

Trabalho de PDS2

Gerado por Doxygen 1.13.2

1 Índice Hierárquico	1
1.1 Hierarquia de Classes	1
2 Índice dos Componentes	3
2.1 Lista de Classes	3
3 Índice dos Arquivos	5
3.1 Lista de Arquivos	5
4 Classes	7
4.1 Referência da Classe Cadastro	7
4.1.1 Descrição detalhada	7
4.1.2 Documentação das funções	7
4.1.2.1 adicionarJogador()	7
4.1.2.2 exibirJogadores()	8
4.1.2.3 getJogadores()	8
4.1.2.4 obterJogador()	8
4.1.2.5 removerJogador()	9
4.2 Referência da Classe Gerente	9
4.2.1 Descrição detalhada	10
4.2.2 Construtores e Destrutores	10
4.2.2.1 Gerente()	10
4.2.3 Documentação das funções	10
4.2.3.1 carregarDados()	10
4.2.3.2 salvarDados()	10
4.3 Referência da Classe Jogador	11
4.3.1 Descrição detalhada	11
4.3.2 Construtores e Destrutores	11
4.3.2.1 Jogador()	11
4.3.3 Documentação das funções	12
4.3.3.1 desserializar()	12
4.3.3.2 getApelido()	12
4.3.3.3 getDerrotas()	12
4.3.3.4 getNome()	13
4.3.3.5 getVitorias()	13
4.3.3.6 registrarDerrota()	13
4.3.3.7 registrarVitoria()	13
4.3.3.8 resetJogador()	14
4.3.3.9 serializar()	14
4.3.3.10 setDerrotas()	14
4.3.3.11 setVitorias()	14
4.4 Referência da Classe JogoDaVelha	15
4.4.1 Descrição detalhada	16

4.4.2 Construtores e Destrutores	16
4.4.2.1 JogoDaVelha()	16
4.4.3 Documentação das funções	16
4.4.3.1 getNome()	16
4.4.3.2 jogar()	16
4.4.3.3 verificarVitoria()	17
4.5 Referência da Classe JogoDeTabuleiro	17
4.5.1 Descrição detalhada	18
4.5.2 Construtores e Destrutores	18
4.5.2.1 JogoDeTabuleiro()	18
4.5.3 Documentação das funções	18
4.5.3.1 exibirTabuleiro()	18
4.5.3.2 getNome()	18
4.5.3.3 jogar() [1/2]	18
4.5.3.4 jogar() [2/2]	19
4.5.3.5 verificarVitoria()	19
4.6 Referência da Classe Liga4	20
4.6.1 Descrição detalhada	21
4.6.2 Construtores e Destrutores	21
4.6.2.1 Liga4()	21
4.6.3 Documentação das funções	21
4.6.3.1 getNome()	21
4.6.3.2 jogar()	21
4.6.3.3 verificarVitoria()	22
4.7 Referência da Classe Reversi	22
4.7.1 Descrição detalhada	23
4.7.2 Construtores e Destrutores	23
4.7.2.1 Reversi()	23
4.7.3 Documentação das funções	24
4.7.3.1 exibirTabuleiro()	24
4.7.3.2 getNome()	24
4.7.3.3 jogar()	24
4.7.3.4 verificarVitoria()	24
5 Arquivos	25
5.1 Referência do Arquivo include/Cadastro.hpp	25
5.1.1 Descrição detalhada	25
5.2 Cadastro.hpp	25
5.3 Referência do Arquivo include/Gerente.hpp	26
5.3.1 Descrição detalhada	26
5.4 Gerente.hpp	26
5.5 Referência do Arquivo include/Jogador.hpp	27

5.5.1 Descrição detalhada	27
5.6 Jogador.hpp	27
5.7 Referência do Arquivo include/JogoDaVelha.hpp	28
5.7.1 Descrição detalhada	28
5.8 JogoDaVelha.hpp	28
5.9 Referência do Arquivo include/Liga4.hpp	28
5.9.1 Descrição detalhada	29
5.10 Liga4.hpp	29
5.11 Referência do Arquivo include/Reversi.hpp	29
5.11.1 Descrição detalhada	29
5.12 Reversi.hpp	30
5.13 Referência do Arquivo include/tabuleiro.hpp	30
5.13.1 Descrição detalhada	30
5.14 tabuleiro.hpp	30
5.15 Referência do Arquivo src/cadastro.cpp	31
5.15.1 Descrição detalhada	31
5.16 Referência do Arquivo src/gerente.cpp	31
5.16.1 Descrição detalhada	31
5.17 Referência do Arquivo src/jogador.cpp	32
5.17.1 Descrição detalhada	32
5.18 Referência do Arquivo src/liga4.cpp	32
5.18.1 Descrição detalhada	32
5.19 Referência do Arquivo src/main.cpp	32
5.19.1 Descrição detalhada	33
5.19.2 Funções	33
5.19.2.1 jogarJogo()	33
5.19.2.2 main()	33
5.19.2.3 selecionarJogo()	33
5.20 Referência do Arquivo src/Reversi.cpp	33
5.20.1 Descrição detalhada	34
5.21 Referência do Arquivo src/tabuleiro.cpp	34
5.21.1 Descrição detalhada	34
5.22 Referência do Arquivo src/velha.cpp	34
5.22.1 Descrição detalhada	34
Índice Remissivo	35

Capítulo 1

Índice Hierárquico

1.1 Hierarquia de Classes

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

Cadastro	7
Gerente	9
Jogador	11
JogoDeTabuleiro	17
JogoDaVelha	15
Liga4	20
Reversi	22

Capítulo 2

Índice dos Componentes

2.1 Lista de Classes

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

Cadastro	Classe responsável por gerenciar os jogadores, incluindo adição, remoção e listagem	7
Gerente	Classe responsável por salvar e carregar os dados dos jogadores	9
Jogador	Classe que representa um jogador, registrando suas vitórias, derrotas e estatísticas gerais em diferentes jogos	11
JogoDaVelha	Classe que gerencia o jogo da velha, incluindo as regras e a verificação de vitórias	15
JogoDeTabuleiro	Classe base abstrata para criar e gerenciar jogos de tabuleiro	17
Liga4	Classe que gerencia o jogo Liga4 , incluindo as regras e a verificação de vitórias	20
Reversi	Classe que gerencia o jogo Reversi , incluindo as regras e a verificação de vitórias	22

