Projeto Final

Gerado por Doxygen 1.12.0

Capítulo 1

UI

1.1 Objetivo

O objetivo da UI do projeto é melhorar a experiância do usuárioa ao interagir com o código em C++.
O código em C++ pode ser tanto rodado usando a UI como diretamente no terminal. Isso decorre do fato de que a integração foi feita usando arquivos WASM (pré-compilados a partir dos arquivos C++ modificados). Ou seja, de certa forma é como se o projeto com a UI seja um executável separado do projeto sem a UI.
Para executar o código com a UI, siga as instruções disponóveis mais a frente.

1.2 Solução

A solução usada para desenvolver a UI foi o REACT. Usando código em javascript, desenvolvemos uma interface gráfica para o projeto que pode ser executada diretamente em qualquer navegador.
Podemos dividir, de uma forma geral, a UI entre telas e componentes.

1.2.1 O worflow REACT

O react trabalha usando html, CSS, e JavaScript para criar páginas web.
O diferencial que o usa do REACT proporciona é a capacidade de criar "componentes": Funções que podem ser chamadas como tags html. A criação de componentes proporciona uma facilidade imensa em criar partes reutilizáveis para o site.
Além disso, o REACT proporciona diversos outros beneficios, como a renderização condicional (usando "state variables" é possível re-renderizar apenas as partes do site que sofreram mudanças, melhorando o desempenho da aplicação) e as "routes" (forma de navegar entre as páginas do site sem ter que recarregar a janela).

1.2.2 Telas

- #### Seleção de jogadores (PlayerSelection): A tela inicial da UI é a seleção de jogadores. Nela, você pode criar, selecionar e deletar jogadores.
  Essa tela usa o "Local Storage" do navegador para guardar os dados dos jogadores e mostrá-los na tela (vitórias, derrotas e empates).

- #### Seleção de jogos (Menu): Após a seleção de jogadores, uma tela para escolher os jogos é apresentada. Nessa tela, os jogadores também tem acesso a uma descrição de cada jogo.
  Além disso é possível escolher a dificuldade da IA, caso o jogador 2 escolhido seja um bot.

- #### Tabuleiro (Board): Essa tela renderiza de forma dinâmica um tabuleiro para cada jogo. Além disso, nela são chamadas as implementações das regras de cada jogo para verificar vitórias ou derrotas.
  A tela controla a verificação de vitória e derrota, as jogadas (tanto do jogador como da IA), e as regras de cada jogo.
  A verificação de vitória, derrota e as regras foram implementadas em C++ e conectadas à tela depois de serem compiladas em WASM (WebAssembly).

- #### Outras telas (DebugMenu): Por fim, a UI contém algumas telas que foram usadas para testar o código durante a fase de desenvolvimento, e não são acessíveis ao usuário.

1.2.3 Componentes

- #### Casa do tabuleiro (BoardCell): Componente usado na renderização do tabuleiro durante os jogos. Exibi uma "casa" circular que pode estar vazia, ou colorida em verde ou vermelho.
  É capaz de guardar qual jogador clicou nele, e de executar uma função personalizada quando clicado.

- #### Banner de jogo (MenuBanner): Um banner que registra uma imagem, um nome e uma descrição de um jogo.
  Tem um botão que executa uma função personalizada, e várias animações inclusas.

- #### Banner de jogador (PlayerBanner): Um banner que mostra um apelido, um nome, e uma pontuação em vários jogos.
  É capaz de executar uma função personalizada quando clicado, e tem vários parâmetros que permitem customizações adicionais (como playerType="nameless" que esconde o nome do jogador).

- #### Seletor (Selector): Um seletor que usa setas para circular entre seus conteúdos. Cada contéudo selecionado atualiza uma variável de "state" customizável.
  Cada contúdo do seletor é um objeto, que contém um nome (mostrado ao usuário), um valor (usado no código e guardado na variável de "state") e um "index" (referente a ordem que os conteúdos aparecerão no seletor).

- #### "Overlay" de vitória (WinOverlay): Um "overlay" semi-transparente que aparece sobre o tabuleiro quando um jogo acaba.
  Exibe o nome do vencedor (ou "Empate") e duas opções: Voltar à tela de seleção de jogadores ou jogar novamente.

- #### Outros componentes (dropdown): Alguns outros componentes foram uasdos apenas nas telas não acessíveis para o usuário.

1.3 Problemas encontrados durante o desenvolvimento

- #### Arquivos locais não podem ser modificados: Como a UI é executada diretamente no navegador como um site, modificar arquivos locais é impossível, devido à medidas de segurança implementadas nos navegadores e no próprio REACT. Modificar o arquivo de texto local usado para armazenar jogadores depois de cada partida é uma falha de segurança grave, pois seria equivalente a permitir que um site modifique arquivos em seu computador.
  Logo, a solução encontrada foi usar o "Local Storage": Dados em formato JSON que ficam armazenados diretamente no navegador do usuário, e podem ser acessados e modificados facilmente pelo site que os armazenou.

- #### Integração direta com o código em C++: Integrar o código do REACT com o código em C++ se provou uma dificuldade maior do que o esperado.
  Isso decorre do fato de que o C++ é uma linguagem feita para ser compilada para computadores. E todos os compildares mais conhecidos (como o G++) compilam C++ em arquivos feitos para ser executados em sistemas operacionais, diferentemente do JavaScript e do REACT, que são compilados com o objetivo de serem executados em navegadores.
  Logo a solução encontrada foi pré-compilar os arquivos de C++ em WebAssembly (WASM) usando o Emscripten. Após a compilação em arquivos WASM, os códigos em c++ poderiam ser chamadas pelo site e integrados com a UI.

- #### WebAssembly e VITE: O Vite é uma "build tool" moderna usada para testar o site durante o desenvolvimento. Ele possibilita que um servidor local aja como "Host" para o site, fazendo que não seja necessário pagar para manter um site na rede apenas para executá-lo, e foi uma ferramenta essencial durante o desenvolvimento da UI para este projeto.
  Além de possibilitar a execução do projeto, ele também é dinâmico e altera o site à medida que a "codebase" é modificada de forma quase instantânea. Entretanto, ele não possui suporte para os "wrappers" em javascript (não possuit suporte para "ES Modules" e WASM) gerados pelo Emscripten.
  Assim, foi necessário lidar diretamente com os arquivos WASM, o que gerou várias restrições na integração do C++ com o REACT. A principal delas é que passar qualquer variável mais complexa que um inteiro é extremamente difícil, forçando uma refatoração das funções que iriam ser integradas com a UI.

1.4 Como executar o projeto com a UI?

1- Instale o package managaer NPM na sua máquina.
2- Abra um terminal no diretório base desse repositório.
3- Digite `cd UI` e pressione enter.
2- Digite `npm install` e pressione enter.
3- No mesmo terminal, digite `npm run dev` e pressione enter.
4- Copie o link resultante e cole-o em um navegador de sua escolha.

IMPORTANTE: não feche a janela do terminal enquanto estiver usando a UI

Capítulo 2

Índice Hierárquico

2.1 Hierarquia de Classes

Esta lista de hierarquias está parcialmente ordenada (ordem alfabética):

Capítulo 3

Índice dos Componentes

3.1 Lista de Classes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

Capítulo 4

Índice dos Arquivos

4.1 Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

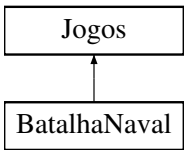
Capítulo 5

Classes

5.1 Referência da Classe BatalhaNaval

`#include <BatalhaNaval.hpp>`

Diagrama de hierarquia da classe BatalhaNaval:



Membros Públicos

- BatalhaNaval ()
- void Jogar (Jogador &Jogador1, Jogador &Jogador2) override

Membros Públicos herdados de Jogos

- virtual void mostrarTabuleiro ()

