miniLaska

Generato da Doxygen 1.8.18

Capitolo 1

Gioco miniLaska

1.1 Introduzione

Progetto di Introduzione alla Programmazione 2020/2021, gioco miniLaska

1.2 Descrizione del gioco

Gioco miniLaska con le regole definite nel sito `http://www.lasca.org/` e le seguenti variazioni:

1. Limite di una pedina mangiata per mossa
2. L'altezza massima delle torri è di 3, nel caso in cui ci siano pedine in eccesso verranno rimosse a partire dal basso

Capitolo 2

Indice dei tipi composti

2.1 Elenco dei tipi composti

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

Capitolo 3

Indice dei file

3.1 Elenco dei file

Questo è un elenco di tutti i file con una loro breve descrizione:

Capitolo 4

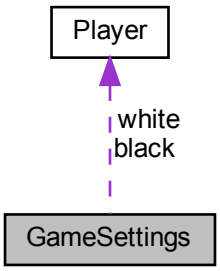
Documentazione delle classi

4.1 Riferimenti per la struct GameSettings

Impostazioni del gioco.

`#include <utility.h>`

Diagramma di collaborazione per GameSettings:



Attributi pubblici

- Player white

  Giocatore che gioca con le pedine bianche.
- Player black

  Giocatore che gioca con le pedine nere.
- bool helpAllowed

  Vero se gli aiuti sono ammessi, falso altrimenti.
- bool clearConsole

  Vero se l'utente vuole che il terminale venga pulito dopo ogni mossa.

4.1.1 Descrizione dettagliata

Impostazioni del gioco.

4.1.2 Documentazione dei membri dato

4.1.2.1 black

```
Player GameSettings::black
```

Giocatore che gioca con le pedine nere.

4.1.2.2 clearConsole

```
bool GameSettings::clearConsole
```

Vero se l'utente vuole che il terminale venga pulito dopo ogni mossa.

4.1.2.3 helpAllowed

```
bool GameSettings::helpAllowed
```

Vero se gli aiuti sono ammessi, falso altrimenti.

4.1.2.4 white

```
Player GameSettings::white
```

Giocatore che gioca con le pedine bianche.

La documentazione per questa struct è stata generata a partire dal seguente file:

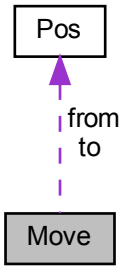
- src/utility.h

4.2 Riferimenti per la struct Move

Rappresenta una mossa.

`#include <utility.h>`

Diagramma di collaborazione per Move:



Attributi pubblici

- Pos from
- Pos to

4.2.1 Descrizione dettagliata

Rappresenta una mossa.

4.2.2 Documentazione dei membri dato

4.2.2.1 from

`Pos Move::from`

4.2.2.2 to

`Pos Move::to`

La documentazione per questa struct è stata generata a partire dal seguente file:

- src/utility.h

4.3 Riferimenti per la struct Piece

Rappresenta un pezzo sulla scacchiera.

`#include <utility.h>`

Attributi pubblici

- Color color [MAX_HEIGHT]

  Colore della peina.
- bool promoted

  Vero se la pedina è promossa, falso altrimenti.
- int height

  Altezza della pedina.

4.3.1 Descrizione dettagliata

Rappresenta un pezzo sulla scacchiera.

4.3.2 Documentazione dei membri dato

4.3.2.1 color

`Color Piece::color[MAX_HEIGHT]`

Colore della peina.

0 - colore in cima, 1 - colore in mezzo, 2 - colore in fondo. UNDEFINED se non c'è nessuna pedina

4.3.2.2 height

`int Piece::height`

Altezza della pedina.

4.3.2.3 promoted

`bool Piece::promoted`

Vero se la pedina è promossa, falso altrimenti.

La documentazione per questa struct è stata generata a partire dal seguente file:

- src/utility.h

4.4 Riferimenti per la struct Player

Rappresenta un giocatore.

`#include <utility.h>`

Attributi pubblici

- PlayerType type
  Tipo del giocatore.
- ComputerLevel level
  Livello del giocatore nel caso fosse di tipo computer.

4.4.1 Descrizione dettagliata

Rappresenta un giocatore.

4.4.2 Documentazione dei membri dato

4.4.2.1 level

`ComputerLevel Player::level`

Livello del giocatore nel caso fosse di tipo computer.

4.4.2.2 type

`PlayerType Player::type`

Tipo del giocatore.

La documentazione per questa struct è stata generata a partire dal seguente file:

- src/utility.h

4.5 Riferimenti per la struct Pos

Rappresenta una posizione nella scacchiera.

`#include <utility.h>`

Attributi pubblici

- int c
- int r

4.5.1 Descrizione dettagliata

Rappresenta una posizione nella scacchiera.

4.5.2 Documentazione dei membri dato

4.5.2.1 c

```
int Pos::c
```

4.5.2.2 r

```
int Pos::r
```

La documentazione per questa struct è stata generata a partire dal seguente file:

- src/utility.h

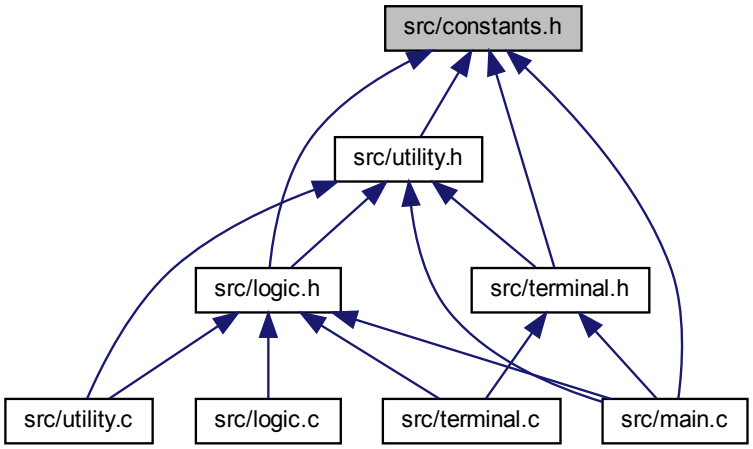
Capitolo 5

Documentazione dei file

5.1 Riferimenti per il file src/constants.h

Dichiarazione delle costanti utilizzate nel gioco.

Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:



Definizioni

- #define ROWS 7
- #define COLUMNS 7
- #define BOARD_SIZE 25
- #define MAX_HEIGHT 3
- #define EASY_DEPTH 2
- #define MEDIUM_DEPTH 4
- #define HARD_DEPTH 6
- #define max(a, b) ((a)>(b)?(a):(b))
- #define min(a, b) ((a)<(b)?(a):(b))
- #define bool int
- #define true 1
- #define false 0

5.1.1 Descrizione dettagliata

Dichiarazione delle costanti utilizzate nel gioco.