Capítulo 3

Índice dos Arquivos

3.1 Lista de Arquivos

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

include/ Cadastro.hpp	
Definição da classe Cadastro , que gerencia jogadores	25
include/ Gerente.hpp	
Definição da classe Gerente , responsável pelo manejo de dados dos jogadores	26
include/ Jogador.hpp	
Definição da classe Jogador , que gerencia informações sobre os jogadores e suas estatísticas	27
include/ JogoDaVelha.hpp	
Definição da classe JogoDaVelha , que implementa as regras do jogo da velha	28
include/ Liga4.hpp	
Definição da classe Liga4 , que implementa as regras do jogo Liga4	28
include/ Reversi.hpp	
Definição da classe Reversi , que implementa as regras do jogo Reversi	29
include/ tabuleiro.hpp	
Definição da classe base JogoDeTabuleiro para criação de jogos de tabuleiro genéricos	30
src/ cadastro.cpp	
Implementação das funções relacionadas ao cadastro de jogadores	31
src/ gerente.cpp	
Implementação das funções de gerenciamento relacionadas ao cadastro de jogadores	31
src/ jogador.cpp	
Implementação das funções relacionadas aos jogadores	32
src/ liga4.cpp	
Implementação das regras do jogo Liga4	32
src/ main.cpp	
Arquivo principal do programa, gerencia o fluxo de execução	32
src/ Reversi.cpp	
Implementação do jogo Reversi	33
src/ tabuleiro.cpp	
Implementação genérica do tabuleiro para jogos de tabuleiro	34
src/ velha.cpp	
Implementação do jogo Jogo da Velha	34

Capítulo 4

Classes

4.1 Referência da Classe Cadastro

Classe responsável por gerenciar os jogadores, incluindo adição, remoção e listagem.

```
#include <Cadastro.hpp>
```

Membros Públicos

- **Cadastro ()**
Construtor padrão da classe [Cadastro](#).
- bool [adicionarJogador](#) (const std::string &apelido, const std::string &nome)
Adiciona um novo jogador ao cadastro.
- bool [removerJogador](#) (const std::string &apelido)
Remove um jogador do cadastro com base no apelido.
- void **listarJogadores ()** const
Exibe a lista de jogadores cadastrados.
- [Jogador](#) * [obterJogador](#) (const std::string &apelido)
Obtém um jogador com base no apelido.
- std::map< std::string, [Jogador](#) > & [getJogadores](#) ()
Obtém todos os jogadores cadastrados.
- void [exibirJogadores](#) () const
Exibe os detalhes de todos os jogadores cadastrados.

4.1.1 Descrição detalhada

Classe responsável por gerenciar os jogadores, incluindo adição, remoção e listagem.

4.1.2 Documentação das funções

4.1.2.1 adicionarJogador()

```
bool Cadastro::adicionarJogador (  
    const std::string & apelido,  
    const std::string & nome)
```

Adiciona um novo jogador ao cadastro.

Parâmetros

<i>apelido</i>	O apelido do jogador.
<i>nome</i>	O nome completo do jogador.

Retorna

true se o jogador foi adicionado com sucesso, false caso o apelido já exista.

Parâmetros

<i>apelido</i>	O apelido do jogador.
<i>nome</i>	O nome completo do jogador.

Retorna

true se o jogador foi adicionado com sucesso, false caso contrário.

4.1.2.2 `exibirJogadores()`

```
void Cadastro::exibirJogadores () const
```

Exibe os detalhes de todos os jogadores cadastrados.

Exibe a lista de jogadores cadastrados.

4.1.2.3 `getJogadores()`

```
std::map< std::string, Jogador > & Cadastro::getJogadores ()
```

Obtém todos os jogadores cadastrados.

Obtém o mapa de jogadores cadastrados.

Retorna

Referência ao mapa contendo os jogadores.

Esta função é utilizada pelo [Gerente](#) para acessar todos os jogadores.

Retorna

Referência ao mapa contendo os jogadores.

4.1.2.4 `obterJogador()`

```
Jogador * Cadastro::obterJogador (  
    const std::string & apelido)
```

Obtém um jogador com base no apelido.

Obtém um jogador específico com base no apelido.

Parâmetros

<i>apelido</i>	O apelido do jogador.
----------------	-----------------------

Retorna

Ponteiro para o jogador, ou nullptr se não encontrado.

4.1.2.5 removerJogador()

```
bool Cadastro::removerJogador (
    const std::string & apelido)
```

Remove um jogador do cadastro com base no apelido.

Remove um jogador do cadastro.

Parâmetros

<i>apelido</i>	O apelido do jogador a ser removido.
----------------	--------------------------------------

Retorna

true se o jogador foi removido com sucesso, false caso o apelido não seja encontrado.

Parâmetros

<i>apelido</i>	O apelido do jogador a ser removido.
----------------	--------------------------------------

Retorna

true se o jogador foi removido com sucesso, false caso contrário.

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

- include/[Cadastro.hpp](#)
- src/[cadastro.cpp](#)

4.2 Referência da Classe Gerente

Classe responsável por salvar e carregar os dados dos jogadores.

```
#include <Gerente.hpp>
```

Membros Públicos

- [Gerente](#) ([Cadastro](#) &cadastro)
Construtor da classe [Gerente](#).
- void [salvarDados](#) () const
Salva os dados dos jogadores em um arquivo.
- void [carregarDados](#) ()
Carrega os dados dos jogadores de um arquivo.

4.2.1 Descrição detalhada

Classe responsável por salvar e carregar os dados dos jogadores.

4.2.2 Construtores e Destrutores

4.2.2.1 Gerente()

```
Gerente::Gerente (
    Cadastro & cadastro)
```

Construtor da classe [Gerente](#).

Parâmetros

cadastro	Referência ao objeto Cadastro utilizado para gerenciar os jogadores.
cadastro	Referência ao objeto Cadastro utilizado para manipulação dos jogadores.

4.2.3 Documentação das funções

4.2.3.1 carregarDados()

```
void Gerente::carregarDados ()
```

Carrega os dados dos jogadores de um arquivo.

