

ARCADE:

BATALHA NAVAL

Robert de Souza




$$\frac{\sim}{\sim)} \frac{\sim}{\sim)} \frac{\sim}{\sim)}$$

Digite (N) para jogar
 --- BATALHA NAVAL ---

Digite (S) para sair

Digite o comando (V/N/S):

MAIN

```
cout << endl << endl << spc << "Digite o comando (V/N/S): ";
char res; cin >> res;

switch (res)
{
case 'V': case 'v':
    JOGO_DA_VELHA();
    break;
case 'N': case 'n':
    while(BATALHA_NAVAL());
    break;
case 'S': case 's':
    return 0;
default:
    cout << spc << "Digite um comando válido: ";
    cin >> res;
    break;
}
```



01

INTERFACES E ANIMAÇÕES

FUNÇÕES DE INTERFACES DO JOGO



```
// INTERFACES //
> void regras() // TUTORIAL DE DIGITACAO...

> void comoJogar() // EXPLICACAO DO JOGO...

> int inicio(bool animacao) // ANIMACAO/TELA INICIAL ...

> int definicaoJogadores(DadosJogador jogador[], bool multiplayer) // ESCOLHA DOS NOMES DO JOGADORES ...

> void tabuleiroImprimir(char tabuleiro[][TAM], DadosJogador jogador[], int dificuldade) // IMPRESSÃO DO TABULEIRO...

> void bomba(int num) // ANIMACAO DE UM NAVIO DESTRUÍDO ...

> void bombinha(char tabuleiro[][TAM], DadosJogador jogador[], int dificuldade, int coord[]) // ANIMACAO NO TABULEIRO...

> void final(DadosJogador jogador[]) // TELA FINAL ...
```



FUNÇÕES PARA AS ANIMAÇÕES

```
// FUNCOES PARA AS ANIMACOES //  
void sleep(int milliseconds) // DELAY  
{  
    std::this_thread::sleep_for(std::chrono::milliseconds(milliseconds));  
}  
  
void clear() // LIMPAR O TERMINAL  
{  
    #ifdef _WIN32  
        system("cls"); // Comando para limpar o terminal no Windows  
    #else  
        system("clear"); // Comando para limpar o terminal em sistemas bas  
    #endif  
}
```

EXEMPLOS DE INTERFACES



EXEMPLOS DE INTERFACES

BATALHA NAVAL - TABULEIRO

	1	2	3	4	5	6	7	8	9	10
A
B
C
D
E
F
G
H
I
J

Dificuldade: 1

Pontuação de Robert: 0

Pontuação de Computador: 0

Para mudar o nível de dificuldade do jogo, digite (N).

Caso esteja com 3 vidas de como jogar digite (C).

Para saber as regras de digite (R).

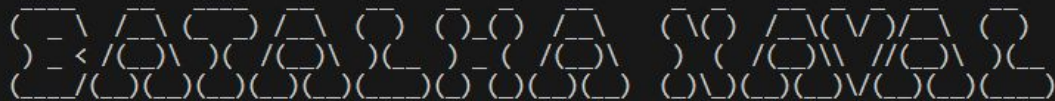
Digite (S) para sair do jogo

1a rodada:

Vez do Jogador 1. Digite a coordenada ou o comando:



EXEMPLOS DE INTERFACES



Parabens, o 2o barco destruido!



EXEMPLOS DE INTERFACES

BATALHA NAVAL - REGRAS

Cada jogador deve jogar somente na sua vez, se ele acertar uma barco ele tem direito a jogar novamente, se acertar de novo, ele ganhará mais uma chance, até acabar errando ou o jogo finalizar.

Durante a digitação das coordenadas você deve escrever ou um dos códigos disponíveis ou a coordenada desejada. Para os códigos você pode escrever tanto em maiúsculo quanto em minúsculo.

Para as coordenadas você pode escrever de quatro formas, por exemplo:

A1 ou a1 ou 1A ou 1a

Código oculto:

- Digite (G) sempre que desejar ver o gabarito (onde estão os navios) do tabuleiro.

