Programação I

Trabalho Prático 2021/2022

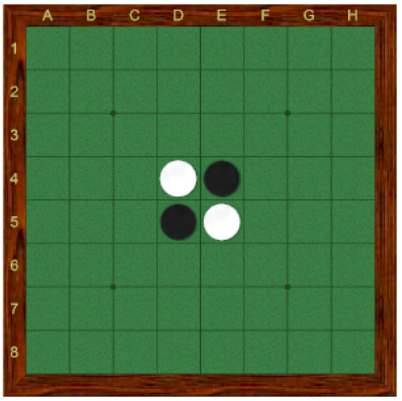
(v1.0)

→ Othello ←

1. Othello

O Othello[1] é um jogo de tabuleiro de estratégia para dois jogadores. O tabuleiro tem 8 linhas e 8 colunas e um conjunto de peças. As peças são discos com uma face clara e outra escura, cada cor pertencendo a um jogador.

O objetivo do jogador é ter a maioria das suas peças à mostra no final do jogo, virando o máximo possível de peças do adversário.



O jogo tem início com 2 peças brancas e 2 pretas dispostas no tabuleiro como indicado na figura.

Um movimento consiste em flanquear as peças do oponente trocando, de seguida, a cor das peças flanqueadas. Flanquear significa colocar uma peça no tabuleiro de modo que a linha (ou linhas) com peças do oponente seja delimitada, em cada extremidade, por uma peça sua (uma "linha" pode ser composta por uma ou mais peças).

Por exemplo, se peça A e as peças pretas estiverem no tabuleiro e for a vez do jogador das peças brancas jogar, ao colocar a peça B, as três peças pretas são flanqueadas, e como tal, a sua cor é trocada, passando a linha a ter a configuração.

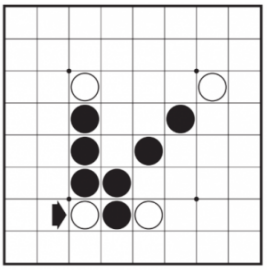
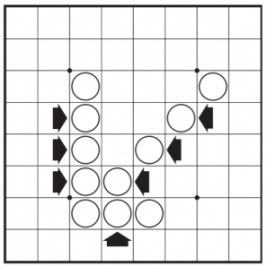
antes: 

depois: 

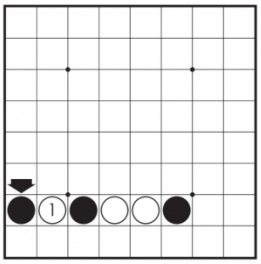
Regras

1. As pretas jogam sempre primeiro
2. Se numa jogada não for possível flanquear o adversário e virar pelo menos uma peça do oponente, a vez é perdida e o oponente joga novamente. No entanto, se existir um movimento disponível o jogador é obrigado a jogar.
3. Todas as peças flanqueadas em qualquer movimento devem ser viradas (mesmo que seja vantajoso para o jogador não virá-las). O exemplo seguinte mostra a mudança de cor ao colocar a peça branca assinalada.

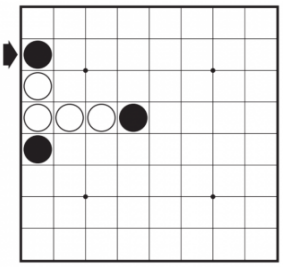
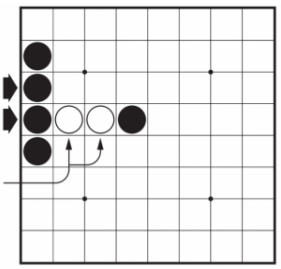
[1] https://www.worldothello.org/about/about-othello

antes:  depois: 

4. Os jogadores não podem pular a(s) sua(s) peça(s) para flanquear o oponente. Por exemplo, ao jogar a peça preta apenas a peça branca identificada com "1" é virada.



5. A(s) peça(s) só podem ser flanqueadas como resultado direto de um movimento e devem estar na linha direta da peça colocada. No exemplo, ao jogar a peça preta apenas as peças brancas na vertical são flanqueadas (e viradas)

antes:  depois: 

6. Se uma peça é colocada num quadrado, já não pode ser movida para outro quadrado mais tarde no jogo.

7. Quando não for possível nenhum dos jogadores jogarem o jogo termina. As peças são contadas e o jogador com o maior número de peças é o vencedor.

   Nota: É possível um jogo terminar antes dos 64 quadrados estarem preenchidos.

2. Descrição do trabalho

O trabalho consiste em desenvolver uma aplicação, implementada em linguagem C, para jogar Othello com o computador.

Em cada instante do jogo cada posição do tabuleiro está **livre**, tem uma peça **branca** ou uma peça **preta**. Uma posição livre é identificada pelo carácter '**.**' e as posições ocupadas por peças brancas ou pretas são identificadas pelos caracteres 'o' e 'x', respectivamente.