Lê os dados do arquivo e atualiza o cadastro com as informações recuperadas.

Lê os dados serializados do arquivo "jogadores_data.txt" e os adiciona ao cadastro.

4.2.3.2 salvarDados()

```
void Gerente::salvarDados () const
```

Salva os dados dos jogadores em um arquivo.

Os dados são serializados e escritos em um arquivo para posterior recuperação.

Os dados dos jogadores são serializados e escritos no arquivo "jogadores_data.txt".

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

- include/[Gerente.hpp](#)
- src/[gerente.cpp](#)

4.3 Referência da Classe Jogador

Classe que representa um jogador, registrando suas vitórias, derrotas e estatísticas gerais em diferentes jogos.

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

Membros Públicos

- **Jogador** ()
Construtor padrão da classe [Jogador](#).
- **Jogador** (const std::string &apelido, const std::string &nome)
Construtor que inicializa o jogador com apelido e nome.
- std::string **getApelido** () const
Obtém o apelido do jogador.
- std::string **getNome** () const
Obtém o nome completo do jogador.
- int **getVitorias** () const
Obtém o total de vitórias do jogador.
- void **setVitorias** (int vitorias)
Define o total de vitórias do jogador.
- int **getDerrotas** () const
Obtém o total de derrotas do jogador.
- void **setDerrotas** (int derrotas)
Define o total de derrotas do jogador.
- void **registrarVitoria** (const std::string &jogo)
Registra uma vitória para o jogador em um jogo específico.
- void **registrarDerrota** (const std::string &jogo)
Registra uma derrota para o jogador em um jogo específico.
- void **resetJogador** ()
Reseta as estatísticas do jogador, zerando vitórias e derrotas.
- void **exibirJogador** () const
Exibe as estatísticas do jogador.
- std::string **serializar** () const
Serializa os dados do jogador em uma string.
- void **desserializar** (const std::string &dados)
Desserializa os dados do jogador a partir de uma string.

4.3.1 Descrição detalhada

Classe que representa um jogador, registrando suas vitórias, derrotas e estatísticas gerais em diferentes jogos.

4.3.2 Construtores e Destrutores

4.3.2.1 Jogador()

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

Construtor que inicializa o jogador com apelido e nome.

Construtor da classe [Jogador](#).

Parâmetros

<i>apelido</i>	Apelido do jogador.
<i>nome</i>	Nome completo do jogador.
<i>apelido</i>	O apelido do jogador.
<i>nome</i>	O nome completo do jogador.

4.3.3 Documentação das funções

4.3.3.1 desserializar()

```
void Jogador::desserializar (  
    const std::string & dados)
```

Desserializa os dados do jogador a partir de uma string.

Parâmetros

<i>dados</i>	String contendo os dados do jogador.
<i>dados</i>	A string contendo os dados do jogador.

4.3.3.2 getApelido()

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

Obtém o apelido do jogador.

Retorna

Apelido do jogador.

O apelido do jogador.

4.3.3.3 getDerrotas()

```
int Jogador::getDerrotas () const
```

Obtém o total de derrotas do jogador.

Retorna

Total de derrotas.

O total de derrotas.

4.3.3.4 getNome()

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

Obtém o nome completo do jogador.

Retorna

Nome completo do jogador.

O nome completo do jogador.

4.3.3.5 getVitorias()

```
int Jogador::getVitorias () const
```

Obtém o total de vitórias do jogador.

Retorna

Total de vitórias.

O total de vitórias.

4.3.3.6 registrarDerrota()

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

Registra uma derrota para o jogador em um jogo específico.

Parâmetros

<i>jogo</i>	Nome do jogo ("Reversi", "Liga4", ou "JogoDaVelha").
<i>jogo</i>	O nome do jogo ("Reversi", "Liga4", ou "JogoDaVelha").

4.3.3.7 registrarVitoria()

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

Registra uma vitória para o jogador em um jogo específico.

Parâmetros

<i>jogo</i>	Nome do jogo ("Reversi", "Liga4", ou "JogoDaVelha").
<i>jogo</i>	O nome do jogo ("Reversi", "Liga4", ou "JogoDaVelha").

4.3.3.8 resetJogador()

```
void Jogador::resetJogador ()
```

Reseta as estatísticas do jogador, zerando vitórias e derrotas.

Reseta os dados do jogador, zerando vitórias e derrotas.

4.3.3.9 serializar()

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

Serializa os dados do jogador em uma string.

Retorna

String contendo os dados do jogador.

A string contendo os dados do jogador.

4.3.3.10 setDerrotas()

```
void Jogador::setDerrotas (  
    int derrotas)
```

Define o total de derrotas do jogador.

Parâmetros

<i>derrotas</i>	Novo total de derrotas.
<i>derrotas</i>	O novo total de derrotas.

4.3.3.11 setVitorias()

```
void Jogador::setVitorias (  
    int vitorias)
```

Define o total de vitórias do jogador.

Parâmetros

<i>vitorias</i>	Novo total de vitórias.
<i>vitorias</i>	O novo total de vitórias.

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

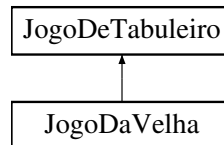
- [include/Jogador.hpp](#)
- [src/jogador.cpp](#)

4.4 Referência da Classe JogoDaVelha

Classe que gerencia o jogo da velha, incluindo as regras e a verificação de vitórias.

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

Diagrama de hierarquia da classe JogoDaVelha:



Membros Públicos

- **JogoDaVelha** ()
Construtor do jogo da velha.
- bool **jogar** (int linha, int coluna, char jogador) override
Realiza uma jogada no jogo da velha.
- bool **verificarVitoria** () const override
Verifica se houve um vencedor no jogo da velha.
- std::string **getNome** () const override
Obtém o nome do jogo.

