Trabalho de Programação III
2022/23

Conversor de notação para jogos de xadrez (v1)

Enunciado

O programa deve aceitar as jogadas dum jogo de xadrez, em sequência, numa das seguintes notações:

1. Algébrica (ver aqui)
2. Descritiva (ver aqui; usar a descrição em Inglês dado que a em Português está off...)
3. Postal (ver aqui)

O programa deverá ler o *standard input*, que consistirá em linhas de texto consecutivas (sem "." outros delimitadores), com as jogadas de xadrez conforme a notação escolhida. O *end-of-file* denota o fim das jogadas.

O programa deverá aceder à linha de comando para determinar o que fazer e obedecer à seguinte especificação:

`xadrez FORMAT ACTION`...

em que **`FORMAT`** é um string que descreve o formato das jogadas no input (**`algebrica`**, **`descritiva`** ou **`postal`**) e **`ACTION`**... é uma sequência não vazia de ações a realizar. Cada ação poderá ser:

`algebrica`	transcrever o jogo em notação algébrica
`descritiva`	transcrever o jogo em notação descritiva(*)
`postal`	transcrever o jogo em notação postal
`mostrar`	imprimir o tabuleiro no formato final
`estado`	indica se algum jogador está em xeque, e qual (**`brancas`** ou **`pretas`**)

O programa deverá funcionar sem nenhuma interação ou mensagem que não a solicitada.
Requisitos:

- Validar o input, i.e. não permitir jogadas ilegais (por serem movimentos impossíveis para a peça). Uma jogada ilegal será ignorada com uma mensagem de erro.
- Ao detetar 3 ou mais jogadas ilegais, o programa deve sair com uma mensagem de erro explicativa.
- Considera-se que o end-of-file no input denota o fim das jogadas.
- As ações "algebrica", "descritiva" e "postal" devem apresentar um registo do jogo, à medida que vai sendo jogado.
- As ações "mostrar" e "estado" aplicam-se ao jogo no final.

Sugestões (para Prolog)

Input/Output

Use os predicados de I/O de texto, `get/1`, `get0/1`, etc. e também o output formatado `format/2` e `format/3`.

Note-se que o end-of-file dá -1 para o predicado get0/1, ie.:

```
| ?- get0(X).
^D
X = -1

yes
| ?-
```

Pode-se usar um predicado gets/1 que lê o stdin até ao fim da linha, e retorna um string (lista de inteiros) com o conteúdo da lista:

```
gets(S) :- get0(C), gets([], C, S).

gets(S, 10, S).       % 10 é o newline
gets(S, -1, S).       % -1 é o end-of-file
gets(I, C, [C|O]) :- get0(CC), gets(I, CC, O).
```

Exemplo de utilização:

```
| ?- gets(X), format(":~s:", [X]).
ola
:ola:

X = [111,108,97] ?
```

```
yes
| ?-
```

Linha de Comando

Pode-se usar os predicados **`argument_list/1`** em que **`argument_list(LISTA)`** unifica **`LISTA`** com os argumentos colocados na linha de comando. Ver a documentação para mais formas de usar.

Goal inicial

Um programa Prolog (compilado com **`gplc`**) pode incluir uma diretiva

```
:- initialization(MAIN).
```

que faz com que o goal **`MAIN`** seja executado quando o programa for iniciado. Por exemplo, se quisermos correr o goal xpto e terminar logo, podemos dar como diretiva de inicialização:

```
:- initialization((xpto, halt)).
```

Observe-se que como o goal tem dois subgoals (**`xpto`** e **`halt`**), precisamos de por parêntesis à volta, senão seria interpretado como como uma chamada a **`initialization/2`** em vez de **`initialization/1`**.

Componentes

O trabalho consiste em:

1. Um programa (em Prolog, CLP(FD) ou OCaml) que satisfaça o pedido acima.
2. Um relatório breve, que:
    a. Explique as opções tomadas (representação das jogadas, representação do tabuleiro, como é que se codificam as jogadas permitidas, etc.)
    b. Explique a organização do código
    c. Indique os pontos fortes e fraquezas do programa
    d. Discuta quais poderiam ser as evoluções caso tivesse mais 1 mês para dedicar ao trabalho

Datas

Data limite de entrega: 2023.01.07

Data para discussões: a marcar caso-a-caso, após a entrega