Projeto Nº 2: Época Normal

Inteligência Artificial
Escola Superior de Tecnologia de Setúbal
2025/2026

Prof. Joaquim Filipe
Eng. Filipe Mariano

1. Descrição do Jogo - Solitário 2 (versão 2 jogadores)

O Solitário 2 é um jogo de tabuleiro para dois jogadores composto por uma base em forma de cruz, geralmente com 33 cavidades, onde os pinos do `Jogador 1` são colocados nas 6 posições no topo do tabuleiro, enquanto que os pinos do `Jogador 2` são colocados nas 6 posições do fundo do tabuleiro.

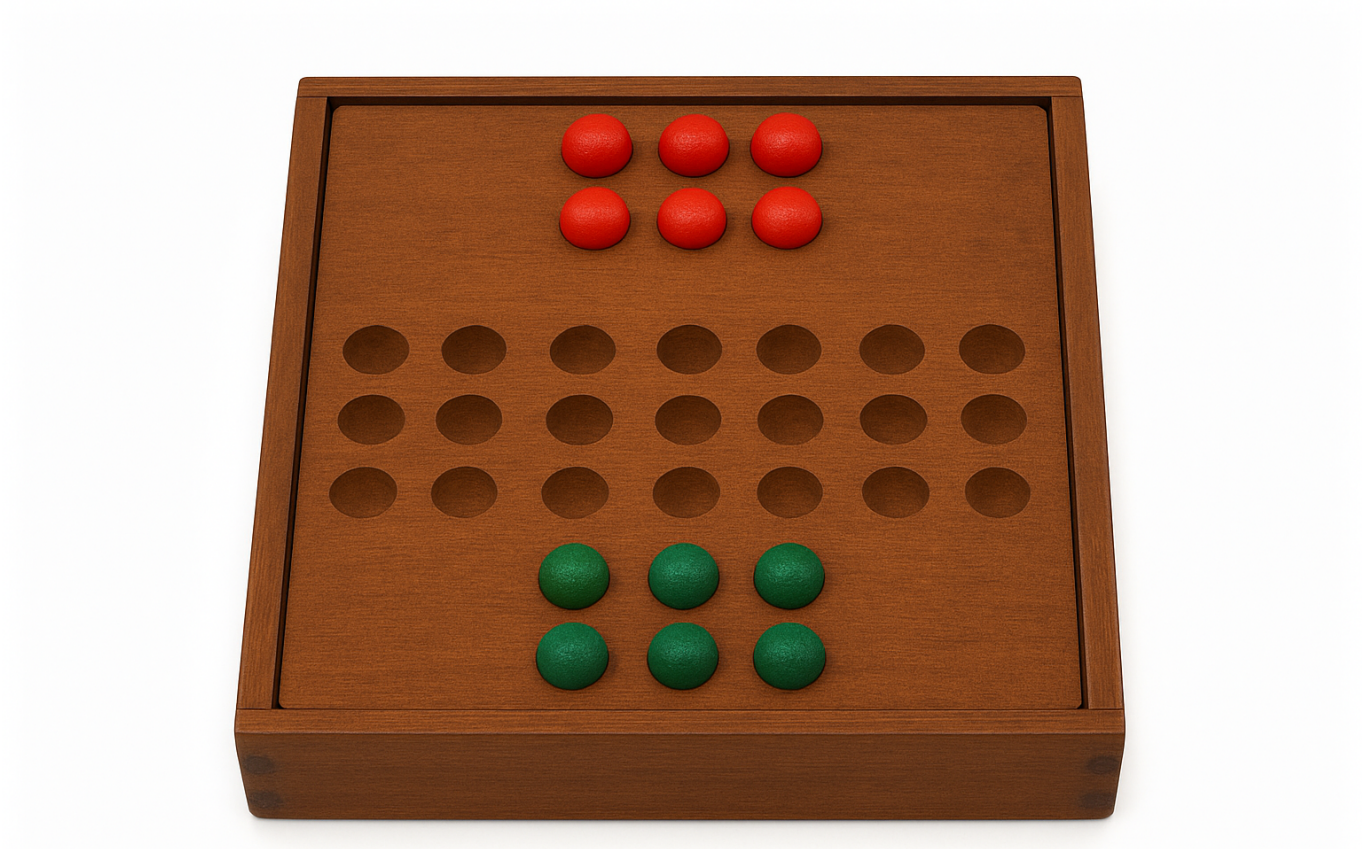

Figura 1: Tabuleiro real do Solitário 2.