Digite qualquer tecla para voltar:





02

GERAÇÃO

DOS NAVIOS

```

switch(dificuldade)
{
case 1: // FACIL
    for(int i = 0; i < 10; i++)
    {
        switch(i)
        {
            case 0 ... 2:
                escolhaBarco(4, tabuleiro, gabarito, barco[0], barcoTam, i);
                break;
            case 3 ... 5:
                escolhaBarco(3, tabuleiro, gabarito, barco[1], barcoTam, i);
                break;
            case 6 ... 7:
                escolhaBarco(2, tabuleiro, gabarito, barco[2], barcoTam, i);
                break;
            case 8 ... 9:
                escolhaBarco(1, tabuleiro, gabarito, barco[3], barcoTam, i);
                break;
        }
    }
    break;

```

```

case 3: case 5: // DIFICIL OU SURPREENDA-ME NO DIFICIL
    for(int i = 0; i < 5; i++)
    {
        switch(i)
        {
            case 0:
                escolhaBarco(3, tabuleiro, gabarito, barco[1], barcoTam, i);
                break;
            case 1 ... 2:
                escolhaBarco(2, tabuleiro, gabarito, barco[2], barcoTam, i);
                break;
            case 3 ... 4:
                escolhaBarco(1, tabuleiro, gabarito, barco[3], barcoTam, i);
                break;
        }
    }
    break;

```

```

case 2: case 4: // MEDIO OU SURPREENDA-ME NO MEDIO
    for(int i = 0; i < numBarcos; i++)
    {
        switch(i)
        {
            case 0:
                escolhaBarco(4, tabuleiro, gabarito, barco[0], barcoTam, i);
                break;
            case 1 ... 2:
                escolhaBarco(3, tabuleiro, gabarito, barco[1], barcoTam, i);
                break;
            case 3 ... 5:
                escolhaBarco(2, tabuleiro, gabarito, barco[2], barcoTam, i);
                break;
            case 6 ... 9:
                escolhaBarco(1, tabuleiro, gabarito, barco[3], barcoTam, i);
                break;
        }
    }

```

```

int barco[4][4] = {{0, 1, 2, 3}, // NAVIO DE 4
                  {0, 1, 2, 0}, // NAVIO DE 3
                  {0, 1, 0, 0}, // NAVIO DE 2
                  {0, 0, 0, 0}}; // NAVIO DE 1

```

```

if(dificuldade == 4)
    if(numBarcos <= 5)
        dificuldade = 5;

```



```

void escolhaBarco(int tamBarco, char tabuleiro[][TAM], char gabarito[][TAM], int barco[4], DadosCoordBarco barcoTam[][4], int numBarco)
{
    int coordH = rand()%10;
    int coordV = rand()%10;
    int orientacao = rand()%2;

    if(orientacao == 0) // Horizontal
    {
        while( !validacao(tabuleiro, gabarito, coordH, coordV + barco[0]) ||
               !validacao(tabuleiro, gabarito, coordH, coordV + barco[1]) ||
               !validacao(tabuleiro, gabarito, coordH, coordV + barco[2]) ||
               !validacao(tabuleiro, gabarito, coordH, coordV + barco[3])) // Enquanto nao for valido
        {
            coordH = rand()%100;
            coordV = rand()%100;
        }

        for(int i = 0; i < tamBarco; i++) // Cadastrando as coordenadas
        {
            gabarito[coordH][coordV+i] = BARCO;

            barcoTam[numBarco][i].coord[0] = coordH;
            barcoTam[numBarco][i].coord[1] = coordV+i;
        }
    }
    else // Vertical
    {

```

```

struct DadosCoordBarco // DADOS DOS BARCOS
{
    int coord[2] = {-1, -1}; // 0: Horizontal 1: Vertical
    bool atacado;
    int jogadorAtacante;
};

