Projeto Final

Gerado por Doxygen 1.9.8

Capítulo 1

Índice Hierárquico

1.1 Hierarquia de Classes

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

Capítulo 2

Índice dos Componentes

2.1 Lista de Classes

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

Capítulo 3

Índice dos Arquivos

3.1 Lista de Arquivos

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

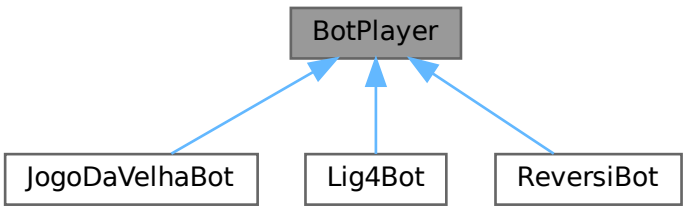
Capítulo 4

Classes

4.1 Referência da Classe BotPlayer

`#include <botplayer.hpp>`

Diagrama de hierarquia da classe BotPlayer:



Membros Públicos

- virtual ~BotPlayer ()=default
- virtual std::pair< int, int > calcularProximaJogada (const JogosDeTabuleiro &jogo, int jogadorAtual)=0

Membros protegidos

- BotPlayer ()=default
- BotPlayer (const BotPlayer &)=delete
- BotPlayer & operator= (const BotPlayer &)=delete

4.1.1 Construtores e Destrutores

4.1.1.1 ~BotPlayer()

`virtual BotPlayer::~BotPlayer ( )  [virtual], [default]`

4.1.1.2 BotPlayer() [1/2]

```
BotPlayer::BotPlayer ( )  [protected], [default]
```

4.1.1.3 BotPlayer() [2/2]

```
BotPlayer::BotPlayer (
            const BotPlayer &  )  [protected], [delete]
```

4.1.2 Documentação das funções

4.1.2.1 calcularProximaJogada()

```
virtual std::pair< int, int > BotPlayer::calcularProximaJogada (
            const JogosDeTabuleiro & jogo,
            int jogadorAtual )  [pure virtual]
```

Implementado por ReversiBot, Lig4Bot e JogoDaVelhaBot.

4.1.2.2 operator=()

```
BotPlayer & BotPlayer::operator= (
            const BotPlayer &  )  [protected], [delete]
```

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

- include/botplayer.hpp

4.2 Referência da Classe Cadastro

Gerencia o cadastro de jogadores, incluindo adição, remoção e persistência de dados.

`#include <cadastro.hpp>`

Membros Públicos

- const std::vector< std::unique_ptr< Jogador > > & get_jogadores () const
- void adicionarJogador (const Jogador &alvo)

  Adiciona um novo jogador ao cadastro.
- void mostrarJogadores () const

  Exibe todos os jogadores cadastrados.
- void import (const std::string &caminho)

  Importa dados de jogadores de um arquivo.
- void save (const std::string &caminho)

  Salva todos os jogadores em um arquivo.
- void removeJogador (const Jogador &alvo)

  Remove um jogador do cadastro.
- bool check (const Jogador &alvo) const

  Verifica se um jogador está cadastrado.

4.2.1 Descrição detalhada

Gerencia o cadastro de jogadores, incluindo adição, remoção e persistência de dados.

4.2.2 Documentação das funções

4.2.2.1 adicionarJogador()

```
void Cadastro::adicionarJogador (
            const Jogador & alvo )
```

Adiciona um novo jogador ao cadastro.

Parâmetros

alvo	Jogador a ser adicionado

Exceções

std::invalid_argument	Se o jogador já estiver cadastrado
std::runtime_error	Se houver falha na alocação de memória

4.2.2.2 check()

```
bool Cadastro::check (
            const Jogador & alvo ) const
```

Verifica se um jogador está cadastrado.

Parâmetros

alvo	Jogador a ser verificado

Retorna

true se o jogador estiver cadastrado, false caso contrário

Exceções

std::runtime_error	Se houver erro durante a verificação

4.2.2.3 get_jogadores()

```
const std::vector< std::unique_ptr< Jogador > > & Cadastro::get_jogadores ( ) const  [inline]
```

4.2.2.4 import()

```
void Cadastro::import (
            const std::string & caminho )
```

Importa dados de jogadores de um arquivo.

Parâmetros

caminho	Caminho do arquivo a ser lido

Exceções

std::runtime_error	Se houver erro na leitura do arquivo
std::invalid_argument	Se o arquivo estiver mal formatado

4.2.2.5 mostrarJogadores()

```
void Cadastro::mostrarJogadores ( ) const
```

Exibe todos os jogadores cadastrados.

Exceções

std::runtime_error	Se houver erro ao acessar os dados dos jogadores

4.2.2.6 removeJogador()

```
void Cadastro::removeJogador (
            const Jogador & alvo )
```

Remove um jogador do cadastro.

Parâmetros

alvo	Jogador a ser removido

Exceções

std::invalid_argument	Se o jogador não for encontrado
std::runtime_error	Se houver erro durante a remoção

4.2.2.7 save()

```
void Cadastro::save (
            const std::string & caminho )
```

Salva todos os jogadores em um arquivo.

Parâmetros

caminho	Caminho do arquivo onde os dados serão salvos

Exceções

std::runtime_error	Se houver erro na escrita do arquivo

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

- include/cadastro.hpp
- src/cadastro.cpp

4.3 Referência da Classe Jogador

Representa um jogador com informações como nome, apelido e desempenho em diferentes jogos.

`#include <cadastro.hpp>`

Membros Públicos

- Jogador (const std::string &nome="", const std::string &apelido="", int vitorias1=0, int derrotas1=0, int vitorias2=0, int derrotas2=0, int vitorias3=0, int derrotas3=0)
- ~Jogador ()
- std::string getNome () const
- std::string getApelido () const
- const Winrate & getLig4 () const
- const Winrate & getReversi () const
- const Winrate & getVelha () const
- int getVitorias (const Winrate &jogo) const
- int getDerrotas (const Winrate &jogo) const
- void setNome (const std::string &nome)
- void setApelido (const std::string &apelido)
- void setVitorias (Winrate &jogo, int vitorias)
- void setDerrotas (Winrate &jogo, int derrotas)
- std::string serializar () const

  Serializa os dados do jogador em uma string formatada.

Membros públicos estáticos

- static Jogador deserializar (const std::string &linha)

  Cria um objeto Jogador a partir de uma string serializada.

4.3.1 Descrição detalhada

Representa um jogador com informações como nome, apelido e desempenho em diferentes jogos.

4.3.2 Construtores e Destrutores

4.3.2.1 Jogador()

```
Jogador::Jogador (
            const std::string & nome = "",
            const std::string & apelido = "",
            int vitorias1 = 0,
            int derrotas1 = 0,
            int vitorias2 = 0,
            int derrotas2 = 0,
            int vitorias3 = 0,
            int derrotas3 = 0 )  [inline]
```

4.3.2.2 ~Jogador()

```
Jogador::~Jogador ( )  [inline]
```

4.3.3 Documentação das funções

4.3.3.1 deserializar()

```
Jogador Jogador::deserializar (
            const std::string & linha )  [static]
```

Cria um objeto Jogador a partir de uma string serializada.

