name(Peca,[75,CorN]),cemiterio(Peca,MaxNivel),Valor is 2000);
    (posicao_actual('KW',_,_,MaxNivel),posicao_actual('KB',_,_,MaxNivel),
    (posicao_actual(Peca,_,_,MaxNivel),(Peca\='KB',Peca\='KW',!,fail));Valor is 0).

fim_JogoAB(Cor,MaxNivel,Valor):-
    (name(Peca,[75,Cor]),
    cemiterio(Peca,MaxNivel),Valor is -1);
    ((Cor=66,CorN is 87,name(Peca,[75,CorN]),cemiterio(Peca,MaxNivel),Valor is 400);
    (Cor=87,CorN is 66,name(Peca,[75,CorN]),cemiterio(Peca,MaxNivel),Valor is 2));
    %   (Cor=87,CorN is 66,name(Peca,[75,CorN]),cemiterio(Peca,MaxNivel),Valor is 2));
    (posicao_actual('KW',_,_,MaxNivel),posicao_actual('KB',_,_,MaxNivel),
    (posicao_actual(Peca,_,_,MaxNivel),(Peca\='KB',Peca\='KW',!,fail));Valor is 0).

valor_tabuleiro(Cor,Valor,MaxNivel):-
    ((fim_JogoAB(Cor,MaxNivel,Valor));
    (listagem_Pecas(Cor,[],ListaValores,[],MaxNivel),
    soma(ListaValores,Valor,0)
    )).
```

```
listagem_Pecas(Cor,L,R,LCoord,MaxNivel):-
    posicao_actual(Peca,XLetra,Y,MaxNivel),
    name(Peca,[_,Cor]),
    name(XLetra,[Num]),
    X is Num-64,
    Elemento=[X,Y],
    not(member(Elemento,LCoord)),
    pontos(Peca,Valor,X,Y,MaxNivel),
    listagem_Pecas(Cor,[Valor|L],R,[Elemento|LCoord],MaxNivel).

listagem_Pecas(_,Lista,Lista,_,_):-!.

pontos(Peca,Valor,X,Y,MaxNivel):-
    ASCII is 64+X,
    char_code(Letra,ASCII),
    posicao_actual(Peca,Letra,Y,MaxNivel),
    bonus_Defendido_Por(Peca,ValorN,X,Y,MaxNivel),
    bonus_Defendido_Por_Rainha(Peca,ValorN1,X,Y,MaxNivel),
    bonus_Por_Dist_Pos_Inicial(Peca,ValorN2,_,Y),
    bonus_Por_Dist_Centro(Peca,ValorN3,X,Y),
    bonus_Por_Dist_Rei(Peca,ValorN4,X,Y,MaxNivel),
    name(Peca,[P,_]),
    valor(P,ValorN5),!,
    Valor is ValorN+ValorN1+ValorN2+ValorN3+ValorN4+ValorN5.

bonus_Defendido_Por(Peca,Valor,X,Y,MaxNivel):-
    defendido_MPeca(Peca,X,Y,MaxNivel),!,
    tabela_Pontos_Defendido_MPeca(Peca,Valor).

bonus_Defendido_Por(_,0,_,_,_):-!.

bonus_Defendido_Por_Rainha(Peca,Valor,X,Y,MaxNivel):-
```

```
    peca_defendida_Rainha(Peca,X,Y,MaxNivel),
    Valor is 25,!.

bonus_Defendido_Por_Rainha(_,0,_,_,_):-!.

bonus_Por_Dist_Pos_Inicial(Peca,Valor,_,Y):-
    name(Peca,PecaArray),
    PecaArray=[80,Cor],!,
    ((Cor=87,Valor is 10*(7-Y));
    (Cor=66,Valor is 10*(Y-2))).

bonus_Por_Dist_Pos_Inicial(_,0,_,_):-!.

