



# JOGO DA VELHA EM C++





# Descrição do Projeto

- **Objetivo:** Criação de um jogo da velha jogável, na linguagem c++, por meio de um compilador.
- **Características:** Dois jogadores (X e O), verificação de vitórias ou empate, opção de jogar novamente.
- **Tecnologias utilizadas:**
  1. Aplicativo: "Dev-C++" como compilador;
  2. Linguagem : C++ 5.11;
  3. Bibliotecas: Padrão da linguagem (iostream) e biblioteca (cstdlib) que permite a limpeza da tela do arquivo exe;
  4. Funções : void e bool e main;
  5. Uso de loops e condicionais;
  6. Matrizes como representação do tabuleiro.



# ORIGEM

Existem registros dele desde o século 14, no Egito, porém ele se popularizou já no século 19, na Inglaterra, onde o jogo ficou popular. Era a brincadeira favorita da hora do chá das mulheres idosas, que adoravam praticar esse passatempo pra relaxar, e é daí que vem o nome: jogo da velha!

# BENEFÍCIOS

Além de ser um passatempo ótimo, o jogo da velha também tem várias vantagens, como a melhora do raciocínio lógico, muito importante durante a vida, e da noção espacial e visual! Quando estimuladas desde cedo, crianças desenvolvem habilidades incríveis de um jeito bem rápido, ficando cada vez mais espertas.





# Descrição

## Definições Iniciais

```
#include <iostream>
#include <cstdlib>
using namespace std;

#define linhas 3
#define colunas 3
#define nada '_'

char ganhador = 'E';
char jogo[3][3];
int local;
char jogadorX = 'X';
char jogadorO = 'O';
int tabuleirotam = linhas * colunas;
```

### Definição:

- Bibliotecas: iostream, cstdlib
- Constantes e Variáveis:
  - linhas, colunas: tamanho da grade;
  - nada: caractere vazio;
  - Variáveis globais: ganhador, jogo, local, jogadorX, jogadorO, tabuleirotam;

# Descrição

FUNÇÕES PARA VERIFICAR A VITÓRIA DOS JOGADORES

01

- Objetivo: Verificar se o jogador O ou X ganhou.
- Implementação: Verifica todas as combinações possíveis de vitória (linhas, colunas, diagonais).

```
void verifvitO(){  
    if (jogo[0][0] == jogadorO && jogo[0][0] == jogo[0][2] && jogo[0][0] == jogo[0][1]){  
        cout << "\n PARABENS!! O Jogador O GANHOU!! \n";  
        ganhador = 'O';  
    }  
    // Condições adicionais para verificar outras possibilidades de vitória do jogador O  
}
```

```
void verifvitX(){  
    if (jogo[0][0] == jogadorX && jogo[0][0] == jogo[0][2] && jogo[0][0] == jogo[0][1]){  
        cout << "\n PARABÉNS!! O Jogador X GANHOU!! \n";  
        ganhador = 'X';  
    }  
    // Condições adicionais para verificar outras possibilidades de vitória do jogador X  
}
```



## INICIALIZAÇÃO DA MATRIZ DO JOGO

```
void matriz() {  
    for (int l = 0; l < 3; l++) {  
        for (int c = 0; c < 3; c++) {  
            jogo[l][c] = '_';  
        }  
    }  
}
```

- Objetivo: Inicializar a matriz do jogo com valores vazios (\_).
- Implementação: Usa um laço duplo para preencher todas as posições da matriz.

## FUNÇÃO PARA IMPRIMIR O TABULEIRO

- Objetivo: Imprimir o estado atual do tabuleiro.
- Implementação: Exibe a matriz jogo de forma organizada.

```
void imprime() {  
    cout << "=====JOGO ATUAL=====\\n";  
    cout << " " << jogo[0][0] << " | " << jogo[0][1] << " | " << jogo[0][2]  
    cout << "-----\\n";  
    cout << " " << jogo[1][0] << " | " << jogo[1][1] << " | " << jogo[1][2]  
    cout << "-----\\n";  
    cout << " " << jogo[2][0] << " | " << jogo[2][1] << " | " << jogo[2][2]  
}
```



## FUNÇÃO PARA JOGAR NOVAMENTE

```
bool jogardenovo(){
    char opcao;
    cout << "\n DESEJA JOGAR NOVAMENTE? \n";
    cin >> opcao;
    return (opcao == 'S' || opcao == 's');
}
```

- Objetivo: Perguntar ao usuário se deseja jogar novamente.
- Implementação: Lê a resposta do usuário e retorna true se a resposta for S ou s.

## DESCRICAÇÃO

## FUNÇÃO PRINCIPAL Parte 1: Configuração Inicial

```
int main() {
    do {
        matriz();
        int contjogada = 0;
        char jogador = jogadorX;
        while (true) {
            // Solicita ao jogador para selecionar uma posição
    }
}
```

- Objetivo: Configurar o jogo e iniciar o loop principal.
- Implementação: Inicializa a matriz e define o jogador atual.



