

MiniLaska Gruppo 4

1.0.0

Generato da Doxygen 1.8.13

Contents

Chapter 1

Indice dei moduli

1.1 Moduli

Questo è l'elenco di tutti i moduli:

Funzioni ausiliarie	??
Funzioni di gestione della memoria	??
Funzioni di input	??
Funzioni di output	??
Funzioni delle logiche di gioco	??

Chapter 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:

cella	??
-----------------------	-------	----

Chapter 3

Indice dei file

3.1 Elenco dei file

Questo è un elenco dei file documentati con una loro breve descrizione:

ml_lib.h	Header della libreria ml_lib	??
ml_main.c	Il main di MiniLaska	??

Chapter 4

Documentazione dei moduli

4.1 Funzioni ausiliarie

Funzioni

- void `set_id_player` (`pedina *p`, `id_p value`)
*Imposta l'id_player value della pedina indicata dal puntatore *p.*
- `id_p get_id_player` (`pedina *p`)
*Ritorna id_player dalla pedina *p specificata.*
- void `set_board_value` (`pedina **board`, unsigned `x`, unsigned `y`, `pedina *value`)
Imposta la pedina value nella posizione x , y nella scacchiera board.
- `pedina * get_board_value` (`pedina **board`, unsigned `x`, unsigned `y`)
Ritorna la pedina contenuta nella posizione x , y di board.
- `pedina * get_board_value_middle` (`pedina **board`, unsigned `x`, unsigned `y`)
Ritorna la pedina "middle" contenuta nella posizione x , y di board.
- `pedina * get_board_value_down` (`pedina **board`, unsigned `x`, unsigned `y`)
Ritorna la pedina "down" contenuta nella posizione x, y di board.
- void `set_grade` (`pedina *p`, `gr value`)
Imposta il grado value della pedina indicata dal puntatore p.
- `gr get_grade` (`pedina *p`)
Ritorna il grado value della pedina indicata dal puntatore p.

4.1.1 Descrizione dettagliata

4.1.2 Documentazione delle funzioni

4.1.2.1 `get_board_value()`

```
pedina* get_board_value (
    pedina ** board,
    unsigned x,
    unsigned y )
```

Ritorna la *pedina* contenuta nella posizione *x* , *y* di *board*.

Parametri

<i>board</i>	matrice linearizzata della scacchiera
<i>x</i>	coordinata x della cella desiderata
<i>y</i>	coordinata y della cella desiderata

Ritorna il puntatore alla pedina nella posizione x,y di board.

4.1.2.2 `get_board_value_down()`

```
pedina* get_board_value_down (
    pedina ** board,
    unsigned x,
    unsigned y )
```

Ritorna la *pedina* "down" contenuta nella posizione *x*, *y* di *board*.

Parametri

<i>board</i>	matrice linearizzata della scacchiera
<i>x</i>	coordinata x della cella desiderata
<i>y</i>	coordinata y della cella desiderata

Ritorna il valore della pedina down nella posizione indicata nella scacchiera.

4.1.2.3 `get_board_value_middle()`

```
pedina* get_board_value_middle (
    pedina ** board,
    unsigned x,
    unsigned y )
```

Ritorna la *pedina* "middle" contenuta nella posizione *x*, *y* di *board*.

Parametri

<i>board</i>	matrice linearizzata della scacchiera
<i>x</i>	coordinata x della cella desiderata
<i>y</i>	coordinata y della cella desiderata

Ritorna il valore della pedina middle nella posizione x,y di board.

4.1.2.4 `get_grade()`

```
gr get_grade (
    pedina * p )
```

Ritorna il grado *value* della pedina indicata dal puntatore *p*.

Parametri

<i>p</i>	puntatore ad una pedina
----------	-------------------------

Ritorna il grado di una pedina.

4.1.2.5 `get_id_player()`

```
id_p get_id_player (
    pedina * p )
```

Ritorna *id_player* dalla pedina **p* specificata.

Parametri

<i>p</i>	puntatore ad una pedina
----------	-------------------------

Ritorna il proprietario della pedina.

4.1.2.6 `set_board_value()`

```
void set_board_value (
    pedina ** board,
    unsigned x,
    unsigned y,
    pedina * value )
```

Imposta la pedina *value* nella posizione *x* , *y* nella scacchiera *board*.

Parametri

<i>board</i>	matrice linearizzata della scacchiera
<i>x</i>	coordinata x della cella desiderata
<i>y</i>	coordinata y della cella desiderata
<i>value</i>	la pedina da inserire

Imposta il valore *value* nella posizione indicata nella scacchiera.