Membros Públicos herdados de JogoDeTabuleiro

- **JogoDeTabuleiro** (int linhas, int colunas)
Construtor da classe JogoDeTabuleiro.
- virtual ~**JogoDeTabuleiro** ()
Destrutor virtual da classe JogoDeTabuleiro.
- virtual bool **jogar** (int coluna, char jogador)
Realiza uma jogada baseada apenas na coluna (implementação padrão).
- virtual void **exibirTabuleiro** () const
Exibe o estado atual do tabuleiro.
- int **getLinhas** () const
Métodos para obter o número de linhas, colunas e o conteúdo de uma posição do tabuleiro.(para uso em testes)
- int **getColunas** () const
- char **getPosicao** (int linha, int coluna) const

Outros membros herdados

Atributos Protegidos herdados de JogoDeTabuleiro

- std::vector< std::vector< char > > **tabuleiro**
Matriz que representa o tabuleiro do jogo.
- int **linhas**
Número de linhas do tabuleiro.
- int **colunas**
Número de colunas do tabuleiro.

4.4.1 Descrição detalhada

Classe que gerencia o jogo da velha, incluindo as regras e a verificação de vitórias.

4.4.2 Construtores e Destrutores

4.4.2.1 JogoDaVelha()

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

Construtor do jogo da velha.

Construtor do jogo Jogo da Velha.

Inicializa o tabuleiro do jogo com dimensões de 3x3.

Inicializa o tabuleiro com dimensões de 3x3.

4.4.3 Documentação das funções

4.4.3.1 getNome()

```
std::string JogoDaVelha::getNome () const [inline], [override], [virtual]
```

Obtém o nome do jogo.

Retorna

Uma string com o nome "JogoDaVelha".

Reimplementa [JogoDeTabuleiro](#).

4.4.3.2 jogar()

```
bool JogoDaVelha::jogar (  
    int linha,  
    int coluna,  
    char jogador) [override], [virtual]
```

Realiza uma jogada no jogo da velha.

Realiza uma jogada no jogo Jogo da Velha.

Parâmetros

<i>linha</i>	A linha onde o jogador deseja jogar.
<i>coluna</i>	A coluna onde o jogador deseja jogar.
<i>jogador</i>	O símbolo do jogador (ex.: 'X' ou 'O').

Retorna

true se a jogada foi válida, false caso contrário.

Implementa [JogoDeTabuleiro](#).

4.4.3.3 verificarVitoria()

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

Verifica se houve um vencedor no jogo da velha.

Verifica se houve um vencedor no jogo Jogo da Velha.

Retorna

true se houve um vencedor, false caso contrário.

Implementa [JogoDeTabuleiro](#).

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

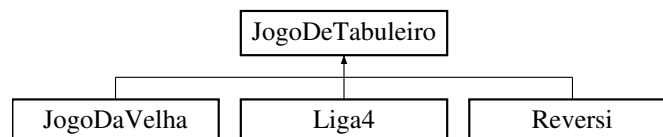
- [include/JogoDaVelha.hpp](#)
- [src/velha.cpp](#)

4.5 Referência da Classe JogoDeTabuleiro

Classe base abstrata para criar e gerenciar jogos de tabuleiro.

```
#include <tabuleiro.hpp>
```

Diagrama de hierarquia da classe JogoDeTabuleiro:



Membros Públicos

- [JogoDeTabuleiro](#) (int [linhas](#), int [colunas](#))
Construtor da classe [JogoDeTabuleiro](#).
- virtual \sim **JogoDeTabuleiro** ()
Destrutor virtual da classe [JogoDeTabuleiro](#).
- virtual bool [jogar](#) (int linha, int coluna, char jogador)=0
Realiza uma jogada em uma posição específica do tabuleiro.
- virtual bool [jogar](#) (int coluna, char jogador)
Realiza uma jogada baseada apenas na coluna (implementação padrão).
- virtual bool [verificarVitoria](#) () const =0
Verifica se houve um vencedor no jogo.
- virtual void [exibirTabuleiro](#) () const
Exibe o estado atual do tabuleiro.
- int **getLinhas** () const
Metodos para obter o número de linhas, colunas e o conteúdo de uma posição do tabuleiro.(para uso em testes)
- int **getColunas** () const
- char **getPosicao** (int linha, int coluna) const
- virtual std::string [getNome](#) () const
Obtém o nome do jogo.

Atributos Protegidos

- `std::vector< std::vector< char > > tabuleiro`
Matriz que representa o tabuleiro do jogo.
- `int linhas`
Número de linhas do tabuleiro.
- `int colunas`
Número de colunas do tabuleiro.

4.5.1 Descrição detalhada

Classe base abstrata para criar e gerenciar jogos de tabuleiro.

4.5.2 Construtores e Destrutores

4.5.2.1 JogoDeTabuleiro()

```
JogoDeTabuleiro::JogoDeTabuleiro (
    int linhas,
    int colunas)
```

Construtor da classe [JogoDeTabuleiro](#).

Parâmetros

<i>linhas</i>	O número de linhas do tabuleiro.
<i>colunas</i>	O número de colunas do tabuleiro.

4.5.3 Documentação das funções

4.5.3.1 exibirTabuleiro()

```
void JogoDeTabuleiro::exibirTabuleiro () const [virtual]
```

Exibe o estado atual do tabuleiro.

Reimplementado por [Reversi](#).

4.5.3.2 getNome()

```
virtual std::string JogoDeTabuleiro::getNome () const [inline], [virtual]
```

Obtém o nome do jogo.

Retorna

Uma string com o nome "JogoDeTabuleiro" (padrão).

Reimplementado por [JogoDaVelha](#), [Liga4](#) e [Reversi](#).

4.5.3.3 jogar() [1/2]

```
virtual bool JogoDeTabuleiro::jogar (
    int coluna,
    char jogador) [inline], [virtual]
```

Realiza uma jogada baseada apenas na coluna (implementação padrão).

Parâmetros

<i>coluna</i>	A coluna onde o jogador deseja jogar.
<i>jogador</i>	O símbolo do jogador (ex.: 'X' ou 'O').

Retorna

true se a jogada foi válida, false caso contrário.

4.5.3.4 jogar() [2/2]

```
virtual bool JogoDeTabuleiro::jogar (  
    int linha,  
    int coluna,  
    char jogador) [pure virtual]
```

Realiza uma jogada em uma posição específica do tabuleiro.

Parâmetros

<i>linha</i>	A linha onde o jogador deseja jogar.
<i>coluna</i>	A coluna onde o jogador deseja jogar.
<i>jogador</i>	O símbolo do jogador (ex.: 'X' ou 'O').