Membros protegidos

- void anunciarInicioPartida (Jogador &Jogador1, Jogador &Jogador2, bool &turno) override
- bool checarVencedor (std::vector< std::pair< int, int > > &jogadas, Jogador &vencedor, Jogador &perdedor) override
- bool checarVencedor (std::vector< std::pair< int, int > > &jogadasAtacante, std::vector< std::pair< int, int > > &barcosOponente, Jogador &vencedor, Jogador &perdedor)
- std::pair< int, int > lerJogada (std::vector< std::vector< char > > &tabuleiroJogador)
- void lerBarcos (std::vector< std::pair< int, int > > &barcosJogador, Jogador &Jogador)
- void inserirBarcos (std::vector< std::pair< int, int > > &barcosJogador, char tipo, int linhaInicial, int colunaInicial, int linhaFinal, int colunaFinal)
- bool quantidadeBarcosDisponiveis (std::map< char, int > &countBarcos, char tipo)
- bool verificarEntrada (char tipo, int linhaInicial, int colunaInicial, int linhaFinal, int colunaFinal)
- bool verificarTamanhodoBarco (char tipo, int linhaInicial, int colunaInicial, int linhaFinal, int colunaFinal)
- void checarPosicaoValida (std::vector< std::vector< char > > &tabuleiro)
- void marcarTabuleiro (std::pair< int, int > &jogada, bool &turno, std::vector< std::vector< char > > &tabuleiroJogador, std::vector< std::pair< int, int > > &barcosJogador)
- void mostrarTabuleiro (const std::vector< std::vector< char > > &tabuleiroJogador)
- bool checarEmpate (int numeroJogadas, Jogador &jogador_01, Jogador &jogador_02) override

Membros protegidos herdados de Jogos

- virtual void marcarTabuleiro (std::pair< int, int > &jogada, bool &turno)
- virtual void limparTabuleiro ()
- void anunciarTurnoJogador (Jogador &Jogador)
- virtual bool sorteioTurno ()
- virtual bool checarJogadaExistente (std::vector< std::pair< int, int > > &jogadas, int linha, int coluna)
- virtual bool checarPosicaoValida (int linha, int coluna)
- std::string gerarDivisoriaTabuleiro ()
- virtual std::pair< int, int > lerJogada ()

Outros membros herdados

Atributos Protegidos herdados de Jogos

- std::vector< std::vector< char > > tabuleiro

5.1.1 Construtores e Destrutores

5.1.1.1 BatalhaNaval()

```
BatalhaNaval::BatalhaNaval ()
```

5.1.2 Documentação das funções

5.1.2.1 anunciarInicioPartida()

```
void BatalhaNaval::anunciarInicioPartida (
            Jogador & Jogador1,
            Jogador & Jogador2,
            bool & turno)  [override], [protected], [virtual]
```

Implementa Jogos.

5.1.2.2 checarEmpate()

```
bool BatalhaNaval::checarEmpate (
            int numeroJogadas,
            Jogador & jogador_01,
            Jogador & jogador_02)  [inline], [override], [protected], [virtual]
```

Implementa Jogos.

5.1.2.3 checarPosicaoValida()

```
void BatalhaNaval::checarPosicaoValida (
            std::vector< std::vector< char > > & tabuleiro)  [protected]
```

5.1.2.4 checarVencedor() [1/2]

```
bool BatalhaNaval::checarVencedor (
            std::vector< std::pair< int, int > > & jogadas,
            Jogador & vencedor,
            Jogador & perdedor)  [inline], [override], [protected], [virtual]
```

Implementa Jogos.

5.1.2.5 checarVencedor() [2/2]

```
bool BatalhaNaval::checarVencedor (
            std::vector< std::pair< int, int > > & jogadasAtacante,
            std::vector< std::pair< int, int > > & barcosOponente,
            Jogador & vencedor,
            Jogador & perdedor)  [protected]
```

5.1.2.6 inserirBarcos()

```
void BatalhaNaval::inserirBarcos (
            std::vector< std::pair< int, int > > & barcosJogador,
            char tipo,
            int linhaInicial,
            int colunaInicial,
            int linhaFinal,
            int colunaFinal)  [protected]
```

5.1.2.7 Jogar()

```
void BatalhaNaval::Jogar (
            Jogador & Jogador1,
            Jogador & Jogador2)  [override], [virtual]
```

Reimplementa Jogos.

5.1.2.8 lerBarcos()

```
void BatalhaNaval::lerBarcos (
            std::vector< std::pair< int, int > > & barcosJogador,
            Jogador & Jogador)  [protected]
```

5.1.2.9 lerJogada()

```
std::pair< int, int > BatalhaNaval::lerJogada (
            std::vector< std::vector< char > > & tabuleiroJogador)  [protected]
```

5.1.2.10 marcarTabuleiro()

```
void BatalhaNaval::marcarTabuleiro (
            std::pair< int, int > & jogada,
            bool & turno,
            std::vector< std::vector< char > > & tabuleiroJogador,
            std::vector< std::pair< int, int > > & barcosJogador)  [protected]
```

5.1.2.11 mostrarTabuleiro()

```
void BatalhaNaval::mostrarTabuleiro (
            const std::vector< std::vector< char > > & tabuleiroJogador)  [protected]
```

5.1.2.12 quantidadeBarcosDisponiveis()

```
bool BatalhaNaval::quantidadeBarcosDisponiveis (
            std::map< char, int > & countBarcos,
            char tipo)  [protected]
```

5.1.2.13 verificarEntrada()

```
bool BatalhaNaval::verificarEntrada (
            char tipo,
            int linhaInicial,
            int colunaInicial,
            int linhaFinal,
            int colunaFinal)  [protected]
```

5.1.2.14 verificarTamanhodoBarco()

```
bool BatalhaNaval::verificarTamanhodoBarco (
            char tipo,
            int linhaInicial,
            int colunaInicial,
            int linhaFinal,
            int colunaFinal)  [protected]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/BatalhaNaval.hpp
- src/BatalhaNaval.cpp

5.2 Referência da Classe CentralDeJogos

```
#include <CentralDeJogos.hpp>
```

Membros Públicos

- CentralDeJogos ()
- ~CentralDeJogos ()
- std::string validarEntrada ()
- bool buscarJogador (std::string &apelido)
- void cadastrarJogador (std::string &apelido, std::string &nome)
- void removerJogador (std::string &apelido)
- void ordenarJogadores ()
- void listarJogadores ()
- void executarPartida ()

Atributos Privados

- std::vector< Jogador > jogadoresCadastrados
- JogoDaVelhaAi Ai
- JogoDaVelha velha
- Lig4 lig4
- Reversi reversi
- BatalhaNaval batalha

5.2.1 Construtores e Destrutores

5.2.1.1 CentralDeJogos()

```
CentralDeJogos::CentralDeJogos ()
```

5.2.1.2 ~CentralDeJogos()

```
CentralDeJogos::~CentralDeJogos ()
```

5.2.2 Documentação das funções

5.2.2.1 buscarJogador()

```
bool CentralDeJogos::buscarJogador (
            std::string & apelido)
```

5.2.2.2 cadastrarJogador()

```
void CentralDeJogos::cadastrarJogador (
            std::string & apelido,
            std::string & nome)
```

5.2.2.3 executarPartida()

```
void CentralDeJogos::executarPartida ()
```

5.2.2.4 listarJogadores()

```
void CentralDeJogos::listarJogadores ()
```

5.2.2.5 ordenarJogadores()

```
void CentralDeJogos::ordenarJogadores ()
```

5.2.2.6 removerJogador()

```
void CentralDeJogos::removerJogador (
            std::string & apelido)
```

5.2.2.7 validarEntrada()

```
std::string CentralDeJogos::validarEntrada ()
```

5.2.3 Atributos

5.2.3.1 Ai

```
JogoDaVelhaAi CentralDeJogos::Ai  [private]
```

5.2.3.2 batalha

```
BatalhaNaval CentralDeJogos::batalha  [private]
```

5.2.3.3 jogadoresCadastrados

```
std::vector<Jogador> CentralDeJogos::jogadoresCadastrados  [private]
```

5.2.3.4 lig4

```
Lig4 CentralDeJogos::lig4  [private]
```

5.2.3.5 reversi

```
Reversi CentralDeJogos::reversi  [private]
```

5.2.3.6 velha

```
JogoDaVelha CentralDeJogos::velha  [private]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/CentralDeJogos.hpp
- src/CentralDeJogos.cpp

5.3 Referência da Classe Estatisticas

```
#include <Estatisticas.hpp>
```

Membros Públicos

- Estatisticas ()
- Estatisticas (int vitorias, int derrotas, int empates)
- void registrarVitoria ()
- void registrarDerrota ()
- void registrarEmpate ()
- int getVitorias () const
- int getDerrotas () const
- int getEmpates () const
- std::vector< char > getHistorico () const
- void mostrarEstatisticas () const

Atributos Privados

- int vitorias
- int derrotas
- int empates

5.3.1 Construtores e Destrutores

5.3.1.1 Estatisticas() [1/2]

```
Estatisticas::Estatisticas ()
```

5.3.1.2 Estatisticas() [2/2]

```
Estatisticas::Estatisticas (
            int vitorias,
            int derrotas,
            int empates)
```