Parâmetros

linha	String contendo os dados do jogador

Retorna

Objeto Jogador construído com os dados da string

Exceções

std::invalid_argument	Se a string tiver formato inválido
std::runtime_error	Se houver erro na conversão dos dados

4.3.3.2 getApelido()

```
std::string Jogador::getApelido ( ) const  [inline]
```

4.3.3.3 getDerrotas()

```
int Jogador::getDerrotas (
            const Winrate & jogo ) const  [inline]
```

4.3.3.4 getLig4()

```
const Winrate & Jogador::getLig4 ( ) const  [inline]
```

4.3.3.5 getNome()

```
std::string Jogador::getNome ( ) const  [inline]
```

4.3.3.6 getReversi()

```
const Winrate & Jogador::getReversi ( ) const  [inline]
```

4.3.3.7 getVelha()

```
const Winrate & Jogador::getVelha ( ) const  [inline]
```

4.3.3.8 getVitorias()

```
int Jogador::getVitorias (
            const Winrate & jogo ) const  [inline]
```

4.3.3.9 serializar()

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

Serializa os dados do jogador em uma string formatada.

Retorna

String contendo os dados do jogador separados por vírgula

Exceções

std::runtime_error	Se houver falha na serialização

4.3.3.10 setApelido()

```
void Jogador::setApelido (
```

```
            const std::string & apelido )  [inline]
```

4.3.3.11 setDerrotas()

```
void Jogador::setDerrotas (
            Winrate & jogo,
            int derrotas )  [inline]
```

4.3.3.12 setNome()

```
void Jogador::setNome (
            const std::string & nome )  [inline]
```

4.3.3.13 setVitorias()

```
void Jogador::setVitorias (
            Winrate & jogo,
            int vitorias )  [inline]
```

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

- include/cadastro.hpp
- src/cadastro.cpp

4.4 Referência da Classe JogoDaVelha

```
#include <jogos.hpp>
```

Diagrama de hierarquia da classe JogoDaVelha:

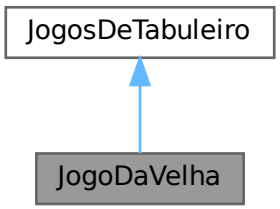
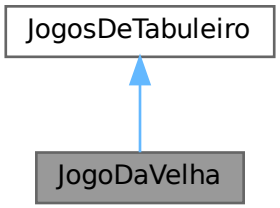


Diagrama de colaboração para JogoDaVelha:



Membros Públicos

- JogoDaVelha (int linhas=3, int colunas=3)
- int ler_jogada (int linha, int coluna, int jogador) override
- bool verificar_jogada (int linha, int coluna, int jogador) const override
- bool testar_condicao_de_vitoria () const override
- int imprimir_vetor (int jogador) const override
- void anunciar_vencedor () const
- int determinar_vencedor () const

Membros Públicos herdados de JogosDeTabuleiro

- JogosDeTabuleiro (int linhas=0, int colunas=0)
- virtual ~JogosDeTabuleiro ()=default
- int getLinhas () const
- int getColunas () const
- std::vector< std::vector< int > > get_tabuleiro () const
- char get_casa (int linha, int coluna) const
- void setLinhasColunas (int linha, int coluna)

Outros membros herdados

Atributos Protegidos herdados de JogosDeTabuleiro

- int linhas_
- int colunas_
- std::vector< std::vector< int > > Tabuleiro_

4.4.1 Construtores e Destrutores

4.4.1.1 JogoDaVelha()

```
JogoDaVelha::JogoDaVelha (
            int linhas = 3,
            int colunas = 3 )
```

4.4.2 Documentação das funções

4.4.2.1 anunciar_vencedor()

```
void JogoDaVelha::anunciar_vencedor ( ) const
```

4.4.2.2 determinar_vencedor()

```
int JogoDaVelha::determinar_vencedor ( ) const
```

4.4.2.3 imprimir_vetor()

```
int JogoDaVelha::imprimir_vetor (
            int jogador ) const  [override], [virtual]
```

Implementa JogosDeTabuleiro.

4.4.2.4 ler_jogada()

```
int JogoDaVelha::ler_jogada (
            int linha,
            int coluna,
            int jogador )  [override], [virtual]
```

Implementa JogosDeTabuleiro.

4.4.2.5 testar_condicao_de_vitoria()

```
bool JogoDaVelha::testar_condicao_de_vitoria ( ) const  [override], [virtual]
```

Implementa JogosDeTabuleiro.

4.4.2.6 verificar_jogada()

```
bool JogoDaVelha::verificar_jogada (
            int linha,
            int coluna,
            int jogador ) const  [override], [virtual]
```

Implementa JogosDeTabuleiro.

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

- include/jogos.hpp
- src/jogos.cpp

4.5 Referência da Classe JogoDaVelhaBot

`#include <botplayer.hpp>`

Diagrama de hierarquia da classe JogoDaVelhaBot:

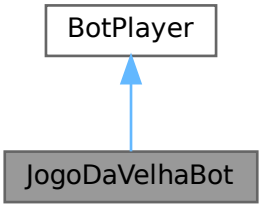
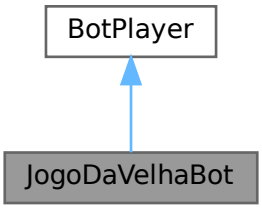


Diagrama de colaboração para JogoDaVelhaBot:



Membros Públicos

- std::pair< int, int > calcularProximaJogada (const JogosDeTabuleiro &jogoBase, int jogadorAtual) override

Membros Públicos herdados de BotPlayer

- virtual ~BotPlayer ()=default

Outros membros herdados

Membros protegidos herdados de BotPlayer

- BotPlayer ()=default
- BotPlayer (const BotPlayer &)=delete
- BotPlayer & operator= (const BotPlayer &)=delete

4.5.1 Documentação das funções

4.5.1.1 calcularProximaJogada()

```
std::pair< int, int > JogoDaVelhaBot::calcularProximaJogada (
            const JogosDeTabuleiro & jogoBase,
            int jogadorAtual )  [override], [virtual]
```

Implementa BotPlayer.

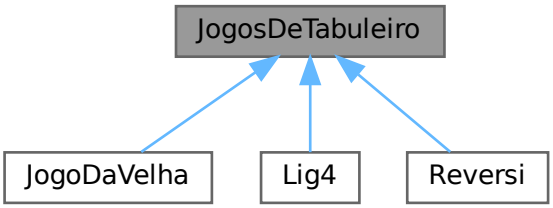
A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- include/botplayer.hpp
- src/botplayer.cpp

4.6 Referência da Classe JogosDeTabuleiro

```
#include <jogos.hpp>
```

Diagrama de hierarquia da classe JogosDeTabuleiro:



Membros Públicos

- JogosDeTabuleiro (int linhas=0, int colunas=0)
- virtual ~JogosDeTabuleiro ()=default
- int getLinhas () const
- int getColunas () const
- std::vector< std::vector< int > > get_tabuleiro () const
- char get_casa (int linha, int coluna) const
- void setLinhasColunas (int linha, int coluna)
- virtual int imprimir_vetor (int jogador) const =0
- virtual int ler_jogada (int linha, int coluna, int jogador)=0
- virtual bool verificar_jogada (int linha, int coluna, int jogador) const =0
- virtual bool testar_condicao_de_vitoria () const =0

Atributos Protegidos

- int linhas_
- int colunas_
- std::vector< std::vector< int > > Tabuleiro_

4.6.1 Construtores e Destrutores

4.6.1.1 JogosDeTabuleiro()

```
JogosDeTabuleiro::JogosDeTabuleiro (
            int linhas = 0,
            int colunas = 0 )
```

4.6.1.2 ~JogosDeTabuleiro()

```
virtual JogosDeTabuleiro::~JogosDeTabuleiro ( )  [virtual], [default]
```

4.6.2 Documentação das funções

4.6.2.1 get_casa()

```
char JogosDeTabuleiro::get_casa (
            int linha,
            int coluna ) const
```

4.6.2.2 get_tabuleiro()

```
std::vector< std::vector< int > > JogosDeTabuleiro::get_tabuleiro ( ) const
```

4.6.2.3 getColunas()

```
int JogosDeTabuleiro::getColunas ( ) const
```

4.6.2.4 getLinhas()

```
int JogosDeTabuleiro::getLinhas ( ) const
```

4.6.2.5 imprimir_vetor()

```
virtual int JogosDeTabuleiro::imprimir_vetor (
            int jogador ) const  [pure virtual]
```

Implementado por Reversi, JogoDaVelha e Lig4.