Retorna

true se a jogada foi válida, false caso contrário.

Implementado por [JogoDaVelha](#), [Liga4](#) e [Reversi](#).

4.5.3.5 verificarVitoria()

```
virtual bool JogoDeTabuleiro::verificarVitoria () const [pure virtual]
```

Verifica se houve um vencedor no jogo.

Retorna

true se houve um vencedor, false caso contrário.

Implementado por [JogoDaVelha](#), [Liga4](#) e [Reversi](#).

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

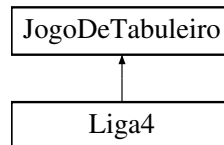
- [include/tabuleiro.hpp](#)
- [src/tabuleiro.cpp](#)

4.6 Referência da Classe Liga4

Classe que gerencia o jogo [Liga4](#), incluindo as regras e a verificação de vitórias.

```
#include <Liga4.hpp>
```

Diagrama de hierarquia da classe Liga4:



Membros Públicos

- [Liga4](#) (int [linhas](#)=6, int [colunas](#)=7)
Construtor do jogo [Liga4](#).
- virtual bool [jogar](#) (int linha, int coluna, char jogador) override
Realiza uma jogada no jogo [Liga4](#).
- virtual bool [verificarVitoria](#) () const override
Verifica se houve um vencedor no jogo [Liga4](#).
- std::string [getNome](#) () const override
Obtém o nome do jogo.

Membros Públicos herdados de [JogoDeTabuleiro](#)

- [JogoDeTabuleiro](#) (int [linhas](#), int [colunas](#))
Construtor da classe [JogoDeTabuleiro](#).
- virtual ~[JogoDeTabuleiro](#) ()
Destrutor virtual da classe [JogoDeTabuleiro](#).
- virtual void [exibirTabuleiro](#) () const
Exibe o estado atual do tabuleiro.
- int [getLinhas](#) () const
Metodos para obter o número de linhas, colunas e o conteúdo de uma posição do tabuleiro.(para uso em testes)
- int [getColunas](#) () const
- char [getPosicao](#) (int linha, int coluna) const

Outros membros herdados

Atributos Protegidos herdados de [JogoDeTabuleiro](#)

- std::vector< std::vector< char > > [tabuleiro](#)
Matriz que representa o tabuleiro do jogo.
- int [linhas](#)
Número de linhas do tabuleiro.
- int [colunas](#)
Número de colunas do tabuleiro.

4.6.1 Descrição detalhada

Classe que gerencia o jogo [Liga4](#), incluindo as regras e a verificação de vitórias.

4.6.2 Construtores e Destrutores

4.6.2.1 Liga4()

```
Liga4::Liga4 (  
    int linhas = 6,  
    int colunas = 7)
```

Construtor do jogo [Liga4](#).

Construtor da classe [Liga4](#).

Parâmetros

<i>linhas</i>	O número de linhas do tabuleiro (padrão: 6).
<i>colunas</i>	O número de colunas do tabuleiro (padrão: 7).
<i>linhas</i>	O número de linhas do tabuleiro.
<i>colunas</i>	O número de colunas do tabuleiro.

4.6.3 Documentação das funções

4.6.3.1 getNome()

```
std::string Liga4::getNome () const [inline], [override], [virtual]
```

Obtém o nome do jogo.

Retorna

Uma string com o nome "Liga4".

Reimplementa [JogoDeTabuleiro](#).

4.6.3.2 jogar()

```
bool Liga4::jogar (  
    int linha,  
    int coluna,  
    char jogador) [override], [virtual]
```

Realiza uma jogada no jogo [Liga4](#).

Realiza uma jogada no jogo [Liga4](#) especificando a linha e a coluna.

Parâmetros

<i>linha</i>	Ignorado (necessário para satisfazer a interface da classe base).
<i>coluna</i>	A coluna onde o jogador deseja jogar.
<i>jogador</i>	O símbolo do jogador (ex.: 'X' ou 'O').

Retorna

true se a jogada foi válida, false caso contrário.

Parâmetros

<i>linha</i>	A linha onde o jogador deseja jogar (não utilizado em Liga4).
<i>coluna</i>	A coluna onde o jogador deseja jogar.
<i>jogador</i>	O símbolo do jogador (ex.: 'X' ou 'O').

Retorna

true se a jogada foi válida, false caso contrário.

Implementa [JogoDeTabuleiro](#).

4.6.3.3 verificarVitoria()

```
bool Liga4::verificarVitoria () const [override], [virtual]
```

Verifica se houve um vencedor no jogo [Liga4](#).

Retorna

true se houve um vencedor, false caso contrário.

Implementa [JogoDeTabuleiro](#).

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

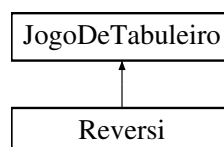
- [include/Liga4.hpp](#)
- [src/liga4.cpp](#)

4.7 Referência da Classe Reversi

Classe que gerencia o jogo [Reversi](#), incluindo as regras e a verificação de vitórias.

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

Diagrama de hierarquia da classe Reversi:



Membros Públicos

- **Reversi** ()
*Construtor do jogo **Reversi**.*
- bool **jogar** (int linha, int coluna, char jogador) override
*Realiza uma jogada no jogo **Reversi**.*
- bool **verificarVitoria** () const override
*Verifica se o jogo **Reversi** foi concluído.*
- void **exibirTabuleiro** () const override
*Exibe o estado atual do tabuleiro do jogo **Reversi**.*
- std::string **getNome** () const override
Obtém o nome do jogo.

Membros Públicos herdados de **JogoDeTabuleiro**

- **JogoDeTabuleiro** (int linhas, int colunas)
*Construtor da classe **JogoDeTabuleiro**.*
- virtual ~**JogoDeTabuleiro** ()
*Destrutor virtual da classe **JogoDeTabuleiro**.*
- virtual bool **jogar** (int coluna, char jogador)
Realiza uma jogada baseada apenas na coluna (implementação padrão).
- int **getLinhas** () const
Metodos para obter o número de linhas, colunas e o conteúdo de uma posição do tabuleiro.(para uso em testes)
- int **getColunas** () const
- char **getPosicao** (int linha, int coluna) const

Outros membros herdados

Atributos Protegidos herdados de **JogoDeTabuleiro**

- std::vector< std::vector< char > > **tabuleiro**
Matriz que representa o tabuleiro do jogo.
- int **linhas**
Número de linhas do tabuleiro.
- int **colunas**
Número de colunas do tabuleiro.