```

```

// CRIACAO DOS BARCOS //
bool validacao(char tabuleiro[][TAM], char gabarito[][TAM], int coordH, int coordV) // SE EH VALIDO
{
    if ((coordH >= 0 && coordH < TAM) && (coordV >= 0 && coordV < TAM)) //Se esta dentro da matriz
        if (tabuleiro[coordH][coordV] == PONTO) //Se esta vazia
        {
            for (int i = -1; i <= 1; i++) //Se ha X nas laterais e diagonais
                for (int j = -1; j <= 1; j++)
                    if ((coordH + i >= 0 && coordH + i < TAM) && (coordV + j >= 0 && coordV + j < TAM))
                        if (gabarito[coordH + i][coordV + j] == BARCO)
                            return false; // NAO VALIDO
            return true; // VALIDO
        }
    return false; // NAO VALIDO
}

```



03

JOGADAS



ESCOLHAS INICIAIS

```
int numBarcos;
switch (dificuldade)
{
case 1:
    numBarcos = 10;
    break;
case 2:
    numBarcos = 10;
    break;
case 3:
    numBarcos = 5;
    break;
case 4:
    numBarcos = rand()%9+1;
    break;
}
```

```
cout << "    Qual o número de jogadores (1/2)? ";
char res[4]; cin >> res;
```

```
while(!(res[0] == '1' || res[0] == '2')) ...
```

```
bool multiplayer = true;
if(res[0] == '1') ...
```

```
if(definicaoJogadores(jogador, multiplayer) == 0)
    return 0;
```

```
struct DadosJogador // DADOS DOS JOGADORES
{
    int NumJogador;
    char Nome[50] = "    ";
    int pontos = 0;
};
```




COMANDO DO JOGADOR

```
while(!fim(barcosDestruídos, numBarcos))
{
    switch(conversao(tabuleiro, res, coord, jogador[par].Nome))
    {
        case 0:
            if(multiplayer || par == 0) ...
            break;
        case 1: //RESPOSTA FOI UMA COORDENADA
            if(jogada(par, tabuleiro, coord, barcoTam, jogador, barcosDestruídos, NumBarcoDestruído, dificuldade, numBarcos))
            {
                rodada++;
                par = (rodada % 2 == 0);

                tabuleiroImprimir(tabuleiro, jogador, dificuldade);
                cout << rodada << "a Rodada:\n" << "Vez do Jogador " << par+1 << ". Digite a coordenada ou o comando: ";
            }
            break;
        case 2:
            regras();
            cout << endl << "Para retornar digite (V): ";
            break;
        case 3:
```



COMANDO DO JOGADOR

```
if(res[1] == '\0')
    switch (res[0])
    {
        case 'R': case 'r': // REGRAS
            return 2;
            break;
        case 'S': case 's': // SAIR
            return 3;
            break;
        case 'V': case 'v': // VOLTAR
            return 4;
            break;
        case 'G': case 'g': // GABARITO
            return 5;
            break;
        case 'N': case 'n': // NIVEL
            return 6;
            break;
        case 'C': case 'c':
            return 7;
            break;
        default: // ERRO
            return 0;
            break;
    }
```


```
if(res[0] == '1' && res[1] == '0' && ((res[2] >= 'A' && res[2] <= 'J') || (res[2] >= 'a' && res[2] <= 'j')))
{
    if(res[2] >= 'a' && res[2] <= 'j')
        res[2] = res[2] - 32; // Converter para maiuscula
    coord[0] = res[2] - 'A';
    coord[1] = 9;

    if(tabuleiro[coord[0]][coord[1]] != PONTO)
        return 0;
    return 1; // JOGADA
}

if(res[1] == '1' && res[2] == '0' && ((res[0] >= 'A' && res[0] <= 'J') || (res[0] >= 'a' && res[0] <= 'j')))
{
    if(res[0] >= 'a' && res[0] <= 'j')
        res[0] = res[0] - 32; // Converter para maiuscula
    coord[0] = res[0] - 'A';
    coord[1] = 9;

    if(tabuleiro[coord[0]][coord[1]] != PONTO)
        return 0;
    return 1; // JOGADA
}
```