5.3.2 Documentação das funções

5.3.2.1 getDerrotas()

```
int Estatisticas::getDerrotas () const
```

5.3.2.2 getEmpates()

```
int Estatisticas::getEmpates () const
```

5.3.2.3 getHistorico()

```
std::vector< char > Estatisticas::getHistorico () const
```

5.3.2.4 getVitorias()

```
int Estatisticas::getVitorias () const
```

5.3.2.5 mostrarEstatisticas()

```
void Estatisticas::mostrarEstatisticas () const
```

5.3.2.6 registrarDerrota()

```
void Estatisticas::registrarDerrota ()
```

5.3.2.7 registrarEmpate()

```
void Estatisticas::registrarEmpate ()
```

5.3.2.8 registrarVitoria()

```
void Estatisticas::registrarVitoria ()
```

5.3.3 Atributos

5.3.3.1 derrotas

```
int Estatisticas::derrotas  [private]
```

5.3.3.2 empates

```
int Estatisticas::empates  [private]
```

5.3.3.3 vitorias

```
int Estatisticas::vitorias  [private]
```

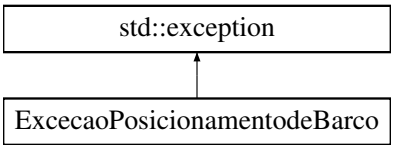
A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/Estatisticas.hpp
- src/Estatisticas.cpp

5.4 Referência da Classe ExcecaoPosicionamentodeBarco

```
#include <BatalhaNaval.hpp>
```

Diagrama de hierarquia da classe ExcecaoPosicionamentodeBarco:



Membros Públicos

- const char ∗ what () const override throw ()

5.4.1 Documentação das funções

5.4.1.1 what()

```
const char * ExcecaoPosicionamentodeBarco::what () const throw ( )   [inline], [override]
```

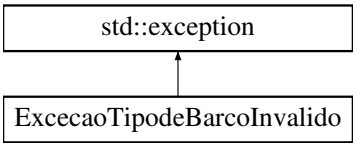
A documentação para essa classe foi gerada a partir do seguinte arquivo:

- include/BatalhaNaval.hpp

5.5 Referência da Classe ExcecaoTipodeBarcoInvalido

```
#include <BatalhaNaval.hpp>
```

Diagrama de hierarquia da classe ExcecaoTipodeBarcoInvalido:



Membros Públicos

- const char * what () const override throw ()

5.5.1 Documentação das funções

5.5.1.1 what()

```
const char * ExcecaoTipodeBarcoInvalido::what () const throw ( )   [inline], [override]
```

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- include/BatalhaNaval.hpp

5.6 Referência da Classe Jogador

```
#include <Jogador.hpp>
```

Membros Públicos

- Jogador (const std::string &apelido, const std::string &nome)
- Jogador (const std::string &apelido, const std::string &nome, int vitoriasJogoDaVelha, int derrotasJogo↩ DaVelha, int empatesJogoDavelha, int vitoriasLig4, int derrotasLig4, int empatesLig4, int vitoriasReversi, int derrotasReversi, int empatesReversi, int vitoriasBatalhaNaval, int derrotasBatalhaNaval, int empates↩ BatalhaNaval)
- void registrarVitoria (const std::string &nomeJogo)
- void registrarDerrota (const std::string &nomeJogo)
- void registrarEmpate (const std::string &nomeJogo)
- std::string getApelido () const
- std::string getNome () const
- int getVitorias (std::string jogo)
- int getDerrotas (std::string jogo)
- int getEmpates (std::string jogo)
- void mostrarEstatisticas (const std::string &nomeJogo) const

Atributos Privados

- std::string apelido
- std::string nome
- std::unordered_map< std::string, Estatisticas > estatisticasPorJogo

5.6.1 Construtores e Destrutores

5.6.1.1 Jogador() [1/2]

```
Jogador::Jogador (
            const std::string & apelido,
            const std::string & nome)
```

5.6.1.2 Jogador() [2/2]

```
Jogador::Jogador (
            const std::string & apelido,
            const std::string & nome,
            int vitoriasJogoDaVelha,
            int derrotasJogoDaVelha,
            int empatesJogoDavelha,
            int vitoriasLig4,
            int derrotasLig4,
            int empatesLig4,
            int vitoriasReversi,
            int derrotasReversi,
            int empatesReversi,
            int vitoriasBatalhaNaval,
            int derrotasBatalhaNaval,
            int empatesBatalhaNaval)
```

5.6.2 Documentação das funções

5.6.2.1 getApelido()

```
std::string Jogador::getApelido () const
```

5.6.2.2 getDerrotas()

```
int Jogador::getDerrotas (
            std::string jogo)
```

5.6.2.3 getEmpates()

```
int Jogador::getEmpates (
            std::string jogo)
```

5.6.2.4 getNome()

```
std::string Jogador::getNome () const
```

5.6.2.5 getVitorias()

```
int Jogador::getVitorias (
            std::string jogo)
```

5.6.2.6 mostrarEstatisticas()

```
void Jogador::mostrarEstatisticas (
            const std::string & nomeJogo) const
```

5.6.2.7 registrarDerrota()

```
void Jogador::registrarDerrota (
            const std::string & nomeJogo)
```

5.6.2.8 registrarEmpate()

```
void Jogador::registrarEmpate (
            const std::string & nomeJogo)
```

5.6.2.9 registrarVitoria()

```
void Jogador::registrarVitoria (
            const std::string & nomeJogo)
```

5.6.3 Atributos

5.6.3.1 apelido

```
std::string Jogador::apelido  [private]
```

5.6.3.2 estatisticasPorJogo

```
std::unordered_map<std::string, Estatisticas> Jogador::estatisticasPorJogo  [private]
```

5.6.3.3 nome

```
std::string Jogador::nome  [private]
```

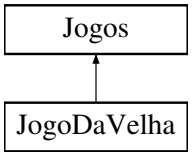
A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/Jogador.hpp
- src/Jogador.cpp

5.7 Referência da Classe JogoDaVelha

```
#include <JogoDaVelha.hpp>
```

Diagrama de hierarquia da classe JogoDaVelha:



Membros Públicos

- JogoDaVelha ()
- JogoDaVelha (int tamanhoTabuleiro)

Membros Públicos herdados de Jogos

- virtual void mostrarTabuleiro ()
- virtual void Jogar (Jogador &Jogador1, Jogador &Jogador2)

Membros protegidos

- void anunciarInicioPartida (Jogador &Jogador1, Jogador &Jogador2, bool &turno) override
- std::pair< int, int > lerJogada () override
- bool checarDiagonal (std::vector< std::pair< int, int > > &jogadas)
- bool checarColunas (std::vector< std::pair< int, int > > &jogadas)
- bool checarLinhas (std::vector< std::pair< int, int > > &jogadas)
- bool checarVencedor (std::vector< std::pair< int, int > > &jogadas, Jogador &vencedor, Jogador &perdedor) override
- bool checarEmpate (int numeroJogadas, Jogador &jogador_01, Jogador &jogador_02) override

Membros protegidos herdados de Jogos

- virtual void marcarTabuleiro (std::pair< int, int > &jogada, bool &turno)
- virtual void limparTabuleiro ()
- void anunciarTurnoJogador (Jogador &Jogador)
- virtual bool sorteioTurno ()
- virtual bool checarJogadaExistente (std::vector< std::pair< int, int > > &jogadas, int linha, int coluna)
- virtual bool checarPosicaoValida (int linha, int coluna)
- std::string gerarDivisoriaTabuleiro ()

Amigos

- class JogoDaVelhaAi

Outros membros herdados

Atributos Protegidos herdados de Jogos

- std::vector< std::vector< char > > tabuleiro

5.7.1 Construtores e Destrutores

5.7.1.1 JogoDaVelha() [1/2]

```
JogoDaVelha::JogoDaVelha ()
```

5.7.1.2 JogoDaVelha() [2/2]

```
JogoDaVelha::JogoDaVelha (
            int tamanhoTabuleiro)
```

5.7.2 Documentação das funções

5.7.2.1 anunciarInicioPartida()

```
void JogoDaVelha::anunciarInicioPartida (
            Jogador & Jogador1,
            Jogador & Jogador2,
            bool & turno)  [override], [protected], [virtual]
```

Implementa Jogos.

5.7.2.2 checarColunas()

```
bool JogoDaVelha::checarColunas (
            std::vector< std::pair< int, int > > & jogadas)  [protected]
```

5.7.2.3 checarDiagonal()

```
bool JogoDaVelha::checarDiagonal (
            std::vector< std::pair< int, int > > & jogadas)  [protected]
```

5.7.2.4 checarEmpate()

```
bool JogoDaVelha::checarEmpate (
            int numeroJogadas,
            Jogador & jogador_01,
            Jogador & jogador_02)  [override], [protected], [virtual]
```

Implementa Jogos.