4.7.1 Descrição detalhada

Classe que gerencia o jogo **Reversi**, incluindo as regras e a verificação de vitórias.

4.7.2 Construtores e Destrutores

4.7.2.1 **Reversi**()

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

Construtor do jogo **Reversi**.

Inicializa o tabuleiro com a configuração inicial padrão do jogo.

Inicializa o tabuleiro com a configuração padrão do jogo.

4.7.3 Documentação das funções

4.7.3.1 `exibirTabuleiro()`

```
void Reversi::exibirTabuleiro () const [override], [virtual]
```

Exibe o estado atual do tabuleiro do jogo [Reversi](#).

Reimplementa [JogoDeTabuleiro](#).

4.7.3.2 `getNome()`

```
std::string Reversi::getNome () const [inline], [override], [virtual]
```

Obtém o nome do jogo.

Retorna

Uma string com o nome "Reversi".

Reimplementa [JogoDeTabuleiro](#).

4.7.3.3 `jogar()`

```
bool Reversi::jogar (  
    int linha,  
    int coluna,  
    char jogador) [override], [virtual]
```

Realiza uma jogada no jogo [Reversi](#).

Parâmetros

<i>linha</i>	A linha onde o jogador deseja jogar.
<i>coluna</i>	A coluna onde o jogador deseja jogar.
<i>jogador</i>	O símbolo do jogador (ex.: 'X' ou 'O').

Retorna

true se a jogada foi válida, false caso contrário.

Implementa [JogoDeTabuleiro](#).

4.7.3.4 `verificarVitoria()`

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

Verifica se o jogo [Reversi](#) foi concluído.

Retorna

true se o jogo terminou, false caso contrário.

Implementa [JogoDeTabuleiro](#).

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

- include/[Reversi.hpp](#)
- src/[Reversi.cpp](#)

Capítulo 5

Arquivos

5.1 Referência do Arquivo include/Cadastro.hpp

Definição da classe [Cadastro](#), que gerencia jogadores.

```
#include "Jogador.hpp"
#include <map>
#include <string>
```

Componentes

- class [Cadastro](#)

Classe responsável por gerenciar os jogadores, incluindo adição, remoção e listagem.

5.1.1 Descrição detalhada

Definição da classe [Cadastro](#), que gerencia jogadores.

5.2 Cadastro.hpp

[Ir para a documentação desse arquivo.](#)

```
00001
00005
00006 #ifndef CADASTRO_HPP
00007 #define CADASTRO_HPP
00008 #define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN
00009
00010
00011 #include "Jogador.hpp"
00012 #include <map>
00013 #include <string>
00014
00019 class Cadastro {
00020 private:
00024     std::map<std::string, Jogador> jogadores;
00025
00026 public:
00030     Cadastro();
00031
```

```

00038     bool adicionarJogador(const std::string& apelido, const std::string& nome);
00039
00045     bool removerJogador(const std::string& apelido);
00046
00050     void listarJogadores() const;
00051
00057     Jogador* obterJogador(const std::string& apelido);
00058
00065     std::map<std::string, Jogador>& getJogadores();
00066
00070     void exibirJogadores() const;
00071 };
00072
00073 #endif // CADASTRO_HPP

```

5.3 Referência do Arquivo include/Gerente.hpp

Definição da classe [Gerente](#), responsável pelo manejo de dados dos jogadores.

```

#include "Cadastro.hpp"
#include <fstream>

```

Componentes

- class [Gerente](#)

Classe responsável por salvar e carregar os dados dos jogadores.

5.3.1 Descrição detalhada

Definição da classe [Gerente](#), responsável pelo manejo de dados dos jogadores.

5.4 Gerente.hpp

[Ir para a documentação desse arquivo.](#)

```

00001
00005
00006 #ifndef GERENTE_HPP
00007 #define GERENTE_HPP
00008 #define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN
00009
00010
00011 #include "Cadastro.hpp"
00012 #include <fstream>
00013
00018 class Gerente {
00019 private:
00023     Cadastro& cadastro;
00024
00025 public:
00030     Gerente(Cadastro& cadastro);
00031
00037     void salvarDados() const;
00038
00044     void carregarDados();
00045 };
00046
00047 #endif // GERENTE_HPP

```


5.5 Referência do Arquivo include/Jogador.hpp

Definição da classe `Jogador`, que gerencia informações sobre os jogadores e suas estatísticas.

```
#include <locale.h>
#include <string>
```

Componentes

- class `Jogador`

Classe que representa um jogador, registrando suas vitórias, derrotas e estatísticas gerais em diferentes jogos.

5.5.1 Descrição detalhada

Definição da classe `Jogador`, que gerencia informações sobre os jogadores e suas estatísticas.

5.6 Jogador.hpp

[Ir para a documentação desse arquivo.](#)

```
00001
00005
00006 #ifndef JOGADOR_HPP
00007 #define JOGADOR_HPP
00008 #define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN
00009
00010
00011 #include <locale.h>
00012 #include <string>
00013
00018 class Jogador {
00019 private:
00023     std::string Apelido;
00024
00028     std::string Nome;
00029
00033     int vitoriaReversi, derrotaReversi;
00034
00038     int vitoriaLiga4, derrotaLiga4;
00039
00043     int vitoriaVelha, derrotaVelha;
00044
00048     int totalVitorias;
00049     int totalDerrotas;
00050
00051 public:
00055     Jogador();
00056
00062     Jogador(const std::string& apelido, const std::string& nome);
00063
00068     std::string getApelido() const;
00069
00074     std::string getNome() const;
00075
00080     int getVitorias() const;
00081
00086     void setVitorias(int vitorias);
00087
00092     int getDerrotas() const;
00093
00098     void setDerrotas(int derrotas);
00099
00104     void registrarVitoria(const std::string& jogo);
00105
00110     void registrarDerrota(const std::string& jogo);
00111
00115     void resetJogador();
```

```

00116
00120     void exibirJogador() const;
00121
00126     std::string serializar() const;
00127
00132     void desserializar(const std::string& dados);
00133 };
00134
00135 #endif // JOGADOR_HPP

```

5.7 Referência do Arquivo include/JogoDaVelha.hpp

Definição da classe [JogoDaVelha](#), que implementa as regras do jogo da velha.

```
#include "tabuleiro.hpp"
```

Componentes

- class [JogoDaVelha](#)
Classe que gerencia o jogo da velha, incluindo as regras e a verificação de vitórias.

