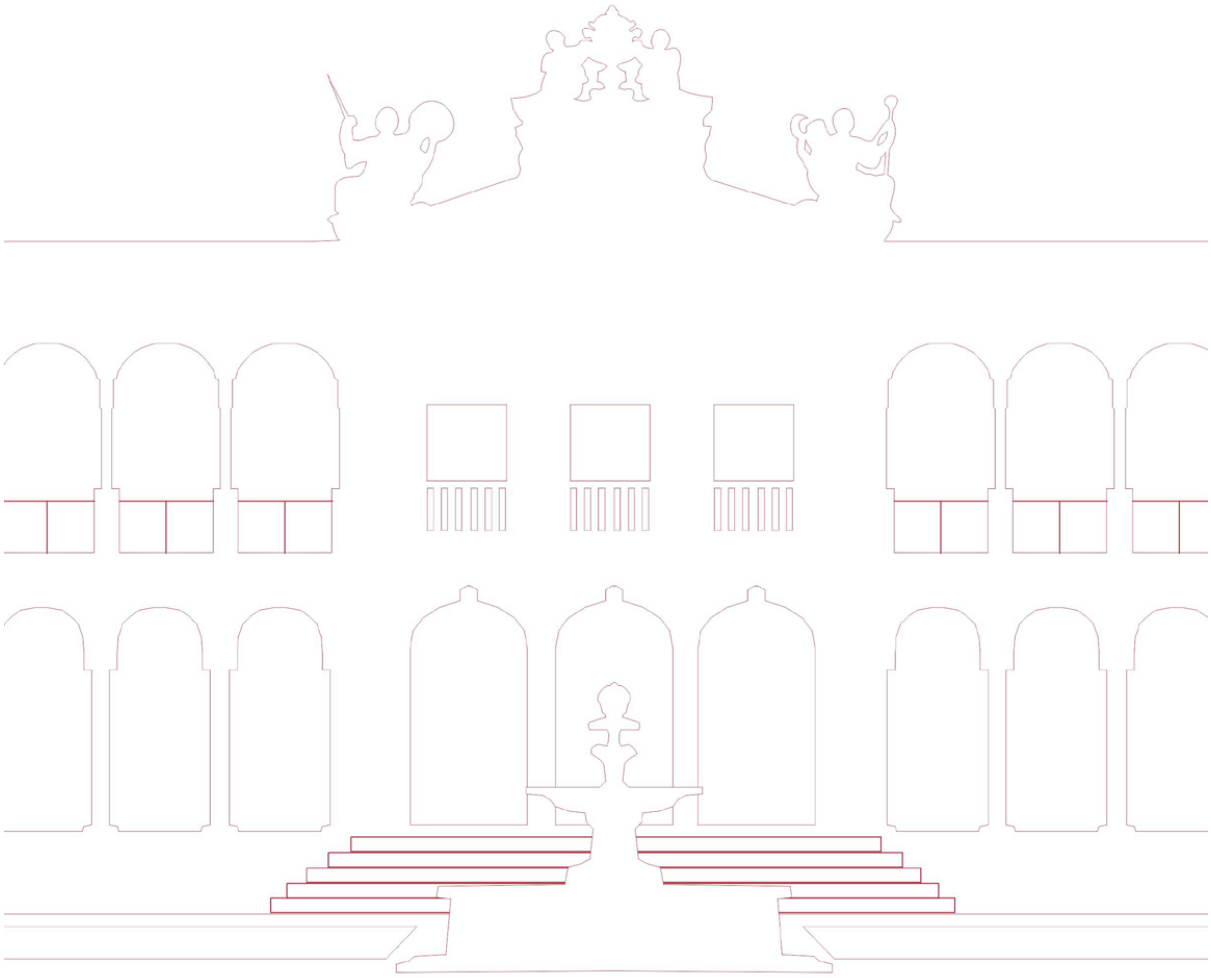
# Relatório trabalho prático
# Conversor Notação de Xadrez

UNIVERSIDADE DE ÉVORA
ESCOLA DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