SINGLEPLAYER



```
// ORGANIZACAO DAS RESPOSTAS //  
void jogadaPC(char res[4]) // JOGADA PC  
{  
    char letra = rand()%10+'A';  
    int num = rand()%10+1;  
  
    res[0] = letra;  
    if(num < 10)  
        res[1] = num + '0';  
    if(num == 10)  
    {  
        res[1] = '1';  
        res[2] = '0';  
    }  
}
```

```
int conversao(char tabuleiro[][TAM], char  
{  
    if(strcmp(nome, "Computador") == 0)  
        jogadaPC(Res);  
    else  
        cin >> Res;
```


JOGADA DA RODADA



```
while(acerto)
{
    tabuleiro[coord[0]][coord[1]] = ERRO;

    acerto = false;
    for(int i = 0; i < numBarcos; i++)
    {
        if(barcosDestruídos[i] == -1)
        {
            for(int j = 0; j < 4; j++)
            {
                if(barcoTam[i][j].coord[0] == coord[0] && barcoTam[i][j].coord[1] == coord[1])
                {
                    acerto = true;
                    jogador[num].pontoss++;
                    NumBarcoDestruído[1]++;

                    barcoTam[i][j].atacado = true;
                    barcoTam[i][j].jogadorAtacante = jogador[num].NumJogador;

                    bomba(NumBarcoDestruído[1]);

                    bombinha(tabuleiro, jogador, dificuldade, coord);
                }
            }
        }
    }
}
```


JOGADA DA RODADA



```
if( (barcoTam[i][0].atacado == true || barcoTam[i][0].coord[0] == -1) &&
    (barcoTam[i][1].atacado == true || barcoTam[i][1].coord[0] == -1) &&
    (barcoTam[i][2].atacado == true || barcoTam[i][2].coord[0] == -1) &&
    (barcoTam[i][3].atacado == true || barcoTam[i][3].coord[0] == -1))
{
    tabuleiroImprimir(tabuleiro, jogador, dificuldade);
    jogador[num].pontos++;

    if( (barcoTam[i][0].jogadorAtacante == jogador[num].NumJogador || barcoTam[i][0].coord[0] == -1) &&
        (barcoTam[i][1].jogadorAtacante == jogador[num].NumJogador || barcoTam[i][1].coord[0] == -1) &&
        (barcoTam[i][2].jogadorAtacante == jogador[num].NumJogador || barcoTam[i][2].coord[0] == -1) &&
        (barcoTam[i][3].jogadorAtacante == jogador[num].NumJogador || barcoTam[i][3].coord[0] == -1))
        jogador[num].pontos++;

    barcosDestruídos[i] = 0;
    NumBarcoDestruído[0]++;
}

cout << "Parabens, " << jogador[num].Nome << "! Voce acertou uma parte de um navio, agora tera mais uma chance.\n\n";
cout << "Sua vez, " << jogador[num].Nome << ". Digite uma coordenada ou um comando: ";

return 0;
}
```

```
return 1;
```

```
case 1: //RESPOSTA FOI UMA COORDENADA
if(jogada(par, tabuleiro, coord, barcoTam, jogador, barcosDestruídos, NumBarcoDestruído, dificuldade, numBarcos))
{
    rodada++;
    par = (rodada % 2 == 0);

    tabuleiroImprimir(tabuleiro, jogador, dificuldade);
    cout << rodada << "a Rodada:\n" << "Vez do Jogador " << par+1 << ". Digite a coordenada ou o comando: ";
}
break;
```

FINALIZAÇÃO DO LOOP



```
bool fim(int barcosDestruídos[], int numBarcos) // FINALIZACAO DO LOOP
{
    int fim = 0;

    for(int i = 0; i < numBarcos; i++)
    {
        if(barcosDestruídos[i] == 0)
            fim++;
    }

    if(fim == numBarcos)
        return true;
    else
        return false;
}
```

