Projeto Prático de
LabII e Lab Algoritmos
1º Ano de MIEI e de LCC
O Jogo do *Reversi*

Desde os primórdios da computação sempre houve a ambição de produzir máquinas inteligentes. Uma interessante demonstração dessa inteligência pode ser vista quando temos máquinas a defrontar humanos em jogos de estratégia, como por exemplo Xadrez, Damas, Go, Reversi, etc.

O objetivo deste projeto prático é o desenvolvimento de um programa capaz de demonstrar o tão esperado comportamento i.e. produzir um programa capaz de jogar um jogo de estratégia contra humanos. O jogo escolhido é o conhecido *Reversi* (jogo encontrado muitos vezes nos nossos telemóveis) que pode ser descrito sucintamente como sendo um jogo com dois adversários que colocam fichas (peças) de duas cores diferentes num tabuleiro de oito linhas por oito colunas.

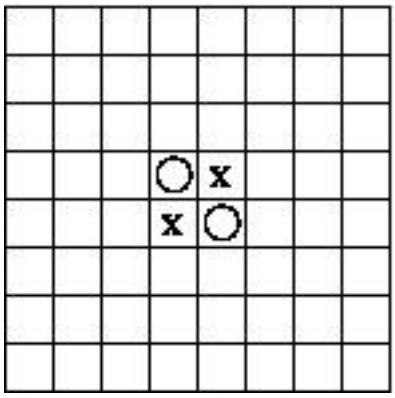

Figure 1: Tabuleiro Inicial

Inicialmente cada jogador tem duas peças já colocadas no tabuleiro. Essas peças estão localizadas no centro do tabuleiro em posição diagonal (ver figura do tabuleiro inicial). O objetivo do jogo é ter o maior número possível de peças no tabuleiro. Assim, ganha o jogador que no final do jogo tenha mais peças. O jogo termina quando todas as posições do tabuleiro estejam preenchidos ou quando não for mais possível fazer uma jogada válida.
Uma jogada para ser válida tem de obedecer a um princípio de colocação de peças no tabuleiro. Só é válido colocar uma ficha numa posição do tabuleiro se esta colocação finalizar um cerco de um conjunto de peças do oponente. Este cerco pode ser em todas as direções. Ou seja, ao longo da linha onde a ficha é colocada, ao longo da coluna ou na diagonal.

Esta colocação tem dois efeitos: colocar mais uma ficha da cor do jogador no tabuleiro e tornar as peças cercadas do oponente em peças da cor do jogador. Vejamos o efeito do jogador com a peça 'X' jogar na linha 6 e coluna 5:

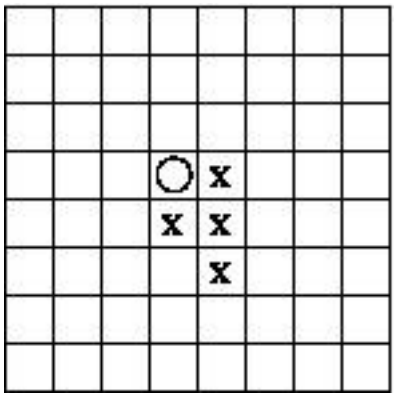

Figure 2: Tabuleiro após jogar ficha X na posição 6,5.

O desenvolvimento do programa a implementar pode ser dividido em duas sub-tarefas:
A primeira é produzir um programa capaz de simular o jogo com a matriz do tabuleiro (8 colunas por 8 linhas), aceitar e validar jogadas do utilizador e verificar se há uma posição final do jogo (indicando quem venceu e contando o número de peças de cada jogador).
Notar que o joga termina quando:

- O tabuleiro está completamente preenchido nas suas 64 posições. Ganha quem tiver mais peças no tabuleiro
- Nenhum jogador tem uma jogada válida disponível. Ganho o jogador com mais peças no tabuleiro
- Só há peças no tabuleiro de um jogador. Consequentemente o oponente não tem jogadas válidas. O jogador também não pode cercar peças do oponente! Logo o jogo termina e ganha este jogador (caso especial da situação anterior).

A segunda tarefa prende-se com a geração de jogadas pelo computador, que é como quem diz, programar um bom jogador de *Reversi*. Este jogador eletrónico deve ser capaz de reagir às jogadas do adversário produzindo estratégias (jogadas) capazes de o derrotar.