4.6.2.6 ler_jogada()

```
virtual int JogosDeTabuleiro::ler_jogada (
            int linha,
            int coluna,
            int jogador )  [pure virtual]
```

Implementado por Reversi, JogoDaVelha e Lig4.

4.6.2.7 setLinhasColunas()

```
void JogosDeTabuleiro::setLinhasColunas (
            int linha,
            int coluna )
```

4.6.2.8 testar_condicao_de_vitoria()

```
virtual bool JogosDeTabuleiro::testar_condicao_de_vitoria ( ) const  [pure virtual]
```

Implementado por Reversi, JogoDaVelha e Lig4.

4.6.2.9 verificar_jogada()

```
virtual bool JogosDeTabuleiro::verificar_jogada (
            int linha,
            int coluna,
            int jogador ) const  [pure virtual]
```

Implementado por Reversi, JogoDaVelha e Lig4.

4.6.3 Atributos

4.6.3.1 colunas_

```
int JogosDeTabuleiro::colunas_  [protected]
```

4.6.3.2 linhas_

```
int JogosDeTabuleiro::linhas_  [protected]
```

4.6.3.3 Tabuleiro_

```
std::vector<std::vector<int> > JogosDeTabuleiro::Tabuleiro_  [protected]
```

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

- include/jogos.hpp
- src/jogos.cpp

4.7 Referência da Classe Lig4

`#include <jogos.hpp>`

Diagrama de hierarquia da classe Lig4:

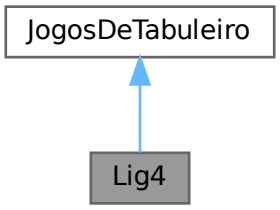
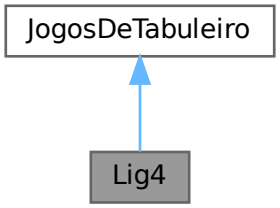


Diagrama de colaboração para Lig4:



Membros Públicos

- Lig4 (int linhas=6, int colunas=7)
- int ler_jogada (int linha, int coluna, int jogador) override
- bool verificar_jogada (int linha, int coluna, int jogador) const override
- bool testar_condicao_de_vitoria () const override
- int imprimir_vetor (int jogador) const override

Membros Públicos herdados de JogosDeTabuleiro

- JogosDeTabuleiro (int linhas=0, int colunas=0)
- virtual ~JogosDeTabuleiro ()=default
- int getLinhas () const
- int getColunas () const
- std::vector< std::vector< int > > get_tabuleiro () const
- char get_casa (int linha, int coluna) const
- void setLinhasColunas (int linha, int coluna)

Outros membros herdados

Atributos Protegidos herdados de JogosDeTabuleiro

- int linhas_
- int colunas_
- std::vector< std::vector< int > > Tabuleiro_

4.7.1 Construtores e Destrutores

4.7.1.1 Lig4()

```
Lig4::Lig4 (
            int linhas = 6,
            int colunas = 7 )
```

4.7.2 Documentação das funções

4.7.2.1 imprimir_vetor()

```
int Lig4::imprimir_vetor (
            int jogador ) const  [override], [virtual]
```

Implementa JogosDeTabuleiro.

4.7.2.2 ler_jogada()

```
int Lig4::ler_jogada (
            int linha,
            int coluna,
            int jogador )  [override], [virtual]
```

Implementa JogosDeTabuleiro.

4.7.2.3 testar_condicao_de_vitoria()

```
bool Lig4::testar_condicao_de_vitoria ( ) const  [override], [virtual]
```

Implementa JogosDeTabuleiro.

4.7.2.4 verificar_jogada()

```
bool Lig4::verificar_jogada (
            int linha,
            int coluna,
            int jogador ) const  [override], [virtual]
```

Implementa JogosDeTabuleiro.

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

- include/jogos.hpp
- src/jogos.cpp

4.8 Referência da Classe Lig4Bot

`#include <botplayer.hpp>`

Diagrama de hierarquia da classe Lig4Bot:

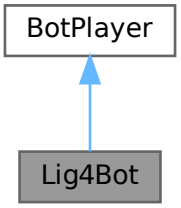
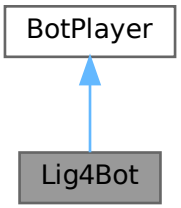


Diagrama de colaboração para Lig4Bot:



Membros Públicos

- std::pair< int, int > calcularProximaJogada (const JogosDeTabuleiro &jogoBase, int jogadorAtual) override

Membros Públicos herdados de BotPlayer

- virtual ~BotPlayer ()=default

Outros membros herdados

Membros protegidos herdados de BotPlayer

- BotPlayer ()=default
- BotPlayer (const BotPlayer &)=delete
- BotPlayer & operator= (const BotPlayer &)=delete

4.8.1 Documentação das funções

4.8.1.1 calcularProximaJogada()

```
std::pair< int, int > Lig4Bot::calcularProximaJogada (
            const JogosDeTabuleiro & jogoBase,
            int jogadorAtual )  [override], [virtual]
```

Implementa BotPlayer.

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

- include/botplayer.hpp
- src/botplayer.cpp

4.9 Referência da Classe Partida

```
#include <partida.hpp>
```

Membros Públicos

- Partida (int tipoJogo, Jogador *jogador1)
  Construtor para criar uma partida com um jogador humano e um bot.
- Partida (int tipoJogo, Jogador *jogador1, Jogador *jogador2)
  Construtor para criar uma partida entre dois jogadores humanos.
- ~Partida ()
- bool iniciarPartida ()
  Inicia uma partida com as configurações atuais.
- bool iniciarPartida (int dificuldade)
  Sobrecarga para iniciar uma partida com dificuldade específica.
- void imprimirTabuleiro (int jogadorAtual) const
  Imprime o tabuleiro atual no console.
- bool realizarJogada (int jogadorAtual, int linha=-1, int coluna=-1)
  Realiza a jogada de um jogador ou bot.
- bool verificarFimDeJogo () const
  Verifica se as condições de fim de jogo foram atingidas.
- bool verificarJogadasDisponiveis (int jogadorAtual) const
  Verifica se o jogador atual tem jogadas disponíveis.
- bool isPvP () const
- Jogador * getJogadorAtual (int jogadorNumero) const

4.9.1 Construtores e Destrutores

4.9.1.1 Partida() [1/2]

```
Partida::Partida (
            int tipoJogo,
            Jogador * jogador1 )
```

Construtor para criar uma partida com um jogador humano e um bot.

Parâmetros

tipoJogo	Tipo do jogo (1 - Jogo da Velha, 2 - Lig4, 3 - Reversi).
jogador1	Ponteiro para o primeiro jogador humano.

Exceções

std::invalid_argument	Se o tipo de jogo for inválido.

4.9.1.2 Partida() [2/2]

```
Partida::Partida (
            int tipoJogo,
            Jogador * jogador1,
            Jogador * jogador2 )
```

Construtor para criar uma partida entre dois jogadores humanos.

