

Batalha Naval – preparação do tabuleiro de jogo

O JOGO DA BATALHA NAVAL

A *Batalha Naval* é jogada por dois jogadores. Cada jogador possui um tabuleiro retangular onde posiciona "a sua esquadra". O objetivo do jogo é descobrir onde se encontram os navios do adversário de modo a afundar toda a sua esquadra.

A esquadra de cada jogador é composta por vários tipos de navios. Em geral, são usados tipos como: porta-aviões, fragata, cruzador, submarino e lancha de patrulha.

Os jogadores possuem esquadras iguais. Antes do início do jogo, posicionam a sua esquadra no tabuleiro respetivo, de modo a que nenhum dos navios se sobreponha a outro. Cada navio pode ser colocado na horizontal ou vertical, mas não na diagonal.

Após o posicionamento da esquadra, o jogo desenrola-se por turnos. Em cada turno, o jogador que tem a vez de jogar seleciona uma coordenada do tabuleiro do oponente para lançar um tiro. Se o tiro acertar num navio do opositor, o jogador continua a jogar, selecionando outra posição. Se o tiro falhar passa a vez ao adversário.

O jogo termina quando uma das esquadras for completamente aniquilada.

OBJETIVOS

Este trabalho prático tem como objetivo desenvolver um programa para preparar os tabuleiros usados por cada um dos jogadores. Ao desenvolvê-lo os estudantes terão oportunidade de pôr em prática os seus conhecimentos de programação em C/C++. Não é objetivo deste trabalho prático desenvolver um programa que permita jogar o jogo; esse será o objetivo do trabalho prático nº 2, usando um conjunto de regras que serão indicadas oportunamente.

Ao realizar este trabalho, os estudantes terão oportunidade de aplicar os conhecimentos adquiridos até ao momento, nomeadamente, quanto a:

- desenvolvimento de interfaces simples;
- tratamento de entradas do teclado inválidas;
- formatação de saídas;
- leitura e escrita em ficheiros de texto;
- uso de estruturas de controlo de um programa (seleção e repetição);
- uso de diversos tipos de estruturas de dados (*strings*, *arrays/vectors*, *structs*, *files*);
- uso de funções, com passagem de parâmetros por valor e por referência.

DESENVOLVIMENTO DO PROGRAMA

Funcionamento geral

O programa deve começar por ler um ficheiro de configuração, no qual são especificadas as dimensões do tabuleiro, o número e tipo de navios, bem como as dimensões de cada navio e a cor a usar para representá-lo no ecrã. De seguida, deve dar ao utilizador a possibilidade de escolher o modo de colocação dos navios no tabuleiro: manual ou automático. No modo manual, será apresentado ao utilizador um tabuleiro vazio, onde este colocará, interativamente, os navios. No modo automático, a posição e orientação dos navios serão escolhidas aleatoriamente pelo programa. No final, deve ser gerado um ficheiro de tabuleiro que conterá informação sobre a colocação final dos navios no tabuleiro. O formato dos ficheiros será especificado adiante.

Nota: Apesar de o 1.º trabalho prático não ser classificado, é condição necessária para a obtenção de frequência o desenvolvimento, entrega e apresentação (quando solicitada) deste trabalho. O trabalho será realizado individualmente.

Ficheiro de configuração

O ficheiro de configuração é um ficheiro de texto que pode ser criado com um editor de texto simples como, por exemplo, o Notepad do Windows. Podem existir vários ficheiros de configuração, de forma a possibilitar que o jogo seja jogado em tabuleiros de diferentes dimensões e com diferentes tipos ou número de navios. O programa deve começar por perguntar ao utilizador qual o nome do ficheiro de configuração a usar, terminando com um erro caso o ficheiro indicado não exista.