FINALIZAÇÃO DO LOOP



```
bool fim(int barcosDestruídos[], int numBarcos) // FINALIZACAO DO LOOP
{
    int fim = 0;

    for(int i = 0; i < numBarcos; i++)
    {
        if(barcosDestruídos[i] == 0)
            fim++;
    }

    if(fim == numBarcos)
        return true;
    else
        return false;
}
```

```
final(jogador);
```

```
cout << endl << "Deseja jogar novamente, Sim ou Nao? ";
cin >> res;
if((res[0] == 'S' || res[0] == 's') && res[1] == 'i' && res[2] == 'm')
    return 1;
else
    return 0;
```


JOGO DA VELHA (Com 1 ou 2 jogadores)



```

/*****
/*                                JOGO DA VELHA                                */
*****/

> void impressao(char tabuleiro[][3], char mapa[][3], char jog[]) // IMPRESSÃO DO TABULEIRO...

> char fimDaPartida(char matriz[][3]) // ANÁLISE DO FINAL...

> char posicao(int coord, char matriz[][3]) // CONVERSÃO...

> int ocupacao(char tabuleiro[][3]) // VERIFICAR RESPOSTA...

> void substituicao(char matriz[][3], char jog, int coord) // JOGADA...

> int multiply(char matriz[][3], char jog[]) // GERACAO MULTIPLAYER...

// main() do JOGO DA VELHA
> int JOGO_DA_VELHA() ...

*****/

```


INTERFACE

=====

SOBREVIVAM

Tabuleiro:

---	---	---
---	---	---

Coordenadas:

1	2	3
---	---	---
4	5	6
---	---	---
7	8	9

Jogador 1: O


Jogador 2: X

Digite (S) para sair

=====

Jogada n. 1. Vez do Jogador 1. Digite a coordenada: █





```
int multiply(char matriz[][3], char jog[]) // GERACAO MULTIPLAYER
{
    if((matriz[0][1] == matriz[0][2] && matriz[0][1] == jog[1]) ||
        (matriz[1][0] == matriz[2][0] && matriz[1][0] == jog[1]) ||
        (matriz[1][1] == matriz[2][2] && matriz[1][1] == jog[1])) && posicao(1, matriz) == ' '
        return 1;
    if((matriz[0][0] == matriz[0][2] && matriz[0][0] == jog[1]) ||
        (matriz[1][1] == matriz[2][1] && matriz[1][1] == jog[1])) && posicao(2, matriz) == ' '
        return 2;
    if((matriz[0][0] == matriz[0][1] && matriz[0][0] == jog[1]) ||
        (matriz[1][2] == matriz[2][2] && matriz[1][2] == jog[1]) ||
        (matriz[1][1] == matriz[2][0] && matriz[1][1] == jog[1])) && posicao(3, matriz) == ' '
        return 3;
    if((matriz[0][0] == matriz[2][0] && matriz[0][0] == jog[1]) ||
        (matriz[1][1] == matriz[1][2] && matriz[1][1] == jog[1])) && posicao(4, matriz) == ' '
        return 4;
    if((matriz[1][0] == matriz[1][2] && matriz[1][0] == jog[1]) ||
        (matriz[0][1] == matriz[2][1] && matriz[0][1] == jog[1]) ||
        (matriz[0][0] == matriz[2][2] && matriz[0][0] == jog[1])) && posicao(5, matriz) == ' '
        return 5;
    if((matriz[1][0] == matriz[1][1] && matriz[1][0] == jog[1]) ||
        (matriz[0][2] == matriz[2][2] && matriz[0][2] == jog[1])) && posicao(6, matriz) == ' '
        return 6;
    if((matriz[0][0] == matriz[1][0] && matriz[0][0] == jog[1]) ||
        (matriz[1][1] == matriz[0][2] && matriz[1][1] == jog[1]) ||
        (matriz[2][1] == matriz[2][2] && matriz[2][1] == jog[1])) && posicao(7, matriz) == ' '
        return 7;
    if((matriz[2][0] == matriz[2][2] && matriz[2][0] == jog[1]) ||
        (matriz[1][1] == matriz[0][1] && matriz[1][1] == jog[1])) && posicao(8, matriz) == ' '
        return 8;
    if((matriz[2][0] == matriz[2][1] && matriz[2][0] == jog[1]) ||
        (matriz[1][2] == matriz[0][2] && matriz[1][2] == jog[1]) ||
        (matriz[1][1] == matriz[0][0] && matriz[1][1] == jog[1])) && posicao(9, matriz) == ' '
        return 9;

    int num = rand()%10+1;

    while(posicao(num, matriz) != ' ')
        num = rand()%9+1;

    return num;
}
```

```

#include <iostream>
#include <iomanip>
#include <cstring>
#include <cstdlib>
#include <thread>

// DEFINICAO DAS PARTES DA INTERFACE
#define PONTO '.'
#define BARRA "----"
#define BARRA1 '|'
#define BARRA2 "_"
#define BARCO 'X'
#define ERRO ' '
#define TAM 10
#define MARGEM TAM*TAM-30+(3*6)
#define spc "      "

using namespace std;

/*****
/*                                BATALHA NAVAL

// ESTRUTURAS //
> struct DadosCoordBarco // DADOS DOS BARCOS ...

struct DadosJogador // DADOS DOS JOGADORES
{
    int NumJogador;
    char Nome[50] = "      ";
    int pontos = 0;
};

// FUNCOES PARA AS ANIMACOES //
> void sleep(int milliseconds) // DELAY ...

> void clear() // LIMPAR O TERMINAL ...