bonus_Por_Dist_Centro(Peca,Valor,X,Y):-
    name(Peca,PecaArray),
    ((PecaArray=[72,_],((Y<5,DistY is Y-1);
    (Y>=5,DistY is 8-Y)),((X<5,DistX is X-1);(X>=5,DistX is 8-X)));
    (PecaArray=[80,Cor],((Cor=87,Y>=5,DistY is 8-Y);(Cor=66,Y<5,DistY is Y-1)),
        ((X<5,DistX is X-1);(X>=5,DistX is 8-X)))),
    Valor is (DistY*5+DistX*5),!.

bonus_Por_Dist_Centro(_,0,_,_):-!.

bonus_Por_Dist_Rei(Peca,Valor,X,Y,MaxNivel):-
    name(Peca,PecaArray),
    PecaArray=[P,C],
    (P=72;P=84;P=81),
    name(Rei,[75,C]),
    posicao_actual(Rei,XReiL,YRei,MaxNivel),
    name(XReiL,[ASCII]),
    XRei is ASCII-64,
    ((Y>=YRei,DistY is Y-YRei);(Y<YRei,DistY is YRei-Y)),
```

```
    ((X>=XRei,DistX is X-XRei);(X<XRei,DistX is XRei-X)),
    (((P=72;P=84),Valor is 70-(DistY*2+DistX*2));
    (Valor is 140-(DistY*3+DistX*3))),!.

bonus_Por_Dist_Rei(_,0,_,_,_):-!.

tabela_Pontos_Defendido_MPeca(Peca,Valor):-
    (name(Peca,PecaArray),
    ((PecaArray=[80,_],Valor is 10);
    (PecaArray=[66,_],Valor is 25);
    (PecaArray=[84,_],Valor is 25)
    ));Valor is 0.

defendido_MPeca(Peca,X,Y,MaxNivel):-
    name(Peca,PecaArray),
    ((PecaArray=[Z,66],PecaNC is 87);(PecaArray=[Z,87],PecaNC is 66)),
    PecaN = [Z,PecaNC],
    funcao_geradora(PecaN,X,Y,[],R,MaxNivel),!,
    not(ver_PecaDiferente(Peca,R)).

ver_PecaDiferente(Peca,[[X,Y,Accao,Nivel]|R]):-
    (Accao=1,
    ASCII is 64+X,
    char_code(Letra,ASCII),
    posicao_actual(PecaC,Letra,Y,Nivel),
    Peca=PecaC,!,fail
    );
    ver_PecaDiferente(Peca,R).


ver_PecaDiferente(_,[]):-!.
```

```
peca_defendida_Rainha(Peca,X,Y,MaxNivel):-
    name(Peca,PecaArray),
    (PecaArray=[66,_];PecaArray=[84,_]),
    ((PecaArray=[Z,66],PecaNC is 87);(PecaArray=[Z,87],PecaNC is 66)),
    PecaN = [Z,PecaNC],
    funcao_geradora(PecaN,X,Y,[],R,MaxNivel),!,
    not(ver_PecaRainha(Peca,R)).

ver_PecaRainha(Peca,[[X,Y,Accao,Nivel]|R]):-
    (Accao=1,
    ASCII is 64+X,
    char_code(Letra,ASCII),
    posicao_actual(PecaC,Letra,Y,Nivel),
    name(Peca,[_,PC]),
    name(PecaC,[81,PC])
    ,!,fail
    );
    ver_PecaRainha(Peca,R).


ver_PecaRainha(_,[]).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Função Move Peca para o computador
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

movimento_maximo_valor(Cor,MaxNivel,Xa,Ya,Xd,Yd,Accao):-
    MaxNivelN is MaxNivel-1,
```

```
    listagem([],R,MaxNivelN,Cor,[]),
    listagem_Pecas_Todas(Cor,R,[],ListaValores,MaxNivelN),
    nivela(ListaValores,[],L),
    avaliar(L,Xa,Ya,Xd,Yd,Cor,MaxNivel,Accao).