Parâmetros

tipoJogo	Tipo do jogo (1 - Jogo da Velha, 2 - Lig4, 3 - Reversi).
jogador1	Ponteiro para o primeiro jogador humano.
jogador2	Ponteiro para o segundo jogador humano.

Exceções

std::invalid_argument	Se o tipo de jogo for inválido.

4.9.1.3 ~Partida()

```
Partida::~Partida ( )  [inline]
```

4.9.2 Documentação das funções

4.9.2.1 getJogadorAtual()

```
Jogador * Partida::getJogadorAtual (
            int jogadorNumero ) const  [inline]
```

4.9.2.2 imprimirTabuleiro()

```
void Partida::imprimirTabuleiro (
            int jogador_atual ) const
```

Imprime o tabuleiro atual no console.

Parâmetros

jogador_atual	Indica qual jogador está jogando.

4.9.2.3 iniciarPartida() [1/2]

```
bool Partida::iniciarPartida ( )
```

Inicia uma partida com as configurações atuais.

Retorna

true se a partida foi concluída com sucesso, false em caso de erro.

4.9.2.4 iniciarPartida() [2/2]

```
bool Partida::iniciarPartida (
            int dificuldade )
```

Sobrecarga para iniciar uma partida com dificuldade específica.

Parâmetros

dificuldade	Nível de dificuldade para bots (não implementado).

Retorna

true se a partida foi concluída com sucesso, false em caso de erro.

4.9.2.5 isPvP()

```
bool Partida::isPvP ( ) const  [inline]
```

4.9.2.6 realizarJogada()

```
bool Partida::realizarJogada (
            int jogadorAtual,
            int linha = -1,
            int coluna = -1 )
```

Realiza a jogada de um jogador ou bot.

Parâmetros

jogadorAtual	Indica o jogador que fará a jogada.
linha	Linha escolhida para a jogada (opcional para bots).
coluna	Coluna escolhida para a jogada (opcional para bots).

Retorna

true se a jogada foi válida, false caso contrário.

4.9.2.7 verificarFimDeJogo()

```
bool Partida::verificarFimDeJogo ( ) const
```

Verifica se as condições de fim de jogo foram atingidas.

Retorna

true se o jogo terminou, false caso contrário.

4.9.2.8 verificarJogadasDisponiveis()

```
bool Partida::verificarJogadasDisponiveis (
            int jogadorAtual ) const
```

Verifica se o jogador atual tem jogadas disponíveis.

Parâmetros

jogadorAtual	Indica o jogador que está verificando as jogadas.

Retorna

true se há jogadas disponíveis, false caso contrário.

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

- include/partida.hpp
- src/partida.cpp

4.10 Referência da Classe Reversi

```
#include <jogos.hpp>
```

Diagrama de hierarquia da classe Reversi:

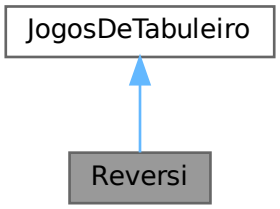
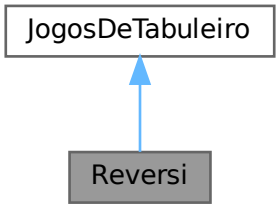


Diagrama de colaboração para Reversi:



Membros Públicos

- Reversi (int linhas=8, int colunas=8)
- int ler_jogada (int linha, int coluna, int jogador) override
- bool verificar_jogada (int linha, int coluna, int jogador) const override
- bool testar_condicao_de_vitoria () const override
- std::vector< std::vector< bool > > atualizar_jogadas_validas (int jogador) const
- int imprimir_vetor (int jogador) const override
- std::pair< int, int > calcular_pontuacao () const
- void mostrar_pontuacao () const
- void anunciar_vencedor () const

Membros Públicos herdados de JogosDeTabuleiro

- JogosDeTabuleiro (int linhas=0, int colunas=0)
- virtual ~JogosDeTabuleiro ()=default
- int getLinhas () const
- int getColunas () const
- std::vector< std::vector< int > > get_tabuleiro () const
- char get_casa (int linha, int coluna) const
- void setLinhasColunas (int linha, int coluna)

Outros membros herdados

Atributos Protegidos herdados de JogosDeTabuleiro

- int linhas_
- int colunas_
- std::vector< std::vector< int > > Tabuleiro_

4.10.1 Construtores e Destrutores

4.10.1.1 Reversi()

```
Reversi::Reversi (
            int linhas = 8,
            int colunas = 8 )
```

4.10.2 Documentação das funções

4.10.2.1 anunciar_vencedor()

```
void Reversi::anunciar_vencedor ( ) const
```

4.10.2.2 atualizar_jogadas_validas()

```
std::vector< std::vector< bool > > Reversi::atualizar_jogadas_validas (
            int jogador ) const
```

4.10.2.3 calcular_pontuacao()

```
std::pair< int, int > Reversi::calcular_pontuacao ( ) const
```

4.10.2.4 imprimir_vetor()

```
int Reversi::imprimir_vetor (
            int jogador ) const  [override], [virtual]
```

Implementa JogosDeTabuleiro.

4.10.2.5 ler_jogada()

```
int Reversi::ler_jogada (
            int linha,
            int coluna,
            int jogador )  [override], [virtual]
```

Implementa JogosDeTabuleiro.

4.10.2.6 mostrar_pontuacao()

```
void Reversi::mostrar_pontuacao ( ) const
```

4.10.2.7 testar_condicao_de_vitoria()

```
bool Reversi::testar_condicao_de_vitoria ( ) const  [override], [virtual]
```

Implementa JogosDeTabuleiro.

4.10.2.8 verificar_jogada()

```
bool Reversi::verificar_jogada (
            int linha,
            int coluna,
            int jogador ) const  [override], [virtual]
```

Implementa JogosDeTabuleiro.

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

- include/jogos.hpp
- src/jogos.cpp

4.11 Referência da Classe ReversiBot

```
#include <botplayer.hpp>
```

Diagrama de hierarquia da classe ReversiBot:

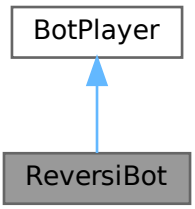
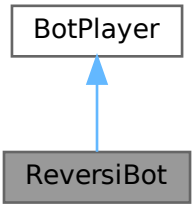


Diagrama de colaboração para ReversiBot:



Membros Públicos

- std::pair< int, int > calcularProximaJogada (const JogosDeTabuleiro &jogo, int jogadorAtual) override

Membros Públicos herdados de BotPlayer

- virtual ~BotPlayer ()=default

Outros membros herdados

Membros protegidos herdados de BotPlayer

- BotPlayer ()=default
- BotPlayer (const BotPlayer &)=delete
- BotPlayer & operator= (const BotPlayer &)=delete

4.11.1 Documentação das funções

4.11.1.1 calcularProximaJogada()

```
std::pair< int, int > ReversiBot::calcularProximaJogada (
            const JogosDeTabuleiro & jogo,
            int jogadorAtual )  [override], [virtual]
```

Implementa BotPlayer.

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

- include/botplayer.hpp
- src/botplayer.cpp

4.12 Referência da Estrutura Winrate

Estrutura para armazenar vitórias e derrotas de um jogador em um jogo específico.

`#include <cadastro.hpp>`

Atributos Públicos

- int _vitorias
- int _derrotas

4.12.1 Descrição detalhada

Estrutura para armazenar vitórias e derrotas de um jogador em um jogo específico.

Parâmetros

_vitorias	Número de vitórias.
_derrotas	Número de derrotas.

