



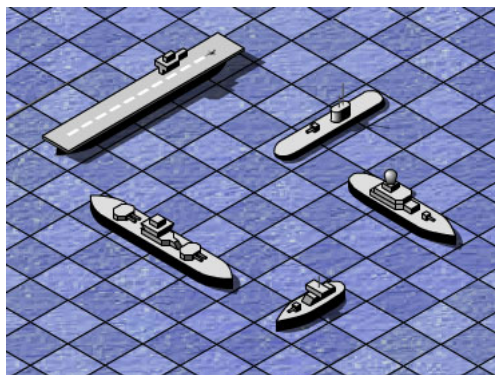
## Batalha Naval

Evan Sá  
up201707418

Margarida Costa  
up201706103

3 de Junho de 2020

Laboratório de Programação  
Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos e  
Licenciatura em Ciência de Computadores



# 1 Introdução

Este trabalho tem como objetivo a implementação do jogo Batalha Naval na linguagem C. Para isso foi feito um estudo sobre as regras do jogo no qual nos baseamos para a realização do mesmo, sendo estas as seguintes:

- Cada jogador tem o mesmo número e tipo de navios, variando consoante o tamanho do tabuleiro de jogo (definido no início);
- Cada jogador define o seu modo de definição e inserção do navio no tabuleiro(manual, random ou ambos);
- Para o início da partida, é necessário que ambos os jogadores escolham um número, o que escolher o maior, será o primeiro a jogar;
- Depois de iniciada a partida, cada jogador à vez, deve escolher uma posição para acertar num navio do outro. Independentemente do jogador acertar ou não em uma posição de um navio, passa a vez ao adversário;
- O jogo termina, quando um dos jogadores tiver os seus navios todos afundados.

## 2 Implementação

Para o funcionamento do jogo foram implementadas duas estruturas de dados: **matriz** e **quadtree**.

### 2.1 Estruturas de dados

Ambas as estruturas de dados a serem apresentadas foram criadas de modo a poderem receber qualquer **tipo** (tipos novos concebidos para um propósito ou pré-definidos da linguagem C). Neste caso as duas estruturas guardam nas suas posições um valor do tipo Cell (no código abaixo, as variáveis ***data*** representam o tipo **Cell**).

#### 2.1.1 MATRIX

```
typedef struct matrix_ Matrix;

struct matrix_
{
    int size;
    void ***data;
};
```

### 2.1.2 QUADTREE

```
typedef enum {NODELEAF, NODEFATHER} nodeType;
typedef enum {FIRST, SECOND, THIRD, FOURTH} quadChilds;

typedef struct Quad_ QuadTree;
typedef struct QuadPoint_ QuadPoint;
typedef struct QuadChilds_ QuadChilds;

#define BOUNDARY 1;

struct QuadPoint_ {

    Point *position;
    void *data;

};

struct Quad_ {

    Point *middlePoint;
    int dimension;

    nodeType type;

    union {
        QuadTree **children;
        QuadPoint *data;
    } contentNode;

};
```

## 2.2 Bibliotecas usadas da linguagem C

As bibliotecas utilizadas foram as seguintes:

- **stdio.h**: define alguns tipos de variáveis, várias macros e várias funções para executar entrada e saída;
- **stdlib.h**: define alguns tipos de variáveis, várias macros e várias funções para executar funções gerais;
- **unistd.h**: define várias constantes e tipos simbólicos e declara funções diversas; esta foi utilizada essencialmente para implementação da função **sleep**. Esta função serve para suspender a execução do código, permitindo assim uma leitura mais perceptível.

## 2.3 Structs

As structs servem para definir, para uma ou mais variáveis, um novo tipo de dados. Para cada struct utilizamos o `typedef struct` para dar um nome à mesma.

- **struct bitmap**\_: define uma matriz de *unsigned char*, com o número de linhas e colunas e também guarda o ponto de referência (refx, refy) de inserção no tabuleiro do jogo;
- **struct ship**\_: define um navio, sendo que este tem um tipo associado. O número de linhas e o número de colunas é definido dependendo do tipo do mesmo. A bitmap inicial(**bp**) é alterada, também pelo tipo de navio, ou seja, o desenho criado pelas linhas e colunas, é retratado com a modificação do valor '0' para '1'. A rotação e a translação escolhida pelo jogador, também influenciará a definição da bitmap. O navio também tem guardado uma variável (*int shotCount* que guarda o número de células que o navio ocupa na bitmap. Cada vez que um jogador acerta numa célula do navio, a variável é decrementada - quando chega a 0, então o barco foi afundado;
- **struct listNode**\_: define qualquer tipo (neste caso o tipo SHIP) e guarda o próximo nó;
- **struct list**\_: define uma lista que tem associada o seu tamanho e a referência do nó inicial(*ListNode head*);
- **struct cell**\_: define uma célula que tem um pointer para o navio, uma variável (*unsigned char value*) que é um ícone ('.', '\*', '+' e '#') apresentado visualmente:
  - '.' : célula do tabuleiro oculta;
  - '\*' : shot acertou numa célula do navio, no entanto o **shotCount** é diferente de 0;
  - '+' : shot acertou numa célula vazia;

