

# LP1 2024/2025 – Mini Projeto: Radar de Barcos

Pedro Arroz Serra, Daniel Silveira, Giosuè Muratore, Martijn Kuipers

Deadline: 06/04/2025 @ 23:59hrs no deisi-moodle

27-03-2025 v1.3

## 1: Introdução

Na resolução deste projecto deve ser utilizada a Linguagem de Programação C. Para além da correta implementação do objectivo, tenha em conta os seguintes aspetos:

- O código apresentado deve ser bem indentado.
- Tenha em atenção os nomes dados das variáveis, para que sejam indicadores daquilo que as mesmas vão conter.
- Não é permitida a utilização de variáveis globais ou estáticas
- O trabalho deve ser desenvolvido e submetido de forma individual.
- O programa deve ser devidamente comentado incluindo as funções, parametros, variaveis (ver exemplo função fornecida como exemplo)
- O programa tem de verificar todas as situações de dados inválidos

O não cumprimento deste aspectos incorre uma penalização de 50% da nota.

O código deve compilar sem erros ou warnings, neste caso a penalização é 100% e aluno não é avaliado.

Todos os trabalhos serão comparados utilizando um sistema de deteção de plágio. Em caso de plágio o plagiado e plagiante tem uma **penalização de 100%** e não serão avaliados.

Todos os trabalhos com código compilado e não plagiado serão revistos e submetidos a avaliação oral

**Importante:** Entregue apenas o ficheiro *main.c* no moodle no [DEISI](#)

## 2: Objetivo

Desenvolver um programa em C que simule um radar de barcos num mapa de 20 linhas por 80 colunas. O radar capta duas leituras (antes e depois) com **5 barcos identificados pelas letras A, B, C, D e E**.

O programa deverá realizar várias operações usando **funções, arrays e structs**, sem utilizar ponteiros.

### 2.1 Descrição Geral

O programa irá:

1. Ler duas tabelas de radar (matrizes de 20x80) que representam a posição dos barcos **antes e depois**. Cada célula pode conter:
  - '.' (água)
  - 'A', 'B', 'C', 'D' ou 'E' (um dos barcos)

2. Apresentar ao utilizador o seguinte **menu**:

```
1 - Imprimir tabela "antes"
2 - Imprimir tabela "depois"
3 - Consultar as coordenadas de um barco
4 - Listar barcos que se moveram (esquerda, direita, cima, baixo)
5 - Sair
```

> ⚠ Cada opção do menu deverá corresponder obrigatoriamente a uma função separada.

—

## 2.2 Funcionalidades detalhadas

### 2.2.a 1) Imprimir tabela “antes”

Função que imprime no ecrã a tabela da primeira leitura.

### 2.2.b 2) Imprimir tabela “depois”

Função que imprime no ecrã a tabela da segunda leitura.

### 2.2.c 3) Consultar as coordenadas de um barco

- Perguntar qual tabela (1 ou 2) o utilizador deseja consultar.
- Perguntar qual barco (A, B, C, D ou E) deseja localizar.
- Imprimir as coordenadas (linha e coluna) onde o barco se encontra nessa tabela. **(a tabela começa na linha 0 e coluna 0)**

### 2.2.d 4) Listar barcos que se moveram

- Comparar as duas tabelas.
- Para cada barco, verificar se mudou de posição:
  - Se sim, indicar o movimento: **cima**, **baixo**, **esquerda** ou **direita** (movimento de uma célula).
  - Caso o barco não se tenha movido, indicar “não se moveu”.

### 2.2.e 5) Sair

Termina o programa.

—

## 2.3 Regras obrigatórias

- Criar uma função distinta para cada opção do menu.
- Utilizar **struct Barco** obrigatoriamente:

```
1 struct Barco {
2     char id;        // 'A', 'B', 'C', 'D' ou 'E'
3     int linha;
4     int coluna;
5 };
```

## 2.4 Exemplo visual de uma tabela (5x10 para simplificação):

Tabela Antes:

.....
....A....
.....
.C.....E..
....B.D...

Tabela Depois:

.....
....A....
.....
.C.....E..
....B.D...

Neste exemplo, o barco A moveu-se uma célula para a direita.