5.7.1 Descrição detalhada

Definição da classe [JogoDaVelha](#), que implementa as regras do jogo da velha.

5.8 JogoDaVelha.hpp

[Ir para a documentação desse arquivo.](#)

```

00001
00005
00006 #ifndef JOGODAVELHA_HPP
00007 #define JOGODAVELHA_HPP
00008 #define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN
00009
00010
00011 #include "tabuleiro.hpp"
00012
00017 class JogoDaVelha : public JogoDeTabuleiro {
00018 public:
00024     JogoDaVelha();
00025
00033     bool jogar(int linha, int coluna, char jogador) override;
00034
00039     bool verificarVitoria() const override;
00040
00045     std::string getNome() const override {
00046         return "JogoDaVelha";
00047     }
00048 };
00049
00050 #endif // JOGODAVELHA_HPP

```

5.9 Referência do Arquivo include/Liga4.hpp

Definição da classe [Liga4](#), que implementa as regras do jogo [Liga4](#).

```
#include "tabuleiro.hpp"
```

Componentes

- class [Liga4](#)

Classe que gerencia o jogo [Liga4](#), incluindo as regras e a verificação de vitórias.

5.9.1 Descrição detalhada

Definição da classe [Liga4](#), que implementa as regras do jogo [Liga4](#).

5.10 Liga4.hpp

[Ir para a documentação desse arquivo.](#)

```
00001
00005
00006 #ifndef LIGA4_HPP
00007 #define LIGA4_HPP
00008 #define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN
00009
00010
00011 #include "tabuleiro.hpp"
00012
00017 class Liga4 : public JogoDeTabuleiro {
00018 public:
00024     Liga4(int linhas = 6, int colunas = 7);
00025
00033     virtual bool jogar(int linha, int coluna, char jogador) override;
00034
00039     virtual bool verificarVitoria() const override;
00040
00045     std::string getNome() const override {
00046         return "Liga4";
00047     }
00048
00049 private:
00056     bool jogar(int coluna, char jogador);
00057 };
00058
00059 #endif // LIGA4_HPP
```

5.11 Referência do Arquivo include/Reversi.hpp

Definição da classe [Reversi](#), que implementa as regras do jogo [Reversi](#).

```
#include "tabuleiro.hpp"
```

Componentes

- class [Reversi](#)

Classe que gerencia o jogo [Reversi](#), incluindo as regras e a verificação de vitórias.

5.11.1 Descrição detalhada

Definição da classe [Reversi](#), que implementa as regras do jogo [Reversi](#).

5.12 Reversi.hpp

[Ir para a documentação desse arquivo.](#)

```
00001
00005
00006 #ifndef REVERSI_HPP
00007 #define REVERSI_HPP
00008 #define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN
00009
00010 #include "tabuleiro.hpp"
00011
00016 class Reversi : public JogoDeTabuleiro {
00017 public:
00023     Reversi();
00024
00032     bool jogar(int linha, int coluna, char jogador) override;
00033
00038     bool verificarVitoria() const override;
00039
00043     void exibirTabuleiro() const override;
00044
00049     std::string getNome() const override {
00050         return "Reversi";
00051     }
00052
00053 private:
00059     bool daPraandar(char player) const;
00060 };
00061
00062 #endif // REVERSI_HPP
```

5.13 Referência do Arquivo include/tabuleiro.hpp

Definição da classe base [JogoDeTabuleiro](#) para criação de jogos de tabuleiro genéricos.

```
#include <vector>
#include <iostream>
```

Componentes

- class [JogoDeTabuleiro](#)
Classe base abstrata para criar e gerenciar jogos de tabuleiro.

5.13.1 Descrição detalhada

Definição da classe base [JogoDeTabuleiro](#) para criação de jogos de tabuleiro genéricos.

5.14 tabuleiro.hpp

[Ir para a documentação desse arquivo.](#)

```
00001
00005
00006 #ifndef TABULEIRO_HPP
00007 #define TABULEIRO_HPP
00008 #define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN
00009
00010
00011 #include <vector>
00012 #include <iostream>
```

```
00013
00018 class JogoDeTabuleiro {
00019 protected:
00023     std::vector<std::vector<char> > tabuleiro;
00024
00028     int linhas;
00029
00033     int colunas;
00034
00035 public:
00041     JogoDeTabuleiro(int linhas, int colunas);
00042
00046     virtual ~JogoDeTabuleiro() {}
00047
00055     virtual bool jogar(int linha, int coluna, char jogador) = 0;
00056
00063     virtual bool jogar(int coluna, char jogador) { return false; }
00064
00069     virtual bool verificarVitoria() const = 0;
00070
00074     virtual void exibirTabuleiro() const;
00075
00076     int getLinhas() const { return linhas; }
00077     int getColunas() const { return colunas; }
00078     char getPosicao(int linha, int coluna) const { return tabuleiro[linha][coluna]; }
00079
00084     virtual std::string getNome() const { return "JogoDeTabuleiro"; }
00085 };
00086
00087 #endif
```

5.15 Referência do Arquivo src/cadastro.cpp

Implementação das funções relacionadas ao cadastro de jogadores.

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

5.15.1 Descrição detalhada

Implementação das funções relacionadas ao cadastro de jogadores.

5.16 Referência do Arquivo src/gerente.cpp

Implementação das funções de gerenciamento relacionadas ao cadastro de jogadores.

```
#include "Gerente.hpp"
#include <fstream>
#include <iostream>
#include <sstream>
```

5.16.1 Descrição detalhada

Implementação das funções de gerenciamento relacionadas ao cadastro de jogadores.