4.1.2.7 `set_grade()`

```
void set_grade (
    pedina * p,
    gr value )
```

Imposta il grado *value* della pedina indicata dal puntatore *p*.

Parametri

<i>p</i>	puntatore ad una pedina
<i>value</i>	il valore da settare

Imposta il grado di una pedina.

4.1.2.8 set_id_player()

```
void set_id_player (
    pedina * p,
    id_p value )
```

Imposta l'id_player *value* della pedina indicata dal puntatore **p*.

Parametri

<i>p</i>	puntatore ad una pedina
<i>value</i>	il valore da settare

Imposta il proprietario della pedina.

4.2 Funzioni di gestione della memoria

Funzioni

- `pedina** createMatrix ()`
Funzione che crea la matrice della scacchiera.
- `void destroyMatrix (pedina **board)`
Distrugge la matrice della scacchiera.
- `void fillBoard (pedina **board)`
Riempie la scacchiera con le pedine.

4.2.1 Descrizione dettagliata

4.2.2 Documentazione delle funzioni

4.2.2.1 createMatrix()

```
pedina** createMatrix ( )
```

Funzione che crea la matrice della scacchiera.

Ritorna un puntatore di tipo `pedina**` ad una matrice bidimensionale di puntatori a pedina linearizzata.

4.2.2.2 destroyMatrix()

```
void destroyMatrix (
    pedina ** board )
```

Distrugge la matrice della scacchiera.

Parametri

<code>board</code>	matrice linearizzata della scacchiera
--------------------	---------------------------------------

Funzione che dealloca la memoria della matrice della scacchiera.

4.2.2.3 fillBoard()

```
void fillBoard (
    pedina ** board )
```

Riempie la scacchiera con le pedine.

Parametri

<i>board</i>	matrice linearizzata della scacchiera
--------------	---------------------------------------

Riempie la scacchiera con le pedine. Il giocatore 1 (*UserOne*) sarà posizionato nella parte bassa della scacchiera.

4.3 Funzioni di input

Funzioni

- int `catchInput` (int *`cord`, `pedina` **`board`)

Legge l'input da tastiera.

4.3.1 Descrizione dettagliata

4.3.2 Documentazione delle funzioni

4.3.2.1 `catchInput()`

```
int catchInput (
    int * cord,
    pedina ** board )
```

Legge l'input da tastiera.

Parametri

<i>cord</i>	array contenente le coordinate di partenza e destinazione della pedina
<i>board</i>	matrice linearizzata della scacchiera

Legge l'input dall'utente e traduce le coordinate in int, che vengono inseriti in un array apposito.

4.4 Funzioni di output

Funzioni

- void `printPedina` (`pedina *p`)
Stampa una lettera rappresentante la pedina.
- void `printMatrix` (`pedina **board`)
Stampa la scacchiera.
- void `printStatus` (unsigned `turn`)
Stampa lo stato del gioco.
- void `printRules` ()
Stampa le regole del gioco.
- void `victory` (`id_p` winner)
Schermata di vittoria.
- void `inputError` ()
Schermata di errore di input.

4.4.1 Descrizione dettagliata

4.4.2 Documentazione delle funzioni

4.4.2.1 `inputError()`

```
void inputError ( )
```

Schermata di errore di input.

Fornisce informazioni in caso di inserimento dati scorretto.

4.4.2.2 `printMatrix()`

```
void printMatrix (  
    pedina ** board )
```

Stampa la scacchiera.

Parametri

<code>board</code>	matrice linearizzata della scacchiera
--------------------	---------------------------------------

Funzione che stampa la scacchiera con una cornice che definisce le coordinate.

4.4.2.3 `printPedina()`

```
void printPedina (
```

```
pedina * p )
```

Stampa una lettera rappresentante la pedina.

Parametri

<i>p</i>	puntatore alla pedina
----------	-----------------------

Stampa un carattere ASCII identificativo del contenuto della casella p:

- b/n se il giocatore è bianco o nero (*UserOne* / *UserTwo*).
- maiuscola/minuscola se la pedina è ufficiale/soldato.

4.4.2.4 printRules()

```
void printRules ( )
```

Stampa le regole del gioco.

Stampa le regole del gioco.

4.4.2.5 printStatus()

```
void printStatus (
    unsigned turn )
```

Stampa lo stato del gioco.