O objetivo é chegar primeiro uma peça a uma das casas inicialmente ocupadas pelo adversário, ou seja, para o Jogador 1 vencer terá de colocar uma peça nas casas (6 3), (6 4), (6 5), (7 3), (7 4) ou (7 5), enquanto que para o Jogador 2 vencer terá de colocar uma das suas peças nas casas (1 3), (1 4), (1 5), (2 3), (2 4) ou (2 5). Para tal, cada jogador alternadamente move uma das suas peças utilizando quaisquer um dos 4 operadores anteriormente conhecidos captura-direita (cd), captura-esquerda (ce), captura-cima (cc), captura-baixo (cb), assim como 4 novos operadores que são:

- `d`: Move uma peça para uma casa para a direita

- e: Move uma peça para uma casa para a esquerda
- c: Move uma peça para uma casa para cima
- b: Move uma peça para uma casa para baixo

Nota: Para realizar algum destes movimentos a casa de destino terá de estar vazia ou estar dentro dos limites do tabuleiro.

O tabuleiro pode ser visto como uma matriz 7x7 (7 linhas e 7 colunas) onde ● = pino do Jogador 1, ■ = pino do Jogador 2, ○ = casa vazia e (espaço) = posição inválida fora da cruz.

```
   1 2 3 4 5 6 7
1      ● ● ●
2      ● ● ●
3  ○ ○ ○ ○ ○ ○ ○
4  ○ ○ ○ ○ ○ ○ ○
5  ○ ○ ○ ○ ○ ○ ○
6      ■ ■ ■
7      ■ ■ ■
```

Desta forma, pode se constatar que as seguintes posições fora da cruz são inválidas (`(x,y)` - coordenada da casa no tabuleiro em que `x` representa a linha e `y` a coluna):

- Linha 1 → (1,1), (1,2), (1,6), (1,7)
- Linha 2 → (2,1), (2,2), (2,6), (2,7)
- Linha 6 → (6,1), (6,2), (6,6), (6,7)
- Linha 7 → (7,1), (7,2), (7,6), (7,7)

1.1. Regras

O jogo desenrola-se da seguinte forma:

- O Jogador 1 começa por mover uma das suas peças, sendo que na primeira jogada apenas será válido utilizar o movimento b (baixo), movendo uma das suas 3 peças na 2ªlinha para a casa imediatamente a baixo.
- O Jogador 2 de seguida move uma das suas peças, sendo que na primeira jogada apenas será válido utilizar o movimento c (cima), movendo uma das suas 3 peças na 6ªlinha para a casa imediatamente a cima.
- De seguida volta a jogar o Jogador 1 e alternadamente com o Jogador 2 vão movendo as suas peças até que algum dos jogadores consiga mover uma das suas peças para uma das casas que originalmente possuia uma das suas peças antes do jogo iniciar.
- É possível realizar os movimentos de captura (cd, ce, cc e cb) se a peça a ser capturada for do adversário.

1.2. Exemplo de Jogada

Para exemplificar uma jogada no puzzle, podemos observar o tabuleiro como uma matriz 7x7 onde ● = pino, ■ = pino do Jogador 2, ○ = casa vazia e (espaço) = posição inválida fora da cruz. Na representação em baixo as linhas e colunas estão numeradas para facilitar a descrição da jogada.

Jogada exemplo 1: Jogador 1 move o pino da posição (2,3) para baixo `(b 2 3)`

Tabuleiro exemplo

```
   1 2 3 4 5 6 7  ← colunas
1      ● ● ●
2      ● ● ●
3  ○ ○ ○ ○ ○ ○ ○
4  ○ ○ ○ ○ ○ ○ ○
5  ○ ○ ○ ○ ○ ○ ○
6      ■ ■ ■
7      ■ ■ ■
```

Passos:

1. **Escolher o pino que vai mover**: o pino em (2,3) (linha 2, coluna 3).
2. **Verificar a casa de destino**: (3,3) é ○ (vazia) — pode receber o pino.
3. **Executar o movimento**:
    - O pino de (2,3) move-se para (3,3).
    - A casa (2,3) fica vazia.