4.12.2 Atributos

4.12.2.1 _derrotas

```
int Winrate::_derrotas
```

4.12.2.2 _vitorias

```
int Winrate::_vitorias
```

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

- include/cadastro.hpp

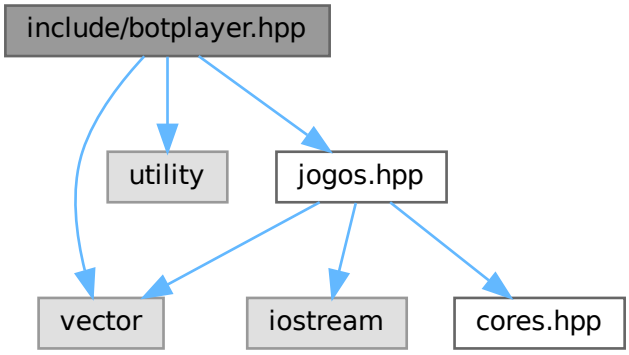
Capítulo 5

Arquivos

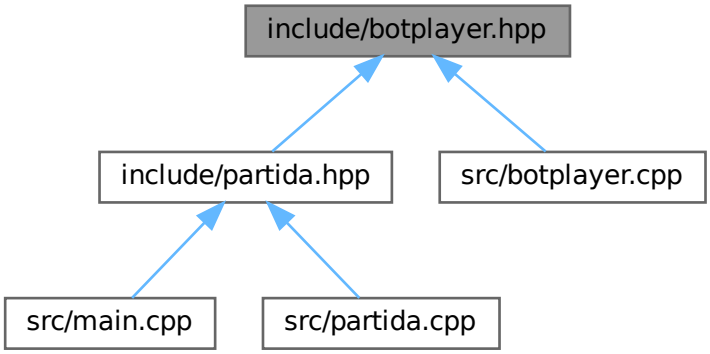
5.1 Referência do Arquivo include/botplayer.hpp

```
#include <vector>
#include <utility>
#include "jogos.hpp"
```

Gráfico de dependência de inclusões para botplayer.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class BotPlayer
- class ReversiBot
- class Lig4Bot
- class JogoDaVelhaBot

5.2 botplayer.hpp

Ir para a documentação desse arquivo.

```
00001 #ifndef BOTPLAYER_HPP
00002 #define BOTPLAYER_HPP
00003 #include <vector>
00004 #include <utility>
00005 #include "jogos.hpp"
00006
00007 // Classe base abstrata para representar bots em jogos de tabuleiro
00008 class BotPlayer {
00009 public:
00010     virtual ~BotPlayer() = default;
00011
00012     // Método abstrato para calcular a próxima jogada do bot
00013     virtual std::pair<int, int> calcularProximaJogada(
00014         const JogosDeTabuleiro& jogo,
00015         int jogadorAtual
00016     ) = 0;
00017
00018 protected:
00019     BotPlayer() = default;
00020     BotPlayer(const BotPlayer&) = delete;
00021     BotPlayer& operator=(const BotPlayer&) = delete;
00022 };
00023
00024 // Classe para implementar o bot do jogo Reversi
00025 class ReversiBot : public BotPlayer {
00026 public:
00027     std::pair<int, int> calcularProximaJogada(
00028         const JogosDeTabuleiro& jogo,
00029         int jogadorAtual
00030     ) override;
00031
00032 private:
00033     int _minimax(Reversi& jogo, int profundidade, bool maximizando, int jogadorAtual);
00034     int _avaliarTabuleiro(const Reversi& jogo, int jogadorAtual);
```

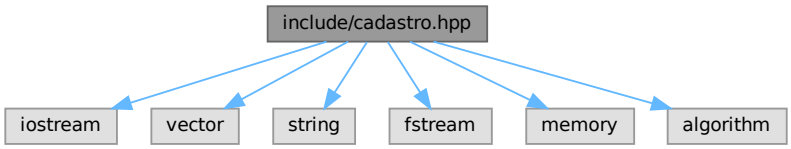
```
00035 };
00036
00037 // Classe para implementar o bot do jogo Lig 4
00038 class Lig4Bot : public BotPlayer {
00039 public:
00040     std::pair<int, int> calcularProximaJogada(
00041         const JogosDeTabuleiro& jogoBase,
00042         int jogadorAtual
00043     ) override;
00044
00045 private:
00046     int _minimax(Lig4& jogo, bool maximizando, int jogadorAtual);
00047 };
00048
00049 // Classe para implementar o bot do jogo da velha
00050 class JogoDaVelhaBot : public BotPlayer {
00051 public:
00052     std::pair<int, int> calcularProximaJogada(
00053         const JogosDeTabuleiro& jogoBase,
00054         int jogadorAtual
00055     ) override;
00056
00057 private:
00058     void minimax(const JogoDaVelha& jogo, int& melhorLinha, int& melhorColuna, int jogador);
00059     int _minimax(JogoDaVelha& jogo, int profundidade, bool maximizando, int jogador);
00060 };
00061
00062 #endif
```

5.3 Referência do Arquivo include/cadastro.hpp

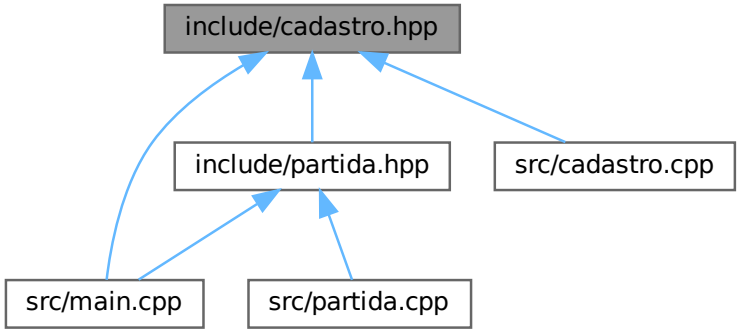
Declaração das classes e métodos relacionados ao gerenciamento de jogadores e seus cadastros.

```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <memory>
#include <algorithm>
```

Gráfico de dependência de inclusões para cadastro.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- struct Winrate

  Estrutura para armazenar vitórias e derrotas de um jogador em um jogo específico.
- class Jogador

  Representa um jogador com informações como nome, apelido e desempenho em diferentes jogos.
- class Cadastro

  Gerencia o cadastro de jogadores, incluindo adição, remoção e persistência de dados.

5.3.1 Descrição detalhada

Declaração das classes e métodos relacionados ao gerenciamento de jogadores e seus cadastros.

5.4 cadastro.hpp

Ir para a documentação desse arquivo.