5.1.2 Documentazione delle definizioni

5.1.2.1 BOARD_SIZE

```
#define BOARD_SIZE 25
```

5.1.2.2 bool

```
#define bool int
```

5.1.2.3 COLUMNS

```
#define COLUMNS 7
```

5.1.2.4 EASY_DEPTH

```
#define EASY_DEPTH 2
```

5.1.2.5 false

```
#define false 0
```

5.1.2.6 HARD_DEPTH

```
#define HARD_DEPTH 6
```

5.1.2.7 max

```
#define max(
             a,
             b ) ((a)>(b)?(a):(b))
```

5.1.2.8 MAX_HEIGHT

```
#define MAX_HEIGHT 3
```

5.1.2.9 MEDIUM_DEPTH

```
#define MEDIUM_DEPTH 4
```

5.1.2.10 min

```
#define min(
             a,
             b ) ((a)<(b)?(a):(b))
```

5.1.2.11 ROWS

```
#define ROWS 7
```

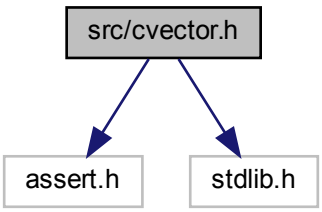
5.1.2.12 true

```
#define true 1
```

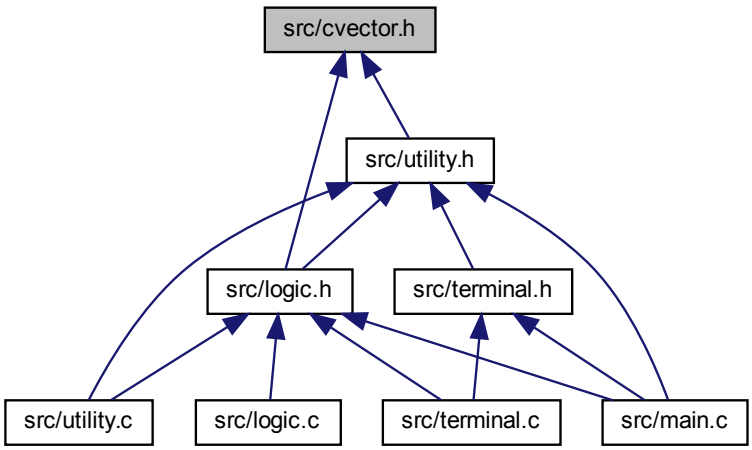
5.2 Riferimenti per il file src/cvector.h

```
#include <assert.h>
#include <stdlib.h>
```

Grafo delle dipendenze di inclusione per cvector.h:



Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:



Definizioni

- #define cvector_vector_type(type) type ∗

  cvector_vector_type - The vector type used in this library
- #define cvector_set_capacity(vec, size)

  cvector_set_capacity - For internal use, sets the capacity variable of the vector
- #define cvector_set_size(vec, size)

  cvector_set_size - For internal use, sets the size variable of the vector
- #define cvector_capacity(vec) ((vec) ? ((size_t ∗)(vec))[-1] : (size_t)0)

cvector_capacity - gets the current capacity of the vector

- #define cvector_size(vec) ((vec) ? ((size_t *)(vec))[-2] : (size_t)0)

  cvector_size - gets the current size of the vector
- #define cvector_empty(vec) (cvector_size(vec) == 0)

  cvector_empty - returns non-zero if the vector is empty
- #define cvector_grow(vec, count)

  cvector_grow - For internal use, ensures that the vector is at least <count> elements big
- #define cvector_pop_back(vec)

  cvector_pop_back - removes the last element from the vector
- #define cvector_erase(vec, i)

  cvector_erase - removes the element at index i from the vector
- #define cvector_free(vec)

  cvector_free - frees all memory associated with the vector
- #define cvector_begin(vec) (vec)

  cvector_begin - returns an iterator to first element of the vector
- #define cvector_end(vec) ((vec) ? &((vec)[cvector_size(vec)]) : NULL)

  cvector_end - returns an iterator to one past the last element of the vector
- #define cvector_push_back(vec, value)

  cvector_push_back - adds an element to the end of the vector
- #define cvector_copy(from, to)

  cvector_copy - copy a vector

5.2.1 Documentazione delle definizioni

5.2.1.1 cvector_begin

```
#define cvector_begin(
              vec ) (vec)
```

cvector_begin - returns an iterator to first element of the vector

Parametri

vec	- the vector

Restituisce

a pointer to the first element (or NULL)

5.2.1.2 cvector_capacity

```
#define cvector_capacity(
              vec ) ((vec) ?  ((size_t *)(vec))[-1] :  (size_t)0)
```

cvector_capacity - gets the current capacity of the vector

Parametri

vec	- the vector

Restituisce

the capacity as a size_t

5.2.1.3 cvector_copy

```
#define cvector_copy(
              from,
              to )
```

Valore:

```
    do {                                                    \
        size_t i;                                           \
        for(i = 0; i < cvector_size(from); i++) {           \
            cvector_push_back(to, from[i]);                 \
        }                                                   \
    } while (0)                                             \
```

cvector_copy - copy a vector

Parametri

from	- the original vector
to	- destination to which the function copy to

Restituisce

void

5.2.1.4 cvector_empty

```
#define cvector_empty(
              vec ) (cvector_size(vec) == 0)
```

cvector_empty - returns non-zero if the vector is empty

Parametri

vec	- the vector

Restituisce

non-zero if empty, zero if non-empty

5.2.1.5 cvector_end

```
#define cvector_end(
              vec ) ((vec) ?  &((vec)[cvector_size(vec)]) :  NULL)
```

cvector_end - returns an iterator to one past the last element of the vector

Parametri

vec	- the vector

Restituisce

a pointer to one past the last element (or NULL)

5.2.1.6 cvector_erase

```
#define cvector_erase(
              vec,
              i )
```

Valore:

```
    do {                                                        \
        if (vec) {                                              \
            const size_t cv_sz = cvector_size(vec);             \
            if ((i) < cv_sz) {                                  \
                cvector_set_size((vec), cv_sz - 1);             \
                size_t cv_x;                                    \
                for (cv_x = (i); cv_x < (cv_sz - 1); ++cv_x) {  \
                    (vec)[cv_x] = (vec)[cv_x + 1];              \
                }                                               \
            }                                                   \
        }                                                       \
    } while (0)
```

cvector_erase - removes the element at index i from the vector

Parametri

vec	- the vector
i	- index of element to remove

Restituisce

void

5.2.1.7 cvector_free

```
#define cvector_free(
              vec )
```

Valore:

```
    do {                                          \
        if (vec) {                                \
            size_t *p1 = &((size_t *)(vec))[-2]; \
            free(p1);                             \
        }                                         \
    } while (0)
```

cvector_free - frees all memory associated with the vector

Parametri

vec	- the vector

Restituisce

void

5.2.1.8 cvector_grow

```
#define cvector_grow(
              vec,
              count )
```

