Uma imagem com texto, quadro-preto

Descrição gerada automaticamente

**Jogo do Galo**

**Ano Letivo:** 2021/2022

**Curso:** Engenharia Informática – 1º Ano, 2º Semestre

**Unidade Curricular:** Programação

**Discente:** Marco Costa

**Índice**

[**Introdução 3**](#_Toc106570885)

[**1. Manual de Utilização 4**](#_Toc106570886)

[1.1. Jogador vs. PC 5](#_Toc106570887)

[1.2. Jogador vs. Jogador 7](#_Toc106570888)

[1.3. Carregar último jogo 7](#_Toc106570889)

[1.4. Verificação de condições 8](#_Toc106570890)

[**2. Estruturas dinâmicas e organização dos ficheiros 9**](#_Toc106570891)

[2.1. Jogadas 9](#_Toc106570892)

[2.2. *Board* 9](#_Toc106570893)

[2.3. Ficheiros 9](#_Toc106570894)

[**3. Justificação do método de implementação utilizado 10**](#_Toc106570895)

[**Conclusão 10**](#_Toc106570896)

# 

# Introdução

Este trabalho foi proposto no âmbito da unidade curricular de Programação, do 2º semestre do 1º ano da Licenciatura de Engenharia Informática do Instituto Superior de Engenharia de Coimbra.

Tendo-me sido proposto o desenvolvimento de uma adaptação digital do famoso jogo de tabuleiro, Jogo do Galo. Esta será uma versão para 2 utilizadores em que efetuam jogadas de forma alternada, até que um deles vença ou que se verifique um empate. O jogo foi desenvolvido no programa *Code Blocks*.

O presente relatório apresenta-se estruturado em 3 partes. Numa primeira fase é realizada uma apresentação global do jogo desenvolvido.

Segundamente, serão apresentadas as estruturas dinâmicas implementadas e a organização dos ficheiros utilizados pelo programa, onde será justificado o motivo das escolhas feitas.

Por fim, no último tópico, serão justificadas as decisões tomadas no que diz respeito à implementação, nomeadamente na definição de algumas regras do jogo.

# Manual de Utilização

O programa apresenta uma utilização muito simples e intuitiva. Ao iniciar o programa somos, de imediato, apresentados com uma regra especial do jogo, e com o pedido de seleção do modo de jogo desejado. Para realizar a seleção basta digitar 1,2 ou 3 tendo em conta o modo que o jogador pretende jogar, uma vez que a cada número corresponde um modo de jogo diferente.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Introdução do Programa \_Fonte própria

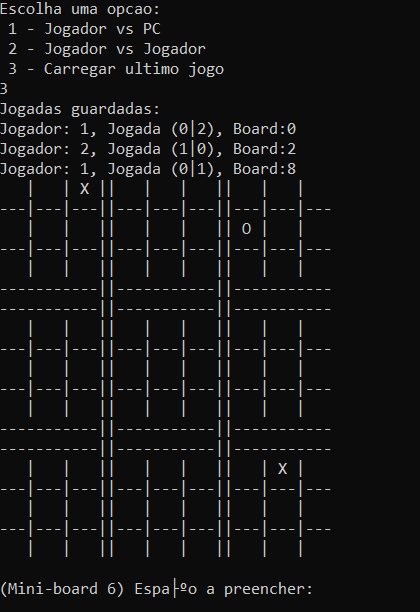
 Consoante a escolha, o programa irá mostrar um novo ecrã. No entanto, as diferenças, entre cada um dos modos oferecidos, são mínimas. Visualmente, a única opção que apresenta algumas mudanças é o modo 3, que corresponde ao continuar do último jogo que não obteve um vencedor ou um empate para a partida, ao ir buscar os dados do jogo anterior que se encontram guardados num ficheiro, apresentando-os ao atualizar a tabela principal.

Figura - Tabela com lugares preenchidos\_ Fonte própria

Uma imagem com texto

Descrição gerada automaticamente

Figura - Tabela de Jogo\_ Fonte Própria

De seguida, serão apresentadas as características correspondentes a cada um dos diversos modos disponíveis.

## Jogador vs. PC

Na primeira opção de modo de jogo, o jogador defronta-se contra o próprio computador.

