Instituto Superior Técnico
DEEC
Programação (LEAer e LEE) 2022/2023
2º Trabalho (v2)

1 Introdução

Neste trabalho pretende-se desenvolver um um jogo utilizando a linguagem de programação C (opção -ansi do compilador gcc) semelhante ao jogo de estratégia **reversi,** cujas regras estão descritas em https://en.wikipedia.org/wiki/Reversi. A aplicação deve ter o nome **flipflop** e será avaliada no ambiente Linux.

Neste jogo existem dois jogadores, um jogador é humano, o outro é um jogador IA (inteligência artificial) que executa um algoritmo para ganhar. As peças do jogo têm duas faces, uma de cor branca, representada por 'O' e a outra face a cor preta, representada por 'X'. As peças são colocadas num tabuleiro que tem 8 por 8 lugares. Um jogador utiliza a cor branca ('O') e o outro a cor preta (´X´), os lugares vazios estão marcados com '.' . Os jogadores jogam de forma alternada.

A configuração inicial do tabuleiro esta representada na figura 1. A representação do tabuleiro é realizada num janela em modo texto (existem versões do jogo que utilizam uma interface gráfica **kreversi** no linux) a qual é mais simples e mais fácil de programar.

Em cada jogada, o programa deve indicar o jogador que vai jogar (H-humano e IA – jogador de IA) e o símbolo que vai ser colocado no tabuleiro ('O' ou 'X'). O programa deve pedir a linha e a coluna ao jogador humano. Cada jogador só pode colocar uma peça com a sua cor. O jogador deve colocar uma peça, junto das existentes no tabuleiro, de modo a transformar (virar) as peças do outro jogador. Para isso é necessário que as peças do outro jogador formem uma linha, entre a peça que foi colocada e uma outra peça, com a cor da que foi colocada.

```
  0 1 2 3 4 5 6 7
0 . . . . . . . .
1 . . . . . . . .
2 . . . . . . . .
3 . . . O X . . .
4 . . . X O . . .
5 . . . . . . . .
6 . . . . . . . .
7 . . . . . . . .
Jogador 1 (O) (linha e coluna) =
```

Figura 1: Representação do tabuleiro em modo texto com a configuração inicial.

Realizada a jogada, o jogador passa a vez ao outro jogador, mesmo que não conseguia colocar uma peça no tabuleiro porque não consegue transformar/virar as peças do outro jogador.

O jogador vencedor é o que obtiver o maior número de peças da sua cor quando o tabuleiro estiver totalmente preenchido ou, se durante o jogo, o outro jogador ficar sem peças da sua cor.

2 Funcionalidades do programa

Funcionalidades a serem realizadas:

1. A ordem de execução do programa é dada através da linha de comando na qual são

seleccionadas opções, que são acessíveis no programa através dos argumentos da função main.

Opções	**Descrição**
-f nome_ficheiro	Guarda num ficheiro (cujo nome é escrito na linha de comando) todas as jogadas e ocupação/estado do tabuleiro.
-p X	O jogador humano escolhe a peça X (ou o O se optar por -p O)
-u	Permite que o jogador humano efectue o "undo" de jogadas durante o jogo. Para isso deve pressionar a tecla u. Pode recuar nas jogadas (até à jogada inicial) e depois continuar a jogar. Não há a opção "redo".

a) Exemplos que descrevem a utilização das opções:

```
1. ./flipflop
2. ./flipflop -f jogo1.txt
3. ./flipflop  -p O -f jogo1.txt
4. ./flipflop  -u -p O -f jogo1.txt
```

Nota: As opções podem ser escritas na linha de comando por qualquer ordem.

2. A opção de **undo** deve ser realizada com **memória dinâmica e estruturas ligadas**.
3. As decisões do jogador IA, ao jogar uma peça, devem ser realizadas de acordo com o algoritmo (regras/estratégia) que estão descritas nas alíneas seguintes:
   a) Deve ser realizada uma procurar sistemática de todas as posições que podem dar origem a troca de peças do adversário. Para cada uma dessas possibilidades deve contabilizar o número de trocas das peças do adversário.
   b) O programa deve apresentar no ecrã, no tabuleiro, as posições possíveis que o jogador IA pode escolher. Essas posições devem ser assinaladas com o 'P'. Esta funcionalidade serve para verificar/testar o funcionamento do programa.
   c) Deve escolher a posição que permite obter o maior número de trocas de peças do adversário. **Nota**: Se existirem posições que dão o mesmo número de trocas de peças, o computador deve escolher a posição que tem a menor coordenada x (coluna), e a menor coordenada y (linha).
   d) A concretização da **estratégia do jogador IA deve ser realizada utilizando memória dinâmica e estruturas ligadas**.

3 Entrega e avaliação

1. A **data entrega** está afixada na página da cadeira. É necessário consultar a página da disciplina para verificar se há actualizações da data ou do enunciado do trabalho.
2. A aplicação será avaliada num computador com o sistema operativo Linux. Deverá compilar

o programa com as opções `-Wall -Werror -pedantic -ansi`

3. Todos os ficheiros com o código do programa devem estar identificados com os nomes dos alunos e os respectivos números.
4. Para realizar o upload no fénix deve colocar todos os ficheiros (*.c , *.h makefile) necessários para gerar o executável do programa num ficheiro zip. O nome do ficheiro deve ter trab2 seguido dos números dos alunos, exemplo **trab2_99998_100100.zip**
5. Elementos a serem avaliados nesta fase:
   a) Processamento das opções da linha de comando (-f, -u, -p), validação das mesmas e realização correcta, detecção do fim do jogo (**5.5 valores**). Na opção undo é necessário utilizar memória dinâmica e estruturas ligadas.
   b) Realização de acções associadas ao jogador humano, validar as coordenadas da posição seleccionada, adjacência, determinação das trocas, e actualização do tabuleiro (**5.0 valores**).
   c) Realização das acções associadas ao jogador IA, escolher a posição que vai dar o maior número de trocas, mostrar no ecrã as posições admissíveis e a posição que vai ser escolhida pelo jogador IA , actualização do tabuleiro e utilização de memória dinâmica e estruturas ligadas (**5.5 valores**).
   d) Organização do código, dimensão das funções, argumentos, tipos, alinhamento de código, funções documentadas, reutilização de funções desenvolvidas pelo programador, compilação separa de módulos, utilização do utilitário make, documentação das funções (**4.0 valores**).
   e) A **ocorrência de erros** durante a execução do programa como **“segmentation fault”, “stack smashing”**, … , **limita a nota** ao valor máximo a 15 valores.
   f) A **não realização do algoritmo do jogador IA** e a sua substituição por um jogador humano **limita a nota** do trabalho a 14 valores.

4 Dúvidas

As dúvidas podem ser esclarecidas no início e no fim das aulas, e no horário de esclarecimento de dúvidas.

Nota: Na definição inicial das funcionalidades de uma aplicação, existem elementos que não estão completamente especificados. Nessas situações, o programador pode especificar os elementos em falta utilizando para o efeito justificações lógicas.

Um bom jogo!