Apresenta-se a seguir um possível conteúdo de um ficheiro de configuração.

```
tabuleiro: 10 x 10
1 - Porta-avioes - 5 - P - Vermelho_claro
1 - Fragata - 4 - F - Magenta_claro
2 - Cruzador - 3 - C - Verde_claro
3 - Submarino - 2 - S - Castanho
4 - Lancha - 1 - L - Azul_claro
```

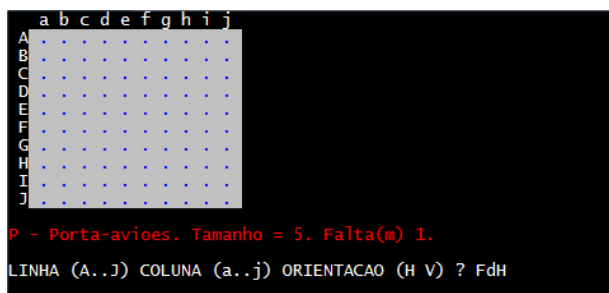
Exemplo de um ficheiro de configuração.

A primeira linha do ficheiro indica as dimensões do tabuleiro, no formato "número_de_linhas x número_de_colunas" (10 linhas e 10 colunas, neste caso). Cada uma das linhas seguintes indica a seguinte informação acerca de cada tipo de navio: a quantidade de navios desse tipo a colocar no tabuleiro, o nome por extenso, o comprimento (a largura de todos os navios é de 1 unidade), o símbolo (letra **P**, **F**, **C**, **S** ou **L**, a indicar porta-aviões, fragata, cruzador, submarino ou lancha, respetivamente) e a cor a usar para a sua representação no ecrã. Os valores de cada linha estão separados entre si por " - ".

Poderão ser escolhidos outros tipos de navios, desde que se tenha o cuidado de usar diferentes símbolos para a representação de cada um deles. O número, o tamanho (comprimento) e a cor dos navios também podem escolhidos arbitrariamente, desde que o tabuleiro tenha espaço para a sua colocação. Notar que em aplicações que funcionem em "modo consola" o número de cores é muito limitado (ver anexo). Não se pretende que o programa tenha a capacidade de fazer a verificação de existência de espaço para colocação dos navios.

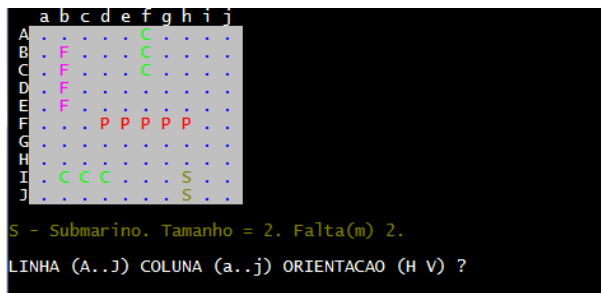
Colocação dos navios no tabuleiro

Apresenta-se a seguir um exemplo de uma possível interface do programa e da interação com utilizador, durante a fase de colocação dos navios, em "modo manual". A interface deve ser feita em modo texto.



a)

a) Tabuleiro vazio, no início da colocação manual ('.' = célula vazia).



b)

b) Tabuleiro após a colocação de alguns navios ('.' = célula vazia; P/F/C/S/L = célula ocupada por um navio).

Os navios são colocados pela ordem estabelecida no ficheiro de configuração, indicando as coordenadas do canto superior esquerdo da posição do navio e a sua orientação. As coordenadas são indicadas por duas letras: uma maiúscula para indicar a linha e uma minúscula para indicar a coluna. A orientação é indicada por uma das letras **H** (horizontal) ou **V** (vertical). Nos navios de comprimento unitário também será necessário indicar uma orientação, apesar de esta ser irrelevante no que diz respeito às regras habituais do jogo.

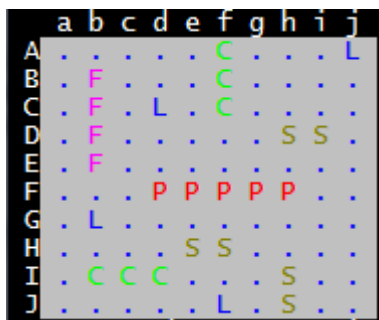
Por cada navio a colocar, deve ser indicado o seu tipo, o seu comprimento e o número de navios desse tipo que falta colocar no tabuleiro (ver exemplo, acima).

No "modo automático", as coordenadas e a orientação de cada navio devem ser geradas de forma aleatória, sem qualquer intervenção do utilizador. No final, deve ser mostrado o tabuleiro resultante.