Tabuleiro após a jogada

```
   1 2 3 4 5 6 7
1      ● ● ●
2      ○ ● ●
3  ○ ○ ● ○ ○ ○ ○
4  ○ ○ ○ ○ ○ ○ ○
5  ○ ○ ○ ○ ○ ○ ○
6      ■ ■ ■
7      ■ ■ ■
```

Jogada exemplo 2: Jogador 2 faz o movimento captura-cima no pino da posição (5,3) `(cc 5 3)`

Tabuleiro exemplo

```
   1 2 3 4 5 6 7
1        ● ● ●
2        ○ ● ●
3  ○ ○ ○ ○ ○ ○ ○
4  ○ ○ ● ○ ○ ○ ○
5  ○ ○ ■ ○ ○ ○ ○
6        ○ ■ ■
7        ■ ■ ■
```

Passos:

1. **Escolher o pino que vai mover**: o pino em (5,3) (linha 5, coluna 3).
2. **Verificar a casa intermédia**: entre (5,3) e (3,3) está (4,3); tem lá um pino do **adversário**? Sim (●) — portanto é possível passar por cima.
3. **Verificar a casa de destino**: (3,3) é ○ (vazia) — pode receber o pino.
4. **Executar o movimento**:
    - O pino de (5,3) move-se para (3,3).
    - O pino que estava em (4,3) é retirado do tabuleiro.
    - A casa (5,3) fica vazia.

Tabuleiro após a jogada

```
   1 2 3 4 5 6 7
1        ● ● ●
2        ○ ● ●
3  ○ ○ ■ ○ ○ ○ ○
4  ○ ○ ○ ○ ○ ○ ○
5  ○ ○ ○ ○ ○ ○ ○
6        ○ ■ ■
7        ■ ■ ■
```

2. Objetivo do Projeto

No âmbito deste projeto vamos considerar o Solitário na versão de dois jogadores, possibilitando um enquadramento teórico-prático com os conhecimentos adquiridos no âmbito da Teoria de Jogos. Neste caso aplicam-se as regras de jogo descritas na Secção 1.1.

Tal como na 1ª fase do Projecto, pretende-se que os alunos desenvolvam um programa, em Lisp. O programa deverá implementar o algoritmo AlfaBeta ou Negamax com cortes alfa-beta, e as funções auxiliares que permitirão realizar as partidas do jogo.

Pretende-se que o programa permita ao computador vencer o jogador humano ou um outro computador, pelo que deverá funcionar em dois modos:

1. Humano vs Computador
2. Computador vs Computador

No modo 1, no início de cada partida, o utilizador deve decidir quem começa como Jogador 1 (peças no topo), humano ou computador, e qual o tempo limite para o computador jogar, enquanto que no modo 2 apenas é necessário definir o tempo limite.

3. Formulação do Problema

3.1. Tabuleiro

O tabuleiro é representado sob a forma de uma lista de listas em lisp em que [nil] representa um espaço ilegal no tabuleiro, [0] representa uma casa vazia, [1] representa uma peça do Jogador 1 e [2] uma peça do Jogador 2. Cada uma dessas listas é composta por 7 elementos, cujo valor de cada casa está supramencionado.

```
(
 (nil nil 1 1 1 nil nil)
 (nil nil 1 1 1 nil nil)
 ( 0   0  0 0 0  0   0)
 ( 0   0  0 0 0  0   0)
 ( 0   0  0 0 0  0   0)
 (nil nil 2 2 2 nil nil)
 (nil nil 2 2 2 nil nil)
)
```

3.2. Operadores

Para este problema a estrutura de cada movimento pode ser representado da seguinte forma:

```
(TipoDeMovimento PosiçãoHorizontal PosiçãoVertical)
```

- **PosiçãoHorizontal** e **PosiçãoVertical** indicam as coordenadas da casa de origem do pino a ser movido.
- **TipoDeMovimento** define a direção e o tipo de ação de captura, visto que se pode movimentar na horizontal ou vertical.

Código	Significado	Descrição
d	Direita	O pino move-se uma casa para a direita.
e	Esquerda	O pino move-se uma casa para a esquerda.
c	Cima	O pino move-se uma casa para a cima.

Código	Significado	Descrição
b	Baixo	O pino move-se uma casa para baixo.
cd	Captura Direita	O pino move-se duas casas à direita, passando por cima de outro pino.
ce	Captura Esquerda	O pino move-se duas casas à esquerda, passando por cima de outro pino.
cc	Captura Cima	O pino move-se duas casas para cima, passando por cima de outro pino.
cb	Captura Baixo	O pino move-se duas casas para baixo, passando por cima de outro pino.