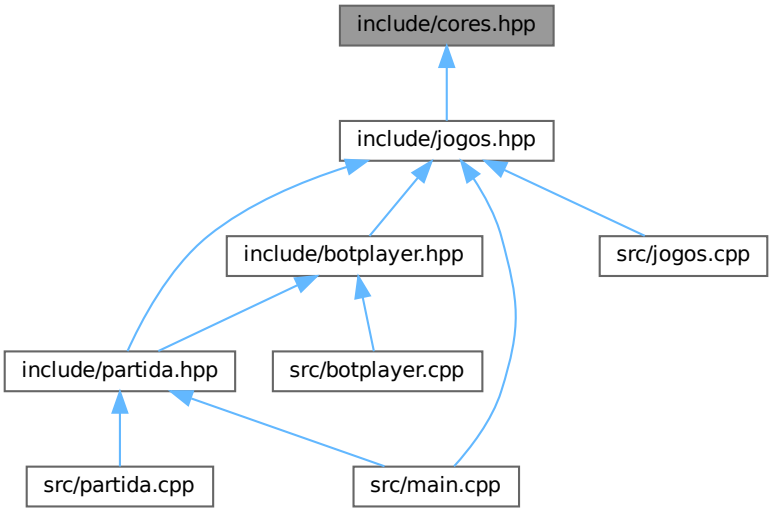
```
00001 #ifndef CADASTRO_HPP
00002 #define CADASTRO_HPP
00003
00009 #include <iostream>
00010 #include <vector>
00011 #include <string>
00012 #include <fstream>
00013 #include <memory>
00014 #include <algorithm>
00015
00022 struct Winrate {
00023     int _vitorias;
00024     int _derrotas;
00025 };
00026
00031 class Jogador {
00032 private:
00033     Winrate Velha;
00034     Winrate Lig4;
00035     Winrate Reversi;
00036     //adicionar outros jogos aqui.
00037     std::string _nome;
00038     std::string _apelido;
```

```
00039
00040 public:
00041     // construtor
00042     Jogador(const std::string& nome = "", const std::string& apelido = "", int vitorias1 = 0, int
      derrotas1 = 0, int vitorias2 = 0, int derrotas2 = 0, int vitorias3 = 0, int derrotas3 = 0)
00043         : _nome(nome), _apelido(apelido),
00044         Velha{vitorias1, derrotas1},
00045         Lig4{vitorias2, derrotas2},
00046         Reversi{vitorias3, derrotas3}
00047         //adicionar outros jogos aqui.
00048         {}
00049
00050     //destrutor
00051     ~Jogador() {}
00052
00053     // métodos de acesso
00054     std::string getNome() const { return _nome; }
00055     std::string getApelido() const { return _apelido; }
00056     const Winrate& getLig4() const { return Lig4; }
00057     const Winrate& getReversi() const { return Reversi; }
00058     const Winrate& getVelha() const { return Velha; }
00059     //possível adicionar outros jogos aqui.
00060     int getVitorias(const Winrate& jogo) const { return jogo._vitorias; }
00061     int getDerrotas(const Winrate& jogo) const { return jogo._derrotas; }
00062
00063     void setNome(const std::string& nome) { _nome = nome; }
00064     void setApelido(const std::string& apelido) { _apelido = apelido; }
00065     void setVitorias(Winrate& jogo, int vitorias) { jogo._vitorias = vitorias; }
00066     void setDerrotas(Winrate& jogo, int derrotas) { jogo._derrotas = derrotas; }
00067
00068     std::string serializar() const;
00069     static Jogador deserializar(const std::string& linha);
00070 };
00071
00076 class Cadastro {
00077 private:
00078     std::vector<std::unique_ptr<Jogador> _jogadores;
00079
00080 public:
00084     const std::vector<std::unique_ptr<Jogador>& get_jogadores() const { return _jogadores; }
00085     void adicionarJogador(const Jogador& alvo);
00086     void mostrarJogadores() const;
00087     void import(const std::string& caminho);
00088     void save(const std::string& caminho);
00089     void removeJogador(const Jogador& alvo);
00090     bool check(const Jogador& alvo) const;
00091 };
00092
00093 #endif
```

5.5 Referência do Arquivo include/cores.hpp

Define macros para cores de texto, fundo e estilos para uso em saídas no terminal.

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Definições e Macros

- #define VERMELHO "\033[31m"
- #define VERDE "\033[32m"
- #define AMARELO "\033[33m"
- #define AZUL "\033[34m"
- #define MAGENTA "\033[35m"
- #define CIANO "\033[36m"
- #define BRANCO "\033[37m"
- #define FUNDO_VERMELHO "\033[41m"
- #define FUNDO_VERDE "\033[42m"
- #define FUNDO_AMARELO "\033[43m"
- #define FUNDO_AZUL "\033[44m"
- #define FUNDO_BRANCO "\033[47m"
- #define FUNDO_PRETO "\033[40m"
- #define NEGRITO "\033[1m"
- #define SUBLINHADO "\033[4m"
- #define RESETAR "\033[0m"

5.5.1 Descrição detalhada

Define macros para cores de texto, fundo e estilos para uso em saídas no terminal.

5.5.2 Definições e macros

5.5.2.1 AMARELO

```
#define AMARELO "\033[33m"
```

5.5.2.2 AZUL

```
#define AZUL "\033[34m"
```

5.5.2.3 BRANCO

```
#define BRANCO "\033[37m"
```

5.5.2.4 CIANO

```
#define CIANO "\033[36m"
```

5.5.2.5 FUNDO_AMARELO

```
#define FUNDO_AMARELO "\033[43m"
```

5.5.2.6 FUNDO_AZUL

```
#define FUNDO_AZUL "\033[44m"
```

5.5.2.7 FUNDO_BRANCO

```
#define FUNDO_BRANCO "\033[47m"
```

5.5.2.8 FUNDO_PRETO

```
#define FUNDO_PRETO "\033[40m"
```

5.5.2.9 FUNDO_VERDE

```
#define FUNDO_VERDE "\033[42m"
```

5.5.2.10 FUNDO_VERMELHO

```
#define FUNDO_VERMELHO "\033[41m"
```

5.5.2.11 MAGENTA

```
#define MAGENTA "\033[35m"
```

5.5.2.12 NEGRITO

```
#define NEGRITO "\033[1m"
```

5.5.2.13 RESETAR

```
#define RESETAR "\033[0m"
```

5.5.2.14 SUBLINHADO

```
#define SUBLINHADO "\033[4m"
```

5.5.2.15 VERDE

```
#define VERDE "\033[32m"
```

5.5.2.16 VERMELHO

```
#define VERMELHO "\033[31m"
```

5.6 cores.hpp

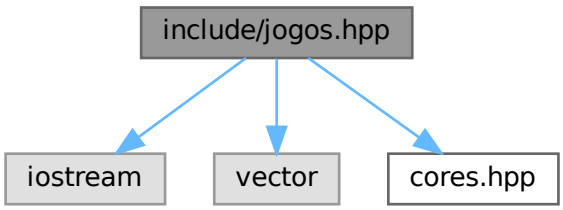
Ir para a documentação desse arquivo.

```
00001 #ifndef cores_hpp
00002 #define cores_hpp
00003
00009 #define VERMELHO    "\033[31m"
00010 #define VERDE       "\033[32m"
00011 #define AMARELO     "\033[33m"
00012 #define AZUL        "\033[34m"
00013 #define MAGENTA     "\033[35m"
00014 #define CIANO       "\033[36m"
00015 #define BRANCO      "\033[37m"
00016 #define FUNDO_VERMELHO    "\033[41m"
00017 #define FUNDO_VERDE       "\033[42m"
00018 #define FUNDO_AMARELO     "\033[43m"
00019 #define FUNDO_AZUL        "\033[44m"
00020 #define FUNDO_BRANCO      "\033[47m"
00021 #define FUNDO_PRETO     "\033[40m"
00022
00023 // Define estilos
00024 #define NEGRITO           "\033[1m"
00025 #define SUBLINHADO        "\033[4m"
00026 #define RESETAR           "\033[0m"
00027
00028 #endif
```

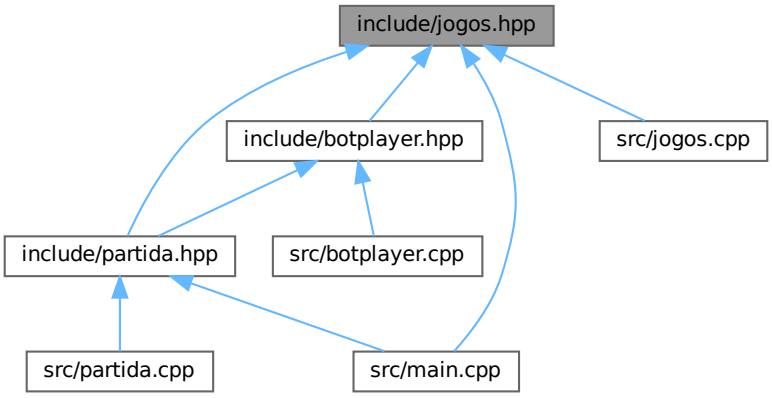
5.7 Referência do Arquivo include/jogos.hpp

```
#include <iostream>
#include <vector>
#include "cores.hpp"
```

Gráfico de dependência de inclusões para jogos.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class JogosDeTabuleiro
- class Reversi
- class JogoDaVelha
- class Lig4

5.8 jogos.hpp

Ir para a documentação desse arquivo.