5.7.2.5 checarLinhas()

```
bool JogoDaVelha::checarLinhas (
            std::vector< std::pair< int, int > > & jogadas)  [protected]
```

5.7.2.6 checarVencedor()

```
bool JogoDaVelha::checarVencedor (
            std::vector< std::pair< int, int > > & jogadas,
            Jogador & vencedor,
            Jogador & perdedor)  [override], [protected], [virtual]
```

Implementa Jogos.

5.7.2.7 lerJogada()

```
std::pair< int, int > JogoDaVelha::lerJogada ()  [override], [protected], [virtual]
```

Reimplementa Jogos.

5.7.3 Documentação dos símbolos amigos e relacionados

5.7.3.1 JogoDaVelhaAi

```
friend class JogoDaVelhaAi  [friend]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/JogoDaVelha.hpp
- src/JogoDaVelha.cpp

5.8 Referência da Classe JogoDaVelhaAi

```
#include <JogoDaVelhaAi.hpp>
```

Membros Públicos

- JogoDaVelhaAi ()
  Construtor que inicializa o tabuleiro com células vazias.
- void Jogar (Jogador &Jogador1, Jogador &Jogador2)
  Controla o fluxo principal do jogo.

Membros privados

- bool checarVitoria (char jogador) const
  Verifica se o jogador especificado alcançou uma condição de vitória.
- bool isTabuleiroCheio () const
  Verifica se todas as posições do tabuleiro estão ocupadas.
- int minimax (bool isMaximizador, int profundidade)
  Implementa o algoritmo Minimax para avaliação de jogadas.
- int getMelhorMovimento ()
  Calcula a melhor jogada para a IA usando Minimax.
- std::pair< int, int > jogarHumano (bool turno)
  Processa e registra uma jogada do jogador humano.
- std::pair< int, int > jogarAI (bool turno)
  Calcula e executa a jogada da IA.

Atributos Privados

- int MAX_PROFUNDIDADE
- JogoDaVelha jogo
- std::vector< char > tabuleiro

5.8.1 Construtores e Destrutores

5.8.1.1 JogoDaVelhaAi()

```
JogoDaVelhaAi::JogoDaVelhaAi ()
```

Construtor que inicializa o tabuleiro com células vazias.

5.8.2 Documentação das funções

5.8.2.1 checarVitoria()

```
bool JogoDaVelhaAi::checarVitoria (
            char jogador) const  [private]
```

Verifica se o jogador especificado alcançou uma condição de vitória.

Checa todas as combinações possíveis de vitória (linhas, colunas e diagonais)

Parâmetros

player	Jogador a ser verificado (JOGADOR_X ou JOGADOR_O)

Retorna

true Se o jogador tem três símbolos consecutivos em alguma linha/coluna/diagonal

false Caso não haja vitória

5.8.2.2 getMelhorMovimento()

```
int JogoDaVelhaAi::getMelhorMovimento ()  [private]
```

Calcula a melhor jogada para a IA usando Minimax.

Utiliza uma ordem otimizada de verificação de movimentos (cantos primeiro, centro depois, bordas por último) para acelerar a busca pela jogada ideal e para evitar jogadas subótimas, uma vez que, como jogadores perfeitos sempre empatam, uma má ordenação dos movimentos pode dificultar a IA na escolha da melhor jogada.

Retorna

int Índice da melhor jogada no tabuleiro (0-8)

5.8.2.3 isTabuleiroCheio()

```
bool JogoDaVelhaAi::isTabuleiroCheio () const  [private]
```

Verifica se todas as posições do tabuleiro estão ocupadas.

Retorna

true Se não há mais espaços vazios no tabuleiro

false Se há pelo menos um espaço vazio restante

5.8.2.4 Jogar()

```
void JogoDaVelhaAi::Jogar (
            Jogador & Jogador1,
            Jogador & Jogador2)
```

Controla o fluxo principal do jogo.

Gerencia todo o ciclo de vida do jogo, incluindo:

- Configuração inicial de jogadores e dificuldade
- Alternância de turnos entre jogadores
- Verificação de condições de término (vitória/empate)
- Reinicialização do jogo ao final

Parâmetros

Jogador1	Primeiro jogador (normalmente humano)
Jogador2	Segundo jogador (normalmente IA)

5.8.2.5 jogarAI()

```
std::pair< int, int > JogoDaVelhaAi::jogarAI (
            bool turno)  [private]
```

Calcula e executa a jogada da IA.

A jogada é marcada no tabuleiro da AI e no objeto JogoDaVelha, uma vez que a exibição do jogo depende do estado do tabuleiro do objeto JogoDaVelha.

Parâmetros

turno	Indica de qual jogador é o turno (não utilizado na implementação atual)

Retorna

std::pair<int, int> Coordenadas (linha, coluna) da jogada

5.8.2.6 jogarHumano()

```
std::pair< int, int > JogoDaVelhaAi::jogarHumano (
            bool turno)  [private]
```

Processa e registra uma jogada do jogador humano.

A jogada é marcada no tabuleiro da AI e no objeto JogoDaVelha, uma vez que a exibição do jogo depende do estado do tabuleiro do objeto JogoDaVelha.

Parâmetros

turno	Indica de qual jogador é o turno (não utilizado na implementação atual)

Retorna

std::pair<int, int> Coordenadas (linha, coluna) da jogada

5.8.2.7 minimax()

```
int JogoDaVelhaAi::minimax (
            bool isMaximizador,
            int profundidade)  [private]
```

Implementa o algoritmo Minimax para avaliação de jogadas.

Avalia recursivamente todas as jogadas possíveis até a profundidade máxima configurada, alternando entre jogadores de maximização (IA) e minimização (jogador humano)

Parâmetros

isMaximizador	Indica se é o turno do jogador maximizador (IA)
profundidade	Profundidade atual da recursão

Retorna

int Valor heurístico da posição (1 para vitória IA, -1 para derrota, 0 para neutro)

5.8.3 Atributos

5.8.3.1 jogo

```
JogoDaVelha JogoDaVelhaAi::jogo  [private]
```

5.8.3.2 MAX_PROFUNDIDADE

```
int JogoDaVelhaAi::MAX_PROFUNDIDADE  [private]
```

5.8.3.3 tabuleiro

```
std::vector<char> JogoDaVelhaAi::tabuleiro  [private]
```

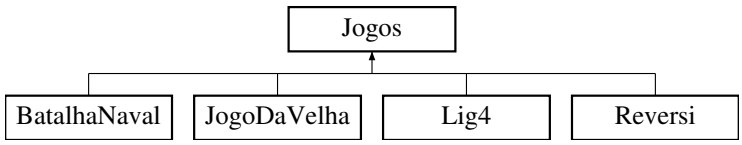
A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/JogoDaVelhaAi.hpp
- src/JogoDaVelhaAi.cpp

5.9 Referência da Classe Jogos

`#include <Jogos.hpp>`

Diagrama de hierarquia da classe Jogos:



Membros Públicos

- virtual void mostrarTabuleiro ()
- virtual void Jogar (Jogador &Jogador1, Jogador &Jogador2)

Membros protegidos

- virtual bool checarVencedor (std::vector< std::pair< int, int > > &jogadas, Jogador &vencedor, Jogador &perdedor)=0
- virtual bool checarEmpate (int numeroJogadas, Jogador &jogador_01, Jogador &jogador_02)=0
- virtual void anunciarInicioPartida (Jogador &Jogador1, Jogador &Jogador2, bool &turno)=0
- virtual void marcarTabuleiro (std::pair< int, int > &jogada, bool &turno)
- virtual void limparTabuleiro ()
- void anunciarTurnoJogador (Jogador &Jogador)
- virtual bool sorteioTurno ()
- virtual bool checarJogadaExistente (std::vector< std::pair< int, int > > &jogadas, int linha, int coluna)
- virtual bool checarPosicaoValida (int linha, int coluna)
- std::string gerarDivisoriaTabuleiro ()
- virtual std::pair< int, int > lerJogada ()

Atributos Protegidos

- std::vector< std::vector< char > > tabuleiro

5.9.1 Documentação das funções

5.9.1.1 anunciarInicioPartida()

```
virtual void Jogos::anunciarInicioPartida (
            Jogador & Jogador1,
            Jogador & Jogador2,
            bool & turno)  [protected], [pure virtual]
```

Implementado por BatalhaNaval, JogoDaVelha, Lig4 e Reversi.

5.9.1.2 anunciarTurnoJogador()