3.2.1. Direita — d

Exemplo: **(d 4 2)**
Descrição: O pino da posição (4,2) move-se uma casa para a direita, desde que a casa na posição (4,3) esteja vazia ou sejá válida (dentro dos limites do tabuleiro).

Antes:

```
Linha 4: ○ ● ○ ○ ○ ○ ○
           ↑ ↑
           | |
         (4,2)(4,3)
```

Depois:

```
Linha 4: ○ ○ ● ○ ○ ○ ○
```

3.2.2. Esquerda — e

Exemplo: **(e 4 3)**
Descrição: O pino da posição (4,3) move-se uma casa para a esquerda, desde que a casa na posição (4,2) esteja vazia ou sejá válida (dentro dos limites do tabuleiro).

Antes:

```
Linha 4: ○ ○ ● ○ ○ ○ ○
           ↑ ↑
           | |
         (4,2)(4,3)
```

Depois:

```
Linha 4: ○ ● ○ ○ ○ ○ ○
```

3.2.3. Cima — c

Exemplo: **(c 6 4)**
Descrição: O pino da posição (6,4) move-se uma casa para cima, ocupando a casa (5,4), desde que essa casa esteja vazia ou sejá válida (dentro dos limites do tabuleiro).

Antes:

```
Coluna 4:
 ○
 ○
 ○
 ○
 ○
 ●
 ○
```

Depois:

```
Coluna 4:
 ○
 ○
 ○
 ○
 ●
 ○
 ○
```

3.2.3. Baixo — b

Exemplo: **(b 5 4)**
Descrição: O pino da posição (5,4) move-se uma casa para baixo, ocupando a casa (6,4), desde que essa casa esteja vazia ou sejá válida (dentro dos limites do tabuleiro).

Antes:

```
Coluna 4:
 ○
 ○
 ○
 ○
 ●
```

```
 ○
 ○
```

Depois:

```
Coluna 4:
 ○
 ○
 ○
 ○
 ○
 ●
 ○
```

3.2.5. Captura Direita — cd

Exemplo: **(cd 4 2)**
Descrição: O pino da posição (4,2) move-se duas casas à direita, saltando sobre o pino em (4,3) e ocupando a casa (4,4).

Antes:

```
Linha 4: ● ● ● ○ ● ● ●
           ↑ ↑
           | |
         (4,2)(4,3)
```

Depois:

```
Linha 4: ● ○ ○ ● ● ● ●
```

3.2.6. Captura Esquerda — ce

Exemplo: **(ce 4 6)**
Descrição: O pino da posição (4,6) move-se duas casas à esquerda, saltando sobre o pino em (4,5) e ocupando a casa (4,4).

Antes:

```
Linha 4: ● ● ● ○ ● ● ●
                 ↑ ↑
                 | |
               (4,5)(4,6)
```

Depois:

```
Linha 4: ● ● ● ● ○ ○ ●
```

3.2.7. Captura Cima — cc

Exemplo: **(cc 6 4)**
Descrição: O pino da posição (6,4) move-se duas casas para cima, saltando sobre o pino em (5,4) e ocupando a casa (4,4).

Antes:

```
Coluna 4:
 ●
 ●
 ●
 ○
 ●
 ●
 ●
```

Depois:

```
Coluna 4:
 ●
 ●
 ●
 ●
 ○
 ○
 ●
```

3.2.8. Captura Baixo — cb

Exemplo: **(cb 2 4)**
Descrição: O pino da posição (2,4) move-se duas casas para baixo, saltando sobre o pino em (3,4) e ocupando a casa (4,4).

Antes:

```
Coluna 4:
 ●
```

```
●
●
○
●
●
●
```

Depois:

```
Coluna 4:
 ●
 ○
 ○
 ●
 ●
 ●
 ●
```

3.3. Estrutura do Programa

O programa deverá estar dividido em três partes, cada uma num ficheiro diferente:

1. Uma parte para o algoritmo AlfaBeta ou Negamax (`algoritmo.lisp`).
2. Outra que deve conter as funções que permitem escrever e ler em ficheiros e tratar da interação com o utilizador (`jogo.lisp`).
3. E a terceira parte corresponde aos operadores do jogo (`puzzle.lisp`).

Enquanto que a primeira parte do programa deverá ser genérica para qualquer jogo que recorre ao algoritmo AlfaBeta ou Negamax com cortes alfa-beta (independente do domínio de aplicação), a segunda e a terceira parte são específicas do jogo Solitário 2 (dependente do domínio de aplicação).

