Hangman

1.0

Generato da Doxygen 1.9.5

1 Indice dei tipi composti

1.1 Elenco dei tipi composti

Queste sono le classi, le struct, le union e le interfacce con una loro breve descrizione:

2 Documentazione delle classi

2.1 Riferimenti per la struct Server::HangmanServer::address

Contiene l'indirizzo IP e la porta del server.

2.1.1 Descrizione dettagliata

Contiene l'indirizzo IP e la porta del server.

2.2 Riferimenti per la classe Client::HangmanClient

Composti

- struct server_address

  Struttura contenente le informazioni del server.

Membri pubblici

- HangmanClient (const char address[ ], const char port[ ])
- HangmanClient (const std::string &address, int port)
- HangmanClient (const std::string &address, const std::string &port)
- ~HangmanClient ()
- void join (const char username[ ])
- void loop ()
- void run (bool verbose=true)
- void close ()

Membri protetti

- bool _getLetter ()
- bool _getShortPhrase ()
- void _waitAction ()
- void _printYourTurn ()
- void _printOtherTurn (Server::OtherOneTurnMessage *message)
- void _printPlayerList (Server::UpdateUserMessage *message)
- void _printShortPhrase (Server::UpdateShortPhraseMessage *message)
- void _printAttempts (Server::UpdateAttemptsMessage *message)
- void _printHangman (int mistakes)
- void _printWin ()
- void _printLose ()

Membri privati

- template<typename TypeMessage >
  bool _send (TypeMessage &message)
- template<typename TypeMessage >
  bool _receive (TypeMessage &message)

Attributi privati

- int **sockfd**

  Descrittore del socket del server.
- char **attempts** [26] {}

  Rappresenta i tentativi fatti fino al momento di accessi durante la partita.
- int **attempts_count** {}

  Rappresenta la lunghezza dell'array attempts.
- char **short_phrase** [SHORTPHRASE_LENGTH] {}

  Rappresenta la frase da indovinare.
- char **blocked_letters** [6] = "AEIOU"

  Rappresenta le lettere che non si possono usare all'inizio.
- uint8_t **blocked_letters_round** = 3

  Rappresenta il numero di turni dopo i quali si possono usare le lettere bloccate.
- uint8_t **players_count** = 0

  Contiene il numero di giocatori connessi al server.
- bool **game_over** = true

  Contiene se la partita è finita.

2.2.1 Descrizione dettagliata

Questa classe rappresenta il client del gioco dell'impiccato.

Il client si occupa di connettersi al server e inviare le richieste di gioco rispettando il protocollo.

Nota

Questa classe non è thread-safe

Autori

Genero Albis Federico, Amato Davide, Toniutti John

2.2.2 Documentazione dei costruttori e dei distruttori

2.2.2.1 HangmanClient() [1/3]

```
Client::HangmanClient::HangmanClient (
        const char address[],
        const char port[] )
```

Costruttore della classe

Parametri

address	L'indirizzo ip del server
port	La porta del server

2.2.2.2 HangmanClient() [2/3] Client::HangmanClient::HangmanClient (const std::string & *address*, int *port*) [inline]

Costruttore della classe

Parametri

address	L'indirizzo ip del server
port	La porta del server

2.2.2.3 HangmanClient() [3/3] Client::HangmanClient::HangmanClient (const std::string & *address*, const std::string & *port*) [inline]

Costruttore della classe

Parametri

address	L'indirizzo ip del server
port	La porta del server

2.2.2.4 ~HangmanClient() Client::HangmanClient::~HangmanClient ( )

Chiude e libera il socket

2.2.3 Documentazione delle funzioni membro

2.2.3.1 _getLetter() bool Client::HangmanClient::_getLetter ( ) [protected]

Questa funzione si occupa di ricevere una lettere in input da tastiera e di inviarla al server

Restituisce

True se è andato tutto a buon fine, false altrimenti

Valori di ritorno

True	se è andato tutto a buon fine
False	se si ha raggiunto il timeout

2.2.3.2 _getShortPhrase() `bool Client::HangmanClient::_getShortPhrase ( ) [protected]`