```
void Jogos::anunciarTurnoJogador (
            Jogador & Jogador)  [protected]
```

5.9.1.3 checarEmpate()

```
virtual bool Jogos::checarEmpate (
            int numeroJogadas,
            Jogador & jogador_01,
            Jogador & jogador_02)  [protected], [pure virtual]
```

Implementado por BatalhaNaval, JogoDaVelha, Lig4 e Reversi.

5.9.1.4 checarJogadaExistente()

```
bool Jogos::checarJogadaExistente (
            std::vector< std::pair< int, int > > & jogadas,
            int linha,
            int coluna)  [protected], [virtual]
```

5.9.1.5 checarPosicaoValida()

```
bool Jogos::checarPosicaoValida (
            int linha,
            int coluna)  [protected], [virtual]
```

5.9.1.6 checarVencedor()

```
virtual bool Jogos::checarVencedor (
            std::vector< std::pair< int, int > > & jogadas,
            Jogador & vencedor,
            Jogador & perdedor)  [protected], [pure virtual]
```

Implementado por BatalhaNaval, JogoDaVelha, Lig4 e Reversi.

5.9.1.7 gerarDivisoriaTabuleiro()

```
std::string Jogos::gerarDivisoriaTabuleiro ()  [protected]
```

5.9.1.8 Jogar()

```
void Jogos::Jogar (
            Jogador & Jogador1,
            Jogador & Jogador2)  [virtual]
```

Reimplementado por BatalhaNaval e Reversi.

5.9.1.9 lerJogada()

```
virtual std::pair< int, int > Jogos::lerJogada ()  [inline], [protected], [virtual]
```

Reimplementado por JogoDaVelha e Lig4.

5.9.1.10 limparTabuleiro()

```
void Jogos::limparTabuleiro ()  [protected], [virtual]
```

Reimplementado por Reversi.

5.9.1.11 marcarTabuleiro()

```
void Jogos::marcarTabuleiro (
            std::pair< int, int > & jogada,
            bool & turno)  [protected], [virtual]
```

Reimplementado por Reversi.

5.9.1.12 mostrarTabuleiro()

```
void Jogos::mostrarTabuleiro ()  [virtual]
```

5.9.1.13 sorteioTurno()

```
bool Jogos::sorteioTurno ()  [protected], [virtual]
```

5.9.2 Atributos

5.9.2.1 tabuleiro

```
std::vector<std::vector<char> > Jogos::tabuleiro  [protected]
```

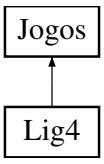
A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/Jogos.hpp
- src/Jogos.cpp

5.10 Referência da Classe Lig4

```
#include <Lig4.hpp>
```

Diagrama de hierarquia da classe Lig4:



Membros Públicos

- Lig4 (int tamanhoTabuleiro)
- Lig4 ()

Membros Públicos herdados de Jogos

- virtual void mostrarTabuleiro ()
- virtual void Jogar (Jogador &Jogador1, Jogador &Jogador2)

Membros protegidos

- void anunciarInicioPartida (Jogador &Jogador1, Jogador &Jogador2, bool &turno) override
- bool checarDiagonal (std::vector< std::pair< int, int > > &jogadas)
- bool checarColunas (std::vector< std::pair< int, int > > &jogadas)
- bool checarLinhas (std::vector< std::pair< int, int > > &jogadas)
- bool checarVencedor (std::vector< std::pair< int, int > > &jogadas, Jogador &vencedor, Jogador &perdedor) override
- bool checarEmpate (int numeroJogadas, Jogador &jogador_01, Jogador &jogador_02) override
- std::pair< int, int > lerJogada () override

Membros protegidos herdados de Jogos

- virtual void marcarTabuleiro (std::pair< int, int > &jogada, bool &turno)
- virtual void limparTabuleiro ()
- void anunciarTurnoJogador (Jogador &Jogador)
- virtual bool sorteioTurno ()
- virtual bool checarJogadaExistente (std::vector< std::pair< int, int > > &jogadas, int linha, int coluna)
- virtual bool checarPosicaoValida (int linha, int coluna)
- std::string gerarDivisoriaTabuleiro ()

Outros membros herdados

Atributos Protegidos herdados de Jogos

- std::vector< std::vector< char > > tabuleiro

5.10.1 Construtores e Destrutores

5.10.1.1 Lig4() [1/2]

```
Lig4::Lig4 (
            int tamanhoTabuleiro)
```

5.10.1.2 Lig4() [2/2]

```
Lig4::Lig4 ()
```

5.10.2 Documentação das funções

5.10.2.1 anunciarInicioPartida()

```
void Lig4::anunciarInicioPartida (
            Jogador & Jogador1,
            Jogador & Jogador2,
            bool & turno)  [override], [protected], [virtual]
```

Implementa Jogos.

5.10.2.2 checarColunas()

```
bool Lig4::checarColunas (
            std::vector< std::pair< int, int > > & jogadas)  [protected]
```

5.10.2.3 checarDiagonal()

```
bool Lig4::checarDiagonal (
            std::vector< std::pair< int, int > > & jogadas)  [protected]
```

5.10.2.4 checarEmpate()

```
bool Lig4::checarEmpate (
            int numeroJogadas,
            Jogador & jogador_01,
            Jogador & jogador_02)  [override], [protected], [virtual]
```

Implementa Jogos.

5.10.2.5 checarLinhas()

```
bool Lig4::checarLinhas (
            std::vector< std::pair< int, int > > & jogadas)  [protected]
```

5.10.2.6 checarVencedor()

```
bool Lig4::checarVencedor (
            std::vector< std::pair< int, int > > & jogadas,
            Jogador & vencedor,
            Jogador & perdedor)  [override], [protected], [virtual]
```

Implementa Jogos.

5.10.2.7 lerJogada()

```
std::pair< int, int > Lig4::lerJogada ()  [override], [protected], [virtual]
```

Reimplementa Jogos.

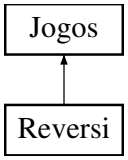
A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/Lig4.hpp
- src/Lig4.cpp

5.11 Referência da Classe Reversi

```
#include <Reversi.hpp>
```

Diagrama de hierarquia da classe Reversi:



Membros Públicos

- Reversi ()
- Reversi (int tamanhoTabuleiro)
- void Jogar (Jogador &Jogador1, Jogador &Jogador2) override

Membros Públicos herdados de Jogos

- virtual void mostrarTabuleiro ()

Membros protegidos

- void anunciarInicioPartida (Jogador &Jogador1, Jogador &Jogador2, bool &turno) override
- bool jogadorInicial (bool &turno)
- std::pair< int, int > lerJogada (bool turno)
- void marcarTabuleiro (std::pair< int, int > &jogada, bool &turno) override
- bool movimentoValido (std::pair< int, int > &jogada, char jogador, std::vector< std::pair< int, int > > &flips)
- bool haMovimentosDisponiveis (char Jogador)
- void limparTabuleiro () override
- bool checarVencedor (std::vector< std::pair< int, int > > &jogadas, Jogador &vencedor, Jogador &perdedor)
- bool checarVencedor (std::vector< std::pair< int, int > > &jogadas, Jogador &jogador1, Jogador &Jogador2, bool &PrimeiroJogador)
- bool checarEmpate (int numeroJogadas, Jogador &jogador_01, Jogador &jogador_02) override

Membros protegidos herdados de Jogos

- void anunciarTurnoJogador (Jogador &Jogador)
- virtual bool sorteioTurno ()
- virtual bool checarJogadaExistente (std::vector< std::pair< int, int > > &jogadas, int linha, int coluna)
- virtual bool checarPosicaoValida (int linha, int coluna)
- std::string gerarDivisoriaTabuleiro ()
- virtual std::pair< int, int > lerJogada ()

Outros membros herdados

Atributos Protegidos herdados de Jogos

- std::vector< std::vector< char > > tabuleiro

5.11.1 Construtores e Destrutores

5.11.1.1 Reversi() [1/2]

```
Reversi::Reversi ()
```

5.11.1.2 Reversi() [2/2]

```
Reversi::Reversi (
            int tamanhoTabuleiro)
```

5.11.2 Documentação das funções

5.11.2.1 anunciarInicioPartida()

```
void Reversi::anunciarInicioPartida (
            Jogador & Jogador1,
            Jogador & Jogador2,
            bool & turno)  [override], [protected], [virtual]
```

Implementa Jogos.

5.11.2.2 checarEmpate()

