

# Índice

[Índice 3](#_Toc71826291)

[Introdução 4](#_Toc71826292)

[Estruturas de Dados 5](#_Toc71826293)

[Estrutura para guardar as informações de uma jogada 5](#_Toc71826294)

[Estrutura para guardar as informações sobre um jogador 6](#_Toc71826295)

[Array dinâmico para guardar o tabuleiro do jogo 6](#_Toc71826296)

[Estruturas Dinâmicas 7](#_Toc71826297)

[Implementações 8](#_Toc71826298)

[Manual de Utilização 9](#_Toc71826299)

[Colocar uma peça 10](#_Toc71826300)

[Utilizar uma habilidade 10](#_Toc71826301)

[Colocar uma pedra 11](#_Toc71826302)

[Expandir o tabuleiro 11](#_Toc71826303)

[Visualizar as K jogadas anteriores 12](#_Toc71826304)

[Sair do jogo 12](#_Toc71826305)

[Conclusão 13](#_Toc71826306)

# Introdução

O trabalho prático de Programação consiste na criação de um jogo de tabuleiro na linguagem de programação C.

O jogo implementado é o **Jogo do Semáforo** em que dois jogadores efetuam uma jogada por turnos. Ganha o jogo o jogador que conseguir formar uma linha, coluna ou diagonal com peças todas da mesma cor.

Vários temas abordados nesta cadeira foram bastante úteis para a criação deste jogo nomeadamente: ponteiros, manipulação de ficheiros para guardar o jogo para ser retomado mais tarde, listas ligadas para armazenar a sucessão do tabuleiro e do jogo, funções recursivas e alocamento dinâmico de memória.

Neste relatório serão abordados alguns pontos essenciais e a minha estratégia desde o dia zero até ao momento em que escrevo estas palavras.



# Estruturas de Dados

Neste trabalho foram criadas 3 estruturas de dados para armazenar informação ao longo do jogo. Essas estruturas são as seguintes:

* Estrutura que guarda as informações de uma jogada
* Estrutura que guarda as informações sobre um jogador
* Array dinâmico para guardar o tabuleiro do jogo

Todas estas estruturas são fundamentais para o bom desempenho do jogo sendo que duas delas são alocadas dinamicamente (a estrutura tab e o array tabuleiro).

## Estrutura para guardar as informações de uma jogada

*tab*

Uma imagem com texto

Descrição gerada automaticamente

## Estrutura para guardar as informações sobre um jogador

*Jogador*

Uma imagem com texto

Descrição gerada automaticamente

## Array dinâmico para guardar o tabuleiro do jogo

*tabuleiro*

Uma imagem com texto

Descrição gerada automaticamente

# Estruturas Dinâmicas

asd

# Implementações

## Gerar o tabuleiro de jogo

Para gerar o tabuleiro do jogo existe a função gerarTabuleiro que tem como parâmetros o número de linhas e colunas com que o tabuleiro irá ser inicializado.

Primeiro é criado um ponteiro para armazenar o tabuleiro (array linearizado) e logo depois é alocada memória para as linhas e colunas. No final, a função preenche os espaços em vazio com underscores ( \_ ) e retorna um ponteiro para o início do tabuleiro.

## Uma imagem com texto Descrição gerada automaticamente

## Verificar se a jogada vai surtir numa vitória/empate

Sempre que uma peça é jogada, a função jogadaVencedora é chamada para verificar se a peça colocada vai fazer com que alguém ganhe o jogo ou acabe num empate. Os parâmetros da função são o ponteiro para o início do tabuleiro, a cor da peça colocada, a posição onde a peça foi colocada, o número de linhas e colunas totais do tabuleiro e por fim, o jogador que efetuou a jogada.

Uma imagem com texto

Descrição gerada automaticamente

A primeira parte da função verifica caso a caso, ou seja, verifica se o jogador ganhou:

* Na horizontal (corre a linha onde a peça foi colocada e verifica se as peças são todas da mesma cor)
* Na vertical (corre a coluna onde a peça foi colocada e verifica se as peças são todas da mesma cor)
* Numa das diagonais (corre para a diagonal principal ([0, 0] [1, 1] [2, 2] ...) e corre na diagonal secundária ([1, 3] [2, 3] [3, 1] ...))

Para além de verificar se alguém ganhou, a função verifica também pela situação de empate correndo o tabuleiro inteiro observando se não há espaços vazios por preencher.

## Colocar uma peça no tabuleiro

A função para colocar uma peça no tabuleiro, para além de colocar a peça, verifica também se o jogador introduziu uma posição válida.

A função é colocarPeca composta por um *switch-case* que consoante a peça que o jogador quer jogar verifica se essa peça pode ser colocada no sítio desejado.