Valore:

```
    do {                                                                    \
        const size_t cv_sz = (count) * sizeof(*(vec)) + (sizeof(size_t) * 2); \
        if (!(vec)) {                                                       \
            size_t *cv_p = malloc(cv_sz);                                   \
            assert(cv_p);                                                   \
            (vec) = (void *)(&cv_p[2]);                                     \
            cvector_set_capacity((vec), (count));                           \
            cvector_set_size((vec), 0);                                     \
        } else {                                                            \
            size_t *cv_p1 = &((size_t *)(vec))[-2];                         \
            size_t *cv_p2 = realloc(cv_p1, (cv_sz));                        \
            assert(cv_p2);                                                  \
            (vec) = (void *)(&cv_p2[2]);                                    \
            cvector_set_capacity((vec), (count));                           \
        }                                                                   \
    } while (0)
```

cvector_grow - For internal use, ensures that the vector is at least <count> elements big

Parametri

vec	- the vector
count	- the new capacity to set

Restituisce

void

5.2.1.9 cvector_pop_back

```
#define cvector_pop_back(
              vec )
```

Valore:

```
    do {                                                \
        cvector_set_size((vec), cvector_size(vec) - 1); \
    } while (0)
```

cvector_pop_back - removes the last element from the vector

Parametri

vec	- the vector

Restituisce

void

5.2.1.10 cvector_push_back

```
#define cvector_push_back(
              vec,
              value )
```

Valore:

```
    do {                                                \
        size_t cv_cap = cvector_capacity(vec);          \
        if (cv_cap <= cvector_size(vec)) {              \
            cvector_grow((vec), cv_cap + 1);            \
        }                                               \
        vec[cvector_size(vec)] = (value);               \
        cvector_set_size((vec), cvector_size(vec) + 1); \
    } while (0)
```

cvector_push_back - adds an element to the end of the vector

Parametri

vec	- the vector
value	- the value to add

Restituisce

void

5.2.1.11 cvector_set_capacity

```
#define cvector_set_capacity(
              vec,
              size )
```

Valore:

```
    do {                                        \
        if (vec) {                              \
            ((size_t *)(vec))[-1] = (size);     \
        }                                       \
    } while (0)
```

cvector_set_capacity - For internal use, sets the capacity variable of the vector

Parametri

vec	- the vector
size	- the new capacity to set

Restituisce

void

5.2.1.12 cvector_set_size

```
#define cvector_set_size(
                vec,
                size )
```

Valore:

```
    do {                                        \
        if (vec) {                              \
            ((size_t *)(vec))[-2] = (size);     \
        }                                       \
    } while (0)
```

cvector_set_size - For internal use, sets the size variable of the vector

Parametri

vec	- the vector
size	- the new capacity to set

Restituisce

void

5.2.1.13 cvector_size

```
#define cvector_size(
                vec ) ((vec) ?  ((size_t *)(vec))[-2] :  (size_t)0)
```

cvector_size - gets the current size of the vector

Parametri

vec	- the vector

Restituisce

the size as a size_t

5.2.1.14 cvector_vector_type

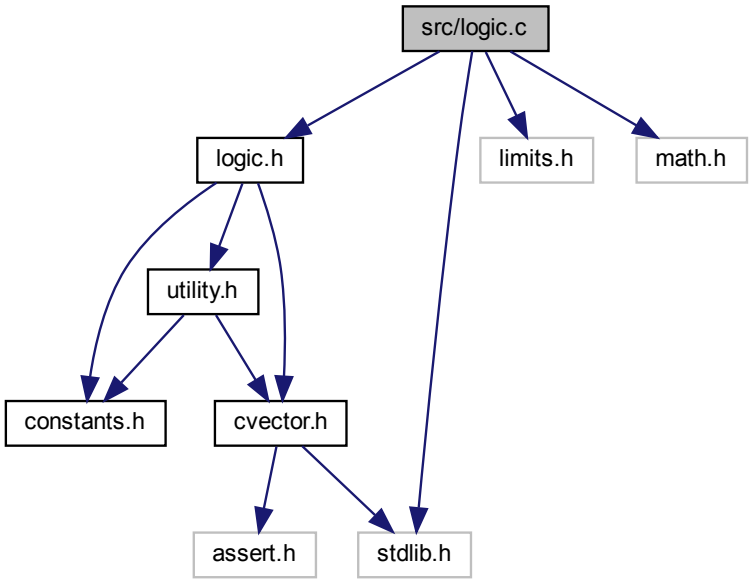
```
#define cvector_vector_type(
              type ) type *
```

cvector_vector_type - The vector type used in this library

5.3 Riferimenti per il file src/logic.c

```
#include "logic.h"
#include <stdlib.h>
#include <limits.h>
#include <math.h>
```

Grafo delle dipendenze di inclusione per logic.c:



Funzioni

- int get_index_from_coordinates (int c, int r)

  Calcola la posizione di una cella all'interno della matrice.
- int get_index_from_pos (Pos pos)

  Calcola la posizione di una cella all'interno della matrice.
- Pos get_pos_from_index (int index)

  Calcola le coordinate della cella dato l'indice.
- void initialize_board (Board board)

  Inizializza la schacchiera allo stato di inizio gioco.
- GameState compute_state (Board board, Color colorToMove)

  Calcola lo stato della scacchiera.
- int calculate_piece_distance (Pos pos1, Pos pos2)
- int compute_score (Board board, Color cpuPlayer, Color colorToMove)
- int minimax (Board board, Color cpuColor, int alpha, int beta, int depth, bool maximize)
- Move best_move_minimax (Board board, Color cpuColor, int depth)

  Calcola la mossa migliore utilizzando l'algoritmo mini-max.
- bool apply_move (Board board, Color colorToMove, Move move)

  Applica la mossa sulla scacchiera se la mossa è valida.
- bool is_move_valid (Board board, Move move)

  Controlla se una mossa è valida.
- bool is_pos_valid (Pos pos)

  Controlla se una posizione è valida, se è all'interno della scacchiera ed è una cella utilizzata nel gioco Lasca.
- bool does_move_eat (Board board, Move move)

  Controlla se una mossa prevede di mangiare una pedina dell'avversario.
- cvector_vector_type (Move)

  Restituisce un array con tutte le mosse che un determinato pezzo può effettuare.

5.3.1 Documentazione delle funzioni

5.3.1.1 apply_move()

```
bool apply_move (
            Board board,
            Color colorToMove,
            Move move )
```

Applica la mossa sulla scacchiera se la mossa è valida.

Parametri

board	scacchiera sulla quale applicare la mossa
colorToMove	colore del giocatore che deve muovere
move	mossa da applicare

Restituisce

bool vero se la mossa è valida, falso altrimenti

5.3.1.2 best_move_minimax()

```
Move best_move_minimax (
            Board board,
            Color cpuColor,
            int depth )
```

Calcola la mossa migliore utilizzando l'algoritmo mini-max.