Também existe uma variável(*unsigned char shot*) que guarda um valor, consoante o shot do jogador no adversário, sendo assim alterado durante a partida:

- '0' : se não houver shot;
  - '1' : se o shot não acertar em nenhum navio;
  - '2' : se o shot acertar em algum navio.
- **struct user**\_: define o jogador, guardando a matriz do mesmo, a lista de barcos que ele definiu(*List shipList*) e também o seu nome.
  - **struct Point**\_: struct para definir um ponto com duas coordenadas (x,y). Foi implementada para utilizar na estrutura de dados QuadTree.

## 2.4 MACROS

As macros, podem ser constantes ou funções:

### 2.4.1 Constantes

- **Na struct BitMap**
  - **define sizeBitMap 5**: define tamanho que uma bitmap tem, neste caso 5;
  - **define PXREF 2**: define o ponto de referência em x para a rotação do barco na bitmap;
  - **define PYREF 2**: define o ponto de referência em y para a rotação do barco na bimap;
- **define MAX\_BUFFER 1024**: define o tamanho máximo do buffer utilizado na manipulação do ficheiro.
- **Na struct Point**
  - **COORDX(P)**: aceder à posição x do ponto.
  - **COORDY(P)**: aceder à posição y do ponto.
  - **SETX(P, x)**: modificar o valor de x.
  - **SETY(P, y)**: modificar o valor de y.

### 2.4.2 Funções

- **define sin\_int(int degrees)**: define uma função para retornar o valor do seno em relação aos ângulos múltiplos de 90;
- **define cos\_int(int degrees)**: define uma função para retornar o valor do cosseno em relação aos ângulos múltiplos de 90;
- **define rotationNumber(number)**: define uma função que recebe um valor de 1 a 4, originário da função **random** da linguagem C, que permite assim saber qual a rotação que o navio deve sofrer;

## 2.5 Enumerações

As enumerações servem para estipular um novo tipo de variável, limitando os valores da mesma:

- **ShipKind**: define o tipo de navio que existe na nossa implementação, como SOLO, TRIAL, SMALL\_QUAD, BIGGEST\_QUAD, L\_GUY;
- **bool**: define um tipo booleano, como true ou false;
- **nodeType**: diferenciar entre uma quadtree do tipo *LEAF* ou *FATHER*;
- **quadChilds**: saber a qual quadrante pertence uma quadtree.

## 2.6 Ficheiros *source*

- **bitmap.c** : funções relacionados com a struct **bitmap**, como a sua inicialização e alteração do valor das células.
- **board.c** : contém várias funções que servem para a demonstração gráfica dos tabuleiros de cada jogador no terminal.
- **cell.c** : diversas funções com o propósito de manipular as células, sendo possível modificar todos os valores atribuídos à mesma.
- **ship.c**: funções relacionadas com a definição do navio, como a sua criação, o cálculo das transformações geométricas (rotações e translações) que este pode sofrer e a eliminação de um navio.
- **list.c**: funções de implementação da lista, como a sua criação e a do nó e a adição de nós à cabeça da lista;
- **matrix.c**: funções relacionadas com o tabuleiro do jogo, como a sua inicialização, a inserção de elementos na matriz, a eliminação dos mesmos;
- **initGame.c**: funções de início do jogo, como a criação dos utilizadores, o cálculo do número de barcos que irá ser usado por cada jogador durante o jogo, sendo este feito pela seguinte fórmula:

$$tamanho do tabuleiro / (sizeBitMap * sizeBitMap),$$

também encontramos a função que permite definir quem inicia o jogo;

- **game.c**: funções relacionadas com o jogo, como o disparo efetuado pelos jogadores (que vai atualizar as variáveis **value** e **shot** da célula da matriz e o valor guardado na célula da bitmap). A cada disparo verifica, se a célula é um pointer para um navio ou não, em caso afirmativo é apurado se esse disparo afunda o navio. Foi implementado uma função em que os jogadores podem escolher se desejam definir e posicionar os seus navios de forma manual, aleatória ou ambas.
- **geral.c**: definido os tipos **MATRIX** e **QUADTREE** e as suas respectivas funções.
- **randomShips.c**: funções para criação de pontos para a translação, a rotação e para a inserção no tabuleiro do jogo, utilizando a função já existente **random()** da linguagem C. Nestas funções, depois da criação, os navios sofrem as alterações geométricas com o valor retornado pelo **random** e também os navios são inseridos.
- **menu.c**: implementada a iteração entre o terminal e o utilizador, demonstrando menus iterativos através da escolha de números, sendo assim possível iniciar um jogo, visualizar as regras e sair do jogo.

