Programação

LEI; LEI-PL; LEI-CE - 2021/22

Trabalho Prático

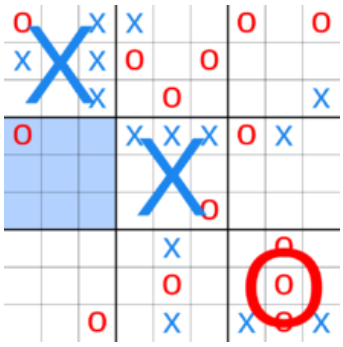
Ultimate Tic-Tac-Toe

Notas prévias:

- O enunciado é possivelmente vago, genérico e incompleto em alguns pontos. O que se pretende é que os alunos avaliem as opções existentes e escolham a que considerarem mais apropriada para cada uma das situações com que se depararem. Todas as escolhas devem ser referidas e justificadas no relatório.
- O programa deve ser implementado em C *standard*, *i.e.*, não deve ter instruções que o tornem específico para um determinado ambiente/plataforma de desenvolvimento. Deverá ser respeitada a norma C99.
- O programa entregue deve ter uma interface simples e amigável, indicando o que pode ser feito em cada situação. Não são valorizados programas com interfaces gráficas e/ou utilização de bibliotecas que não sejam *standard*, por exemplo, para usar cores ou posicionar o cursor no ecrã.
- Deve distribuir o código fonte por vários ficheiros. Para além dos ficheiros com código disponibilizados com este enunciado, deverão existir, no mínimo, outros dois ficheiros com código fonte.
- Deverá utilizar *header files* para gerir as dependências entre os vários ficheiros com código fonte.
- Cada ficheiro deve ter a identificação do aluno (nome completo e número), em comentário, nas linhas iniciais do ficheiro.

1. Introdução

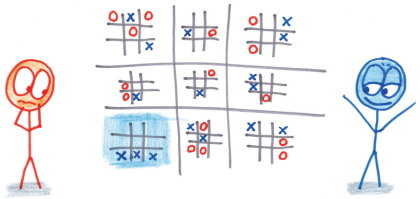
O programa a implementar deverá permitir a realização da versão do jogo do galo descrita a seguir. Esta é uma versão de um jogo de tabuleiro entre 2 pessoas que efetuam jogadas alternadas, até que uma delas vença ou que se verifique um empate. No ponto 3 são descritas as principais funcionalidades a implementar.



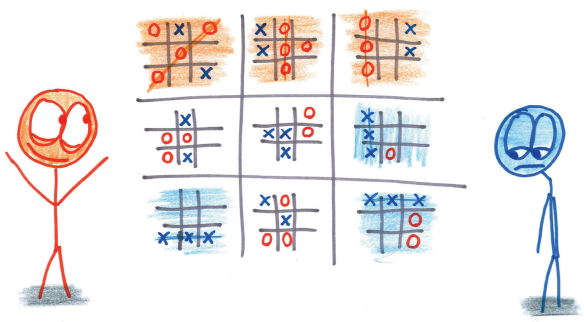
2. Ultimate Tic-Tac-Toe

Existe uma área de jogo no qual estão organizados 9 mini-tabuleiros do jogo do galo, dispostos numa grelha 3×3. O jogo desenrola-se entre 2 jogadores, de acordo com as seguintes regras:

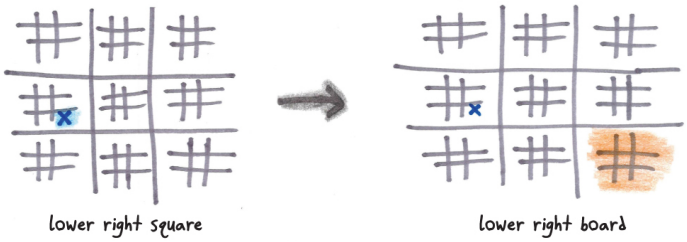
1. Na sua vez, cada jogador coloca a sua peça num dos 9 mini-tabuleiros
2. Um jogador que faça 3 em linha num dos mini-tabuleiros ganha essa secção



3. Um jogador que ganhe 3 mini-tabuleiros em linha (no tabuleiro grande) ganha o jogo



4. A escolha do mini-tabuleiro em que se joga não é livre, sendo determinada pela jogada anterior do adversário. A posição escolhida pelo adversário identifica o mini-tabuleiro onde deve ser continuado o jogo.