Parametri

board	scacchiara su cui calcolare la mossa
cpuColor	colore del giocatore che deve muovere
depth	profondità del calcolo della mossa

Restituisce

Move mossa migliore che può venire giocata

5.3.1.3 calculate_piece_distance()

```
int calculate_piece_distance (
            Pos pos1,
            Pos pos2 )
```

5.3.1.4 compute_score()

```
int compute_score (
            Board board,
            Color cpuPlayer,
            Color colorToMove )
```

5.3.1.5 compute_state()

```
GameState compute_state (
            Board board,
            Color colorToMove )
```

Calcola lo stato della scacchiera.

Parametri

board	scacchiera di cui calcolare lo stato
colorToMove	colore del giocatore che deve muovere

Restituisce

GameState stato della scacchiera

5.3.1.6 cvector_vector_type()

```
cvector_vector_type (
            Move  )
```

Restituisce un array con tutte le mosse che un determinato pezzo può effettuare.

Parametri

board	scacchiera sulla quale applicare la mossa
piecePos	posizione del pezzo di cui calcolare le mosse possibili

Restituisce

Move∗, array di mosse che il pezzo può eseguire

5.3.1.7 does_move_eat()

```
bool does_move_eat (
            Board board,
            Move move )
```

Controlla se una mossa prevede di mangiare una pedina dell'avversario.

Parametri

board	scacchiera sulla quale applicare la mossa
move	mossa da controllare

Restituisce

bool vero se la mossa prevede di mangiare una pedina dell'avversario, falso altrimenti

5.3.1.8 get_index_from_coordinates()

```
int get_index_from_coordinates (
            int c,
            int r )
```

Calcola la posizione di una cella all'interno della matrice.

Parametri

c	indice della colonna
r	indice della riga

Restituisce

int indice della matrice

5.3.1.9 get_index_from_pos()

```
int get_index_from_pos (
            Pos pos )
```

Calcola la posizione di una cella all'interno della matrice.

Parametri

pos	posizione della scacchiera

Restituisce

int indice della matrice

5.3.1.10 get_pos_from_index()

```
Pos get_pos_from_index (
            int index )
```

Calcola le coordinate della cella dato l'indice.

Parametri

index	indice della cella

Restituisce

Pos coordinate della cella

5.3.1.11 initialize_board()

```
void initialize_board (
            Board board )
```

Inizializza la schacchiera allo stato di inizio gioco.

Parametri

board	schacchiera da inizializzare

5.3.1.12 is_move_valid()

```
bool is_move_valid (
            Board board,
            Move move )
```

Controlla se una mossa è valida.

Parametri

board	scacchiera sulla quale applicare la mossa
move	mossa da controllare

Restituisce

bool vero se la mossa è valida, falso altrimenti

5.3.1.13 is_pos_valid()

```
bool is_pos_valid (
            Pos pos )
```

Controlla se una posizione è valida, se è all'interno della scacchiera ed è una cella utilizzata nel gioco Lasca.

Parametri

pos	posizione dellla cella

Restituisce

bool vero se la cella è valida, falso altrimenti

5.3.1.14 minimax()

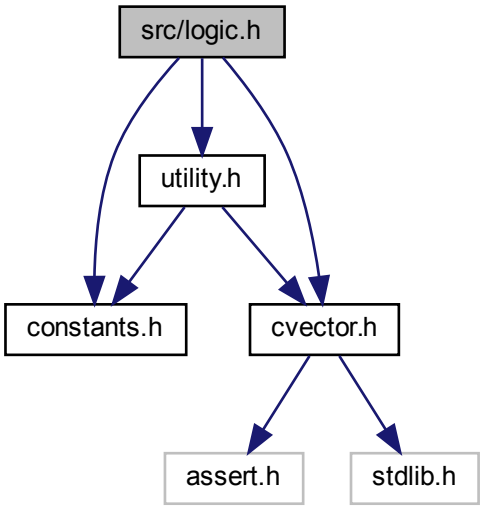
```
int minimax (
            Board board,
            Color cpuColor,
            int alpha,
            int beta,
            int depth,
            bool maximize )
```

5.4 Riferimenti per il file src/logic.h

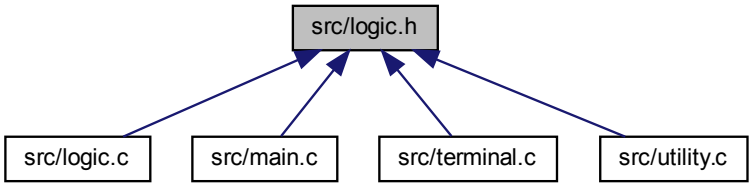
Gestione della logica del gioco miniLaska.

```
#include "constants.h"
#include "cvector.h"
#include "utility.h"
```

Grafo delle dipendenze di inclusione per logic.h:



Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:



Funzioni

- int get_index_from_coordinates (int c, int r)

  Calcola la posizione di una cella all'interno della matrice.
- int get_index_from_pos (Pos pos)

  Calcola la posizione di una cella all'interno della matrice.
- Pos get_pos_from_index (int index)

  Calcola le coordinate della cella dato l'indice.
- void initialize_board (Board board)

  Inizializza la schacchiera allo stato di inizio gioco.
- GameState compute_state (Board board, Color colorToMove)

  Calcola lo stato della scacchiera.
- Move best_move_minimax (Board board, Color cpuColor, int depth)

  Calcola la mossa migliore utilizzando l'algoritmo mini-max.
- bool apply_move (Board board, Color colorToMove, Move move)

  Applica la mossa sulla scacchiera se la mossa è valida.
- bool is_move_valid (Board board, Move move)

  Controlla se una mossa è valida.
- bool is_pos_valid (Pos pos)

  Controlla se una posizione è valida, se è all'interno della scacchiera ed è una cella utilizzata nel gioco Lasca.
- bool does_move_eat (Board board, Move move)

  Controlla se una mossa prevede di mangiare una pedina dell'avversario.
- cvector_vector_type (Move) get_possible_moves_by_color(Board board

  Restituisce un array con tutte le mosse che un determinato colore può effettuare.

Variabili

- Color color
- Pos piecePos

5.4.1 Descrizione dettagliata

Gestione della logica del gioco miniLaska.

5.4.2 Documentazione delle funzioni

5.4.2.1 apply_move()

```
bool apply_move (
            Board board,
            Color colorToMove,
            Move move )
```

Applica la mossa sulla scacchiera se la mossa è valida.

Parametri

board	scacchiera sulla quale applicare la mossa
colorToMove	colore del giocatore che deve muovere
move	mossa da applicare

Restituisce

bool vero se la mossa è valida, falso altrimenti

5.4.2.2 best_move_minimax()

```
Move best_move_minimax (
            Board board,
            Color cpuColor,
            int depth )
```

Calcola la mossa migliore utilizzando l'algoritmo mini-max.

