TP1

Gerado por Doxygen 1.9.8

Capítulo 1

Índice dos Componentes

1.1 Lista de Classes

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

Capítulo 2

Índice dos Arquivos

2.1 Lista de Arquivos

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

Capítulo 3

Classes

3.1 Referência da Classe Cadastro

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.

3.1.1 Documentação das funções

3.1.1.1 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

3.1.1.2 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

3.1.1.3 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

3.1.1.4 mostrarJogadores()

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

Exibe todos os jogadores cadastrados.

Exceções

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

3.1.1.5 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

3.1.1.6 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

3.2 Referência da Classe Jogador

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)

- 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.

3.2.1 Documentação das funções

3.2.1.1 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

3.2.1.2 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

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

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

3.3 Referência da Estrutura Winrate

Atributos Públicos

- int **_vitorias**
- int **_derrotas**

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

- include/cadastro.hpp

Capítulo 4

Arquivos

4.1 cadastro.hpp

```
00001 #ifndef CADASTRO_HPP
00002 #define CADASTRO_HPP
00003
00004 #include <iostream>
00005 #include <vector>
00006 #include <string>
00007 #include <fstream>
00008 #include <memory>
00009 #include <algorithm>
00010
00011 struct Winrate {
00012     int _vitorias;
00013     int _derrotas;
00014
00015 };
00016
00017 class Jogador {
00018 private:
00019     Winrate Velha;  // Primeiro membro
00020     Winrate Lig4;   // Segundo membro
00021     Winrate Reversi; // Terceiro membro
00022     std::string _nome;
00023     std::string _apelido;
00024     //adicionar outros jogos
00025
00026 public:
00027     // construtor
00028     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)
00029         : _nome(nome), _apelido(apelido),
00030         Velha{vitorias1, derrotas1},
00031         Lig4{vitorias2, derrotas2},
00032         Reversi{vitorias3, derrotas3}
00033         //adicionar outros jogos aqui.
00034         {}
00035
00036     //destrutor
00037     ~Jogador() {}
00038
00039     // métodos de acesso
00040     std::string getNome() const { return _nome; }
00041     std::string getApelido() const { return _apelido; }
00042     const Winrate& getLig4() const { return Lig4; }
00043     const Winrate& getReversi() const { return Reversi; }
00044     const Winrate& getVelha() const { return Velha; }
00045     //possível adicionar outros jogos aqui.
00046     int getVitorias(const Winrate& jogo) const { return jogo._vitorias; }
00047     int getDerrotas(const Winrate& jogo) const { return jogo._derrotas; }
00048
00049     void setNome(const std::string& nome) { _nome = nome; }
00050     void setApelido(const std::string& apelido) { _apelido = apelido; }
00051     void setVitorias(Winrate& jogo, int vitorias) { jogo._vitorias = vitorias; }
00052     void setDerrotas(Winrate& jogo, int derrotas) { jogo._derrotas = derrotas; }
00053
00054     // serializar o jogador como string para salvar em arquivo
00055     std::string serializar() const;
00056
00057     //deserializar uma string para criar um jogador
```

```
00058     static Jogador deserializar(const std::string& linha);
00059 };
00060
00061
00062 class Cadastro {
00063 private:
00064     std::vector<std::unique_ptr<Jogador» _jogadores;
00065
00066 public:
00067
00068     const std::vector<std::unique_ptr<Jogador»& get_jogadores() const {
00069     return _jogadores; }
00070
00071     //adiciona um jogador ao vetor de cadastro
00072     void adicionarJogador(const Jogador& alvo);
00073
00074     //mostra o vetor de cadastros
00075     void mostrarJogadores() const;
00076
00077     //importa de um arquivo .txt todos os cadastros
00078     void import(const std::string& caminho) ;
00079
00080     //salva e atualiza os dados de cadastro em um .txt
00081     void save(const std::string& caminho) ;
00082
00083     //remove o jogador alvo do vetor de cadastros
00084     void removeJogador(const Jogador& alvo) ;
00085
00086     //verifica se o jogador alvo está no vetor de cadastros, retorna 0 ou 1.
00087     bool check(const Jogador& alvo) const ;
00088
00089 };
00090
00091 #endif
```