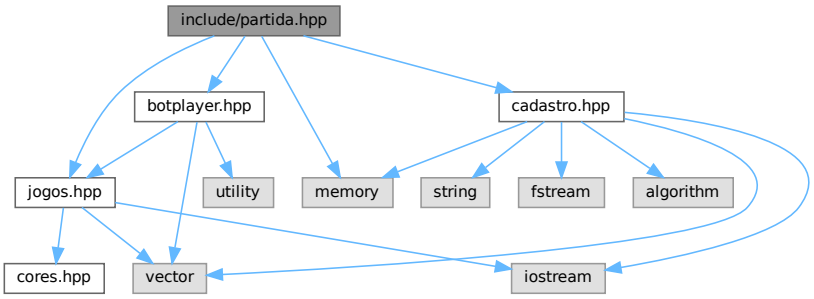
```
00001 #ifndef JOGOS_HPP
00002 #define JOGOS_HPP
00003 #include <iostream>
00004 #include <vector>
00005 #include "cores.hpp"
00006
00007 class JogosDeTabuleiro {
00008 public:
00009     JogosDeTabuleiro(int linhas = 0, int colunas = 0);
00010     virtual ~JogosDeTabuleiro() = default;  // Added virtual destructor
00011
00012     int getLinhas() const;
00013     int getColunas() const;
00014     std::vector<std::vector<int» get_tabuleiro() const;
00015     char get_casa(int linha, int coluna) const;
00016     void setLinhasColunas(int linha, int coluna);
00017
00018     virtual int imprimir_vetor(int jogador) const = 0;
00019     virtual int ler_jogada(int linha, int coluna, int jogador) = 0;
00020     virtual bool verificar_jogada(int linha, int coluna, int jogador) const = 0;
00021     virtual bool testar_condicao_de_vitoria() const = 0;
00022
00023 protected:
00024     int linhas_, colunas_;
00025     std::vector<std::vector<int» Tabuleiro_;
00026 };
00027
00028 class Reversi : public JogosDeTabuleiro {
00029 public:
00030     Reversi(int linhas = 8, int colunas = 8);
00031     int ler_jogada(int linha, int coluna, int jogador) override;
00032     bool verificar_jogada(int linha, int coluna, int jogador) const override;
00033     bool testar_condicao_de_vitoria() const override;
00034     std::vector<std::vector<bool» atualizar_jogadas_validas(int jogador) const;
00035     int imprimir_vetor(int jogador) const override;
00036     std::pair<int, int> calcular_pontuacao() const;
00037     void mostrar_pontuacao() const;
00038     void anunciar_vencedor() const;
00039 private:
00040     static const std::vector<std::pair<int, int» direcoes;
00041     std::vector<std::vector<bool» JogadasValidas_;
00042     bool verificar_direcao(int linha, int coluna, int dLinha, int dColuna, int jogador) const;
00043 };
00044
00045 class JogoDaVelha : public JogosDeTabuleiro {
00046 public:
00047     // Construtor padrão com tamanho 3x3
00048     JogoDaVelha(int linhas = 3, int colunas = 3);
00049
00050     // Métodos herdados e sobrescritos da classe base
00051     int ler_jogada(int linha, int coluna, int jogador) override;
00052     bool verificar_jogada(int linha, int coluna, int jogador) const override;
00053     bool testar_condicao_de_vitoria() const override;
00054     int imprimir_vetor(int jogador) const override;
00055     void anunciar_vencedor() const;
00056     int determinar_vencedor() const; // Retorna 1 para X, 2 para O, 0 para empate
00057 };
00058
00059 class Lig4 : public JogosDeTabuleiro{
00060 public:
00061
00062     Lig4(int linhas = 6, int colunas = 7);
00063
00064     int ler_jogada(int linha, int coluna, int jogador) override;
00065     bool verificar_jogada(int linha, int coluna, int jogador) const override;
00066     bool testar_condicao_de_vitoria() const override;
00067     int imprimir_vetor(int jogador) const override;
00068 };
00069
00070 #endif
```

5.9 Referência do Arquivo include/partida.hpp

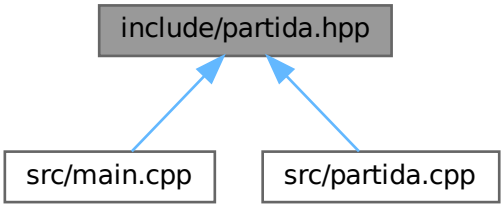
```
#include <memory>
#include "jogos.hpp"
```

```
#include "cadastro.hpp"
#include "botplayer.hpp"
```

Gráfico de dependência de inclusões para partida.hpp:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class Partida

5.10 partida.hpp

Ir para a documentação desse arquivo.

```
00001 #ifndef PARTIDA_HPP
00002 #define PARTIDA_HPP
00003
00004 #include <memory>
00005 #include "jogos.hpp"
00006 #include "cadastro.hpp"
00007 #include "botplayer.hpp"
00008
00009 class Partida {
00010 public:
00011     // Constructors
00012     Partida(int tipoJogo, Jogador* jogador1);  // For PvE
00013     Partida(int tipoJogo, Jogador* jogador1, Jogador* jogador2);  // For PvP
00014
00015     // Destructor to clean up bot pointers
00016     ~Partida() {
```

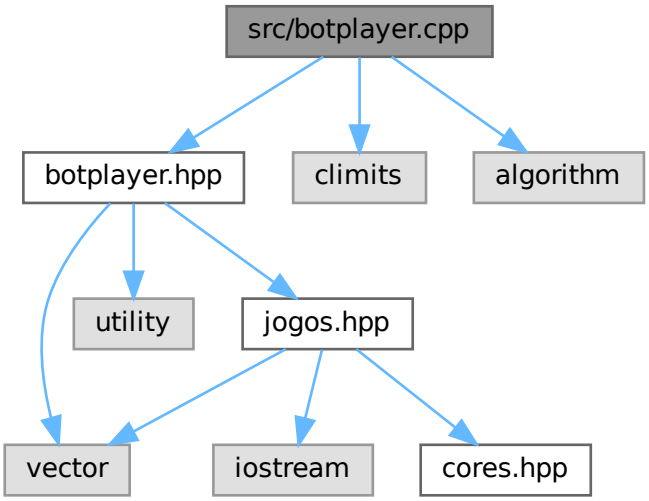
```
00017         delete bot1;
00018         delete bot2;
00019     }
00020
00021     // Game control methods
00022     bool iniciarPartida();
00023     bool iniciarPartida(int dificuldade);  // For future difficulty levels
00024     void imprimirTabuleiro(int jogadorAtual) const;
00025     bool realizarJogada(int jogadorAtual, int linha = -1, int coluna = -1);
00026     bool verificarFimDeJogo() const;
00027     bool verificarJogadasDisponiveis(int jogadorAtual) const;
00028
00029     // Utility methods
00030     bool isPvP() const { return jogador2 != nullptr; }
00031     Jogador* getJogadorAtual(int jogadorNumero) const {
00032         return jogadorNumero == 1 ? jogador1 : jogador2;
00033     }
00034
00035 private:
00036     std::unique_ptr<JogosDeTabuleiro> jogoAtual;
00037     Jogador* jogador1;
00038     Jogador* jogador2;
00039     BotPlayer* bot1;
00040     BotPlayer* bot2;
00041
00042     void finalizarPartida();
00043 };
00044
00045 #endif
```