### 3: Função fornecida: Leitura de tabela do ficheiro

```
1 #include <stdio.h>
2
3 /*
4  * Função para ler uma tabela de um ficheiro e armazená-la numa matriz
5  * bidimensional.
6  * @param tabela      Matriz de caracteres com 20 linhas e 80 colunas,
7  *                    onde os dados do ficheiro serão armazenados.
8  * @param nomeFicheiro Nome do ficheiro a ser lido.
9  */
10 void lerTabelaDeFicheiro(char tabela[20][80], const char *nomeFicheiro) {
11     // Abre o ficheiro para leitura
12     FILE *ficheiro = fopen(nomeFicheiro, "r");
13
14     // Verifica se o ficheiro foi aberto corretamente
15     if (ficheiro == NULL) {
16         printf("Erro ao abrir o ficheiro %s\n", nomeFicheiro);
17         return; // Se não for possível abrir o ficheiro, a função termina
18                 imediatamente
19     }
20     // Percorre a matriz para preencher os 20x80 caracteres com dados do
21     // ficheiro
22     for (int i = 0; i < 20; i++) { // Loop externo para as 20 linhas
23         for (int j = 0; j < 80; j++) { // Loop interno para as 80 colunas
24             char c = fgetc(ficheiro); // Lê um único carácter do ficheiro
25
26             // Se o carácter for uma nova linha, ignora-o e lê o próximo
27             // carácter
28             if (c == '\n') {
29                 c = fgetc(ficheiro);
30             }
31
32             // Armazena o carácter lido na matriz
33             tabela[i][j] = c;
34         }
35     }
36     // Fecha o ficheiro após a leitura
37     fclose(ficheiro);
38 }
```

Exemplo do uso da função:

```
1 char tabelaAntes[20][80];
2 char tabelaDepois[20][80];
3
4 lerTabelaDeFicheiro(tabelaAntes, "tabela_antes.txt");
5 lerTabelaDeFicheiro(tabelaDepois, "tabela_depois.txt");
```

## 4: Dicas

Cria funções auxiliares para modularizar o teu código.

Mantém o código bem organizado e legível.

Podes usar funções como:

```
void imprimirTabela(char tabela[20][80]);
```

```
void consultarCoordenadas(char tabela[20][80]);
```

```
void listarMovimentos(char tabelaAntes[20][80], char tabelaDepois[20][80]);
```

Usa os seguintes printf's:

```
1      printf("\nMenu:\n");
2
3      printf("1) Imprimir tabela 'antes'\n");
4
5      printf("2) Imprimir tabela 'depois'\n");
6
7      printf("3) Consultar as coordenadas de um barco\n");
8
9      printf("4) Listar barcos que se moveram\n");
10
11     printf("5) Sair\n");
12
13     printf("Escolha uma opcao: ");
14
15     printf("Saindo...\n");
16
17     printf("Opcao invalida. Tente novamente.\n");
18
19     printf("%c", tabela[i][j]);
20
21     printf("Digite a letra do barco (A, B, C, D, E): ");
22
23     printf("Linha %d, Coluna %d\n", barcos[i].id, barcos[i].linha,
barcos[i].coluna);
24
25     printf("Barco nao encontrado!\n");
26
27     printf("%c: Nao se moveu\n", antes[i].id);
28
29     printf("%c: Moveu para cima\n", antes[i].id);
30
31     printf("%c: Moveu para baixo\n", antes[i].id);
32
33     printf("%c: Moveu para esquerda\n", antes[i].id);
34
35     printf("%c: Moveu para direita\n", antes[i].id);
```





```
Menu:
1) Imprimir tabela 'antes'
2) Imprimir tabela 'depois'
3) Consultar as coordenadas de um barco
4) Listar barcos que se moveram
5) Sair
Escolha uma opcao: 4
A: Moveu para esquerda
B: Nao se moveu
C: Nao se moveu
D: Moveu para baixo
D: Moveu para direita
E: Nao se moveu

Menu:
1) Imprimir tabela 'antes'
2) Imprimir tabela 'depois'
3) Consultar as coordenadas de um barco
4) Listar barcos que se moveram
5) Sair
Escolha uma opcao: 5
Saindo...

Process finished with exit code 0
```