Pretende-se que a aplicação permita jogar quer com um tabuleiro inicial, quer com um tabuleiro obtido a partir de um conjunto de jogadas lidas de um ficheiro de texto.

Para executar o jogo, deverá executar um dos seguintes comandos:

```
$ ./othello
$ ./othello jogadas.txt
```

O primeiro comando permite jogar interactivamente a partir da posição inicial do jogo (2 peças pretas e 2 brancas). O segundo comando inicializa o jogo do mesmo modo, mas permite ler uma sequência de jogadas (pretas e brancas) a partir de um ficheiro indicado como parâmetro. O jogo passa a interactivo depois de lidas as jogadas do ficheiro.

Antes de iniciar um jogo é escolhido aleatoriamente, entre o jogador e o computador, qual fica com as peças pretas. Isto acontece quer seja lido, ou não, um ficheiro de jogadas.

No início da fase interactiva do jogo, é mostrado o estado do tabuleiro. Depois, durante o jogo, após cada jogada é mostrado o novo estado do tabuleiro já com as peças do adversário "viradas".

O ficheiro de texto tem uma jogada por linha, com a identificação da posição onde colocar a peça; as linhas são identificadas por algarismos (1 a 8); as colunas por letras (A a H). O conteúdo de um desses ficheiros poderia ser, por exemplo:

```
4C
5C
6C
5B
6E
5F
6G
```

Quando o jogo termina (porque o tabuleiro está cheio ou não é possível colocar mais peças) é mostrada uma mensagem como a seguinte:

```
Game Over!
Black: 54 discs, White: 10 discs
You win!
```

Outras resultados possíveis são:

- `You lose!`
- `It's a draw!`

O objetivo principal do trabalho não é dotar o computador de "inteligência" (fazer a melhor jogada dado um tabuleiro) mas implementar a dinâmica do jogo: representar um tabuleiro, verificar a validade de uma jogada, alterar o tabuleiro de acordo com uma jogada válida, etc… Dotar o programa de "inteligência" apenas poderá/deverá ser adicionada após a implementação completa e correta das restantes características.

Trabalhos com implementação de estratégias "inteligentes" (e que cumpram toda a dinâmica do jogo) terão uma sobre-valorização de no máximo 2 valores, sendo promovida uma competição entre os participantes.

3. Exemplos de interação

Para ilustrar a interação com o programa, são mostrados dois exemplos de interação onde **In**, a vermelho, representa *input* do utilizador e **Out**, a azul, o *output* do programa. O tabuleiro é mostrado após cada jogada; na vez do computador é indicada a jogada antes de mostrar o tabuleiro atualizado.

O trabalho deverá implementar **estritamente** a interação a seguir exemplificada (mensagens e respostas, incluindo os espaços e quebras de linha); a imaginação e criatividade deverá ser canalizada para a implementação das funções.

Exemplo 1: geração do tabuleiro

```
$ ./othello
Othello Game

Your discs are Black.

  A B C D E F G H
1 . . . . . . . .
2 . . . . . . . .
3 . . . . . . . .
4 . . . o x . . .
5 . . . x o . . .
6 . . . . . . . .
7 . . . . . . . .
8 . . . . . . . .

Choose your move. For instance: 5F
6D
Invalid move!
Choose your move. For instance: 5F
4C

  A B C D E F G H
1 . . . . . . . .
2 . . . . . . . .
3 . . . . . . . .
4 . . x x x . . .
5 . . . x o . . .
6 . . . . . . . .
7 . . . . . . . .
8 . . . . . . . .

My move: 5C

  A B C D E F G H
1 . . . . . . . .
2 . . . . . . . .
3 . . . . . . . .
4 . . x x x . . .
5 . . o o o . . .
6 . . . . . . . .
```

```
7 . . . . . .
8 . . . . . .

Choose your move. For instance: 5F
6C

  A B C D E F G H
1 . . . . . .
2 . . . . . .
3 . . . . . .
4 . . x x x . . .
5 . . x x o . . .
6 . . x . . . . .
7 . . . . . .
8 . . . . . .

My move: 5B

  A B C D E F G H
1 . . . . . .
2 . . . . . .
3 . . . . . .
4 . . x x x . . .
5 . o o o o . . .
6 . . x . . . . .
7 . . . . . .
8 . . . . . .

Choose your move. For instance: 5F
6E

   A B C D E F G H
1 . . . . . .
2 . . . . . .
3 . . . . . .
4 . . x x x . . .
5 . o o x x . . .
6 . . x . x . . .
7 . . . . . .
8 . . . . . .
```

Exemplo 2: leitura do ficheiro

```
$ ./othello jogadas.txt
Othello game

Your discs are Black.

  A B C D E F G H
1 . . . . . .
2 . . . . . .
3 . . . . . .
4 . . x x x . . .
```

```
5 . o o x x . . .
6 . . x . x . . .
7 . . . . . . . .
8 . . . . . . . .

My move: 5F

  A B C D E F G H
1 . . . . . . . .
2 . . . . . . . .
3 . . . . . . . .
4 . . x x x . . .
5 . o o o o o . .
6 . . x . x . . .
7 . . . . . . . .
8 . . . . . . . .

Choose your move. For instance: 5F
6G

   A B C D E F G H
1 . . . . . . . .
2 . . . . . . . .
3 . . . . . . . .
4 . . x x x . . .
5 . o o o o x . .
6 . . x . x . x .
7 . . . . . . . .
8 . . . . . . . .
```

4. Implementação

É importante dividir o trabalho em duas fases:

1. Desenho da aplicação. Compreende a estruturação em funções com especificação clara dos argumentos e valor de retorno de cada função.