Objetivos

O objetivo do trabalho é desenvolver um programa em linguagem C que implemente as tarefas atrás descritas. Podemos detalhar os objetivos a cumprir na seguinte lista:

- Implementar um programa que permita jogar devidamente o *Reversi* (por vezes também conhecido como Othello). Isto quer dizer que o programa deve aceitar jogadas dos utilizadores (representadas pelo número da linha e coluna onde colocar uma ficha) e validar essa jogada, rejeitando-a se for caso disso. O programa deve ter uma visualização constante da situação do jogo i.e. estado do tabuleiro, conjuntamente com o *score* de cada jogador. Deve ser possível definir qual o jogador que inicia o jogo (se o da peça 'X' ou 'O'). Isto vai permitir ter dois jogadores eletrónicos a jogar um contra o outro.

- O programa tem de gerar jogadas para o jogador representativo do computador. Ou seja, tem de escolher a jogada que beneficia mais a posição do computador. Obviamente estas jogadas têm também de ser validadas.

		X	X	X	X	X	X
			O	X	X	X	X
			O	O	O	X	X
	O	O	O	O	X	O	X
		O	O	X	O	X	X
O	O	O	O	O	O	O	O
O					O		
O					O		

Figure 3: O Jogador 'O' tem de passar jogada pois não existem jogadas válidas disponíveis para a peça 'O' (brancas).

Gravação do estado do Jogo (formato do ficheiro)

Para gravar num ficheiro o estado do tabuleiro em texto vamos usar o seguinte formato:

- Primeira linha indica o modo (**M**anual ou **A**utomático) e o próximo jogador a jogar ('X' ou 'O'). Manual para jogo entre dois humanos e Automático para jogo contra o computador.
- Para cada linha do tabuleiro utiliza-se uma String com os seguintes carateres:
  - 'X' para as peças pretas
  - 'O' para as peças brancas
  - '-' para espaços vazios
  - '.' Para assinalar jogadas válidas para este jogador
- Exemplo:

```
M O
- - - - - - - -
- - - - - - - -
- - . X . - - -
- - - X X - - -
- - . X O - - -
- - - - - - - -
- - - - - - - -
- - - - - - - -
```

Escala das Tarefas

1. Primeiro programa: Jogo com dois jogadores. Ler jogadas via teclado (coordenadas do local onde colocar a peça). Validar jogada. Imprimir tabuleiro com o reflexo dessa jogada. Implementar uma função que dado um tabuleiro e uma peça (brancas ou pretas) produzir uma lista com as coordenadas das jogadas válidas para esse jogador. Imprimir o score de cada jogador (número de peças).
2. Verificar se há situação de fim do jogo. Implementar a situação de "passar a jogada" (ver figura 3).
3. Gravar o estado do tabuleiro em ficheiro com o formato acima descrito.
4. Implementar linha de comandos que aceita e executa:

    - **N <peça>** para novo jogo em que o primeiro a jogar é o jogador com **peça**.
    - **L <ficheiro>** para ler um jogo de ficheiro.
    - **E <ficheiro>** escrever em ficheiro estado do jogo.
    - **J <L,C>** jogar peça atual na posição (l,c).
    - **S** para imprimir um ponto '.' nas posições com jogada válida.
    - **H** para sugestão de jogada. Deve ser colocado um '?' no sitio sugerido.
    - **U** para desfazer a última jogada (Undo). Isto tem de permitir desfazer até ao estado inicial do jogo!
    - **A <peça>** novo jogo contra 'bot' em que o computador joga com a peça **<peça>.** Neste modo quem joga primeiro é sempre o jogador com a peça preta 'X'.
    - **Q** para sair.

    Nota: Cada alteração (comando) implica necessariamente imprimir o tabuleiro! Só as jogadas válidas são aceites.

5. Jogador eletrónico. Implementar um jogador 'bot' que usa uma variante do algoritmo *MiniMax* para a análise de tabuleiros e subsequente geração da melhor jogada.
6. Oferecer vários níveis de dificuldade i.e. oponente básico, médio, bom, profissional, etc.

Considerações

A representação do tabuleiro terá obrigatoriamente de usar a estrutura de dados fornecida (projeto C em anexo). Poderá ser adicionado novos campos à estrutura fornecida.

A operação de desfazer comandos deve ser implementada usando estruturas dinâmicas.

Grupos

Os grupos de trabalho são compostos por 3 elementos e terão necessariamente de ser compostos por pessoas do mesmo turno prático. Nos casos em que o número de elementos no turno não seja divisível por 3 aceitam-se 2 grupos de 2 elementos se o resto da divisão do número por 3 for 1 e 1 grupo de 2 elementos se o resto der 2.