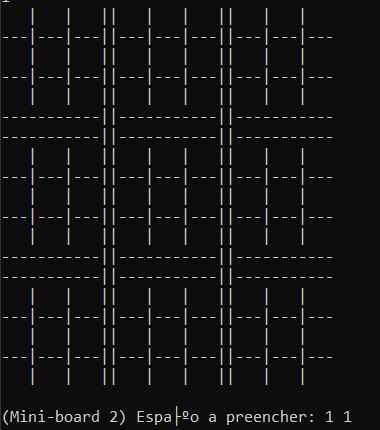
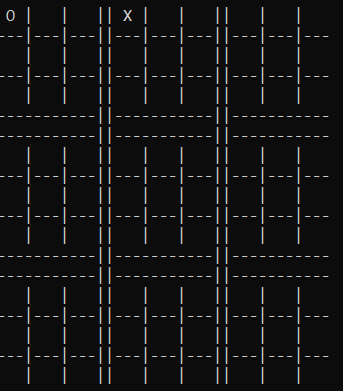
A minitabela inicial, é eleita pelo programa. Para fazer a eleição, o programa gera um número aleatório entre 1 e 10, número esse que será correspondente à minitabela em que o jogo terá o seu início. A partir daqui cada minitabela será definida consoante as jogadas dos jogadores. Antes de cada jogada, o programa pede ao utilizador qual o espaço que deseja preencher.

Figura - Input coordenadas do jogador\_ Fonte Própria

Quando é a vez do computador jogar, o mesmo sorteia novamente um número aleatório, no entanto, desta vez entre 1 e 09, de modo a eleger o espaço onde irá realizar a sua jogada. Posteriormente, deverá verificar se o espaço correspondente ao número do sorteio se já se encontra preenchido ou não. Caso se verifique que não está preenchido o computador pode proceder á jogada, em caso contrário, o computador deverá iniciar novamente o processo de seleção, até obter um espaço livre.

Figura - Tabela com lugares preenchidos\_ Fonte própria

No fim de cada ronda, o programa pergunta ao utilizador, se pretende ter acesso às últimas jogadas realizadas por ambos os jogadores, com um limite máximo de 10 movimentos. Isto é possível, uma vez que todas as jogadas, durante o decorrer do jogo são introduzidas e armazenadas numa lista ligada.

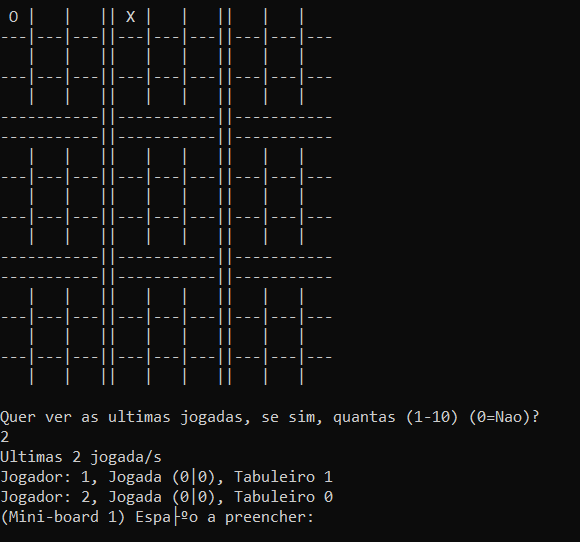
A partir deste ponto, o jogo procede de forma normal. Durante a sua execução, o programa tem de verificar algumas condições com o intuito de funcionar corretamente, para mais informações aconselha-se a leitura do ponto 1.4 do presente relatório.

Figura - Últimas x jogadas\_ Fonte Própria

## Jogador vs. Jogador

O modo de jogo Jogador vs. Jogador, é de forma geral igual ao modo anteriormente apresentado, com a diferença de que agora existem dois jogadores reais a jogar. Não existindo nenhuma alteração efetuada pelo computador.

## Carregar último jogo

Neste modo, o jogo acede ao ficheiro onde estão guardados os dados do último jogo, e recria a tabela principal de acordo com as jogadas feitas.

Entre os dados guardados encontram-se informações como a correspondência entre jogada e jogador e a localização onde foram realizadas as jogadas.

Os dados são guardados de forma automática no fim de cada ronda, por isso, mesmo que o utilizador feche o jogo durante uma partida, dá para recuperar todas as jogadas e continuar o jogo.

## Verificação de condições

Para o bom funcionamento do programa é necessário verificar algumas condições sendo que as principais são:

* verificar se a minitabela escolhida já tem vencedor;
* verificar se algum jogador já ganhou;
* verificar se o espaço a preencher por um dos utilizadores está vazio.