Em qualquer dos modos de funcionamento, o programa deve impedir a colocação de navios em posições inválidas, isto é, navios que fiquem total ou parcialmente fora do tabuleiro e/ou se sobreponham a outros navios. No “modo manual” deve ser mostrada uma mensagem ao utilizador e solicitado que ele indique uma nova posição. No “modo automático” deve ser gerada aleatoriamente uma nova posição.

Ficheiro de tabuleiro

No final, pretende-se que o programa guarde o conteúdo do tabuleiro, isto é, a colocação dos navios, num ficheiro de texto, designado ficheiro de tabuleiro, cujo nome deve ser indicado pelo utilizador no início do programa. O conteúdo desse ficheiro deve ser semelhante ao indicado a seguir.



a) *Aspetto final do tabuleiro, após a colocação de todos os navios.*

```
10 x 10
P 5 Fd H
F 4 Bb V
C 3 Af V
C 3 Ib H
S 2 He H
S 2 Ih V
S 2 Dh H
L 1 Cd H
L 1 Aj V
L 1 Jf V
L 1 Gb H
```

b) *Ficheiro de tabuleiro correspondente ao tabuleiro de a).*

A primeira linha do ficheiro deve conter dois números que indicam, respetivamente, o número de linhas e de colunas do tabuleiro (10 linhas x 10 colunas, no caso do exemplo). Cada uma das restantes linhas indica a colocação de um navio no tabuleiro. Cada linha contém os seguintes dados relativos a um navio: símbolo, comprimento, coordenadas (linha e coluna) do canto superior esquerdo e orientação. Os dados são separados por espaços.

MELHORIAS OPCIONAIS

Apresentam-se a seguir alguns desafios para desenvolvimentos adicionais:

- Avisar o utilizador quando já existe um ficheiro de tabuleiro com o mesmo nome que ele escolheu.
- Verificar a validade do conteúdo de um ficheiro de configuração, impedindo que sejam escolhidos tabuleiros com dimensões demasiado grandes/pequenas, cores inválidas, símbolos ou cores repetidos, ...
- Permitir alterar a posição de navios já colocados no tabuleiro.

NOTAS SOBRE O DESENVOLVIMENTO

Funcionamento geral do programa

O funcionamento geral do programa é o seguinte:

- Começa por pedir ao utilizador que indique o nome do ficheiro de configuração, o nome do ficheiro de tabuleiro e o modo de funcionamento (manual - 'M' ou 'm' - ou automático - 'A' ou 'a').
 - Depois, deve ler o conteúdo do ficheiro de configuração e guardá-lo numa estrutura de dados adequada.
 - Gera um tabuleiro vazio (ver exemplo anterior).
 - Seguidamente, executa um ciclo em que, para cada navio a colocar no tabuleiro:
 - mostra o conteúdo atual do tabuleiro (na primeira iteração mostra um tabuleiro vazio);
 - repete:
 - indica o tipo de navio a colocar no tabuleiro e, no caso de o utilizador ter optado pela colocação manual, pede-lhe que indique a sua localização (coordenadas do canto superior esquerdo e orientação); caso ele tenha optado pela colocação automática, gera a localização de forma aleatória
- até que a localização indicada seja válida, isto é, o navio esteja inteiramente contido no tabuleiro e não fique sobreposto a nenhum outro navio;

- o insere o navio na posição indicada;
- No final, grava o conteúdo do tabuleiro no ficheiro indicado no início do programa pelo utilizador.

No desenvolvimento da interface com o utilizador tenha em conta os exemplos apresentados, procurando reproduzir aproximadamente o modo de introdução dos dados e de apresentação dos resultados. Por exemplo, os dados relativos à localização do navio devem ser introduzidos na mesma linha de entrada de dados.

O desenvolvimento de uma interface colorida, em modo texto, pode ser efetuado recorrendo à biblioteca de funções declaradas em windows.h (ver exemplo, em anexo).

Devem ser tomadas as precauções necessárias para evitar que o programa deixe de funcionar corretamente devido a entradas inválidas do utilizador, nomeadamente, valores fora da gama admissível.