Parametri

<i>turn</i>	numero dei turni passati
-------------	--------------------------

Stampa lo status del gioco (numero del turno e giocatore che deve muovere).

4.4.2.6 victory()

```
void victory (
    id_p winner )
```

Schermata di vittoria.

Stampa il vincitore del gioco.

4.5 Funzioni delle logiche di gioco

Funzioni

- int `can_move` (`pedina **board`, int `x`, int `y`)
Verifica la possibilità di muoversi.
- int `isWinner` (`pedina **board`, int `id_p` player)
Verifica che il giocatore player abbia vinto.
- int `isForbiddenCell` (unsigned `x`, unsigned `y`)
Verifica che la cella sia accessibile.
- int `move` (`pedina **board`, unsigned `from_x`, unsigned `from_y`, unsigned `to_x`, unsigned `to_y`, unsigned `turn`)
Verifica che la mossa selezionata sia legale e la esegue.
- int `distance` (int `from_x`, int `from_y`, int `to_x`, int `to_y`)
Restituisce un codice che descrive la lunghezza della mossa.
- void `capture` (`pedina **board`, unsigned `from_x`, unsigned `from_y`, unsigned `to_x`, unsigned `to_y`)
Esegue la cattura delle pedine.
- int `gradeCheck` (`pedina **board`, unsigned `from_x`, unsigned `from_y`, unsigned `to_y`)
Verifica che la mossa selezionata sia compatibile con il grado della pedina.
- int `can_eat` (`pedina **board`, int `x`, int `y`)
Verifica la possibilità di mangiare.
- int `existMandatory` (`pedina **board`, unsigned `from_x`, unsigned `from_y`, unsigned `to_x`, unsigned `to_y`)
Controlla la presenza di mosse obbligatorie.

4.5.1 Descrizione dettagliata

4.5.2 Documentazione delle funzioni

4.5.2.1 `can_eat()`

```
int can_eat (
    pedina ** board,
    int x,
    int y )
```

Verifica la possibilità di mangiare.

Parametri

<code>board</code>	matrice linearizzata della scacchiera
<code>x</code>	coordinata x della cella
<code>y</code>	coordinata y della cella

Verifica la possibilità della pedina in `x`, `y` di mangiare le pedine avversarie intorno a sé

4.5.2.2 can_move()

```
int can_move (
    pedina ** board,
    int x,
    int y )
```

Verifica la possibilità di muoversi.

Parametri

<i>board</i>	matrice linearizzata della scacchiera
<i>x</i>	coordinata x della cella
<i>y</i>	coordinata y della cella

Verifica la possibilità della pedina in *x* , *y* di muoversi nelle caselle adiacenti

4.5.2.3 capture()

```
void capture (
    pedina ** board,
    unsigned from_x,
    unsigned from_y,
    unsigned to_x,
    unsigned to_y )
```

Esegue la cattura delle pedine.

Parametri

<i>board</i>	matrice linearizzata della scacchiera
<i>from</i> _↔ <i>_x</i>	coordinata x della cella di partenza
<i>from</i> _↔ <i>_y</i>	coordinata y della cella di partenza
<i>to_x</i>	coordinata x della cella di destinazione
<i>to_y</i>	coordinata y della cella di destinazione

Questa funzione si occupa di catturare le pedine indicate. Si assume la correttezza delle coordinate inserite, la legalità della mossa è verificata nella funzione [move\(\)](#).

4.5.2.4 distance()

```
int distance (
    int from_x,
    int from_y,
    int to_x,
    int to_y )
```

Restituisce un codice che descrive la lunghezza della mossa.

Parametri

<i>from</i> ↔ _x	coordinata x della cella di partenza
<i>from</i> ↔ _y	coordinata y della cella di partenza
<i>to_x</i>	coordinata x della cella di destinazione
<i>to_y</i>	coordinata y della cella di destinazione

Restituisce la distanza in modulo tra due punti nella matrice: Se è maggiore di 2, uguale a 0, o la destinazione è in una casella non accessibile restituisce il codice errore -1.

Le coordinate inserite sono corrette (la destinazione non è una casella proibita).

4.5.2.5 existMandatory()

```
int existMandatory (
    pedina ** board,
    unsigned from_x,
    unsigned from_y,
    unsigned to_x,
    unsigned to_y )
```

Controlla la presenza di mosse obbligatorie.