Na parte 2 deverá ser definida uma função com o nome `jogar` com os parâmetros `estado` e `tempo`, e que devolva uma lista em que o primeiro elemento é uma lista com o tipo de movimento e as coordenadas da jogada realizada (`(TipoDeMovimento PosiçãoHorizontal PosiçãoVertical)`) e o segundo elemento o novo estado do jogo resultante da aplicação da jogada.

4. Algoritmo

O algoritmo de jogo a implementar é o AlfaBeta ou Negamax com cortes alfa-beta. Como forma de verificação das diversas jogadas realizadas durante um jogo, antes de devolver cada resultado, o programa deverá escrever num ficheiro `log.dat` e no ecrã qual a jogada realizada, o número de nós analisados, o número de cortes efetuados (de cada tipo) e o tempo gasto. Caso tenha sido escolhido o modo **Humano vs Computador**, o programa deverá ler cada jogada do jogador humano inserida através do teclado e repetir o procedimento até a partida terminar.

```
```

```
O tempo limite para o computador jogar será de X milissegundos. O valor de X
deverá ser um valor compreendido entre 1 e 20 segundos que deverá ser fornecido
pelo utilizador do jogo, nomeadamente lido do teclado caso seja um jogo contra um
humano, ou recebido por parâmetro na função jogar, caso seja contra um computador
(vide subsecção Campeonato na secção 8).
```

5. Grupos

Os grupos deverão ser os mesmos que realizaram a 1ª fase do Projeto, seguindo as regras anteriormente estipuladas. Alterações nos grupos deverão ser comunicadas e aprovadas pelos Professores

6. Datas

Entrega do projeto: 16 de Janeiro de 2025, até as 23:59.

7. Documentação a Entregar

A entrega do projeto e da respetiva documentação deverá ser feita através do Moodle, na zona do evento "Entrega do 2º Projeto". Todos os ficheiros a entregar deverão ser devidamente arquivados num ficheiro comprimido (`ZIP` com um tamanho máximo de `5 MB`), até à data acima indicada. O nome do arquivo deve seguir a estrutura `P2_nomeAluno1_numeroAluno1_nomeAluno2_numeroAluno2`.

7.1. Código fonte

Os ficheiros de código devem ser devidamente comentados e organizados da seguinte forma:

jogo.lisp: Carrega os outros ficheiros de código, escreve e lê de ficheiros e trata da interação com o utilizador.

puzzle.lisp: Código relacionado com o problema.

algoritmo.lisp: Deve conter a implementação do algoritmo de jogo independente do domínio.

7.2. Manuais

No âmbito da Unidade Curricular de Inteligência Artificial pretende-se que os alunos pratiquem a escrita de documentos recorrendo à linguagem de marcação ***Markdown***, que é amplamente utilizada para os ficheiros ***ReadMe*** no GitHub. Na Secção 4 do guia de Laboratório nº 2, encontrará toda a informação relativa à estrutura recomendada e sugestões de ferramentas de edição para ***Markdown***.

Para além de entregar os ficheiros de código e de problemas, é necessário elaborar e entregar 2 manuais (o manual de utilizador e o manual técnico), em formato `PDF` juntamente com os sources em `MD`, incluídos no arquivo acima referido:

ManualTecnico.pdf O Manual Técnico deverá conter:

- O algoritmo implementado, devidamente comentado e respetivas funções auxiliares;
- Descrição dos tipos abstratos usados no programa; Identificação das limitações e opções técnicas;
- Uma análise crítica dos resultados das execuções do programa, onde deverá transparecer a compreensão das limitações do projeto;

- Uma análise estatística acerca de uma execução do programa contra um adversário humano, mencionando o limite de tempo usado e, para cada jogada: o respetivo valor, a profundidade do grafo de jogo e o número de cortes efetuado no processo de análise. Poderão utilizar os dados do ficheiro `log.dat` para isso.

ManualUtilizador.pdf O Manual do Utilizador deverá conter:

- A identificação dos objetivos do programa, juntamente com descrição geral do seu funcionamento;
- Explicação da forma como se usa o programa (acompanhada de exemplos) e descrição da informação necessária e da informação produzida (ecrã/teclado e ficheiros);
- Limitações do programa (do ponto de vista do utilizador, de natureza não técnica).

8. Avaliação

Tabela 1: Grelha de classificação.