Questa funzione si occupa di ricevere una frase in input da tastiera e di inviarla al server

Restituisce

True se è andato tutto a buon fine, false altrimenti

Valori di ritorno

True	se è andato tutto a buon fine
False	se si ha raggiunto il timeout

2.2.3.3 _printAttempts() `void Client::HangmanClient::_printAttempts ( Server::UpdateAttemptsMessage * message ) [protected]`

Questa funzione si occupa di stampare a video i tentativi fatti

Parametri

message	Il messaggio ricevuto dal server

2.2.3.4 _printHangman() `void Client::HangmanClient::_printHangman ( int mistakes ) [protected]`

Questa funzione si occupa di stampare a video lo stato attuale dell'Hangman

Parametri

mistakes	Mostra a video l'Hangman

2.2.3.5 _printLose() `void Client::HangmanClient::_printLose ( ) [protected]`

Questa funzione si occupa di stampare a video che il gioco è finito e i giocatori hanno perso

2.2.3.6 _printOtherTurn() `void Client::HangmanClient::_printOtherTurn ( Server::OtherOneTurnMessage * message ) [protected]`

Questa funzione si occupa di stampare a video che è il turno di un altro giocatore

Parametri

message	Il messaggio ricevuto dal server

2.2.3.7 _printPlayerList() `void Client::HangmanClient::_printPlayerList ( Server::UpdateUserMessage * message ) [protected]`

Questa funzione si occupa di stampare a video i giocatori connessi

Parametri

message	Il messaggio ricevuto dal server

2.2.3.8 _printShortPhrase() `void Client::HangmanClient::_printShortPhrase ( Server::UpdateShortPhraseMessage * message ) [protected]`

Questa funzione si occupa di stampare a video la frase da indovinare

Parametri

message	Il messaggio ricevuto dal server

2.2.3.9 _printWin() `void Client::HangmanClient::_printWin ( ) [protected]`

Questa funzione si occupa di stampare a video che il gioco è finito e i giocatori hanno vinto

2.2.3.10 _printYourTurn() `void Client::HangmanClient::_printYourTurn ( ) [protected]`

Questa funzione si occupa di stampare a video che è il tuo turno

2.2.3.11 _receive() `template<typename TypeMessage > bool Client::HangmanClient::_receive ( TypeMessage & message ) [private]`

Questa funzione si occupa di ricevere un messaggio dal server

Parametri dei template

TypeMessage	Il tipo del messaggio da ricevere

Parametri

message	Dove salvare il messaggio ricevuto

Restituisce

True se la ricezione è andata a buon fine, false altrimenti

Valori di ritorno

True	se la ricezione è andata a buon fine
False	se la ricezione è fallita

Nota

Il messaggio viene inviato in maniera bloccante

2.2.3.12 _send() `template<typename TypeMessage >`

```
bool Client::HangmanClient::_send (
            TypeMessage & message )  [private]
```

Questa funzione si occupa di inviare un messaggio al server

Parametri dei template

TypeMessage	Il tipo del messaggio da inviare

Parametri

message	Il messaggio da inviare

Restituisce

True se l'invio è andato a buon fine, false altrimenti

Valori di ritorno

True	se l'invio è andato a buon fine
False	se l'invio è fallito

Nota

Il messaggio viene inviato in maniera bloccante

2.2.3.13 _waitAction() `void Client::HangmanClient::_waitAction ( ) [protected]`

Questa funzione permette di bloccare l'esecuzione fino a quando non viene ricevuto un messaggio dal server

2.2.3.14 close() `void Client::HangmanClient::close ( )`

Questa funzione si occupa di chiudere la connessione col server

2.2.3.15 join() `void Client::HangmanClient::join ( const char username[] )`

Questa funzione si occupa di connettersi al server

Parametri

username	Lo username dell'utente

2.2.3.16 loop() `void Client::HangmanClient::loop ( )`

Questa funzione si occupa di gestire la partita per il client

2.2.3.17 run() `void Client::HangmanClient::run ( bool verbose = true )`

Questa funzione si occupa di gestire l'ingresso del client nel gioco e di gestire la partita

Parametri

verbose	Se true, verranno stampati i messaggi di errore

2.3 Riferimenti per la classe Server::HangmanServer

Composti

- struct address

  Contiene l'indirizzo IP e la porta del server.