Neste exemplo vê-se que o jogador azul colocou uma peça na posição (3,3) do mini-tabuleiro selecionado para a sua jogada. Na próxima jogada, o oponente tem que colocar uma peça no mini-tabuleiro que se encontra nesta posição na área global do jogo. Pode dar-se o caso do mini-tabuleiro escolhido para continuar o jogo já estar terminado, i.e., já ter um vencedor. Se isso acontecer, fica ao critério do aluno delinear uma regra para determinar o próximo tabuleiro em que continua o jogo.

5. Verifica-se um empate se o jogo chegar ao final sem que nenhum dos jogadores consiga fazer 3 mini-tabuleiros em linha.
6. Fica ao critério do aluno definir as regras para a jogada inicial, i.e., quem deve iniciar o jogo e como é escolhido o mini-tabuleiro em que esta jogada inicial deve ser efetuada.

3. Programa a Implementar

Pretende-se que desenvolva um programa em linguagem C que permita jogar o jogo descrito no ponto anterior. As funcionalidades previstas são descritas nos pontos seguintes.

3.1. Tipos do Jogo

O programa deve permitir duas modalidades:

1. Jogo entre 2 jogadores humanos. Neste caso, recebe e valida as jogadas alternadas dos jogadores e deteta o final do jogo
2. Jogo entre um humano e um jogador automático. Nesta opção, o programa deve ter funcionalidades que simulem as escolhas de um jogador. Não se pretende que desenvolva e implemente uma estratégia inteligente completa para o jogo. Deve simplesmente programar um jogador automático que escolha aleatoriamente uma jogada legal em cada iteração.

3.2. Gestão da informação

O tabuleiro do jogo deve ser mantido num *array* dinâmico. Poderá utilizar as funções que foram disponibilizadas em anexo a este enunciado para efetuar o armazenamento e gestão desta informação. Fica ao critério do aluno decidir como é definida e gerida esta estrutura dinâmica. *Poderá recorrer a um array dinâmico único, ter 9 pequenos tabuleiros ou qualquer outra solução que considere adequada. Além disso, não é obrigatório que utilize*

o código que foi disponibilizado. A única restrição é que, durante a realização do jogo, a informação do tabuleiro deve ser mantida num (ou vários) array(s) dinâmico(s).

A sucessão de jogadas realizadas deve ser mantida numa lista ligada simples. A informação a manter nesta lista corresponde à jogada realizada e respetivo jogador. Poderá guardar informações adicionais, caso considere relevante, mas não se pretende que guarde o tabuleiro completo. Esta lista ligada mantém todas as jogadas realizadas, desde o início do jogo.

Antes de cada jogada, um jogador poderá solicitar a visualização das K jogadas anteriores (K varia entre 1 e 10, sendo o valor indicado pelo jogador que efetua o pedido). A visualização deve mostrar as jogadas ordenadamente, da mais antiga para a mais recente. Deve apenas mostrar as jogadas efetuadas (por exemplo, o jogador X efetuou a jogada Y), não sendo necessário mostrar os tabuleiros completos.

3.3. Interrupção do Jogo

Esta funcionalidade permite que os jogos sejam interrompidos e retomados mais tarde. O programa deve guardar num **ficheiro binário,** com nome "*jogo.bin*", a lista das jogadas que está armazenada na lista ligada e outra informação essencial para retomar o jogo mais tarde. Não se pretende que guarde o tabuleiro completo no ficheiro, uma vez que a sucessão de jogadas deverá ser suficiente para retomar o jogo. Quando a aplicação é reiniciada deverá ser verificada a existência do ficheiro e, caso exista, o utilizador deverá ser questionado sobre se pretende continuar o jogo anterior.