Licenciatura em Eng. Informática
Programação III



**Ricardo Oliveira 42647**

**Docente: Salvador Abreu**

*Évora, 16 de janeiro de 2023*

# Conteúdo

**1 Introdução** **1**

**2 Desenvolvimento e Implementação** **2**
2.1 Decisões tomadas . . . . . 2
2.2 Organização do código . . . . . 2
2.3 Compilação e Execução . . . . . 3

**3 Pontos Fortes e Fraquezas** **3**

**4 Conclusão** **3**

# 1 Introdução

O trabalho prático da Unidade Curricular de Programação III consiste na implementação de um programa na linguagem GNU Prolog que aceita jogadas de um jogo de xadrez. O programa deve ler o *input* do utilizador de modo a receber três diferentes argumentos, dos quais o primeiro deverá ser o *Format* da notação de xadrez (algébrica, descritiva ou postal), de seguida outro argumento que sera a *action* que será a ação propriamente dita a ser executada (algébrica, descritiva, postal, mostrar ou estado) e, por último, o ficheiro de texto que terá os movimentos de um jogo de xadrez. O objetivo deste trabalho passa por aplicar todos os conhecimentos adquiridos no decorrer do semestre de uma forma mais prática.

# 2 Desenvolvimento e Implementação

## 2.1 Decisões tomadas

Inicialmente, de modo a ler o *input* do *standard input* do utilizador, para verificar o tipo de formato, ação e ficheiro de jogo, tomei a escolha de, pela `argument_list(As)`, verificar a *Head* dessa lista e compará-lo com as possíveis opções (descritas e explicadas na descrição), tomando um diferente rumo consoante o seu formato; de seguida, a *Tail* da lista de argumentos é enviada para um novo predicado que verifica a qual ação a *Head* dessa lista corresponde, realizando também diferentes ações consoante a escolha do utilizador; e, finalmente, o último elemento da lista de argumentos, deverá ser o nome do ficheiro de texto correspondente ao jogo que, por sua vez, baseado no troço de código disponibilizado pelo Professor, verifica se existe um ficheiro com esse nome e abre-o para iniciar a leitura, utilizando os predicados predefinidos de Prolog: `file_exists(J)` e `see(J)`, respetivamente.
De seguida, de modo a ler cada jogada do ficheiro, baseado novamente no código disponibilizado pelo Professor, através do predicado `get0(?in_character_code)` foi possível ler os caracteres um a um em código ASCII.
Para determinar cada jogada, tomei a decisão de, quando encontrar um espaço (jogada pelas brancas) ou um enter (jogada pelas pretas), esse movimento chegava ao fim, tendo assim uma lista com os códigos de ASCII correspondentes a cada jogada, até ao *end-of-file (eof)*. Como as jogadas variam o seu número de caracteres, criei um novo predicado que verifica o número de caracteres de cada jogada e chama diferentes predicados para jogadas de dois caracteres (peões apenas), de três caracteres (movimentos de todas as outras peças, p.e.), de quatro ou mais caracteres (movimentos de peças quando duas identicas podem se movimentar para a mesma casa, movimento de uma peça com cheque, p.e.).
Entretanto, defini um predicado para o tabuleiro que é definido como uma lista de oito listas, cada posição representando um quadrado do tabuleiro e criadas as peças nas suas posições iniciais. As peças são definidas com uma primeira letra minúscula 'w' ou 'b', referindo se a peça em questão é *black* ou *white*; o caracter seguinte é a letra em maiúscula da peça a que corresponde (com base na notação algébrica):

- R *Rook* para as Torres;
- N *Knight* para os Cavalos;
- B *Bishop* para os Bispos;
- Q *Queen* para a Dama;
- K *King* para o Rei;
- P *Pawn* para os Peões.

Nota: Os quadrados vazios do tabuleiro, são representados por 'es' (*empty square*).

Posteriormente, após ter cada jogada, temos as "coordenadas"para qual a peça será movida (uma vez que as colunas são representadas por letras de a-h, para as converter para os inteiros correspondentes de 1-8, fiz a subtração de 96 ao seu código ASCII), seria necessário, através das regras de cada peça, procurar pela peça que pode realizar esse movimento no tabuleiro. Assim que a peça for encontrada, temos então as suas coordenadas iniciais e finais, portanto defini um predicado que acede ao tabuleiro (lista dinâmica), seleciona a linha toda correspondente, retira a peça da coluna de onde veio (colocando 'es') e coloca-a nas respetivas linha e coluna de destino efetuando assim, uma jogada.