Uma imagem com texto

Descrição gerada automaticamente

## Expandir o tabuleiro

Podemos expandir o tabuleiro graças à função expandirTabuleiro que recebe como parâmetros um ponteiro para o início do tabuleiro, o tamanho do tabuleiro (com o novo tamanho) e o tipo de expansão.

Uma imagem com texto

Descrição gerada automaticamente

Esta função é bastante idêntica à função gerarTabuleiro só que em vez de usarmos a função MALLOC, usamos a função REALLOC para realocar memória no tabuleiro dinamicamente alocado. A função retorna um ponteiro para o início do tabuleiro expandido.

## Guardar o jogo para retomar mais tarde

Esta função é chamada quando um jogador decide terminar o jogo e guardá-lo para retomar noutra altura. A função guardaJogo recebe como parâmetros um ponteiro para o início do tabuleiro, um ponteiro para a estrutura que guarda as informações do jogador A e um ponteiro para a estrutura que guarda as informações do jogador B.

Uma imagem com texto

Descrição gerada automaticamente

Começa-se por criar o ficheiro jogo.bin e depois abre-se o ficheiro em modo de escrita binária. Em primeiro lugar escreve-se as estruturas dos jogadores e só depois é que cada elemento da lista ligada é escrito. Isto inclui escrever o tabuleiro em todas as jogadas efetuadas até ao momento no ficheiro binário.

## Recuperar um jogo guardado

Caso o jogador queira retomar um jogo ele precisa de ter o ficheiro jogo.bin que foi gerado no momento em que esse jogo foi interrompido. Esse ficheiro tem obrigatoriamente que estar na mesma pasta do jogo.

## Uma imagem com texto Descrição gerada automaticamente

A função começa por criar duas variáveis, a variável **p** que é a lista ligada e a variável **novo** que vai servir para armazenar os valores que vão sendo lidos durante o ciclo *while*. Logo após isso, o ficheiro é aberto em modo de leitura binária e é gerado um tabuleiro com dimensão máxima para poder receber qualquer tamanho de tabuleiro que possa estar guardado.

A leitura é feita pela mesma ordem da escrita. Primeiro é lido a estrutura referente ao jogador e depois é lida a estrutura referente ao jogador B.

O último passo é ler cada elemento da lista ligada que esteja guardado no ficheiro. Para cada elemento lido é chamada a função insere\_final que insere um novo elemento ao final da lista ligada.

A função retorna um ponteiro para o início da lista ligada preenchida com os elementos que estavam guardados no ficheiro binário.

## Inserir um elemento no fim da lista ligada

Os parâmetros da função insere\_final são um ponteiro para o início da lista ligada, um ponteiro para o início do tabuleiro, o número de linhas e colunas totais, o número da jogada em que a função foi chamada, o jogador que efetuou a jogada, a cor da peça jogada e a posição onde essa peça foi jogada.

Uma imagem com texto

Descrição gerada automaticamente

Com essa informação toda, há que alocar memória para o novo elemento que irá ser introduzido no final da lista ligada, por esse motivo, a função MALLOC é chamada.

A função preenche trata de preencher o novo elemento com a informação passada nos parâmetros e o ciclo *while* trata de chegar ao final da lista ligada para que o novo elemento possa ser adicionado nessa posição. Para finalizar, a função retorna um ponteiro para o início da lista ligada com o novo elemento inserido no final da mesma.

# Manual de Utilização

O programa conta com uma interface amigável e intuitiva que apresenta sempre todas as opções possíveis numa linguagem percetível.

Ao iniciar o jogo, o jogador deparar-se-á com este menu principal:

Uma imagem com texto

Descrição gerada automaticamente

No menu principal existem 4 opções:

* Opção 1 – Iniciar um jogo contra um jogador humano
* Opção 2 – Iniciar um jogo contra o computador
* Opção 3 – Ver as instruções do jogo
* Opção 4 – Sair do jogo

O jogo começa sempre com um tabuleiro quadrado, vazio e de dimensão aleatória (entre 3 e 5). O jogador A é sempre o primeiro a jogar.

Uma imagem com texto

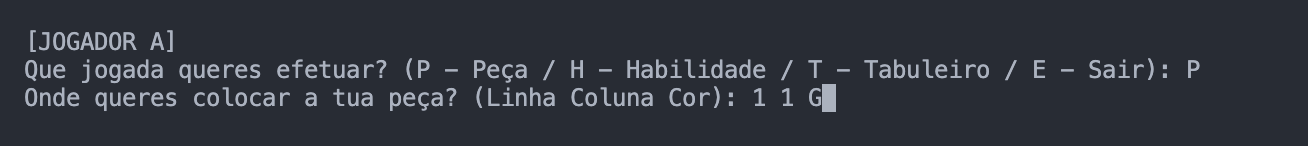
Descrição gerada automaticamente

Neste momento, o jogador é confrontado com 4 possibilidades:

* P – Colocar uma peça colorida no tabuleiro
* H – Utilizar uma habilidade especial
* T – Ver o estado do tabuleiro nas K jogadas anteriores
* E – Sair do jogo com a possibilidade de o retomar mais tarde

## Colocar uma peça

No caso de o jogador ter escolhido colocar uma peça no tabuleiro, agora terá de escolher onde quer colocar essa peça. O programa tratará de verificar se a jogada é válida e se isso se verificar então a peça é colocada.