Membri pubblici

- HangmanServer (const string &_ip="0.0.0.0", uint16_t _port=9090)
- ~HangmanServer ()

  Chiude il socket del server e con i giocatori connessi.
- void start (uint8_t _max_errors=10, const string _start_blocked_letters="AEIOU", uint8_t _blocked_attempts=3, const string &_filename="data/data.txt")
- void run (const bool verbose=true)

  Permette di lasciare la gestione del server alla classe stessa, che si occuperà di avviare il server e gestire il loop di gioco.

Membri protetti

- void _load_short_phrases (const string &filename="data/data.txt")
- void _generate_short_phrase ()
- void _next_turn ()
- void _broadcast_action (Server::Action action)
- void _send_update_short_phrase (Player &player)
- void _send_update_attempts (Player &player)
- void _send_update_players (Player &player)
- bool _send_action (Player *player, Server::Action action)
- void _remove_player (Player *player)
- bool _is_short_phrase_guessed ()
- int _get_letter_from_player (Player *player, int timeout=5)
- int _get_short_phrase_from_player (Player *player, int timeout=10)
- void _check_disconnected_players ()
- void accept ()
- void loop ()

  Si occupa di gestire le connessioni e le richieste dei client, quindi di eseguire il gioco.
- void new_round ()

Membri privati

- template<typename TypeMessage >
  bool _send (Player *player, TypeMessage &message)
- template<class TypeMessage >
  bool _read (Player *player, TypeMessage &message, Client::Action action=Client::Action::GENERIC, int timeout=5)

Attributi privati

- int **sockfd**

  Descrittore del socket del server.
- unsigned int **max_errors** {}

  Rappresenta il numero di errori massimo che i giocatori possono commettere.
- unsigned int **current_errors** {}

  Rappresenta il numero di errori commessi dai giocatori.
- std::vector< char > **attempts**

  Rappresenta i tentativi fatti fin'ora.
- unsigned int **current_attempt** {}

  Rappresenta quanti tentativi sono stati fatti.
- char **short_phrase** [SHORTPHRASE_LENGTH] {}

  Rappresenta la parola o frase da indovinare.
- char **short_phrase_masked** [SHORTPHRASE_LENGTH] {}

  Rappresenta la parola o frase da indovinare con i caratteri non ancora indovinati sostituiti da _.
- char **start_blocked_letters** [26] {}

  Rappresenta le lettere che non si possono indovinare all'inizio.
- unsigned int **blocked_attempts** {}

  Rappresenta il numero di tentativi che devono essere fatti prima di poter usare le lettere bloccate.
- std::vector< Player > **players**

  Lista dei client connessi.
- unsigned int **players_connected** {}

  Rappresenta il numero di giocatori connessi.
- Player * **current_player** {}

  Rappresenta il giocatore corrente.
- std::vector< string > **all_phrases**

  Contiene tutte le possibili frasi da indovinare.

2.3.1 Descrizione dettagliata

Questa classe rappresenta l'intero server del gioco dell'impiccato

Provvede a dare una funzione per eseguire il gioco direttamente e a dare le funzioni per crearsi il proprio loop di gioco nel caso fosse necessario.

Nota

Questa classe non è thread-safe

Avvertimento

Se un client non rispetta il protocollo, questo viene disconnesso

Autore

John Toniutti

2.3.2 Documentazione dei costruttori e dei distruttori

2.3.2.1 HangmanServer() `Server::HangmanServer::HangmanServer (const string & _ip = "0.0.0.0", uint16_t _port = 9090 )`

Costruttore della classe HangmanServer

Parametri

_ip	L'indirizzo IP del server (se lasciato come default usa tutte le interfacce disponibili)
_port	La porta del server

Eccezioni

std::runtime_error	Se non è possibile creare il socket

2.3.2.2 ~HangmanServer() `Server::HangmanServer::~HangmanServer ( )`

Chiude il socket del server e con i giocatori connessi.

