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:

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:

Chapter 3

Indice dei file

3.1 Elenco dei file

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

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, point p, pedina ∗value)

  Imposta la pedina value nella posizione x , y nella scacchiera board.
- pedina ∗ get_board_value (pedina ∗∗board, point p)

  Ritorna la pedina contenuta nella posizione x , y di board.
- pedina ∗ get_board_value_immediate (pedina ∗∗board, int x, int y)

  Ritorna la pedina contenuta nella posizione x , y di board.
- pedina ∗ get_board_value_middle (pedina ∗∗board, point p)

  Ritorna la pedina "middle" contenuta nella posizione x , y di board.
- pedina ∗ get_board_value_down (pedina ∗∗board, point p)

  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.
- int is_inside (int x, int y)

  Indica se x, è dentro alla scacchiera.
- int **right_path** (dir direction, gr grade, id_p player)
- int **is_valid_letter** (char input)
- int **is_valid_number** (char input)
- int **is_valid_input** (char input[5])

4.1.1 Descrizione dettagliata

4.1.2 Documentazione delle funzioni

4.1.2.1 get_board_value()

```
pedina* get_board_value (
            pedina ** board,
            point p )
```

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

Parametri

board	matrice linearizzata della scacchiera
p	punto in cui si trova la pedina nella scacchiera

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,
            point p )
```

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

Parametri

board	matrice linearizzata della scacchiera
p	punto in cui si trova la pedina nella scacchiera

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

4.1.2.3 get_board_value_immediate()

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

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

Parametri

board	matrice linearizzata della scacchiera
x	coordinata x della cella desiderata
y	coordinata y della cella desiderata

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

4.1.2.4 get_board_value_middle()

```
pedina* get_board_value_middle (
            pedina ** board,
            point p )
```

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

Parametri

board	matrice linearizzata della scacchiera
p	punto in cui si trova la pedina nella scacchiera

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

4.1.2.5 get_grade()

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

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

Parametri

p	puntatore ad una pedina

Ritorna il grado di una pedina.

4.1.2.6 get_id_player()

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

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

Parametri

p	puntatore ad una pedina

Ritorna il proprietario della pedina.

4.1.2.7 is_inside()

```
int is_inside (
            int x,
            int y )
```

Indica se *x*, è dentro alla scacchiera.

Parametri

x	coordinata x della cella desiderata
y	coordinata y della cella desiderata

Indica se la posizione desiderata è dentro alla scacchiera

4.1.2.8 set_board_value()

```
void set_board_value (
            pedina ** board,
            point p,
            pedina * value )
```

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

Parametri

board	matrice linearizzata della scacchiera
p	punto in cui si trova la pedina nella scacchiera
value	la pedina da inserire

Imposta il valore value nella posizione indicata nella scacchiera.

4.1.2.9 set_grade()

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

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

Parametri

p	puntatore ad una pedina
value	il valore da settare

Imposta il grado di una pedina.

4.1.2.10 set_id_player()

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

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

Parametri

p	puntatore ad una pedina
value	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

board	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

board	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)

4.3.1 Descrizione dettagliata

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 (int 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

board	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

p	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 (
            int turn )
```

Stampa lo stato del gioco.

Parametri

turn	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 isWinner (pedina **board, id_p player)

  Verifica che il giocatore player abbia vinto.
- int isForbiddenCell (point p)

  Verifica che la cella sia accessibile.
- int move (pedina **board, point from, point to, int turn)

  Verifica che la mossa selezionata sia legale e la esegue.
- int distance (point from, point to)

  Restituisce un codice che descrive la lunghezza della mossa.
- void capture (pedina **board, point from, point to)

  Esegue la cattura delle pedine.
- int gradeCheck (pedina **board, point from, point to)

  Verifica che la mossa selezionata sia compatibile con il grado della pedina.
- int can_eat (pedina **board, point p)

  Verifica la possibilità di mangiare.
- int can_move (pedina **board, point p)

  Verifica la possibilità di muoversi.
- int existMandatory (pedina **board, point from, point to)

  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,
            point p )
```

Verifica la possibilità di mangiare.

Parametri

board	matrice linearizzata della scacchiera
p	punto in cui si trova la pedina nella scacchiera

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,
            point p )
```

Verifica la possibilità di muoversi.