5.11 Referência do Arquivo src/botplayer.cpp

Declaração dos métodos da classe BotPlayer.

```
#include "botplayer.hpp"
#include <climits>
#include <algorithm>
```

Gráfico de dependência de inclusões para botplayer.cpp:



5.11.1 Descrição detalhada

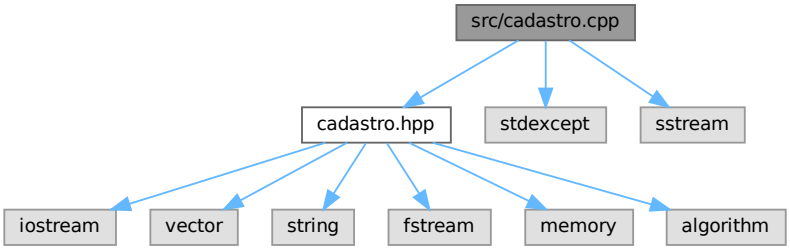
Declaração dos métodos da classe BotPlayer.

5.12 Referência do Arquivo src/cadastro.cpp

Declaração dos métodos das classes Jogador e Cadastro.

```
#include "cadastro.hpp"
#include <stdexcept>
#include <sstream>
```

Gráfico de dependência de inclusões para cadastro.cpp:



5.12.1 Descrição detalhada

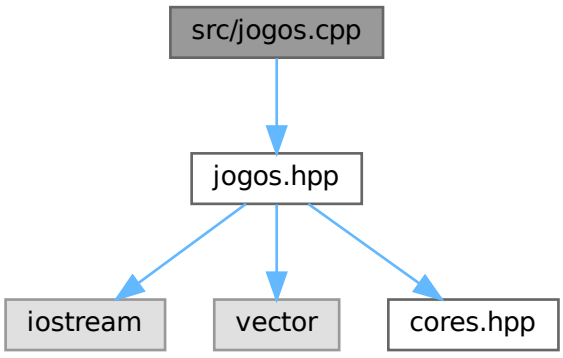
Declaração dos métodos das classes Jogador e Cadastro.

5.13 Referência do Arquivo src/jogos.cpp

Declaração dos métodos da classe JogosDeTabuleiro.

```
#include "jogos.hpp"
```

Gráfico de dependência de inclusões para jogos.cpp:



5.13.1 Descrição detalhada

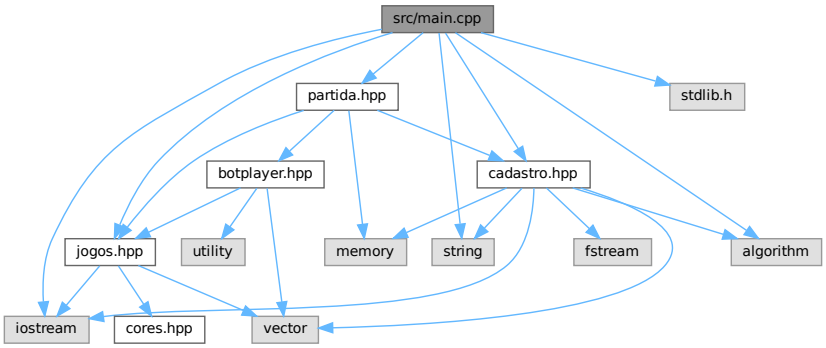
Declaração dos métodos da classe JogosDeTabuleiro.

5.14 Referência do Arquivo src/main.cpp

Sistema de gerenciamento de jogadores e jogos com funcionalidade de cadastro, remoção, busca e partidas.

```
#include <iostream>
#include <stdlib.h>
#include <string>
#include "cadastro.hpp"
#include "jogos.hpp"
#include "partida.hpp"
#include <algorithm>
```

Gráfico de dependência de inclusões para main.cpp:



Funções

- void mostrarMenu ()

  Exibe o menu de comandos do sistema.
- void cadastrarJogador (Cadastro &jogadores)

  Cadastra um novo jogador no sistema.
- void removerJogador (Cadastro &jogadores)

  Remove um jogador do sistema.
- void procurarJogador (const Cadastro &jogadores)

  Procura por um jogador no sistema.
- void iniciarNovaPartida (Cadastro &jogadores)

  Inicia uma nova partida entre dois jogadores ou contra um bot.
- int main ()

  Função principal do programa.

5.14.1 Descrição detalhada

Sistema de gerenciamento de jogadores e jogos com funcionalidade de cadastro, remoção, busca e partidas.

5.14.2 Funções

5.14.2.1 cadastrarJogador()

```
void cadastrarJogador (
            Cadastro & jogadores )
```

Cadastra um novo jogador no sistema.

Parâmetros

jogadores	Referência ao objeto Cadastro para gerenciar jogadores.

5.14.2.2 iniciarNovaPartida()

```
void iniciarNovaPartida (
            Cadastro & jogadores )
```

Inicia uma nova partida entre dois jogadores ou contra um bot.

Parâmetros

jogadores	Referência ao objeto Cadastro para gerenciar jogadores.

5.14.2.3 main()

```
int main ( )
```

Função principal do programa.

Retorna

Retorna 0 em caso de execução bem-sucedida.

5.14.2.4 mostrarMenu()

```
void mostrarMenu ( )
```

Exibe o menu de comandos do sistema.

5.14.2.5 procurarJogador()

```
void procurarJogador (
            const Cadastro & jogadores )
```

Procura por um jogador no sistema.

Parâmetros

jogadores	Referência ao objeto Cadastro para gerenciar jogadores.

5.14.2.6 removerJogador()

```
void removerJogador (
            Cadastro & jogadores )
```

Remove um jogador do sistema.

Parâmetros

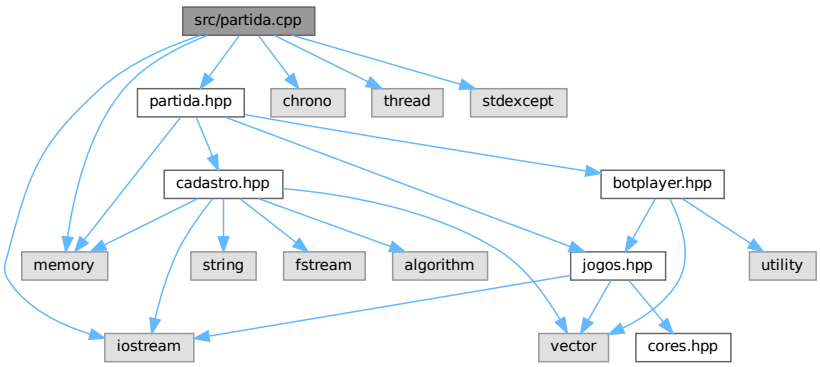
jogadores	Referência ao objeto Cadastro para gerenciar jogadores.

5.15 Referência do Arquivo src/partida.cpp

Declaração dos métodos da classe Partida.

```
#include "partida.hpp"
#include <iostream>
#include <memory>
#include <chrono>
#include <thread>
#include <stdexcept>
```

Gráfico de dependência de inclusões para partida.cpp:



5.15.1 Descrição detalhada

Declaração dos métodos da classe Partida.