Parametri

board	scacchiara su cui calcolare la mossa
cpuColor	colore del giocatore che deve muovere
depth	profondità del calcolo della mossa

Restituisce

Move mossa migliore che può venire giocata

5.4.2.3 compute_state()

```
GameState compute_state (
            Board board,
            Color colorToMove )
```

Calcola lo stato della scacchiera.

Parametri

board	scacchiera di cui calcolare lo stato
colorToMove	colore del giocatore che deve muovere

Restituisce

GameState stato della scacchiera

5.4.2.4 cvector_vector_type()

```
cvector_vector_type (
            Move  )
```

Restituisce un array con tutte le mosse che un determinato colore può effettuare.

Restituisce un array con tutte le mosse che un determinato pezzo può effettuare.

Parametri

board	scacchiera sulla quale applicare la mossa
color	colore del giocatore di cui calcolare le mosse possibili

Restituisce

Move∗, array di mosse che il colore può eseguire

Parametri

board	scacchiera sulla quale applicare la mossa
piecePos	posizione del pezzo di cui calcolare le mosse possibili

Restituisce

Move∗, array di mosse che il pezzo può eseguire

5.4.2.5 does_move_eat()

```
bool does_move_eat (
            Board board,
            Move move )
```

Controlla se una mossa prevede di mangiare una pedina dell'avversario.

Parametri

board	scacchiera sulla quale applicare la mossa
move	mossa da controllare

Restituisce

bool vero se la mossa prevede di mangiare una pedina dell'avversario, falso altrimenti

5.4.2.6 get_index_from_coordinates()

```
int get_index_from_coordinates (
            int c,
            int r )
```

Calcola la posizione di una cella all'interno della matrice.

Parametri

c	indice della colonna
r	indice della riga

Restituisce

int indice della matrice

5.4.2.7 get_index_from_pos()

```
int get_index_from_pos (
            Pos pos )
```

Calcola la posizione di una cella all'interno della matrice.

Parametri

pos	posizione della scacchiera

Restituisce

int indice della matrice

5.4.2.8 get_pos_from_index()

```
Pos get_pos_from_index (
            int index )
```

Calcola le coordinate della cella dato l'indice.

Parametri

index	indice della cella

Restituisce

Pos coordinate della cella

5.4.2.9 initialize_board()

```
void initialize_board (
            Board board )
```

Inizializza la schacchiera allo stato di inizio gioco.

Parametri

board	schacchiera da inizializzare

5.4.2.10 is_move_valid()

```
bool is_move_valid (
            Board board,
            Move move )
```

Controlla se una mossa è valida.

Parametri

board	scacchiera sulla quale applicare la mossa
move	mossa da controllare

Restituisce

bool vero se la mossa è valida, falso altrimenti

5.4.2.11 is_pos_valid()

```
bool is_pos_valid (
            Pos pos )
```

Controlla se una posizione è valida, se è all'interno della scacchiera ed è una cella utilizzata nel gioco Lasca.

Parametri

pos	posizione dellla cella

Restituisce

bool vero se la cella è valida, falso altrimenti

5.4.3 Documentazione delle variabili

5.4.3.1 color

```
Color color
```

5.4.3.2 piecePos

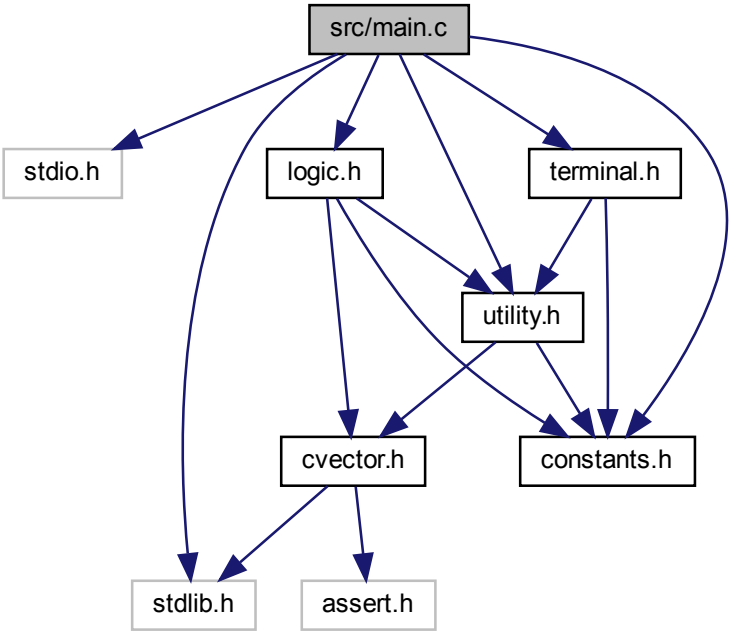
```
Pos piecePos
```

5.5 Riferimenti per il file src/main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "constants.h"
#include "utility.h"
#include "terminal.h"
```

```
#include "logic.h"
```

Grafo delle dipendenze di inclusione per main.c:



Funzioni

- int main ()

5.5.1 Documentazione delle funzioni

5.5.1.1 main()

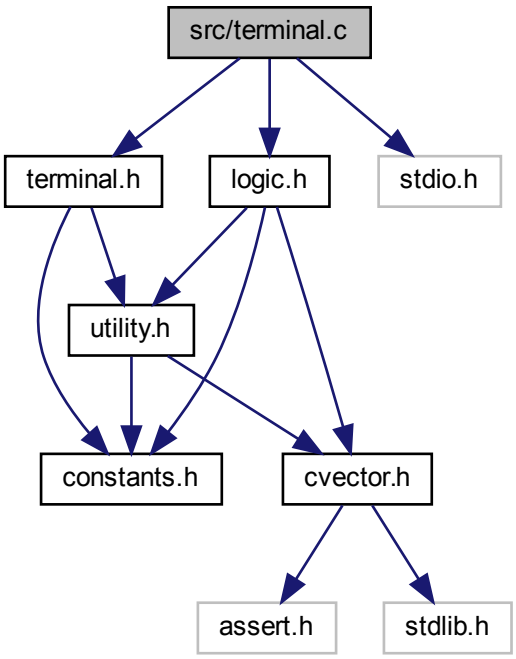
```
int main ( )
```

5.6 Riferimenti per il file src/terminal.c

```
#include "terminal.h"
#include "logic.h"
```

```
#include <stdio.h>
```

Grafo delle dipendenze di inclusione per terminal.c:



Funzioni

- void display_board (Board board)

  Stampa la scacchiera sul terminale.
- void display_last_move (Move move)

  Stampa l'ultima mossa effettuata sul terminale.
- void display_player_to_move (int turn, Color playerToMove)

  Stampa il numero del turno e il colore del giocatore che deve giocare sul terminale.
- GameSettings read_game_settings ()

  Chiede le impostazioni del gioco all'utente tramite terminale.
- Move read_player_move (Board board, Color color, GameSettings settings)

  Legge la mossa dell'utente da terminale.
- void display_winner (GameState state)

  Stampa il vincitore sul terminale.
- bool does_user_want_new_game ()

  Chiede all'utente se vuole giocare una nuova partita.

5.6.1 Documentazione delle funzioni

5.6.1.1 display_board()