**Verificar se a minitabela escolhida já tem vencedor:**

Para uma boa gestão do tabuleiro, o computador, deverá ter em atenção o seguinte: a existência de um *array* principal de estruturas, ou seja, a tabela maior. Cada estrutura contém uma minitabela e um campo para preencher com o vencedor dessa minitabela.

O programa faz um *loop* do *array* até encontrar a minitabela escolhida, quando encontra, acede ao campo do vencedor e verifica se está ou não vazio.

**Verificar se algum jogador já ganhou:**

Nesta fase, o programa faz um *loop* do *array*, e verifica todas as combinações para ganhar um jogo normal, verificando sempre se existe um vencedor em cada minitabela.

Se o programa encontra um vencedor, todas as jogadas são guardadas num ficheiro denominado pelo utilizador.

**Verificar se o espaço a preencher por um dos utilizadores, está vazio.**

O programa nesta etapa verifica se o espaço das coordenadas que o jogador indicou está vazio. Se estiver então esse lugar fica com a letra do jogador correspondente, senão o programa pede novas coordenadas.

# Estruturas dinâmicas e organização dos ficheiros

Neste capítulo será feita uma breve apresentação das diversas estruturas de dados presentes no programa, juntamente com uma breve justificação para a criação das mesmas.

## Jogadas

Esta estrutura dinâmica é responsável por armazenar todas as jogadas desde o início ao fim do jogo. Esta lista é criada logo no início do programa, e guarda as informações logo após uma jogada de um jogador.

As informações armazenadas são:

* Jogador, valor inteiro, que varia entre 1 e 2;
* coordenadas (x,y) da jogada, dois valores inteiros que variam entre 1 e 3;
* mini tabuleiro onde decorreu a jogada, um valor inteiro que varia entre 1 e 10.

No fim de cada ronda, esta lista é armazenada em formato binário para o ficheiro *jogo.bin*, sendo, assim, possível retomar o jogo mais tarde.

## *Board*

Esta estrutura é essencial para o bom funcionamento do jogo, pois guarda toda a informação das tabelas.

Esta estrutura tem os seguintes componentes:

* mini-*board*, um *array* bidimensional que guarda as informações de cada minitabela;
* *winner*, um *single char* que guarda a letra do vencedor nesta minitabela.

No início do jogo é então criado um *array* dinamicamente alocado contendo 9 destas estruturas, sendo assim possível formar a tabela principal do jogo.

## Ficheiros

***main.c***

Ficheiro principal do programa que contém as condições relativas aos vários modos de jogo, servindo como base de controlo para a inicialização do mesmo.

***game.c***

Ficheiro que contem todas as funções para o bom funcionamento do programa.

***game.h***

Ficheiro *header* que contem os protótipos das funções, estruturas e algumas variáveis.

***Jogo.bin***

Ficheiro que guarda toda a informação da lista ligada em binário.

Existe ainda o ficheiro que guarda toda a informação da lista ligada legível ao utilizador.

# Justificação do método de implementação utilizado

A primeira grande decisão que implementei neste trabalho prático foi a estrutura *Board* e maneira como ele trabalha. Visto que existem variadíssimas maneiras de fazer uma tabela, para a versão *ultimate* do jogo do galo, eu queria uma simples e eficaz.

Decidi então criar uma estrutura que representa um mini tabuleiro, e criando um *array* de 9 destas estruturas tinha o tabuleiro principal.

Outro problema, que poderia acontecer ao longo do desenvolvimento do jogo é o programa eleger um mini tabuleiro onde já existe um vencedor, algo que de todo não poderia acontecer, portanto seria crucial arranjar estratégias para o evitar, por isso, sempre que isto acontece é deverá ser escolhida uma nova minitabela, que não tenha vencedor.

# Conclusão

Olhando para o início deste projeto, para os objetivos a que nos propusemos alcançar e para a forma como foi desenvolvida a proposta de trabalho desta unidade curricular, constatei uma evolução substancial na maneira de encarar e resolução dos problemas, algo que acredito ter contribuído em muito para a minha preparação como futuro engenheiro.

A familiarização com o *Code Blocks* a um nível mais prática permitiu-me adquirir bastantes conhecimentos sobre o programa, nomeadamente a forma como o mesmo funciona e para que pode ser utilizado no futuro. O *Code Blocks* revelou-se uma ferramenta bastante poderosa, com uma vasta área de aplicação e implementação no mercado.

Proporcionou-me também alargar os conhecimentos em linguagem C.