Parametri

<i>board</i>	matrice linearizzata della scacchiera
<i>from</i> ↔ _x	coordinata x della cella di partenza
<i>from</i> ↔ _y	coordinata y della cella di partenza
<i>to_x</i>	coordinata x della cella di partenza
<i>to_y</i>	coordinata y della cella di partenza

Verifica se, nel caso di non cattura, esiste una cattura obbligatoria da fare. Restituisce 1 se esiste una mossa obbligatoria non tentata, altrimenti 0.

4.5.2.6 gradeCheck()

```
int gradeCheck (
    pedina ** board,
    unsigned from_x,
    unsigned from_y,
    unsigned to_y )
```

Verifica che la mossa selezionata sia compatibile con il grado della pedina.

Parametri

<i>board</i>	matrice linearizzata della scacchiera
--------------	---------------------------------------

Parametri

<i>from</i> ↔ _x	coordinata x della cella di partenza
<i>from</i> ↔ _y	coordinata y della cella di partenza
<i>to</i> _y	coordinata y della cella di destinazione

Verifica il grado della pedina mossa: restituisce 1 se la mossa è consentita, 0 se non è consentita.

4.5.2.7 isForbiddenCell()

```
int isForbiddenCell (
    unsigned x,
    unsigned y )
```

Verifica che la cella sia accessibile.

Parametri

x	coordinata x della cella
y	coordinata y della cella

Restituisce 1 se la cella non è accessibile (si possono usare solo le celle bianche della scacchiera), altrimenti 0.

4.5.2.8 isWinner()

```
int isWinner (
    pedina ** board,
    id_p idPlayer )
```

Verifica che il giocatore *player* abbia vinto.

Parametri

<i>board</i>	matrice linearizzata della scacchiera
<i>idPlayer</i>	giocatore selezionato

Verifica che il giocatore *idPlayer* abbia vinto. Restituisce 1 se idPlayer ha vinto, altrimenti 0.

4.5.2.9 move()

```
int move (
    pedina ** board,
    unsigned from_x,
    unsigned from_y,
    unsigned to_x,
```

```
    unsigned to_y,  
    unsigned turn )
```

Verifica che la mossa selezionata sia legale e la esegue.

Parametri

<i>board</i>	matrice linearizzata della scacchiera
<i>from</i> _↔ <i>_x</i>	coordinata x della cella di partenza
<i>from</i> _↔ <i>_y</i>	coordinata y della cella di partenza
<i>to_x</i>	coordinata x della cella di destinazione
<i>to_y</i>	coordinata y della cella di destinazione
<i>turn</i>	numero del turno corrente

Restituisce 1 se la mossa è stata fatta, 0 se non è stato possibile. Le coordinate inserite sono corrette in fase di input (sono all'interno della scacchiera e non sono caselle proibite). Verifica che la distanza ed il grado siano compatibili con la mossa.

Chapter 5

Documentazione delle classi

5.1 Riferimenti per la struct cella

```
#include <ml_lib.h>
```

Attributi pubblici

- `id_p id_player`
- `gr grado`
- `pedina * middle`
- `pedina * down`

5.1.1 Descrizione dettagliata

Definizione del tipo pedina

5.1.2 Documentazione dei membri dato

5.1.2.1 down

```
pedina* cella::down
```

Puntatore alla pedina in fondo alla colonna

5.1.2.2 grado

```
gr cella::grado
```

Grado della pedina

5.1.2.3 id_player

`id_p` `cella::id_player`

ID del giocatore proprietario della pedina

5.1.2.4 middle

`pedina*` `cella::middle`

Puntatore alla pedina di mezzo della colonna

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

- [ml_lib.h](#)

Chapter 6

Documentazione dei file

6.1 Riferimenti per il file ml_lib.h

Header della libreria ml_lib.

Composti

- struct [cella](#)

Ridefinizioni di tipo (typedef)

- typedef struct [cella](#) [pedina](#)

Tipi enumerati (enum)

- enum [id_p](#) { **UserOne**, **UserTwo** }
- enum [gr](#) { **Soldier**, **Officer** }

Funzioni

- void [set_id_player](#) ([pedina](#) *p, [id_p](#) value)
*Imposta l'id_player value della pedina indicata dal puntatore *p.*
- [id_p](#) [get_id_player](#) ([pedina](#) *p)
*Ritorna id_player dalla pedina *p specificata.*
- void [set_board_value](#) ([pedina](#) **board, unsigned x, unsigned y, [pedina](#) *value)
Imposta la pedina value nella posizione x , y nella scacchiera board.
- [pedina](#) * [get_board_value](#) ([pedina](#) **board, unsigned x, unsigned y)
Ritorna la pedina contenuta nella posizione x , y di board.
- [pedina](#) * [get_board_value_middle](#) ([pedina](#) **board, unsigned x, unsigned y)
Ritorna la pedina "middle" contenuta nella posizione x , y di board.
- [pedina](#) * [get_board_value_down](#) ([pedina](#) **board, unsigned x, unsigned y)
Ritorna la pedina "down" contenuta nella posizione x, y di board.
- void [set_grade](#) ([pedina](#) *p, [gr](#) value)