```
void display_board (
            Board board )
```

Stampa la scacchiera sul terminale.

Parametri

board	scacchiera da venire visualizzata

5.6.1.2 display_last_move()

```
void display_last_move (
            Move move )
```

Stampa l'ultima mossa effettuata sul terminale.

Parametri

move	mossa da stampare

5.6.1.3 display_player_to_move()

```
void display_player_to_move (
            int turn,
            Color playerToMove )
```

Stampa il numero del turno e il colore del giocatore che deve giocare sul terminale.

Parametri

turn	numero del turno
playerToMove	colore del giocatore

5.6.1.4 display_winner()

```
void display_winner (
            GameState state )
```

Stampa il vincitore sul terminale.

Parametri

state	stato della scacchiera

5.6.1.5 does_user_want_new_game()

```
bool does_user_want_new_game ( )
```

Chiede all'utente se vuole giocare una nuova partita.

Restituisce

bool true se l'utente vuole giocare una nuova partita, falso altrimenti

5.6.1.6 read_game_settings()

```
GameSettings read_game_settings ( )
```

Chiede le impostazioni del gioco all'utente tramite terminale.

Restituisce

GameSettings impostazioni

5.6.1.7 read_player_move()

```
Move read_player_move (
            Board board,
            Color color,
            GameSettings settings )
```

Legge la mossa dell'utente da terminale.

Parametri

board	scacchiera sulla quale deve venire effettuata la mossa
color	colore del giocatore che deve effettuare la mossa
settings	impostazioni del gioco

Restituisce

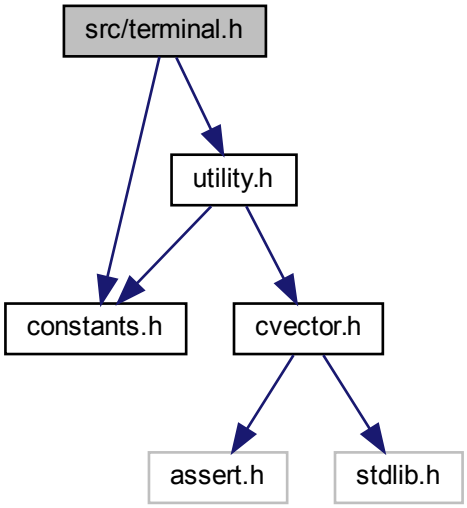
Move Mossa inserita dall'utente

5.7 Riferimenti per il file src/terminal.h

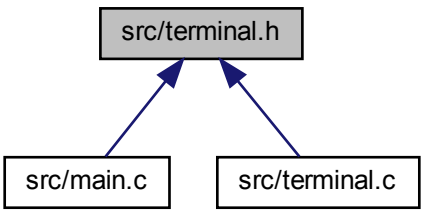
Gestione dell'interfaccia grafica del gioco miniLaska attraverso il terminale.

```
#include "constants.h"
#include "utility.h"
```

Grafo delle dipendenze di inclusione per terminal.h:



Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:



Funzioni

- void display_board (Board board)

  Stampa la scacchiera sul terminale.
- void display_last_move (Move move)

  Stampa l'ultima mossa effettuata sul terminale.
- void display_player_to_move (int turn, Color playerToMove)

  Stampa il numero del turno e il colore del giocatore che deve giocare sul terminale.
- GameSettings read_game_settings ()

  Chiede le impostazioni del gioco all'utente tramite terminale.
- Move read_player_move (Board board, Color color, GameSettings settings)

  Legge la mossa dell'utente da terminale.
- void display_winner (GameState state)

  Stampa il vincitore sul terminale.
- bool does_user_want_new_game ()

  Chiede all'utente se vuole giocare una nuova partita.

5.7.1 Descrizione dettagliata

Gestione dell'interfaccia grafica del gioco miniLaska attraverso il terminale.

5.7.2 Documentazione delle funzioni

5.7.2.1 display_board()

```
void display_board (
            Board board )
```

Stampa la scacchiera sul terminale.

Parametri

board	scacchiera da venire visualizzata

5.7.2.2 display_last_move()

```
void display_last_move (
            Move move )
```

Stampa l'ultima mossa effettuata sul terminale.

Parametri

move	mossa da stampare

5.7.2.3 display_player_to_move()

```
void display_player_to_move (
            int turn,
            Color playerToMove )
```

Stampa il numero del turno e il colore del giocatore che deve giocare sul terminale.

Parametri

turn	numero del turno
playerToMove	colore del giocatore

5.7.2.4 display_winner()

```
void display_winner (
            GameState state )
```

Stampa il vincitore sul terminale.

Parametri

state	stato della scacchiera

5.7.2.5 does_user_want_new_game()

```
bool does_user_want_new_game ( )
```

Chiede all'utente se vuole giocare una nuova partita.

Restituisce

bool true se l'utente vuole giocare una nuova partita, falso altrimenti

5.7.2.6 read_game_settings()

```
GameSettings read_game_settings ( )
```

Chiede le impostazioni del gioco all'utente tramite terminale.

Restituisce

GameSettings impostazioni

5.7.2.7 read_player_move()

```
Move read_player_move (
            Board board,
            Color color,
            GameSettings settings )
```

Legge la mossa dell'utente da terminale.

Parametri

board	scacchiera sulla quale deve venire effettuata la mossa
color	colore del giocatore che deve effettuare la mossa
settings	impostazioni del gioco

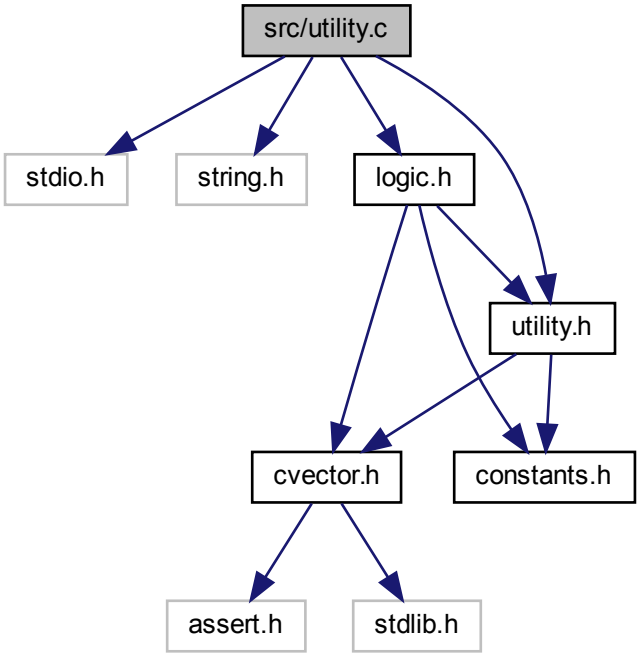
Restituisce

Move Mossa inserita dall'utente

5.8 Riferimenti per il file src/utility.c