avaliar(ListaValores,Xa,Ya,Xd,Yd,Cor,MaxNivel,Accao):-
    avaliarLista(ListaValores,Cor,[],L,MaxNivel),!,
    maxValor(L,0,_,0,0,0,0,Xa,Ya,Xd,Yd,0,Accao).


avaliarLista([[Xantes,Yantes,Xdepois,Ydepois,Accao]|R],Cor,K,L,MaxNivel):-
    criar_tab_aux(MaxNivel),
    mover_PecaNivel(Xantes,Yantes,Xdepois,Ydepois,MaxNivel,Accao),
    valor_tabuleiro(Cor,Valor,MaxNivel),
    ((Cor=66,valor_tabuleiro(87,ValorAdv,MaxNivel));
    (valor_tabuleiro(66,ValorAdv,MaxNivel))),
    apagar_tab_aux(MaxNivel),
    avaliarLista(R,Cor,[[Xantes,Yantes,Xdepois,Ydepois,Accao,Valor,ValorAdv]|K],L,MaxNivel)

avaliarLista([],_,L,L,_).


criar_tab_aux(MaxNivel):-
    NivelAnt is MaxNivel-1,
    (posicao_actual(P,X,Y,NivelAnt),not(posicao_actual(P,X,Y,MaxNivel)),
    assert(posicao_actual(P,X,Y,MaxNivel)),criar_tab_aux(MaxNivel));!.

apagar_tab_aux(MaxNivel):-
    (posicao_actual(_,_,_,MaxNivel),retractall(posicao_actual(_,_,_,MaxNivel)),
    retractall(cemiterio(_,MaxNivel)));!.
```

```
listagem_Pecas_Todas(Cor,[[Peca,X,Y]|L],R,R1,MaxNivel):-
    name(Peca,PecaArray),
    funcao_geradoraM(PecaArray,X,Y,[],M,MaxNivel),
    listagem_Pecas_Todas(Cor,L,[M|R],R1,MaxNivel).

listagem_Pecas_Todas(_,[],R,R,_):-!.

listagem(L,R,MaxNivel,Cor,LCoord):-
    posicao_actual(Peca,XLetra,Y,MaxNivel),
    name(Peca,[_,Cor]),
    name(XLetra,[Num]),
    X is Num-64,
    Elemento=[X,Y],
    not(member(Elemento,LCoord)),
    listagem([[Peca,X,Y]|L],R,MaxNivel,Cor,[Elemento|LCoord]).

listagem(R,R,_,_,_).


listagem_Pecas_TodasAB(Cor,[[Peca,X,Y]|L],R,R1,MaxNivel):-
    name(Peca,PecaArray),
    funcao_geradoraAB(PecaArray,X,Y,[],M,MaxNivel,0),
    listagem_Pecas_TodasAB(Cor,L,[M|R],R1,MaxNivel).

listagem_Pecas_TodasAB(_,[],R,R,_):-!.

listagemAB(L,R,MaxNivel,Cor,LCoord):-
    posicao_actual(Peca,XLetra,Y,MaxNivel),
    name(Peca,[_,Cor]),
```

```
    name(XLetra,[Num]),
    X is Num-64,
    Elemento=[X,Y],
    not(member(Elemento,LCoord)),
    listagemAB([[Peca,X,Y]|L],R,MaxNivel,Cor,[Elemento|LCoord]).

listagemAB(R,R,_,_,_).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Algoritmo Min-Max , com cortes alpha-beta
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

alphabetaT:-
    alphabeta([], -100000, 100000, GoodPos, Val),
    writeln(GoodPos),
    writeln(Val),
    apagar_tab_aux(1),
    apagar_tab_aux(2),
    apagar_tab_aux(3).

alphabeta(Pos, Alpha, Beta, GoodPos, Val):-
    moves(Pos,PostList),!,
    write(Pos),writeln(PostList),
    boundedbest(PostList, Alpha, Beta, GoodPos, Val);
    funcao_avaliadora(Pos, Val).