- Imposta il grado value della pedina indicata dal puntatore p.*
- `gr get_grade (pedina *p)`
Ritorna il grado value della pedina indicata dal puntatore p.
- `pedina ** createMatrix ()`
Funzione che crea la matrice della scacchiera.
- `void destroyMatrix (pedina **board)`
Distrugge la matrice della scacchiera.
- `void fillBoard (pedina **board)`
Riempie la scacchiera con le pedine.
- `int catchInput (int *cord, pedina **board)`
Legge l'input da tastiera.
- `void printPedina (pedina *p)`
Stampa una lettera rappresentante la pedina.
- `void printMatrix (pedina **board)`
Stampa la scacchiera.
- `void printStatus (unsigned turn)`
Stampa lo stato del gioco.
- `void printRules ()`
Stampa le regole del gioco.
- `void victory (id_p winner)`
Schermata di vittoria.
- `void inputError ()`
Schermata di errore di input.
- `int can_move (pedina **board, int x, int y)`
Verifica la possibilità di muoversi.
- `int isWinner (pedina **board, id_p player)`
Verifica che il giocatore player abbia vinto.
- `int isForbiddenCell (unsigned x, unsigned y)`
Verifica che la cella sia accessibile.
- `int move (pedina **board, unsigned from_x, unsigned from_y, unsigned to_x, unsigned to_y, unsigned turn)`
Verifica che la mossa selezionata sia legale e la esegue.
- `int distance (int from_x, int from_y, int to_x, int to_y)`
Restituisce un codice che descrive la lunghezza della mossa.
- `void capture (pedina **board, unsigned from_x, unsigned from_y, unsigned to_x, unsigned to_y)`
Esegue la cattura delle pedine.
- `int gradeCheck (pedina **board, unsigned from_x, unsigned from_y, unsigned to_y)`
Verifica che la mossa selezionata sia compatibile con il grado della pedina.
- `int can_eat (pedina **board, int x, int y)`
Verifica la possibilità di mangiare.
- `int existMandatory (pedina **board, unsigned from_x, unsigned from_y, unsigned to_x, unsigned to_y)`
Controlla la presenza di mosse obbligatorie.

6.1.1 Descrizione dettagliata

Header della libreria `ml_lib`.

Questo file contiene le definizioni di tutte le strutture e delle funzioni che compongono la libreria `ml_lib`

6.1.2 Documentazione delle ridefinizioni di tipo (typedef)

6.1.2.1 pedina

```
typedef struct cella pedina
```

Rinominazione del tipo struct cella in pedina, per praticità di scrittura

6.1.3 Documentazione dei tipi enumerati

6.1.3.1 gr

```
enum gr
```

Definizione dei due possibili gradi della pedina

6.1.3.2 id_p

```
enum id_p
```

Definizione dei due giocatori esistenti

6.2 Riferimenti per il file ml_main.c

Il main di MiniLaska.

Funzioni

- int `main` ()

Variabili

- `pedina` ** `board` = NULL
- int `coordinate` [4]
- int `success_move` = 1
- int `success_input` = 1
- unsigned `turn` = 0

6.2.1 Descrizione dettagliata

Il main di MiniLaska.

Questo file contiene il programma del gioco MiniLaska, che utilizza la libreria ml_lib

6.2.2 Documentazione delle funzioni

6.2.2.1 main()

```
main ( )
```

Funzione principale del gioco

6.2.3 Documentazione delle variabili

6.2.3.1 board

```
pedina** board = NULL
```

La scacchiera

6.2.3.2 coordinate

```
int coordinate[4]
```

Array contenente le coordinate di partenza e di arrivo di ogni mossa

6.2.3.3 success_input

```
int success_input = 1
```

Flag che verifica la correttezza dell'input

6.2.3.4 success_move

```
int success_move = 1
```

Flag che verifica la legalità di una mossa

6.2.3.5 turn

```
unsigned turn = 0
```

Contatore del turno corrente