- **quadtree.c**: implementação da estrutura de dados quadtree com todas as funções necessárias para o seu funcionamento.
- **point.c**: contém funções para inicializar, aceder, modificar ou remover um ponto.
- **main.c**: função principal do jogo.

## 2.7 Manipulação de ficheiros

As regras do jogo estão escritas num ficheiro `.txt`. Como a linguagem C permite ler texto através de ficheiros, utilizamos o `fopen()`, para imprimir as regras no terminal, facilitando assim a leitura das mesmas. O código usado é o seguinte:

```
char buffer [MAX_BUFFER];
int c;

FILE *file;
file = fopen("rules.txt", "r");

if (file) {
    while ((c = getc(file)) != EOF) putchar(c);
    fclose(file);
}
```

## 2.8 Compilação

Para o jogo funcionar, precisa como base, da sua estrutura de dados. Como neste projeto implementamos duas, é necessário diferenciá-las, deste modo foi criado um ficheiro *geral.c*, que atribuiu as funções necessárias para cada uma delas.

```
#ifdef MATRIX                #ifdef QUADTREE
...
#endif                        #endif
```

Na compilação utilizamos um *MakeFile* com um pre-processor, em particular a opção `-D`. Este mesmo distingue os dois tipos, assim sendo, o utilizador escolhe com qual estrutura de dados quer inicializar o jogo.

```
quad:
    gcc -D QUADTREE -o main *.c

matrix:
    gcc -D MATRIX -o main *.c
```

### 3 Interface

### 3.1 Navios

Os navios são desenhados da seguinte forma:

SHIPS'S TYPE:

|      |          |               |   |                                  |                    |                                  |
|------|----------|---------------|---|----------------------------------|--------------------|----------------------------------|
| 1) x | 2) x x x | 3) x x<br>x x | 4) x x x x<br>x x x x<br>x x x x<br>x x x x | 5) x<br>x<br>x<br>x<br>x x x x x | 6) x x x<br>x<br>x | 7) x x x<br>x<br>x<br>x<br>x x x |
|------|----------|---------------|---|----------------------------------|--------------------|----------------------------------|

### 3.1.1 Bitmap default de cada navio

```

BITMAP SHIP (default):

SOLO: 1 0 0 0 0 TRIAL: 1 1 1 0 0 SMALL_QUAD: 1 1 0 0 0 BIGGEST_QUAD: 1 1 1 1 0
      0 0 0 0 0      0 0 0 0 0      1 1 0 0 0      1 1 1 1 0
      0 0 0 0 0      0 0 0 0 0      0 0 0 0 0      1 1 1 1 0
      0 0 0 0 0      0 0 0 0 0      0 0 0 0 0      1 1 1 1 0
      0 0 0 0 0      0 0 0 0 0      0 0 0 0 0      0 0 0 0 0

      L_GUY: 1 0 0 0 0 T_GUY: 1 1 1 0 0 S_GUY: 0 0 1 1 1
            1 0 0 0 0      0 1 0 0 0      0 0 1 0 0
            1 0 0 0 0      0 1 0 0 0      0 0 1 0 0
            1 0 0 0 0      0 0 0 0 0      0 0 1 0 0
            1 1 1 1 1      0 0 0 0 0      1 1 1 0 0

```

### 3.2 Início do jogo

```
-----
                WELCOME TO BATTLESHIP
-----

(1) Start
(2) Rules
(3) Exit

Choose one option: █
```