## 2.2 Organização do código

A organização do código, já foi ligeiramente explicada também no ponto anterior. Todos os predicados implementados têm um breve comentário com argumentos que recebe e uma breve explicação do que o código faz.
Inicialmente, através do predicado predefinido `initialization`, inicia outro predicado `start` que irá proceder ao início automático do jogo aquando da execução do programa. De seguida, defini os predicados que realizam a verificação correta do *input* do utilizador para o *format*, *action* e *file-game.txt*, tal como pedido no enunciado. Posteriormente, defini o tabuleiro como uma lista de listas com as respetivas peças nas posições inicias, bem como os quadrados vazios. Subsequentemente, foram definidos os predicados de

leitura de cada tipo jogada (dois caracteres, três caracteres e assim em diante), bem como os predicados que realizam a jogada em si (acender ao tabuleiro, retirar a peça da posição inicial e movê-la para a posição pretendida). Finalmente, de modo a complementar totalmente as jogadas, foram adicionados vários predicados para a verificação das regras da peça em questão, de modo a obter as coordenadas "anteriores"dada a posição pretendida de movimento e validar esse movimento.

### 2.3 Compilação e Execução

Para uma correta compilação e execução do programa, pode ser obtida através do seguinte comando, corridos no terminal:

```
gplc xadrez.pl
./xadrez [FORMAT] [ACTION] [FILE-GAME.txt]
```

Nota: Os parênteses retos são meramente informativos dos tipos de argumentos que o utilizador deve colocar como input, não devem ser utilizados aquando da execução.

## 3 Pontos Fortes e Fraquezas

O programa lê corretamente e verifica os argumentos passados pelo utilizador no momento da execução. Da mesma maneira, as regras para cada peça estão bem definidas e verificam **sempre** se estão dentro do tabuleiro.
Como fraquezas, poderia implementar um predicado para cada peça atribuindo-lhes as coordenadas das posições iniciais das mesmas e, ao realizar a procura delas no tabuleiro, verificar também, através desses predicados dinâmicos, se a peça realmente se encontra naquela posição, evitando assim que ocorram erros de movimentação no decorres das jogadas. Por outro lado, o programa também não verifica corretamente se existe uma peça no caminho pelo qual outra peça se irá mover (à excepção dos cavalos que podem movimentar-se "através"de outras peças).

## 4 Conclusão

Com a realização do trabalho prático descrito, foi possível adquirir de uma forma mais consolidada toda a matéria lecionada nas aulas, tanto práticas como teóricas. Ao longo do trabalho, foram encaradas algumas diversidades, nomeadamente na definição dos predicados que iriam realizar a troca da peça, porque inicialmente, assim que retirava tanto a peça da linha inicial, como o quadrado vazio ('es'), se uma jogada futura fosse realizada numa posição à direita dessa mesma (isto é numa coluna maior), todas as peças dessa linha iriam andar uma casa para a esquerda, porém, após uma análise mais complexa ao predicado `new_row_handler`, este erro foi corrigido.
Se, eventualmente, tivesse mais tempo para dedicar ao trabalho, iria realizar a conversão entre as três notações de xadrez; algo que poderia ser obtido com o reaproveitamento (e possível adaptação) de predicados já definidos nesta fase do trabalho. Bem como, realizar os movimentos corretos para "jogadas especiais", tais como *Castle*, porém seria necessário verificar que nenhuma das peças (tanto o Rei como a Torre em questão) se teriam movido anteriormente e se o Rei não passaria numa posição onde estaria em cheque; *En Passant* na qual um peão poderá capturar outro, caso este se mova duas casas (caso esteja em posição inicial) e, no entanto, passe por uma casa onde este peão poderia capturar "diretamente"; *Promotion* na qual um peão, ao atingir a última linha do tabuleiro, pode promover para uma peça à escolha do jogador (excluindo o peão e o Rei).