Funcionalidade	**Valores**
Algoritmo AlfaBeta ou Negamax com Cortes alfa-beta	5
Operadores do jogo	2
Função de Avaliação	2
Jogada Humano	1
Calcular e Apresentar estatísticas a cada jogada (ecrã e ficheiro)	2
Implementação do limite de tempo para o computador	1
Procura quiescente	1
Memoização	1
Ordenação dos nós	1
Qualidade do código	2
Manuais (utilizador e técnico)	2
Total	20

A avaliação do projeto levará em linha de conta os seguintes aspectos:

- **Data de entrega final** – Existe uma tolerância de 3 dias em relação ao prazo de entrega, com a penalização de 1 valor por cada dia de atraso. Findo este período a nota do projeto será 0.
- **Correção processual da entrega do projeto** - (Moodle; manuais no formato correto). Anomalias processuais darão origem a uma penalização que pode ir até 3 valores.
- **Qualidade técnica** - Objetivos atingidos; Código correto; Facilidade de leitura e manutenção do programa; Opções técnicas corretas.
- **Qualidade da documentação** - Estrutura e conteúdo dos manuais que acompanham o projeto.
- **Avaliação oral** - Eficácia e eficiência da exposição; Compreensão das limitações e possibilidades de desenvolvimento do programa. Nesta fase poderá haver lugar a uma revisão total da nota de projeto.

Campeonato

Após a entrega dos projetos, será realizado um campeonato entre os programas de cada grupo (os alunos deverão manifestar a sua intenção de participar, fazendo um registo no Moodle por ocasião da entrega do projeto 2). Para participar, será necessário que:

a) o programa esteja totalmente inserido num package com a designação **p<numeros>** em que **<numeros>::=<numero de um elemento do grupo>-<numero do outro elemento do grupo> | <numero do unico elemento do grupo>**.

b) seja definida uma função com o nome `jogar` com os parâmetros `estado` e `tempo` e que devolva `uma lista com a jogada e com o novo estado`.

Exemplo de input:

```
(jogar
   (;estado
         (nil nil 1 1 1 nil nil)
         (nil nil 1 1 1 nil nil)
         ( 0   0  0 0 0  0   0)
         ( 0   0  0 0 0  0   0)
         ( 0   0  0 0 0  0   0)
         (nil nil 2 2 2 nil nil)
         (nil nil 2 2 2 nil nil)
   )
   5000 ;tempo da jogado em ms
)
```

Exemplo de output:

```
(
   (b 2 3) ;coordenadas da jogada baixo na linha 2 coluna 3
   (
         (nil nil 1 1 1 nil nil)
         (nil nil 0 1 1 nil nil)
         ( 0   0  1 0 0  0   0)
         ( 0   0  0 0 0  0   0)
         ( 0   0  0 0 0  0   0)
         (nil nil 2 2 2 nil nil)
         (nil nil 2 2 2 nil nil)
   )
)
```

Todos os participantes neste campeonato recebem um bónus na nota final do projeto 2, com destaque para os dois primeiros lugares:

a) 3 valores para o grupo vencedor.
b) 2 valores para o grupo que acabar na 2ª posição.

c) 1 valor para os restantes grupos que se inscreverem no campeonato e em que o programa esteja corretamente estruturado para participar.

9. Recomendações Finais

Com este projeto pretende–se motivar o paradigma de programação funcional. A utilização de variáveis globais, de instruções de atribuição do tipo set, setq, setf (somente dentro de *closures*), de ciclos, de funções destrutivas ou de quaisquer funções com efeitos laterais é fortemente desincentivada dado que denota normalmente uma baixa qualidade técnica. Exceptua-se a utilização de `setf` em conjugação com `gethash`. A sequenciação só será permitida no contexto das funções de leitura/escrita.

As únicas excepções permitidas a estas regras poderão ser a utilização da instrução loop para implementar funções de escrita ou leitura em ficheiros.

A utilização de ferramentas como **ChatGPT**, **Copilot** ou outras **é permitida**, desde que apenas como auxílio, e não para gerar o código completo do projeto. Todo o código produzido com a ajuda destas ferramentas deve conter um comentário indicando que foi gerado automaticamente. Grupos que não incluam esta indicação ou em que existam suspeitas de código integralmente produzido por ferramentas de IA poderão ser penalizados na nota final ou até ter o projeto anulado.

ATENÇÃO: Suspeitas confirmadas de plágio serão penalizadas com a anulação de ambos os projetos envolvidos (fonte e destino), e os responsáveis ficam sujeitos à instauração de processo disciplinar.