```

```

// INTERFACES //
> void regras() // TUTORIAL DE DIGITACAO ...

> void comoJogar() // EXPLICACAO DO JOGO ...

> int inicio(bool animacao) // ANIMACAO/TELA INICIAL ...

> int definicaoJogadores(DadosJogador jogador[], bool multiplayer) // ESCOLHA DOS NOMES DO JOGADORES ...

> void tabuleiroImprimir(char tabuleiro[][TAM], DadosJogador jogador[], int dificuldade) // IMPRESSÃO DO TABULEIRO ...

> void bomba(int num) // ANIMACAO DE UM NAVIO DESTRUÍDO ...

> void bombinha(char tabuleiro[][TAM], DadosJogador jogador[], int dificuldade, int coord[]) // ANIMACAO NO TABULEIRO ...

> void final(DadosJogador jogador[]) // TELA FINAL ...

// ORGANIZACAO DAS RESPOSTAS //
> void jogadaPC(char res[4]) // JOGADA PC ...

> int conversao(char tabuleiro[][TAM], char Res[4], int coord[], char nome[100]) // GERACAO DAS RESPOSTAS ...

// CRIACAO DOS BARCOS //
> bool validacao(char tabuleiro[][TAM], char gabarito[][TAM], int coordH, int coordV) // SE EH VALIDO ...

> void escolhaBarco(int tamBarco, char tabuleiro[][TAM], char gabarito[][TAM], int barco[4], DadosCoordBarco barcoCoord, DadosJogador jogador[]) // ESCOLHA DO BARCO ...

> void criacao(char tabuleiro[TAM][TAM], char gabarito[TAM][TAM], DadosCoordBarco barcoTam[][4], int numBarcos, DadosJogador jogador[]) // CRIACAO DOS BARCOS ...

// JOGABILIDADE //
> int jogada(int num, char tabuleiro[][TAM], int coord[], DadosCoordBarco barcoTam[][4], DadosJogador jogador[], DadosJogador jogador2[]) // JOGADA ...

> bool fim(int barcosDestruídos[], int numBarcos) // FINALIZACAO DO LOOP ...

// main() do jogo BATALHA NAVAL
> int BATALHA_NAVAL() ...

```


LINK PARA JOGAR:

<https://www.onlinegdb.com/caa6m1lCY>