# Descrição

```
01 cout << "Jogador " << jogador << ", selecione em qual local (numero) do tabuleiro voce deseja jogar.\n";
cout << "-----\n";
cout << " 1 | 2 | 3\n";
cout << "-----\n";
cout << " 4 | 5 | 6\n";
cout << "-----\n";
cout << " 7 | 8 | 9\n";
cin >> local;

if (!(local >= 1 && local <= 9)) {
    cout << "Posicao invalida. Escolha novamente!\n";
    continue;
}
```

## Função Principal Parte 2: Loop Principal

- Objetivo: Solicitar ao jogador para selecionar uma posição no tabuleiro.
- Implementação: Exibe um modelo do tabuleiro e lê a posição escolhida pelo jogador.



# Descrição

- Objetivo: Processar a jogada, atualizar o tabuleiro, verificar vitórias e trocar jogadores.
- Implementação: Usa switch para definir a posição no tabuleiro, verifica vitórias e alterna entre os jogadores.

## Função Principal Parte 3: Processamento das Jogadas

```
switch (local) {
    // Cada caso representa uma posição no tabuleiro
}

system("cls");
imprime();
verifvitX();
verifvitO();
contjogada += 1;

if (ganhador != 'E') {
    break;
}

if (contjogada == 9 && ganhador == 'E') {
    cout << "\nDeu velha!!\n";
    break;
}

jogador = (contjogada % 2 == 0) ? jogadorX : jogadorO;

} while (jogardenovo());

return 0;
}
```



# MENU

1. Selecionar em qual coordenada cada jogador fará a jogada;
2. Tabuleiro atualizado;
3. Jogar novamente.

1

2

3

Jogador X, selecione em qual local (numero) do tabuleiro deseja jogar.

MODELO DE TABULEIRO PARA ESCOLHER AS POSICOES

1		2		3
<hr/>				
4		5		6
<hr/>				
7		8		9

JOGO ATUAL

-	X	-
<hr/>		
-	0	-
<hr/>		
-		-

DESEJA JOGAR NOVAMENTE?



Jogador X, selecione em qual local (numero) do tabuleiro deseja jogar.

MODELO DE TABULEIRO PARA ESCOLHER AS POSICOES

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

Jogada invalida!! Escolha outra posicao que nao esteja ocupada!

Jogador 0, selecione em qual local (numero) do tabuleiro deseja jogar.

DESEJA JOGAR NOVAMENTE?



PARABENS!! O Jogador X GANHOU!!

Deu velha!!

# FUNCIONALIDADES

1.Jogadas - Cada jogador, iniciando pelo X, poderá selecionar um número de 1 a 9 para colocar o seu respectivo símbolo.

2.Identificação de jogada em coordenadas iguais, permitindo uma nova jogada “válida”. Caso um jogador selecione uma casa ja ocupada ou um número inválido ele poderá jogar novamente.

3.Opção de jogar o jogo novamente;

4.Reconhecimento de vencedores e empate (velha);

# CONTEÚDOS ABORDADOS

```
void verifvitX();
```

```
void verifvitO();
```

## Funções

Utilizou-se funções para imprimir o tabuleiro, substituir as coordenadas por X ou O e verificar os vencedores.

```
void matriz() {  
    for (int l = 0; l < 3; l++) {  
        for (int c = 0; c < 3; c++) {  
            jogo[l][c] = '_';  
        }  
    }  
}
```

## Matrizes

As matrizes foram utilizadas para a montagem do tabuleiro e comparações de vitória ou empate e substituição de valores em suas coordenadas.

```
while (true) {  
    :
```

## Loops

Utilizaram-se loops para com o objetivo de estabelecer o inicio e fim do jogo. Ou seja, o jogo só acaba ao fechar o loop do while.

```
if (!(local >= 1 && local <= 9)) {  
    cout << "Posicao invalida. Escolha novamente!\n";  
    continue;  
}  
  
switch (local) {  
    case 1:  
        if (jogo[0][0] != nada) {  
            system ("cls");  
            verifvitO();  
            verifvitX();  
        }  
        break;  
    case 2:  
        if (jogo[0][1] != nada) {  
            system ("cls");  
            verifvitO();  
            verifvitX();  
        }  
        break;
```

## Condicionais

If, else e switch case foram usados para verificar as condições vitoriosas e empate, além do reconhecimento de jogadas inválidas

# CONTEÚDOS ABORDADOS

Também foi utilizada a biblioteca “cstdlib” que permite a limpeza da tela do executável por meio do comando system (“cls”).

```
#include <cstdlib>
```

```
system ("cls");
```



Cada membro do grupo foi designado para exercer uma parcela do projeto computacional do jogo da velha em C++;

- Eduardo Rodrigues Santos (241024491) - Criação do código e documento de apresentação;
- Arthur Félix de Abreu (241024606) - Edição, criação do roteiro do vídeo e narração de algumas partes;
- Fellipe Ilha Gattai (241024508) - Debug do código e parte escrita dos slides;
- Ivam Arthur Adorno (232026853) - Criação dos slides e formatação dos documentos;
- Diogo Blatt (241008489) - Organização do vídeo e do repositório, auxílio na criação do código e criação do vídeo;

## DISTRIBUIÇÃO DO DESENVOLVIMENTO