### 3.3 Tabuleiro do jogo de um utilizador

#### 3.3.1 Tabuleiro depois de inserir os navios todos de um utilizador

|   | Matrix |   |   |   |   |   |   |   |   |   | User2 |   |   |   |   |   |   |   |   |   |
|---|--------|---|---|---|---|---|---|---|---|---|-------|---|---|---|---|---|---|---|---|---|
|   | A      | B | C | D | E | F | G | H | I | J | K     | L | M | N | O | P | Q | R | S | T |
| A | .      | . | . | . | . | . | X | X | X | . | .     | . | . | . | . | . | . | . | . | . |
| B | .      | . | . | . | . | . | . | . | . | . | .     | . | . | X | . | . | . | . | . | . |
| C | .      | . | X | X | X | X | X | . | . | . | .     | . | . | X | . | . | X | . | . | . |
| D | .      | . | . | . | . | . | X | . | . | . | .     | . | X | X | X | . | X | . | . | . |
| E | .      | . | X | X | . | . | X | . | . | . | .     | . | . | . | . | X | X | X | . | . |
| F | .      | . | X | X | . | . | X | . | . | . | .     | . | X | . | . | . | . | . | . | . |
| G | .      | . | . | . | . | X | . | . | . | X | X     | X | . | . | X | X | X | X | . | . |
| H | .      | . | . | X | . | . | . | . | X | . | .     | X | . | . | X | X | X | X | . | . |
| I | .      | . | . | . | . | X | X | . | . | . | .     | . | . | . | X | X | X | X | . | . |
| J | .      | . | . | . | . | X | X | . | . | . | .     | . | . | . | X | X | X | X | . | . |
| K | .      | . | . | . | . | . | . | . | . | . | .     | . | . | . | . | . | . | . | . | . |
| L | .      | X | X | X | . | . | . | . | X | . | .     | . | . | . | . | . | . | . | . | . |
| M | .      | . | . | . | . | . | . | . | X | . | .     | . | . | . | . | X | . | . | . | . |
| N | .      | . | . | . | . | . | . | . | X | X | X     | X | X | . | . | . | . | . | . | . |
| O | .      | . | . | X | X | X | . | . | . | . | .     | . | X | . | . | . | . | . | . | . |
| P | .      | . | . | . | . | . | . | . | . | . | .     | . | X | . | . | . | . | . | . | . |
| Q | .      | . | . | . | . | . | . | . | . | . | .     | . | . | . | . | . | . | . | . | . |
| R | .      | . | . | . | . | . | . | . | . | . | .     | . | . | . | . | . | . | . | . | . |
| S | .      | . | . | . | . | . | . | . | . | . | .     | . | . | X | X | . | . | . | . | . |
| T | .      | . | . | . | . | . | . | . | . | . | .     | . | X | X | . | . | . | . | . | . |

#### 3.3.2 Tabuleiro do utilizador1 depois de 2 jogadas

A jogada efetuada foi no ponto **G,A** e acertou num barco.

A jogada efetuada foi no ponto **D,B** e acertou numa célula vazia.

| Matrix User1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|--------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|              | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T |
| A            | + | . | . | . | . | . | + | . | . | . | . | . | . | . | . | . | . | . | . | . |
| B            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| C            | . | . | . | . | . | . | . | . | . | . | . | . | X | . | . | . | . | . | . | . |
| D            | . | . | . | X | X | X | . | X | X | X | . | . | X | X | X | . | . | . | . | . |
| E            | . | . | . | . | . | . | . | X | . | . | . | . | X | . | . | . | . | . | . | . |
| F            | . | . | . | X | . | . | . | X | . | . | . | . | . | . | . | . | . | . | . | . |
| G            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | X | . | . | . |
| H            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | X | . | . |
| I            | . | . | . | . | . | . | . | . | X | X | X | X | . | . | . | . | X | X | X | . |
| J            | . | . | . | . | . | . | . | . | X | X | X | X | . | . | . | . | X | . | . | . |
| K            | . | . | X | X | X | X | X | . | X | X | X | X | . | . | . | . | X | . | . | . |
| L            | . | . | . | . | . | . | X | . | X | X | X | X | . | . | . | . | X | X | X | X |
| M            | . | . | . | . | . | . | X | . | . | . | . | . | . | . | . | . | . | . | . | X |
| N            | . | . | . | . | . | . | X | . | . | . | . | . | X | X | . | . | . | . | . | X |
| O            | . | . | . | . | . | X | . | . | X | X | . | X | X | . | . | . | . | . | . | . |
| P            | . | . | . | . | . | . | . | . | X | X | . | . | . | . | . | . | . | . | . | . |
| Q            | . | . | X | X | X | . | X | . | . | . | . | X | . | . | . | . | . | . | . | . |
| R            | . | . | . | . | . | . | . | . | . | X | X | . | . | . | . | . | . | . | . | . |
| S            | . | . | . | . | . | . | X | X | X | . | X | X | . | . | . | . | . | . | . | . |
| T            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |

| Matrix User2 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|--------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|              | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T |
| A            | . | . | . | . | . | . | * | . | . | . | . | . | . | . | . | . | . | . | . | . |
| B            | . | . | . | + | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| C            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| D            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| E            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| F            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| G            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| H            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| I            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| J            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| K            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| L            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| M            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| N            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| O            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| P            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| Q            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| R            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| S            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| T            | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |

## 4 Execução do Código

Para a execução do código foi criado um **Makefile**, pois são utilizados vários ficheiros e é mais fácil utilizar pois diferencia diretamente o tipo de estrutura de dados a ser escolhida.

**Compilar escolhendo a estrutura de dados: MATRIX e QUADTREE**

```
$ make matrix
```

```
$ make quad
```

**Para executar:**

```
$ ./main
```

**Para eliminar os ficheiros executáveis:**

```
$ make clean
```