Escrita do código

Na escrita do código devem ser respeitadas as indicações dadas nas aulas, nomeadamente, em relação aos seguintes aspetos:

- Escolha das estruturas de dados mais adequadas para representar os dados do programa.
- Escolha adequada dos identificadores de tipos, variáveis e funções.
- Estrutura modular do código.
- Comentários ao código.

ENTREGA DO TRABALHO

- Criar uma pasta com o nome **TxEYYYYYYYYY**, em que **x** representa o número da turma e **YYYYYYYYYY** representa o número de estudante (sem as letras up ou ei); por exemplo, **T2E201400007**, para o estudante com o número 201400007 da turma 2) e copiar para lá o código-fonte do programa (apenas os ficheiros com a extensão .cpp e .h, se existirem). Incluir também um ficheiro ReadMe.txt (em formato de texto simples) indicando o estado de desenvolvimento do trabalho, isto é, se foram cumpridos todos os objetivos ou, caso contrário, quais os que não foram cumpridos, ou ainda que melhorias foram implementadas, se for esse o caso.
- Compactar o conteúdo desta pasta num ficheiro **TxEYYYYYYYYY.zip** ou **TxEYYYYYYYYY.rar** e submeter este ficheiro na página da disciplina de Programação, no Moodle da FEUP. Não serão aceites entregas por outras vias. **(NOTA: nomes corrigidos em 23/março)**
- Data limite para a entrega: 19/Abr/2015 (às 23:55h).

ANEXO

Exemplo de programa que escreve caracteres coloridos

```
// PROG - MIEIC
// JAS
// Example of a program that prints colored characters on the console (in text mode)

#include <iostream>
#include <ctime>
#include <cstdlib>
#include <windows.h>

using namespace std;

//=====
//COLOR CODES: (alternative: use symbolic const's)
#define BLACK 0
#define BLUE 1
#define GREEN 2
#define CYAN 3
#define RED 4
#define MAGENTA 5
#define BROWN 6
#define LIGHTGRAY 7
#define DARKGRAY 8
#define LIGHTBLUE 9
#define LIGHTGREEN 10
#define LIGHTCYAN 11
#define LIGHTRED 12
#define LIGHTMAGENTA 13
#define YELLOW 14
#define WHITE 15
//=====

void clrscr(void)
{
    COORD coordScreen = { 0, 0 }; // upper left corner
    DWORD cCharsWritten;
    DWORD dwConSize;
    HANDLE hCon = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_SCREEN_BUFFER_INFO csbi;

    GetConsoleScreenBufferInfo(hCon, &csbi);
    dwConSize = csbi.dwSize.X * csbi.dwSize.Y;

    // fill with spaces
    FillConsoleOutputCharacter(hCon, TEXT(' '), dwConSize, coordScreen, &cCharsWritten);
    GetConsoleScreenBufferInfo(hCon, &csbi);
    FillConsoleOutputAttribute(hCon, csbi.wAttributes, dwConSize, coordScreen, &cCharsWritten);

    // cursor to upper left corner
    SetConsoleCursorPosition(hCon, coordScreen);
}

//=====

// Position the cursor at column 'x', line 'y'
// The coordinates of upper left corner of the screen are (x,y)=(0,0)

void gotoxy(int x, int y)
{
    COORD coord;
    coord.X = x;
    coord.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}

//=====

// Set text color
void setcolor(unsigned int color)
{
    HANDLE hcon = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hcon, color);
}

//=====

// Set text color & background
void setcolor(unsigned int color, unsigned int background_color)
{
    HANDLE hCon = GetStdHandle(STD_OUTPUT_HANDLE);
    if (background_color == BLACK)
        SetConsoleTextAttribute(hCon, color);
    else
        SetConsoleTextAttribute(hCon, color | BACKGROUND_BLUE | BACKGROUND_GREEN |
                                BACKGROUND_RED);
}

//=====

// Fill the screen with colored numbers

int main()
{
    clrscr();

    srand((unsigned int)time(NULL));

    for (int x = 0; x < 80; x++)
        for (int y = 0; y < 24; y++)
        {
            gotoxy(x, y);
            if (rand() % 2 == 0)
                setcolor(x % 15 + 1);
            else
                setcolor(y % 15 + 1, rand() % 2);
            cout << x % 10;
        }
}
```