*Neste exemplo o jogador escolheu colocar uma peça verde (G) na posição (1, 1) do tabuleiro.*

Após isso, o tabuleiro é atualizado e é dada a oportunidade de o jogador B efetuar uma jogada. O ciclo continua daí em diante.

Uma imagem com texto

Descrição gerada automaticamente

## Utilizar uma habilidade

O jogador pode escolher utilizar uma das duas habilidades disponíveis que são: colocar uma pedra num espaço vazio ou expandir o tabuleiro numa direção (linha ou coluna).

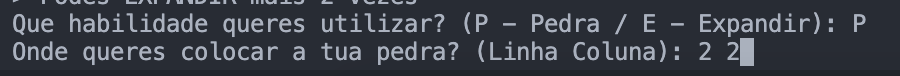
Cada jogador pode colocar 1 pedra por jogo e expandir o tabuleiro 2 vezes.

Uma imagem com texto

Descrição gerada automaticamente

### Colocar uma pedra

Ao escolher colocar uma pedra, o jogador tem de especificar onde a quer colocar indicando assim uma posição.



*Neste exemplo o jogador escolheu colocar uma pedra na posição (2, 2) do tabuleiro.*

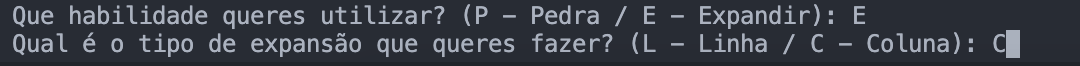
Se o jogador tiver pedras disponíveis para utilizar e se a posição escolhida for válida (ou seja, vazia), então a pedra é colocada e o tabuleiro é atualizado.

***Uma imagem com texto

Descrição gerada automaticamente***

### Expandir o tabuleiro

Uma habilidade especial poderosa é a expansão do tabuleiro. Isto permite que um jogador expanda o tabuleiro na vertical ou na horizontal (apenas expande uma unidade).



*Neste exemplo o jogador expandir as colunas do tabuleiro.*

Se o jogador tiver expansões disponíveis então o tabuleiro é expandido e atualizado.

Uma imagem com texto

Descrição gerada automaticamente

## Visualizar as K jogadas anteriores

Um jogador com memória curta, pode tirar proveito desta funcionalidade para rever o tabuleiro num determinado número de jogadas anteriores.

Para isso, basta indicar quantas jogadas é que quer voltar atrás e o programa tratará de mostrar os tabuleiros desde essa jogada até à jogada atual.

Uma imagem com texto

Descrição gerada automaticamente

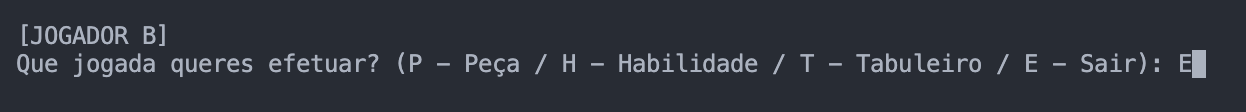
Uma imagem com texto

Descrição gerada automaticamente

*Neste exemplo o jogador escolheu ver as 2 jogadas anteriores.*

## Sair do jogo

O jogo pode ser guardado para ser recuperado mais tarde se os jogadores assim o entenderem. Para isso, basta selecionarem a opção de Sair (E).



Logo de seguida, o jogador terá a oportunidade de escolher se quer ou não guardar o jogo num ficheiro para voltar a jogar mais tarde.

Uma imagem com texto

Descrição gerada automaticamente

Se o jogador tiver escolhido guardar o jogo, será criado um ficheiro chamado jogo.bin que vai ser necessário para retomar o jogo da próxima vez que o programa iniciar.

# Conclusão

Este trabalho prático permitiu-me aplicar os conhecimentos adquiridos nas aulas de Programação e colocar-me à prova relativamente àquilo que eu sabia fazer e àquilo que eu não sabia e desse modo fui obrigado a procurar ou investigar mais sobre determinados assuntos.

Por se tratar de um jogo, algo em mim sempre teve uma grande vontade de fazer este trabalho e aprimorar todos os aspetos do programa. Foi uma oportunidade de juntar tudo o que aprendi nas aulas para construir algo com uma aplicação prática.