5.17 Referência do Arquivo src/jogador.cpp

Implementação das funções relacionadas aos jogadores.

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

5.17.1 Descrição detalhada

Implementação das funções relacionadas aos jogadores.

5.18 Referência do Arquivo src/liga4.cpp

Implementação das regras do jogo [Liga4](#).

```
#include "Liga4.hpp"
#include <iostream>
```

5.18.1 Descrição detalhada

Implementação das regras do jogo [Liga4](#).

5.19 Referência do Arquivo src/main.cpp

Arquivo principal do programa, gerencia o fluxo de execução.

```
#include "Cadastro.hpp"
#include "Gerente.hpp"
#include "JogoDaVelha.hpp"
#include "Reversi.hpp"
#include "Liga4.hpp"
#include <iostream>
#include <memory>
#include <locale.h>
```

Funções

- `std::unique_ptr< JogoDeTabuleiro > selecionarJogo ()`
Permite ao usuário selecionar um jogo.
- `void jogarJogo (JogoDeTabuleiro *jogo, Cadastro &cadastro, Gerente &gerente)`
Gerencia a execução de uma partida do jogo.
- `int main ()`
Função principal do programa.

5.19.1 Descrição detalhada

Arquivo principal do programa, gerencia o fluxo de execução.

5.19.2 Funções

5.19.2.1 jogarJogo()

```
void jogarJogo (
    JogoDeTabuleiro * jogo,
    Cadastro & cadastro,
    Gerente & gerente)
```

Gerencia a execução de uma partida do jogo.

Parâmetros

<i>jogo</i>	Ponteiro para o jogo que será jogado.
<i>cadastro</i>	Referência ao objeto Cadastro para manipular jogadores.
<i>gerente</i>	Referência ao objeto Gerente para salvar e carregar dados.

5.19.2.2 main()

```
int main ()
```

Função principal do programa.

Gerencia o cadastro de jogadores, escolha de jogos e execução das partidas.

Retorna

int Status de saída do programa.

5.19.2.3 selecionarJogo()

```
std::unique_ptr< JogoDeTabuleiro > selecionarJogo ()
```

Permite ao usuário selecionar um jogo.

Retorna

Um ponteiro exclusivo para o jogo selecionado.

5.20 Referência do Arquivo src/Reversi.cpp

Implementação do jogo [Reversi](#).

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

5.20.1 Descrição detalhada

Implementação do jogo [Reversi](#).

5.21 Referência do Arquivo src/tabuleiro.cpp

Implementação genérica do tabuleiro para jogos de tabuleiro.

```
#include "tabuleiro.hpp"
```

5.21.1 Descrição detalhada

Implementação genérica do tabuleiro para jogos de tabuleiro.

5.22 Referência do Arquivo src/velha.cpp

Implementação do jogo Jogo da Velha.

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

5.22.1 Descrição detalhada

Implementação do jogo Jogo da Velha.

Índice Remissivo

adicionarJogador
 Cadastro, 7

Cadastro, 7
 adicionarJogador, 7
 exibirJogadores, 8
 getJogadores, 8
 obterJogador, 8
 removerJogador, 9

carregarDados
 Gerente, 10

desserializar
 Jogador, 12

exibirJogadores
 Cadastro, 8

exibirTabuleiro
 JogoDeTabuleiro, 18
 Reversi, 24

Gerente, 9
 carregarDados, 10
 Gerente, 10
 salvarDados, 10

getApelido
 Jogador, 12

getDerrotas
 Jogador, 12

getJogadores
 Cadastro, 8

getNome
 Jogador, 12
 JogoDaVelha, 16
 JogoDeTabuleiro, 18
 Liga4, 21
 Reversi, 24

getVitorias
 Jogador, 13

include/Cadastro.hpp, 25
include/Gerente.hpp, 26
include/Jogador.hpp, 27
include/JogoDaVelha.hpp, 28
include/Liga4.hpp, 28, 29
include/Reversi.hpp, 29, 30
include/tabuleiro.hpp, 30

Jogador, 11
 desserializar, 12

getApelido, 12
getDerrotas, 12
getNome, 12
getVitorias, 13
Jogador, 11
registrarDerrota, 13
registrarVitoria, 13
resetJogador, 13
serializar, 14
setDerrotas, 14
setVitorias, 14

jogar
 JogoDaVelha, 16
 JogoDeTabuleiro, 18, 19
 Liga4, 21
 Reversi, 24

jogarJogo
 main.cpp, 33
JogoDaVelha, 15
 getNome, 16
 jogar, 16
 JogoDaVelha, 16
 verificarVitoria, 16
JogoDeTabuleiro, 17
 exibirTabuleiro, 18
 getNome, 18
 jogar, 18, 19
 JogoDeTabuleiro, 18
 verificarVitoria, 19

Liga4, 20
 getNome, 21
 jogar, 21
 Liga4, 21
 verificarVitoria, 22

main
 main.cpp, 33
main.cpp
 jogarJogo, 33
 main, 33
 selecionarJogo, 33

obterJogador
 Cadastro, 8

registrarDerrota
 Jogador, 13
registrarVitoria
 Jogador, 13

- removerJogador
 - Cadastro, [9](#)
- resetJogador
 - Jogador, [13](#)
- Reversi, [22](#)
 - exibirTabuleiro, [24](#)
 - getNome, [24](#)
 - jogar, [24](#)
 - Reversi, [23](#)
 - verificarVitoria, [24](#)
- salvarDados
 - Gerente, [10](#)
- selecionarJogo
 - main.cpp, [33](#)
- serializar
 - Jogador, [14](#)
- setDerrotas
 - Jogador, [14](#)
- setVitorias
 - Jogador, [14](#)
- src/cadastro.cpp, [31](#)
- src/gerente.cpp, [31](#)
- src/jogador.cpp, [32](#)
- src/liga4.cpp, [32](#)
- src/main.cpp, [32](#)
- src/Reversi.cpp, [33](#)
- src/tabuleiro.cpp, [34](#)
- src/velha.cpp, [34](#)
- verificarVitoria
 - JogoDaVelha, [16](#)
 - JogoDeTabuleiro, [19](#)
 - Liga4, [22](#)
 - Reversi, [24](#)