```
bool Reversi::checarEmpate (
            int numeroJogadas,
            Jogador & jogador_01,
            Jogador & jogador_02)  [inline], [override], [protected], [virtual]
```

Implementa Jogos.

5.11.2.3 checarVencedor() [1/2]

```
bool Reversi::checarVencedor (
            std::vector< std::pair< int, int > > & jogadas,
            Jogador & jogador1,
            Jogador & Jogador2,
            bool & PrimeiroJogador)  [protected]
```

5.11.2.4 checarVencedor() [2/2]

```
bool Reversi::checarVencedor (
            std::vector< std::pair< int, int > > & jogadas,
            Jogador & vencedor,
            Jogador & perdedor)  [inline], [protected], [virtual]
```

Implementa Jogos.

5.11.2.5 haMovimentosDisponiveis()

```
bool Reversi::haMovimentosDisponiveis (
            char Jogador)  [protected]
```

5.11.2.6 jogadorInicial()

```
bool Reversi::jogadorInicial (
            bool & turno)  [protected]
```

5.11.2.7 Jogar()

```
void Reversi::Jogar (
            Jogador & Jogador1,
            Jogador & Jogador2)  [override], [virtual]
```

Reimplementa Jogos.

5.11.2.8 lerJogada()

```
std::pair< int, int > Reversi::lerJogada (
            bool turno)  [protected]
```

5.11.2.9 limparTabuleiro()

```
void Reversi::limparTabuleiro ()  [override], [protected], [virtual]
```

Reimplementa Jogos.

5.11.2.10 marcarTabuleiro()

```
void Reversi::marcarTabuleiro (
            std::pair< int, int > & jogada,
            bool & turno)  [override], [protected], [virtual]
```

Reimplementa Jogos.

5.11.2.11 movimentoValido()

```
bool Reversi::movimentoValido (
            std::pair< int, int > & jogada,
            char jogador,
            std::vector< std::pair< int, int > > & flips)  [protected]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/Reversi.hpp
- src/Reversi.cpp

Capítulo 6

Arquivos

6.1 Referência do Arquivo include/BatalhaNaval.hpp

```
#include "Jogos.hpp"
```

Componentes

- class BatalhaNaval
- class ExcecaoTipodeBarcoInvalido
- class ExcecaoPosicionamentodeBarco

6.2 BatalhaNaval.hpp

Ir para a documentação desse arquivo.

```
00001 #ifndef BATALHA_NAVAL
00002 #define BATALHA_NAVAL
00003
00004 #include "Jogos.hpp"
00005
00006 class BatalhaNaval : public Jogos
00007 {
00008   public:
00009     BatalhaNaval();
00010     void Jogar(Jogador &Jogador1, Jogador &Jogador2) override;
00011
00012   protected:
00013     void anunciarInicioPartida(Jogador &Jogador1, Jogador &Jogador2, bool &turno) override;
00014
00015     bool checarVencedor(std::vector<std::pair<int, int» &jogadas, Jogador &vencedor, Jogador
      &perdedor) override
00016     {
00017         return false;
00018     };
00019     bool checarVencedor(std::vector<std::pair<int, int» &jogadasAtacante,
00020                         std::vector<std::pair<int, int» &barcosOponente, Jogador &vencedor, Jogador
      &perdedor);
00021     std::pair<int, int> lerJogada(std::vector<std::vector<char» &tabuleiroJogador);
00022
00023     void lerBarcos(std::vector<std::pair<int, int» &barcosJogador, Jogador &Jogador);
00024     void inserirBarcos(std::vector<std::pair<int, int» &barcosJogador, char tipo, int linhaInicial,
      int colunaInicial,
00025                        int linhaFinal, int colunaFinal);
00026     bool quantidadeBarcosDisponiveis(std::map<char, int> &countBarcos, char tipo);
00027     bool verificarEntrada(char tipo, int linhaInicial, int colunaInicial, int linhaFinal, int
      colunaFinal);
```

```
00028     bool verificarTamanhodoBarco(char tipo, int linhaInicial, int colunaInicial, int linhaFinal, int
      colunaFinal);
00029
00030     void checarPosicaoValida(std::vector<std::vector<char> &tabuleiro);
00031     void marcarTabuleiro(std::pair<int, int> &jogada, bool &turno, std::vector<std::vector<char>
      &tabuleiroJogador,
00032                          std::vector<std::pair<int, int> &barcosJogador);
00033     void mostrarTabuleiro(const std::vector<std::vector<char> &tabuleiroJogador);
00034     bool checarEmpate(int numeroJogadas, Jogador &jogador_01, Jogador &jogador_02) override { return
      false; };
00035 };
00036
00037 class ExcecaoTipodeBarcoInvalido : public std::exception
00038 {
00039   public:
00040     const char *what() const throw() override { return "ERRO! Tipo de barco invalido."; }
00041 };
00042
00043 class ExcecaoPosicionamentodeBarco : public std::exception
00044 {
00045   public:
00046     const char *what() const throw() override
00047     {
00048         return "ERRO! Barcos devem ser colocados horizontalmente ou verticalmente.";
00049     }
00050 };
00051
00052 #endif
```

6.3 Referência do Arquivo include/CentralDeJogos.hpp

```
#include "BatalhaNaval.hpp"
#include "JogoDaVelha.hpp"
#include "JogoDaVelhaAi.hpp"
#include "Jogos.hpp"
#include "Lig4.hpp"
#include "Reversi.hpp"
```

Componentes

- class CentralDeJogos

6.4 CentralDeJogos.hpp

Ir para a documentação desse arquivo.

```
00001 #ifndef CENTRALDEJOGOS_HPP
00002 #define CENTRALDEJOGOS_HPP
00003
00004 #include "BatalhaNaval.hpp"
00005 #include "JogoDaVelha.hpp"
00006 #include "JogoDaVelhaAi.hpp"
00007 #include "Jogos.hpp"
00008 #include "Lig4.hpp"
00009 #include "Reversi.hpp"
00010
00011 class CentralDeJogos
00012 {
00013   private:
00014     std::vector<Jogador> jogadoresCadastrados;
00015
00016     JogoDaVelhaAi Ai;
00017     JogoDaVelha velha;
00018     Lig4 lig4;
00019     Reversi reversi;
00020     BatalhaNaval batalha;
00021
00022   public:
```

```
00023     CentralDeJogos();
00024     ~CentralDeJogos();
00025
00026     std::string validarEntrada();
00027
00028     bool buscarJogador(std::string &apelido);
00029
00030     void cadastrarJogador(std::string &apelido, std::string &nome);
00031     void removerJogador(std::string &apelido);
00032     void ordenarJogadores();
00033     void listarJogadores();
00034
00035     void executarPartida();
00036 };
00037
00038 #endif
```

6.5 Referência do Arquivo include/Estatisticas.hpp

```
#include <vector>
```

Componentes

- class Estatisticas

6.6 Estatisticas.hpp

Ir para a documentação desse arquivo.

```
00001 #ifndef ESTATISTICAS_HPP
00002 #define ESTATISTICAS_HPP
00003
00004 #include <vector>
00005
00006 class Estatisticas
00007 {
00008   private:
00009     int vitorias;
00010     int derrotas;
00011     int empates;
00012
00013   public:
00014     Estatisticas();
00015     Estatisticas(int vitorias, int derrotas, int empates);
00016
00017     void registrarVitoria();
00018     void registrarDerrota();
00019     void registrarEmpate();
00020
00021     int getVitorias() const;
00022     int getDerrotas() const;
00023     int getEmpates() const;
00024     std::vector<char> getHistorico() const;
00025     void mostrarEstatisticas() const;
00026 };
00027
00028 #endif
```

6.7 Referência do Arquivo include/Jogador.hpp

```
#include <map>
#include <string>
#include <unordered_map>
#include "Estatisticas.hpp"
```