Parametri

board	matrice linearizzata della scacchiera
p	punto in cui si trova la pedina nella scacchiera

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

4.5.2.3 capture()

```
void capture (
            pedina ** board,
            point from,
            point to )
```

Esegue la cattura delle pedine.

Parametri

board	matrice linearizzata della scacchiera
from	punto di partenza della pedina
to	punto di arrivo della pedina

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 (
            point from,
            point to )
```

Restituisce un codice che descrive la lunghezza della mossa.

Parametri

from	punto di partenza della pedina
to	punto di arrivo della pedina

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,
            point from,
            point to )
```

Controlla la presenza di mosse obbligatorie.

Parametri

board	matrice linearizzata della scacchiera
from	punto di partenza della pedina
to	punto di arrivo della pedina

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,
            point from,
            point to )
```

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

Parametri

board	matrice linearizzata della scacchiera
from	punto di partenza della pedina
to	punto di arrivo della pedina

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 (
            point p )
```

Verifica che la cella sia accessibile.

Parametri

p	punto in cui si trova la pedina nella scacchiera

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

board	matrice linearizzata della scacchiera
idPlayer	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,
            point from,
            point to,
            int turn )
```

Verifica che la mossa selezionata sia legale e la esegue.

Parametri

board	matrice linearizzata della scacchiera
from	punto di partenza della pedina
to	punto di arrivo della pedina
turn	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

5.2 Riferimenti per la struct mossa

Attributi pubblici

- pedina * **ped**
- point **destinazione**

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

- Autoplay.c

5.3 Riferimenti per la struct node

Attributi pubblici

- int **alfa**
- point **destinazione**

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

- Autoplay.c

5.4 Riferimenti per la struct punto

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

Attributi pubblici

- int x
- int y

5.4.1 Descrizione dettagliata

Definizione del tipo punto

5.4.2 Documentazione dei membri dato

5.4.2.1 x

```
int punto::x
```

Coordinata x

5.4.2.2 y

```
int punto::y
```

Coordinata y

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
- struct punto

Ridefinizioni di tipo (typedef)

- typedef struct cella pedina
- typedef struct punto **point**

Tipi enumerati (enum)

- enum id_p { **UserOne**, **UserTwo** }
- enum gr { **Soldier**, **Officer** }
- enum dir { **Up**, **Down** }

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, point p, pedina *value)

  Imposta la pedina value nella posizione x , y nella scacchiera board.
- pedina * get_board_value (pedina **board, point p)

  Ritorna la pedina contenuta nella posizione x , y di board.
- pedina * get_board_value_immediate (pedina **board, int x, int y)

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

- pedina * get_board_value_middle (pedina **board, point p)

  Ritorna la pedina "middle" contenuta nella posizione x , y di board.
- pedina * get_board_value_down (pedina **board, point p)

  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.
- int is_inside (int x, int y)

  Indica se x, è dentro alla scacchiera.
- int **right_path** (dir direction, gr grade, id_p player)
- int **is_valid_letter** (char input)
- int **is_valid_number** (char input)
- int **is_valid_input** (char input[5])
- 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)
- void printPedina (pedina *p)

  Stampa una lettera rappresentante la pedina.
- void printMatrix (pedina **board)

  Stampa la scacchiera.
- void printStatus (int 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 isWinner (pedina **board, id_p player)

  Verifica che il giocatore player abbia vinto.
- int isForbiddenCell (point p)

  Verifica che la cella sia accessibile.
- int move (pedina **board, point from, point to, int turn)

  Verifica che la mossa selezionata sia legale e la esegue.
- int distance (point from, point to)

  Restituisce un codice che descrive la lunghezza della mossa.
- void capture (pedina **board, point from, point to)

  Esegue la cattura delle pedine.
- int gradeCheck (pedina **board, point from, point to)

  Verifica che la mossa selezionata sia compatibile con il grado della pedina.
- int can_eat (pedina **board, point p)

  Verifica la possibilità di mangiare.
- int can_move (pedina **board, point p)

  Verifica la possibilità di muoversi.
- int existMandatory (pedina **board, point from, point to)

  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 dir

```
enum dir
```

Definizione della direzione

6.1.3.2 gr

```
enum gr
```

Definizione dei due possibili gradi della pedina

6.1.3.3 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
- int turn = 0
- point **from**
- point to

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 to

```
point to
```

Segnaposto dei punti di partenza e arrivo di ogni mossa

6.2.3.6 turn

```
int turn = 0
```

Contatore del turno corrente