Distruttore della classe HangmanServer

2.3.3 Documentazione delle funzioni membro

2.3.3.1 _broadcast_action() `void Server::HangmanServer::_broadcast_action ( Server::Action action ) [inline], [protected]`

Permette di inviare a tutti i giocatori un'azione

Parametri

action	L'azione da inviare

2.3.3.2 _check_disconnected_players() `void Server::HangmanServer::_check_disconnected_players ( ) [protected]`

Permette di verificare se uno dei giocatori connessi si è disconnesso

2.3.3.3 _generate_short_phrase() `void Server::HangmanServer::_generate_short_phrase ( ) [protected]`

Permette di generare una nuova frase da indovinare

2.3.3.4 _get_letter_from_player() `int Server::HangmanServer::_get_letter_from_player ( Player * player, int timeout = 5 ) [protected]`

Permette di ricevere da un player un tentativo contenente una lettera

Parametri

player	Il player che deve fare il tentativo
timeout	Il tempo massimo da aspettare in secondi

Restituisce

Un numero che rappresenta il risultato della funzione

Valori di ritorno

1	Se il tentativo è andato a buon fine e ha indovinato
0	Se il tentativo è andato a buon fine e non ha indovinato
-1	Se la lettera è bloccata o se è già stata usata
-2	Se ha mandato un pacchetto sbagliato (in questo caso il client viene disconnesso)

2.3.3.5 _get_short_phrase_from_player() `int Server::HangmanServer::_get_short_phrase_from_player ( Player * player, int timeout = 10 ) [protected]`

Permette di ricevere da un player un tentativo sulla frase da indovinare

Parametri

player	Il player che deve fare il tentativo
timeout	Il tempo massimo da aspettare in secondi

Restituisce

Un numero che rappresenta il risultato della funzione

Valori di ritorno

1	Se il tentativo è andato a buon fine e ha indovinato
0	Se il tentativo è andato a buon fine e non ha indovinato
-2	Se ha mandato un pacchetto sbagliato (in questo caso il client viene disconnesso)

2.3.3.6 _is_short_phrase_guessed() `bool Server::HangmanServer::_is_short_phrase_guessed ( ) [inline], [protected]`

Permette di verificare se la parola o frase è stata indovinata

Restituisce

se la parola o frase è stata indovinata

Valori di ritorno

true	se la parola o la frase è stata indovinata
false	se la parola o la frase non è stata indovinata

2.3.3.7 _load_short_phrases() `void Server::HangmanServer::_load_short_phrases ( const string & filename = "data/data.txt" ) [protected]`

Carica le frasi da un file

Parametri

filename	il nome del file da cui caricare le frasi

Eccezioni

std::runtime_error	se il file non esiste

2.3.3.8 _next_turn() void Server::HangmanServer::_next_turn ( ) [protected]

Permette di passare il turno al giocatore successivo

2.3.3.9 _read() template<class TypeMessage >

```
bool Server::HangmanServer::_read (
            Player * player,
            TypeMessage & message,
            Client::Action action = Client::Action::GENERIC,
            int timeout = 5 ) [private]
```

Permette di leggere un messaggio da un certo giocatore

Parametri dei template

TypeMessage	Un tipo di messaggio di 128 bytes
TypeAction	L'azione che ci si aspetta di ricevere

Parametri

player	Il giocatore da cui aspettarsi il messaggio
message	Dove salvare il messaggio
timeout	Il tempo massimo di attesa per il messaggio (in secondi)

Restituisce

Se la lettura è andata a buon fine

2.3.3.10 _remove_player() void Server::HangmanServer::_remove_player (

```
            Player * player ) [protected]
```

Permette di eliminare un giocatore dalla lista dei giocatori

Parametri

player	Il Player da eliminare

2.3.3.11 _send() template<typename TypeMessage >

```
bool Server::HangmanServer::_send (
            Player * player,
            TypeMessage & message ) [private]
```