2. Implementação da aplicação. Inclui a implementação e teste de cada uma das funções, começando pelas mais simples (e que não chamam outras funções)

Fase 1

O desenho da aplicação inclui:

1. Perceber o problema, identificando inequivocamente o input e output do programa;
2. Pensar numa solução, que inclui dividir o problema noutros mais simples, encontrar a solução para estes e, a partir destas, construir a solução geral

Como resultado desta fase, deverá obter o programa estruturado através de um conjunto de funções, com especificação clara dos argumentos e valor de retorno de cada função.

Comece por decidir qual a representação de um quadrado e do tabuleiro. **A variável que representa o tabuleiro deve ser declarada na função `main()`**.

Estruture o seu programa em funções. Deverá, no mínimo, implementar as seguintes:

- `init_board( board )`
  - Inicializa o tabuleiro (parâmetro board) com as peças brancas e pretas iniciais.
- `print_board( board )`
  - mostra o tabuleiro atual no *standard output* (ecrã).
- `play( board, line, col, color )`
  - coloca uma peça de cor `color` na posição (`line`, `col`) e vira as peças do adversário de acordo com as regras indicadas na descrição do jogo..
- `flanked( board, line, col, color )`
  - Conta o número de peças viradas ao jogar numa certa linha e coluna. Se a jogada for inválida retorna zero.
- `int count_flips_dir(board, line, col, delta_line, delta_col, color)`
  - Conta quantas peças serão viradas, numa certa linha e coluna, e numa certa numa direção (definida por delta_line e delta_col, por exemplo se delta_line=1 e delta_col=1, estamos a considerar a direção "baixo-direita")

Fase 2

Esta fase compreende a codificação e teste de cada uma das funções desenhadas na fase anterior, bem como do programa principal. Nesta fase é essencial comentar o código. Para tal utilize o seguinte formato:

```
#######################################################
# area - Esta funcao calcula a area de um retangulo
#
# Argumentos:
# l - largura, inteiro
# c - comprimento, inteiro
# Valor de retorno:
# area calculada, inteiro
#######################################################
int area(int l, int c){
    return l*c;
}
```

Sugere-se a organização do código nos seguintes ficheiros:

- `othello_func.h`: com os protótipos das funções do programa
- `othello_func.c`: implementação das funções definidas em `othello_func.h`
- `othello.c`: implementação da função `main()`

O comando

```
gcc -c othello.c
```

compila o ficheiro `othello.c` e gera o ficheiro objeto `othello.o`

O comando

```
gcc -o othello othello.o othello_func.o
```

liga os ficheiros objeto (`othello.o` e `othello_func.o`) e cria o executável `othello`

5. Condições Gerais

- O trabalho deverá ser efetuado por grupos de 2 elementos. Em casos excecionais, e acordados com os docentes, poderá ser feito por apenas 1 elemento.
- A entrega será via Moodle e até à data lá indicada.
- O trabalho será discutido em dia e horário a anunciar.
- O relatório deverá conter a identificação dos elementos do grupo assim como uma descrição sucinta quer dos tipos das principais variáveis (incluindo: tabuleiro, quadrado), quer das principais funções (incluindo obrigatoriamente `play()` e `flank()`). Preferencialmente não deve ter código.
- O trabalho será classificado entre 0 e 6 valores (correspondente a 30% da nota final) com os seguintes fatores de avaliação:
    - a clareza dos algoritmos implementados;
    - a correção do código;
    - a legibilidade do código (a qualidade dos identificadores, os comentários que o acompanham, etc);
    - a qualidade do relatório (lendo-o, deve ficar-se com uma ideia precisa do funcionamento do trabalho).
- Será aplicado o código de conduta do Departamento de Informática. Em caso de fraude, para além reprovação à disciplina, a situação será reportada.

6. Entrega

- O *upload* do trabalho é efectuado no Moodle pelo elemento do grupo que tiver o **número menor**.
- Deverá ser um ficheiro **`.tar.gz`** ou **`.zip`** que, ao descomprimir, crie uma pasta **`num1_num2`** em que `num1` e `num2` são os números dos alunos que compõem o grupo por ordem crescente, por exemplo: `123_456`. Nessa pasta deverá existir uma subpasta **`src`** com todo o código fonte necessário para testar o programa. Deverá também existir na directoria principal um ficheiro (**`relatorio`**) em formato texto, HTML ou PDF com o relatório.