boundedbest([Pos|PostList], Alpha, Beta, GoodPos, GoodVal):-
    alphabeta(Pos, Alpha, Beta, _, Val),
    goodenough(PostList, Alpha, Beta, Pos, Val, GoodPos, GoodVal).
```

```
goodenough([], _, _, Pos, Val, Pos, Val).

goodenough(_, Alpha, Beta, [X,Y,Z,T,W,Pos,_], Val, [X,Y,Z,T,W,Pos,_], Val):-
    (max_to_move(Pos), Val>Beta, !);
    (min_to_move(Pos), Val<Alpha, !).

goodenough(PostList, Alpha, Beta, Pos, Val, GoodPos, GoodVal):-
    newbounds(Alpha, Beta, Pos, Val, NewAlpha, NewBeta),
    boundedbest(PostList, NewAlpha, NewBeta, Pos1, Val1),
    betterof(Pos, Val, Pos1,Val1, GoodPos, GoodVal).

newbounds(Alpha, Beta, [_,_,_,_,_,Pos,_], Val, Val, Beta):-
    max_to_move(Pos), Val>Alpha, !.

    %write(Val),write(';'),write(Alpha),write(';'),writeln(Beta).

    %write(Val),write(';'),write(Alpha),write(';'),writeln(Beta).

newbounds(Alpha, Beta, [_,_,_,_,_,Pos,_], Val, Alpha, Val):-
    min_to_move(Pos), Val<Beta, !.

    %write(Val),write(';'),write(Alpha),write(';'),writeln(Beta).

newbounds(Alpha, Beta, _, _, Alpha, Beta).

%:-write(Alpha),write(';'),writeln(Beta).

betterof([X,Y,Z,T,W,Pos,_], Val,_, Val1, [X,Y,Z,T,W,Pos,_], Val):-
    max_to_move(Pos), Val>Val1, !;
    min_to_move(Pos), Val<Val1, !.
```

```
betterof(_, _, Pos1,Val1, Pos1, Val1).

    %:-writeln(Val1).


min_to_move(Pos):-
    Y is Pos mod 2,
    Y=0.
max_to_move(Pos):-
    Y is Pos mod 2,
    Y=1.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Função avaliadora
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

funcao_avaliadora([Xa,Ya,Xd,Yd,Accao,Nivel,_], Val):-
    apagar_tab_aux(Nivel),
    criar_tab_aux(Nivel),
    mover_PecaNivel(Xa,Ya,Xd,Yd,Nivel,Accao),
    ((min_to_move(Nivel),Cor=87,CorA=66);(max_to_move(Nivel),Cor=66,CorA=87)),
    valor_tabuleiro(Cor,ValN,Nivel),
    valor_tabuleiro(CorA,ValA,Nivel),
    ((Nivel>1,Val is (-1)*(ValN-ValA));
    (Val is (ValN-ValA))),
    apagar_tab_aux(Nivel).

moves([Xa,Ya,Xd,Yd,Accao,Nivel,_],L):-
     nivel_prof(Prof),Nivel<Prof,jogadaN(Num),Num>5,
    ((min_to_move(Nivel),Cor=87);(max_to_move(Nivel),Cor=66)),
```

```
    max_Tab_Nivel(MaxNivel),
    (MaxNivel<Nivel;apagar_tab_aux(MaxNivel)),
    criar_tab_aux(Nivel),
    mover_PecaNivel(Xa,Ya,Xd,Yd,Nivel,Accao),
    listagemAB([],R,Nivel,Cor,[]),
    listagem_Pecas_TodasAB(Cor,R,[],ListaValores,Nivel),
    nivela(ListaValores,[],L).


moves([],PosLista):-
    criar_tab_aux(1),
    Cor=66,
    listagemAB([],R,0,Cor,[]),
    listagem_Pecas_TodasAB(Cor,R,[],ListaValores,0),
    nivela(ListaValores,[],PosLista),
    apagar_tab_aux(1).
```