```
#include <stdio.h>
#include <string.h>
#include "utility.h"
#include "logic.h"
```

Grafo delle dipendenze di inclusione per utility.c:



Funzioni

- bool is_piece_null (Piece piece)

  Controlla se un pezzo è nullo, ovvero se non c'è nessun pezzo.
- bool is_move_equal (Move a, Move b)

  Compara due Move e restituisce vero se sono uguali.
- bool is_pos_equal (Pos a, Pos b)

  Compara due Pos e restituisce vero se sono uguali.
- bool is_opposite_color (Color a, Color b)

  Compara due Color e restituisce vero se sono opposti, es.
- bool is_piece_equal (Piece a, Piece b)

  Compara due Piece e resituisce vero se sono uguali.
- Pos initialize_pos (int c, int r)

  Inizializza una struct Pos con la colonna e riga.
- Move initialize_move (Pos from, Pos to)

  Inizializza una struct Move.
- Piece initialize_piece (Color color0, Color color1, Color color2, bool promoted, int height)

  Inizializza una struct Piece.
- Piece initialize_null_piece ()

  Inizializza un Piece vuoto.
- Color get_opposite_color (Color color)

  Dato un colore di un giocatore restituisce l'opposto.
- Board clone_board (Board board)

Esegue una deep-copy della scacchiera.

- cvector_vector_type (Pos)
- int count_pieces (Board board, Color color)

  Conta i pezzi che ci sono nella scacchiera di un determinato colore.

5.8.1 Documentazione delle funzioni

5.8.1.1 clone_board()

```
Board clone_board (
            Board board )
```

Esegue una deep-copy della scacchiera.

Parametri

board	scacchiera da copiare

Restituisce

Piece* - copia della scacchiera

5.8.1.2 count_pieces()

```
int count_pieces (
            Board board,
            Color color )
```

Conta i pezzi che ci sono nella scacchiera di un determinato colore.

Parametri

board	scacchiera sulla quale contare i pezzi
color	colore dei pezzi da contare

Restituisce

int, numero di pezzi del colore dato

5.8.1.3 cvector_vector_type()

```
cvector_vector_type (
            Pos  )
```

5.8.1.4 get_opposite_color()

```
Color get_opposite_color (
            Color color )
```

Dato un colore di un giocatore restituisce l'opposto.

Parametri

color	colore del quale si vuole l'opposto

Restituisce

Color colore opposto

5.8.1.5 initialize_move()

```
Move initialize_move (
            Pos from,
            Pos to )
```

Inizializza una struct Move.

Parametri

from	posizione di inizio della mossa
to	posizione di fine della mossa

Restituisce

Move struct Move inizializzata

5.8.1.6 initialize_null_piece()

```
Piece initialize_null_piece ( )
```

Inizializza un Piece vuoto.

Restituisce

Piece struct Piece vuota

5.8.1.7 initialize_piece()

```
Piece initialize_piece (
            Color color0,
            Color color1,
            Color color2,
            bool promoted,
            int height )
```

Inizializza una struct Piece.

Parametri

color0	colore del pezzo in cima alla pedina
color1	colore del pezzo in mezzo alla pedina
color2	colore del pezzo alla base della pedina
promoted	vero se la pedina è promossa, falso altrimenti
height	altezza della pedina

Restituisce

Piece struct Piece inizializzato

5.8.1.8 initialize_pos()

```
Pos initialize_pos (
            int c,
            int r )
```

Inizializza una struct Pos con la colonna e riga.

Parametri

c	colonna
r	riga

Restituisce

Pos struct Pos inizializzata

5.8.1.9 is_move_equal()

```
bool is_move_equal (
            Move a,
            Move b )
```

Compara due Move e restituisce vero se sono uguali.

Parametri

a	prima Move
b	seconda Move

Restituisce

bool vero se le due Move sono uguali, falso altrimenti

5.8.1.10 is_opposite_color()

```
bool is_opposite_color (
            Color a,
            Color b )
```

Compara due Color e restituisce vero se sono opposti, es.

Bianco e Nero

Parametri

a	primo Color
b	secondo Color

Restituisce

bool vero se i due Color sono opposti, falso altrimenti

5.8.1.11 is_piece_equal()

```
bool is_piece_equal (
            Piece a,
            Piece b )
```

Compara due Piece e resituisce vero se sono uguali.

Parametri

a	primo Piece
b	secondo Piece

Restituisce

bool vero se i due Piece sono uguali, falso altrimenti

5.8.1.12 is_piece_null()

```
bool is_piece_null (
            Piece piece )
```

Controlla se un pezzo è nullo, ovvero se non c'è nessun pezzo.

Parametri

piece	pezzo da controllare

Restituisce

bool vero se il pezzo è nullo, falso altrimenti

5.8.1.13 is_pos_equal()

```
bool is_pos_equal (
            Pos a,
            Pos b )
```

Compara due Pos e restituisce vero se sono uguali.

Parametri

a	prima Pos
b	seconda Pos

Restituisce

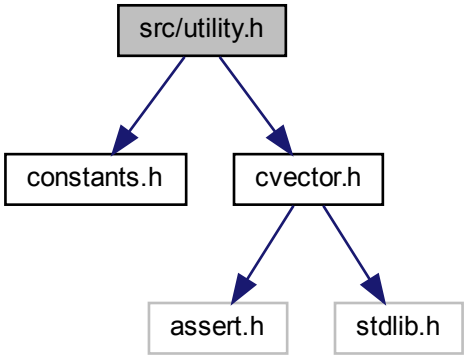
bool vero se le due Pos sono uguali, falso altrimenti

5.9 Riferimenti per il file src/utility.h

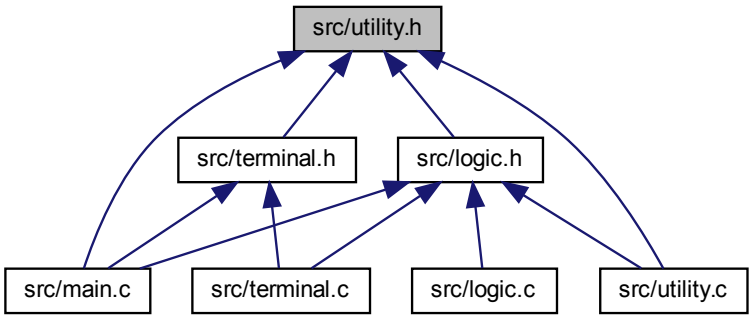
Dichiarazione enum, struts e funzioni di supporto.