Componentes

- class Jogador

6.8 Jogador.hpp

Ir para a documentação desse arquivo.

```
00001 #ifndef JOGADOR_HPP
00002 #define JOGADOR_HPP
00003
00004 #include <map>
00005 #include <string>
00006 #include <unordered_map>
00007
00008 #include "Estatisticas.hpp"
00009
00010 class Jogador
00011 {
00012   private:
00013     std::string apelido;
00014     std::string nome;
00015     std::unordered_map<std::string, Estatisticas> estatisticasPorJogo;
00016
00017   public:
00018     Jogador(const std::string &apelido, const std::string &nome);
00019     Jogador(const std::string &apelido, const std::string &nome, int vitoriasJogoDaVelha, int
      derrotasJogoDaVelha,
00020             int empatesJogoDavelha, int vitoriasLig4, int derrotasLig4, int empatesLig4, int
      vitoriasReversi,
00021             int derrotasReversi, int empatesReversi, int vitoriasBatalhaNaval, int
      derrotasBatalhaNaval,
00022             int empatesBatalhaNaval);
00023
00024     void registrarVitoria(const std::string &nomeJogo);
00025     void registrarDerrota(const std::string &nomeJogo);
00026     void registrarEmpate(const std::string &nomeJogo);
00027
00028     std::string getApelido() const;
00029     std::string getNome() const;
00030     int getVitorias(std::string jogo);
00031     int getDerrotas(std::string jogo);
00032     int getEmpates(std::string jogo);
00033
00034     void mostrarEstatisticas(const std::string &nomeJogo) const;
00035 };
00036
00037 #endif
```

6.9 Referência do Arquivo include/JogoDaVelha.hpp

```
#include "Jogos.hpp"
```

Componentes

- class JogoDaVelha

6.10 JogoDaVelha.hpp

Ir para a documentação desse arquivo.

```
00001 #ifndef JOGODAVELHA_HPP
00002 #define JOGODAVELHA_HPP
00003
00004 #include "Jogos.hpp"
00005
00006 class JogoDaVelha : public Jogos
00007 {
00008   public:
00009     JogoDaVelha();
00010     JogoDaVelha(int tamanhoTabuleiro);
00011
00012   protected:
00013     void anunciarInicioPartida(Jogador &Jogador1, Jogador &Jogador2, bool &turno) override;
00014
00015     std::pair<int, int> lerJogada() override;
00016
00017     bool checarDiagonal(std::vector<std::pair<int, int» &jogadas);
00018     bool checarColunas(std::vector<std::pair<int, int» &jogadas);
00019     bool checarLinhas(std::vector<std::pair<int, int» &jogadas);
00020     bool checarVencedor(std::vector<std::pair<int, int» &jogadas, Jogador &vencedor, Jogador
      &perdedor) override;
00021     bool checarEmpate(int numeroJogadas, Jogador &jogador_01, Jogador &jogador_02) override;
00022
00023     friend class JogoDaVelhaAi;
00024 };
00025
00026 #endif
```

6.11 Referência do Arquivo include/JogoDaVelhaAi.hpp

```
#include "JogoDaVelha.hpp"
#include "Jogos.hpp"
#include <iostream>
#include <vector>
```

Componentes

- class JogoDaVelhaAi

Variáveis

- const int TABULEIRO_SIZE = 9
- const char VAZIO = ' '
- const char JOGADOR_X = 'X'
- const char JOGADOR_O = 'O'

6.11.1 Variáveis

6.11.1.1 JOGADOR_O

```
const char JOGADOR_O = 'O'
```

6.11.1.2 JOGADOR_X

```
const char JOGADOR_X = 'X'
```

6.11.1.3 TABULEIRO_SIZE

```
const int TABULEIRO_SIZE = 9
```

6.11.1.4 VAZIO

```
const char VAZIO = ' '
```

6.12 JogoDaVelhaAi.hpp

Ir para a documentação desse arquivo.

```
00001 #ifndef JOGODAVELHAAI_HPP
00002 #define JOGODAVELHAAI_HPP
00003
00004 #include "JogoDaVelha.hpp"
00005 #include "Jogos.hpp"
00006 #include <iostream>
00007 #include <vector>
00008
00009 const int TABULEIRO_SIZE = 9;
00010 const char VAZIO = ' ';
00011 const char JOGADOR_X = 'X';
00012 const char JOGADOR_O = 'O';
00013 class JogoDaVelhaAi
00014 {
00015   public:
00016     JogoDaVelhaAi();
00017     void Jogar(Jogador &Jogador1, Jogador &Jogador2);
00018
00019   private:
00020     int MAX_PROFUNDIDADE;
00021
00022     JogoDaVelha jogo;
00023     std::vector<char> tabuleiro;
00024
00025     bool checarVitoria(char jogador) const;
00026     bool isTabuleiroCheio() const;
00027     int minimax(bool isMaximizador, int profundidade);
00028     int getMelhorMovimento();
00029
00030     std::pair<int, int> jogarHumano(bool turno);
00031     std::pair<int, int> jogarAI(bool turno);
00032 };
00033
00034 #endif
```

6.13 Referência do Arquivo include/Jogos.hpp

```
#include <algorithm>
#include <ctime>
#include <fstream>
#include <iostream>
#include <limits>
#include <random>
#include <string>
#include <utility>
#include <vector>
#include "Jogador.hpp"
```

Componentes

- class Jogos

6.14 Jogos.hpp

Ir para a documentação desse arquivo.

```
00001 #ifndef JOGOS_HPP
00002 #define JOGOS_HPP
00003
00004 #include <algorithm>
00005 #include <ctime>
00006 #include <fstream>
00007 #include <iostream>
00008 #include <limits>
00009 #include <random>
00010 #include <string>
00011 #include <utility>
00012 #include <vector>
00013
00014 #include "Jogador.hpp"
00015
00016 class Jogos
00017 {
00018   protected:
00019     std::vector<std::vector<char>> tabuleiro;
00020
00021     virtual bool checarVencedor(std::vector<std::pair<int, int>> &jogadas, Jogador &vencedor, Jogador
      &perdedor) = 0;
00022     virtual bool checarEmpate(int numeroJogadas, Jogador &jogador_01, Jogador &jogador_02) = 0;
00023     virtual void anunciarInicioPartida(Jogador &Jogador1, Jogador &Jogador2, bool &turno) = 0;
00024
00025     virtual void marcarTabuleiro(std::pair<int, int> &jogada, bool &turno);
00026
00027     virtual void limparTabuleiro();
00028
00029     void anunciarTurnoJogador(Jogador &Jogador);
00030
00031     virtual bool sorteioTurno();
00032     virtual bool checarJogadaExistente(std::vector<std::pair<int, int>> &jogadas, int linha, int
      coluna);
00033     virtual bool checarPosicaoValida(int linha, int coluna);
00034
00035     std::string gerarDivisoriaTabuleiro();
00036
00037     virtual std::pair<int, int> lerJogada() { return {0, 0}; };
00038
00039   public:
00040     virtual void mostrarTabuleiro();
00041     virtual void Jogar(Jogador &Jogador1, Jogador &Jogador2);
00042 };
00043
00044 #endif
```

6.15 Referência do Arquivo include/Lig4.hpp

```
#include "Jogos.hpp"
```

Componentes

- class Lig4

6.16 Lig4.hpp

Ir para a documentação desse arquivo.

```
00001 #ifndef LIG4_HPP
00002 #define LIG4_HPP
00003
00004 #include "Jogos.hpp"
00005
00006 class Lig4 : public Jogos
00007 {
00008   public:
00009     Lig4(int tamanhoTabuleiro);
00010     Lig4();
00011
00012   protected:
00013     void anunciarInicioPartida(Jogador &Jogador1, Jogador &Jogador2, bool &turno) override;
00014
00015     bool checarDiagonal(std::vector<std::pair<int, int» &jogadas);
00016     bool checarColunas(std::vector<std::pair<int, int» &jogadas);
00017     bool checarLinhas(std::vector<std::pair<int, int» &jogadas);
00018     bool checarVencedor(std::vector<std::pair<int, int» &jogadas, Jogador &vencedor, Jogador
      &perdedor) override;
00019     bool checarEmpate(int numeroJogadas, Jogador &jogador_01, Jogador &jogador_02) override;
00020
00021     std::pair<int, int> lerJogada() override;
00022 };
00023
00024 #endif
```

6.17 Referência do Arquivo include/Reversi.hpp

```
#include "Jogos.hpp"
```

Componentes

- class Reversi

6.18 Reversi.hpp

Ir para a documentação desse arquivo.

```
00001 #ifndef REVERSI_HPP
00002 #define REVERSI_HPP
00003
00004 #include "Jogos.hpp"
00005 class Reversi : public Jogos
00006 {
00007   public:
00008     Reversi();
00009     Reversi(int tamanhoTabuleiro);
00010     void Jogar(Jogador &Jogador1, Jogador &Jogador2) override;
00011
00012   protected:
00013     void anunciarInicioPartida(Jogador &Jogador1, Jogador &Jogador2, bool &turno) override;
00014     bool jogadorInicial(bool &turno);
00015
00016     std::pair<int, int> lerJogada(bool turno);
00017
00018     void marcarTabuleiro(std::pair<int, int> &jogada, bool &turno) override;
00019     bool movimentoValido(std::pair<int, int> &jogada, char jogador, std::vector<std::pair<int, int»
      &flips);
00020     bool haMovimentosDisponiveis(char Jogador);
00021
00022     void limparTabuleiro() override;
00023     bool checarVencedor(std::vector<std::pair<int, int» &jogadas, Jogador &vencedor, Jogador
      &perdedor)
00024     {
00025         return false;
00026     };
```

```
00027     bool checarVencedor(std::vector<std::pair<int, int» &jogadas, Jogador &jogador1, Jogador
      &Jogador2,
00028                         bool &PrimeiroJogador);
00029     bool checarEmpate(int numeroJogadas, Jogador &jogador_01, Jogador &jogador_02) override { return
      false; };
00030 };
00031
00032 #endif
```