3.4 Exportação para Ficheiro

No final do jogo, a sucessão completa das jogadas realizadas deve ser exportada para um **ficheiro de texto**, cujo nome é pedido ao utilizador**.** Neste ficheiro ficará informação detalhada e completa das jogadas que foram efetuadas. O ficheiro deve ser criado e escrito **apenas no final do jogo**, com base na informação armazenada na lista ligada que mantém as jogadas realizadas.

Código Disponibilizado

4.1. Matriz Dinâmica de Caracteres

O ficheiro *matdin.c* contém algumas funções que permitem criar e gerir uma matriz dinâmica de caracteres. Poderão ser usadas para a criação do tabuleiro de jogo.

*char** criaMat(int nLin, int nCol);*
Cria uma matriz dinâmica de caracteres com nLin linhas e nCol colunas

*void libertaMat(char** p, int nLin);*
Liberta a matriz dinâmica de caracteres p com nLin linhas

*void mostraMat(char **p, int nLin, int nCol);*
Mostra na consola o conteúdo de uma matriz de caracteres p com nLin linhas e nCol colunas

*void setPos(char **p, int x, int y, char c);*
Coloca o caracter c na posição (x,y) da matriz dinâmica de caracteres p

*char getPos(char **p, int x, int y);*
Devolve o caracter na posição (x,y) da matriz dinâmica de caracteres p

4.2. Funções Random

O ficheiro *utils.c* contém algumas funções auxiliares que podem ser úteis durante a implementação do trabalho:

void initRandom();

Inicializa o gerador de números aleatórios. Deve ser chamada apenas uma vez, no início da execução do programa.

int intUniformRnd(int a, int b);

Devolve um valor inteiro aleatório distribuído uniformemente no intervalo *[a, b]*.

int probEvento(float prob);

Devolve o valor 1 com probabilidade *prob*. Caso contrário, devolve 0.

O ficheiro *utils.c* contém igualmente uma função *main()* que serve apenas para ilustrar alguns exemplos de chamadas das funções disponibilizadas.

4. Normas para a realização do trabalho

O trabalho deve ser **realizado individualmente**. O trabalho só pode ser entregue uma vez e a nota obtida é válida em todas as épocas de avaliação do ano letivo 2021/22.
Não existirá um novo trabalho ou uma substituição desta componente de avaliação para a época especial de setembro.

Data provisória para entrega do trabalho prático: **23.59** do dia **19 de Junho de 2022.**

Material a entregar:

- Entregar através do Moodle um ficheiro compactado em formato ZIP, contendo o relatório, o código fonte comentado (apenas os ficheiros .c e .h) e os ficheiros de dados necessários para o funcionamento do programa.
- O nome do ficheiro ZIP deve obrigatoriamente ter o seguinte formato: `Prog_Nome_NumAluno.zip`, em que *Nome* e *NumAluno* identificam, respetivamente, o nome e número do do aluno (exemplo: *Prog_AnaSilva_123456789.zip*)

Defesa:

Os trabalhos serão sujeitos a **defesa obrigatória,** em data e formato a anunciar. As defesas poderão incluir:

i) Demonstração do funcionamento do programa
ii) Explicação detalhada do código
iii) Implementação de alterações / novas funcionalidades

Relatório

Deve ser entregue um relatório contemplando os seguintes pontos:

- Descrição genérica da organização do programa
- Identificação do ambiente de desenvolvimento utilizado durante a implementação
- Apresentação das estruturas dinâmicas implementadas e da organização dos ficheiros utilizados pelo programa, justificando as escolhas feitas
- Justificação para as opções tomadas em termos de implementação, nomeadamente na definição de algumas das regras do jogo.

Avaliação

A cotação do trabalho é de **7 valores**.

Esta componente da avaliação não tem nota mínima.

A deteção de plágio parcial ou total implica a anulação imediata de todos os trabalhos envolvidos.

Critérios de Avaliação para as Funcionalidades Implementadas

- Definição das estruturas de dados
- Correção das funcionalidades implementadas
- Manipulação de estruturas dinâmicas
- Manipulação de ficheiros
- Simplicidade/funcionalidade da interface com o utilizador