Programação

LEI; LEI-PL; LEI-CE - 2020/2021

Trabalho Prático

Jogo do Semáforo

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 do ficheiro com código disponibilizado 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.
- Todos os ficheiros devem ter a identificação do aluno (nome completo e número), em comentário, nas linhas iniciais dos ficheiros.

1. Introdução

O programa a implementar deverá permitir a realização do jogo do semáforo. Este é 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. Apresentação do Jogo

O jogo do semáforo desenrola-se num tabuleiro dividido em células. No início, o tabuleiro está vazio. Alternadamente os jogadores vão colocando peças de cor Verde (G), Amarela (Y) ou Vermelha (R). Ganha o jogador que coloque uma peça que permita formar uma linha, coluna ou diagonal completa[1] com peças da mesma cor. As jogadas válidas relativas à colocação de peças são as seguintes:

1. Colocar uma peça Verde numa célula vazia
2. Trocar uma peça Verde que esteja colocada no tabuleiro por uma peça Amarela
3. Trocar uma peça Amarela que esteja colocada no tabuleiro por uma peça Vermelha

Existem duas jogadas adicionais que podem ser efetuadas pelos jogadores:

4. Colocar uma pedra numa célula vazia. Cada jogador pode colocar, no máximo, uma pedra por jogo. A colocação de uma pedra inviabiliza que o jogo possa terminar por preenchimento da linha e coluna afetadas (e, eventualmente também da diagonal ou diagonais).
5. Adicionar uma linha ou uma coluna ao final do tabuleiro. Esta jogada adiciona linhas ou colunas completas e vazias ao tabuleiro de jogo. Cada jogador pode efetuar esta jogada, no máximo, duas vezes por jogo.

Em cada iteração, um jogador escolhe uma destas jogadas para atualizar o tabuleiro. As jogadas 4 e 5 estão sujeitas às restrições indicadas na sua definição. O número de peças de cada cor é ilimitado.

[1] Se o tabuleiro não for quadrado, será impossível terminar o jogo formando uma diagonal toda da mesma cor.

A figura seguinte ilustra algumas jogadas viáveis num tabuleiro 4 × 4.

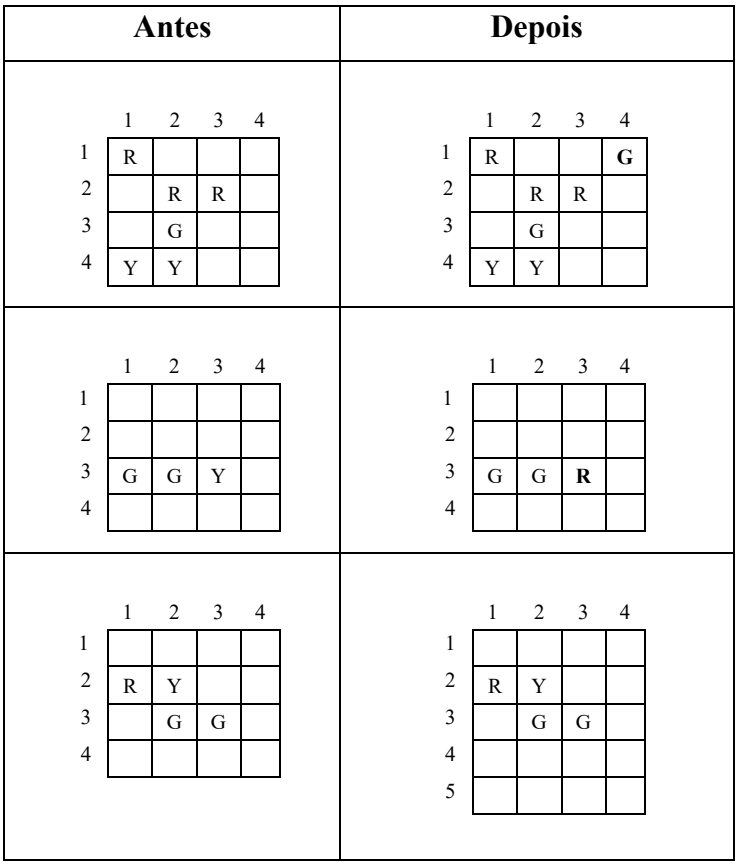

Figura 1

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. Implementação Base

Esta funcionalidade deve permitir efetuar a gestão de um jogo simples entre duas pessoas, identificadas como jogador A e jogador B.

Início do Jogo

A dimensão do tabuleiro é definida no início do jogo. O tabuleiro inicial deve ser quadrado e ter dimensão entre 3 e 5 linhas. O valor é selecionado aleatoriamente. O jogador A é sempre o primeiro a jogar.

Desenrolar do Jogo

Durante a realização do jogo, o tabuleiro deve ser mantido numa estrutura dinâmica, que pode ser **uma tabela alocada dinamicamente ou uma variante de uma lista ligada**. O programa deve receber e validar as jogadas alternadas dos 2 jogadores, efetuando a respetiva atualização do tabuleiro. O programa deve ainda detetar o final do jogo, anunciando o vencedor ou o empate.

Manutenção das Jogadas Realizadas
O programa deve manter informação sobre as jogadas que forem sendo realizadas ao longo de um jogo. Esta informação deve ser mantida numa **estrutura dinâmica do tipo lista ligada**. Antes de efetuar o seu movimento, cada jogador pode pedir para visualizar o estado do tabuleiro nas K jogadas anteriores. Por exemplo, se o utilizador indicar K=3, o programa apresenta na consola a sucessão de estados do tabuleiro nas 3 últimas jogadas.

Exportação para Ficheiro
No final do jogo, a sucessão de estados do tabuleiro 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 estrutura dinâmica que mantém as jogadas realizadas.

3.2. 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*", toda a informação relevante que permita retomar o jogo numa altura posterior. 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.

3.3. Criação de um jogador Automático
Esta funcionalidade permite a um jogador humano (jogador A) realizar um jogo contra o programa. Para isso deve ser implementada uma função que simule as escolhas do jogador B. Nesta opção 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.

4. Código Disponibilizado

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.

5. 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 2020/2021.

Data provisória para entrega do trabalho prático: **23.59** do dia **13 de Junho de 2021**.

Material a entregar:

- Entregar através do Moodle um ficheiro compactado em formato ZIP, contendo o relatório, o código fonte comentado 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:

- Apresentação das principais estruturas de dados, justificando as escolhas feitas
- Apresentação detalhada das estruturas dinâmicas implementadas
- Justificação para as opções tomadas em termos de implementação
- Pequeno manual de utilização.

Avaliação

A cotação do trabalho é de **10 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