```
#include "constants.h"
#include "cvector.h"
```

Grafo delle dipendenze di inclusione per utility.h:



Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:



Composti

- struct Pos

  Rappresenta una posizione nella scacchiera.
- struct Move

  Rappresenta una mossa.
- struct Piece

  Rappresenta un pezzo sulla scacchiera.
- struct Player

  Rappresenta un giocatore.
- struct GameSettings

  Impostazioni del gioco.

Definizioni

- #define Board Piece*

Tipi enumerati (enum)

- enum Color { UNDEFINED = 0, WHITE = 1, BLACK = 2 }

  Rappresenta il colore di un giocatore.
- enum PlayerType { HUMAN = 0, COMPUTER = 1 }

  Rappresenta il tipo di giocatore, umano o computer.
- enum GameState { PLAYING = 0, WHITE_WIN = 1, BLACK_WIN = 2 }

  Stato del gioco.
- enum ComputerLevel { EASY = 0, MEDIUM = 1, HARD = 2 }

  Difficoltà del giocatore Computer.

Funzioni

- bool is_piece_null (Piece piece)

  Controlla se un pezzo è nullo, ovvero se non c'è nessun pezzo.
- bool is_move_equal (Move a, Move b)

  Compara due Move e restituisce vero se sono uguali.
- bool is_pos_equal (Pos a, Pos b)

  Compara due Pos e restituisce vero se sono uguali.
- bool is_opposite_color (Color a, Color b)

  Compara due Color e restituisce vero se sono opposti, es.
- bool is_piece_equal (Piece a, Piece b)

  Compara due Piece e resituisce vero se sono uguali.
- Pos initialize_pos (int c, int r)

  Inizializza una struct Pos con la colonna e riga.
- Move initialize_move (Pos from, Pos to)

  Inizializza una struct Move.
- Piece initialize_piece (Color color0, Color color1, Color color2, bool promoted, int height)

  Inizializza una struct Piece.
- Piece initialize_null_piece ()

  Inizializza un Piece vuoto.
- Color get_opposite_color (Color color)

  Dato un colore di un giocatore restituisce l'opposto.
- Board clone_board (Board board)

  Esegue una deep-copy della scacchiera.
- cvector_vector_type (Pos) get_pieces_pos_by_color(Board board

  Restituisce un array con tutte le posizioni delle pedine di un determinato colore.
- int count_pieces (Board board, Color color)

  Conta i pezzi che ci sono nella scacchiera di un determinato colore.

Variabili

- Color color

5.9.1 Descrizione dettagliata

Dichiarazione enum, struts e funzioni di supporto.

5.9.2 Documentazione delle definizioni

5.9.2.1 Board

```
#define Board Piece*
```

5.9.3 Documentazione dei tipi enumerati

5.9.3.1 Color

```
enum Color
```

Rappresenta il colore di un giocatore.

Valori del tipo enumerato

UNDEFINED	
WHITE	
BLACK	

5.9.3.2 ComputerLevel

```
enum ComputerLevel
```

Difficoltà del giocatore Computer.

Valori del tipo enumerato

EASY	
MEDIUM	
HARD	

5.9.3.3 GameState

```
enum GameState
```

Stato del gioco.

Valori del tipo enumerato

PLAYING	
WHITE_WIN	
BLACK_WIN	

5.9.3.4 PlayerType

```
enum PlayerType
```

Rappresenta il tipo di giocatore, umano o computer.

Valori del tipo enumerato

HUMAN	
COMPUTER	

5.9.4 Documentazione delle funzioni

5.9.4.1 clone_board()

```
Board clone_board (
            Board board )
```

Esegue una deep-copy della scacchiera.

Parametri

board	scacchiera da copiare

Restituisce

Piece* - copia della scacchiera

5.9.4.2 count_pieces()

```
int count_pieces (
            Board board,
            Color color )
```

Conta i pezzi che ci sono nella scacchiera di un determinato colore.

Parametri

board	scacchiera sulla quale contare i pezzi
color	colore dei pezzi da contare

Restituisce

int, numero di pezzi del colore dato

5.9.4.3 cvector_vector_type()

```
cvector_vector_type (
            Pos  )
```

Restituisce un array con tutte le posizioni delle pedine di un determinato colore.

Parametri

board	scacchiera sulla quale cercare le pedine
color	colore del giocatore di cui cercare le pedine

Restituisce

Pos*, array di posizioni

5.9.4.4 get_opposite_color()

```
Color get_opposite_color (
            Color color )
```

Dato un colore di un giocatore restituisce l'opposto.

Parametri

color	colore del quale si vuole l'opposto

Restituisce

Color colore opposto

5.9.4.5 initialize_move()

```
Move initialize_move (
            Pos from,
            Pos to )
```

Inizializza una struct Move.

Parametri

from	posizione di inizio della mossa
to	posizione di fine della mossa

Restituisce

Move struct Move inizializzata

5.9.4.6 initialize_null_piece()

```
Piece initialize_null_piece ( )
```

Inizializza un Piece vuoto.

Restituisce

Piece struct Piece vuota

5.9.4.7 initialize_piece()

```
Piece initialize_piece (
            Color color0,
            Color color1,
            Color color2,
            bool promoted,
            int height )
```

Inizializza una struct Piece.

Parametri

color0	colore del pezzo in cima alla pedina
color1	colore del pezzo in mezzo alla pedina
color2	colore del pezzo alla base della pedina
promoted	vero se la pedina è promossa, falso altrimenti
height	altezza della pedina

Restituisce

Piece struct Piece inizializzato

5.9.4.8 initialize_pos()

```
Pos initialize_pos (
            int c,
            int r )
```

Inizializza una struct Pos con la colonna e riga.

Parametri

c	colonna
r	riga

Restituisce

Pos struct Pos inizializzata

5.9.4.9 is_move_equal()

```
bool is_move_equal (
            Move a,
            Move b )
```

Compara due Move e restituisce vero se sono uguali.

Parametri

a	prima Move
b	seconda Move

Restituisce

bool vero se le due Move sono uguali, falso altrimenti

5.9.4.10 is_opposite_color()

```
bool is_opposite_color (
            Color a,
            Color b )
```

Compara due Color e restituisce vero se sono opposti, es.

Bianco e Nero

Parametri

a	primo Color
b	secondo Color

Restituisce

bool vero se i due Color sono opposti, falso altrimenti

5.9.4.11 is_piece_equal()

```
bool is_piece_equal (
            Piece a,
            Piece b )
```

Compara due Piece e resituisce vero se sono uguali.

Parametri

a	primo Piece
b	secondo Piece

Restituisce

bool vero se i due Piece sono uguali, falso altrimenti

5.9.4.12 is_piece_null()

```
bool is_piece_null (
            Piece piece )
```

Controlla se un pezzo è nullo, ovvero se non c'è nessun pezzo.

Parametri

piece	pezzo da controllare

Restituisce

bool vero se il pezzo è nullo, falso altrimenti

5.9.4.13 is_pos_equal()

```
bool is_pos_equal (
            Pos a,
            Pos b )
```

Compara due Pos e restituisce vero se sono uguali.

Parametri

a	prima Pos
b	seconda Pos

Restituisce

bool vero se le due Pos sono uguali, falso altrimenti

5.9.5 Documentazione delle variabili

5.9.5.1 color

```
Color color
```

Indice analitico