Permette di inviare un messaggio ad un certo giocatore

Parametri dei template

TypeMessage	Un tipo di messaggio di 128 bytes

Parametri

player	Il giocatore a cui inviare il messaggio
message	Il messaggio da inviare

Restituisce

true se l'invio è andato a buon fine, false altrimenti

Valori di ritorno

true	L'invio è andato a buon fine
false	L'invio non è andato a buon fine

Nota

Se l'invio fallisce, il giocatore viene disconnesso

2.3.3.12 _send_action() `bool Server::HangmanServer::_send_action ( Player * player, Server::Action action ) [inline], [protected]`

Permette di inviare un'azione ad un certo giocatore

Parametri

player	Il giocatore a cui inviare l'azione
action	L'azione da inviare

Restituisce

true se l'invio è andato a buon fine, false altrimenti

Valori di ritorno

true	L'invio è andato a buon fine
false	L'invio non è andato a buon fine

2.3.3.13 _send_update_attempts() void Server::HangmanServer::_send_update_attempts (Player & player) [inline], [protected]

Permette di inviare a un giocatore un aggiornamento sul numero di errori commessi

Parametri

player	Il giocatore a cui inviare l'aggiornamento

2.3.3.14 _send_update_players() void Server::HangmanServer::_send_update_players (Server::Player & player) [inline], [protected]

Permette di inviare a un giocatore un aggiornamento sulla lista dei giocatori

Parametri

player	Il giocatore a cui inviare l'aggiornamento

2.3.3.15 _send_update_short_phrase() void Server::HangmanServer::_send_update_short_phrase (Server::Player & player) [inline], [protected]

Permette di inviare a un giocatore un aggiornamento sullo stato del gioco

Parametri

player	Il giocatore a cui inviare l'aggiornamento

2.3.3.16 accept() void Server::HangmanServer::accept ( ) [protected]

Permette di verificare se ci sono nuovi client che vogliono connettersi e di accettarli

2.3.3.17 loop() void Server::HangmanServer::loop ( ) [protected]

Si occupa di gestire le connessioni e le richieste dei client, quindi di eseguire il gioco.

Loop del server

Nota

Deve trovarsi all'interno di un while loop

2.3.3.18 new_round() void Server::HangmanServer::new_round ( ) [protected]

Permette di avviare un nuovo round

2.3.3.19 run() void Server::HangmanServer::run (const bool *verbose = true*)

Permette di lasciare la gestione del server alla classe stessa, che si occuperà di avviare il server e gestire il loop di gioco.

Esegue tutte le funzioni del server

Parametri

verbose	Se deve stampare un resoconto dello stato del server ad ogni loop

2.3.3.20 start() void Server::HangmanServer::start (uint8_t *_max_errors = 10*, const string *_start_blocked_letters = "AEIOU"*, uint8_t *_blocked_attempts = 3*, const string & *_filename = "data/data.txt"*)

Avvia il server

Parametri

_max_errors	Il numero massimo di errori prima che la partita sia persa
_start_blocked_letters	Le lettere che non si possono indovinare all'inizio
_blocked_attempts	Il numero di tentativi che devono essere fatti prima di poter usare le lettere bloccate
_filename	Il nome del file da cui caricare le frasi

Eccezioni

std::runtime_error	Se il server non è stato avviato

2.4 Riferimenti per la struct Server::Player

Attributi pubblici

- int **sockfd**

  Socket del client.
- char **username** [USERNAME_LENGTH] {}

  Nome del client.

2.4.1 Descrizione dettagliata

Rappresenta il giocatore

Contiene il nome del giocatore e il descrittore del suo socket

Autore

John Toniutti

2.5 Riferimenti per la struct Client::HangmanClient::server_address

Struttura contenente le informazioni del server.

2.5.1 Descrizione dettagliata

Struttura contenente le informazioni del server.

2.6 Riferimenti per la struct TerminalSize

2.6.1 Descrizione dettagliata

Struttura che rappresenta le dimensioni del terminale

Indice analitico