6.19 Referência do Arquivo src/BatalhaNaval.cpp

```
#include "BatalhaNaval.hpp"
```

6.20 Referência do Arquivo src/CentralDeJogos.cpp

```
#include <CentralDeJogos.hpp>
```

6.21 Referência do Arquivo src/Estatisticas.cpp

```
#include "Estatisticas.hpp"
#include <iostream>
```

6.22 Referência do Arquivo src/Jogador.cpp

```
#include "Jogador.hpp"
#include <iostream>
```

6.23 Referência do Arquivo src/Jogos.cpp

```
#include "Jogos.hpp"
#include "Jogador.hpp"
```

6.24 Referência do Arquivo src/Lig4.cpp

```
#include "Lig4.hpp"
```

6.25 Referência do Arquivo src/main.cpp

```
#include <CentralDeJogos.hpp>
#include <iostream>
#include <stdexcept>
#include <vector>
```

Funções

- std::string validarEntrada ()
- void exibirMenu ()
- int main ()

6.25.1 Funções

6.25.1.1 exibirMenu()

```
void exibirMenu ()
```

6.25.1.2 main()

```
int main ()
```

6.25.1.3 validarEntrada()

```
std::string validarEntrada ()
```

6.26 Referência do Arquivo src/JogoDaVelha.cpp

```
#include "JogoDaVelha.hpp"
```

6.27 Referência do Arquivo UI/cpp/JogoDaVelha.cpp

```
#include <emscripten.h>
```

Funções

- EMSCRIPTEN_KEEPALIVE bool checarDiagonal (int turno)
- bool checarColunas (int turno)
- bool checarLinhas (int turno)
- bool checarVencedor (int turno)
- EMSCRIPTEN_KEEPALIVE bool jogar (int linha, int coluna, int turno)

Variáveis

- const int linhas = 3
- const int colunas = 3
- int tabuleiro [linhas][colunas] = {0}

6.27.1 Funções

6.27.1.1 checarColunas()

```
bool checarColunas (
            int turno)
```

6.27.1.2 checarDiagonal()

```
EMSCRIPTEN_KEEPALIVE bool checarDiagonal (
            int turno)
```

6.27.1.3 checarLinhas()

```
bool checarLinhas (
            int turno)
```

6.27.1.4 checarVencedor()

```
bool checarVencedor (
            int turno)
```

6.27.1.5 jogar()

```
EMSCRIPTEN_KEEPALIVE bool jogar (
            int linha,
            int coluna,
            int turno)
```

6.27.2 Variáveis

6.27.2.1 colunas

```
const int colunas = 3
```

6.27.2.2 linhas

```
const int linhas = 3
```

6.27.2.3 tabuleiro

```
int tabuleiro[linhas][colunas] = {0}
```

6.28 Referência do Arquivo src/JogoDaVelhaAi.cpp

Implementação da lógica do jogo da velha com IA usando o algoritmo Minimax.

```
#include "JogoDaVelhaAi.hpp"
```

6.28.1 Descrição detalhada

Implementação da lógica do jogo da velha com IA usando o algoritmo Minimax.

Este arquivo contém a implementação das funções da classe JogoDaVelhaAi, incluindo a lógica de verificação da vitória, do empate, e do algoritmo Minimax para a IA.

6.29 Referência do Arquivo UI/cpp/JogoDaVelhaAi.cpp

```
#include <emscripten.h>
```

Funções

- EMSCRIPTEN_KEEPALIVE void colocarPeca (int position, int turno)
- bool checkWin (int player)
- bool isBoardFull ()
- int minimax (bool isMaximizing, int depth)
- EMSCRIPTEN_KEEPALIVE int getBestMove (int difficulty)

Variáveis

- const int lines = 3
- const int columns = 3
- int board [lines ∗columns] ={0}

6.29.1 Funções

6.29.1.1 checkWin()

```
bool checkWin (
            int player)
```

6.29.1.2 colocarPeca()

```
EMSCRIPTEN_KEEPALIVE void colocarPeca (
            int position,
            int turno)
```

6.29.1.3 getBestMove()

```
EMSCRIPTEN_KEEPALIVE int getBestMove (
            int difficulty)
```

6.29.1.4 isBoardFull()

```
bool isBoardFull ()
```

6.29.1.5 minimax()

```
int minimax (
            bool isMaximizing,
            int depth)
```

6.29.2 Variáveis

6.29.2.1 board

```
int board[lines *columns] ={0}
```

6.29.2.2 columns

```
const int columns = 3
```

6.29.2.3 lines

```
const int lines = 3
```

6.30 Referência do Arquivo UI/cpp/Ligue4.cpp

```
#include <emscripten.h>
```

Funções

- EMSCRIPTEN_KEEPALIVE bool checarDiagonal (int turno)
- bool checarLinhas (int turno)
- bool checarColunas (int turno)
- bool checarVencedor (int turno)
- EMSCRIPTEN_KEEPALIVE bool jogar (int linha, int coluna, int turno)

Variáveis

- const int linhas = 6
- const int colunas = 7
- int tabuleiro [linhas][colunas] ={0}

6.30.1 Funções

6.30.1.1 checarColunas()

```
bool checarColunas (
            int turno)
```

6.30.1.2 checarDiagonal()

```
EMSCRIPTEN_KEEPALIVE bool checarDiagonal (
            int turno)
```

6.30.1.3 checarLinhas()

```
bool checarLinhas (
            int turno)
```

6.30.1.4 checarVencedor()

```
bool checarVencedor (
            int turno)
```

6.30.1.5 jogar()

```
EMSCRIPTEN_KEEPALIVE bool jogar (
            int linha,
            int coluna,
            int turno)
```

6.30.2 Variáveis

6.30.2.1 colunas

```
const int colunas = 7
```

6.30.2.2 linhas

```
const int linhas = 6
```

6.30.2.3 tabuleiro

```
int tabuleiro[linhas][colunas] ={0}
```

6.31 Referência do Arquivo src/Reversi.cpp

```
#include "Reversi.hpp"
```

6.32 Referência do Arquivo UI/cpp/Reversi.cpp

```
#include <emscripten.h>
```

Funções

- EMSCRIPTEN_KEEPALIVE void colocarNoTabuleiro (int linha, int coluna, int turno)
- bool checarPosicaoValida (int linha, int coluna)
- EMSCRIPTEN_KEEPALIVE int movimentoValidoY (int turno, int linha, int coluna)
- EMSCRIPTEN_KEEPALIVE int movimentoValidoX (int turno, int linha, int coluna)
- EMSCRIPTEN_KEEPALIVE int checarVencedor (int turno)

Variáveis

- const int linhas = 8
- const int colunas = 8
- int tabuleiro [linhas][colunas] ={0}

6.32.1 Funções

6.32.1.1 checarPosicaoValida()

```
bool checarPosicaoValida (
            int linha,
            int coluna)
```

6.32.1.2 checarVencedor()

```
EMSCRIPTEN_KEEPALIVE int checarVencedor (
            int turno)
```

6.32.1.3 colocarNoTabuleiro()

```
EMSCRIPTEN_KEEPALIVE void colocarNoTabuleiro (
            int linha,
            int coluna,
            int turno)
```

6.32.1.4 movimentoValidoX()

```
EMSCRIPTEN_KEEPALIVE int movimentoValidoX (
            int turno,
            int linha,
            int coluna)
```

6.32.1.5 movimentoValidoY()

```
EMSCRIPTEN_KEEPALIVE int movimentoValidoY (
            int turno,
            int linha,
            int coluna)
```

6.32.2 Variáveis

6.32.2.1 colunas

```
const int colunas = 8
```

6.32.2.2 linhas

```
const int linhas = 8
```

6.32.2.3 tabuleiro

```
int tabuleiro[linhas][colunas] ={0}
```

6.33 Referência do Arquivo UI/README.md

Índice Remissivo