9 Novembre 2021

Programmazione B
Ingegneria e Scienze Informatiche - Cesena
A.A. 2021-2022

Elaborato 7

Data di sottomissione: entro le 20 del 14 Novembre 2021.
Formato di sottomissione: un file compresso con nome `Elaborato7.zip`, contenente un unico file sorgente con nome `board.c`
Codeboard: https://codeboard.io/projects/131850/

Specifiche:

- Sviluppare funzioni di libreria per poter gestire l'oggetto **board** nel gioco *minesweeper* (campo minato).
- Viene fornita l'implementazione dell'intero gioco, tranne l'implementazione della libreria `board.c`. L'implementazione fa uso della libreria `curses`.
- I prototipi delle funzioni da implementare sono dichiarati nell'header `board.h` e allegati alle specifiche.
- Il campo di gioco (board) è rappresentato tramite un array bidimensionale.
- Le funzioni di libreria in `board.c` si occupano di gestire come e quali elementi del campo di gioco debbano essere visualizzati, settando ogni entry con una opportuna costante enumerativa di tipo `enum type`.

Vincoli:

- Le implementazioni devono aderire perfettamente ai prototipi e alle specifiche fornite.
- Le eventuali funzioni di utility della libreria devono essere *nascoste* all'esterno.
- Non è possibile utilizzare puntatori o la notazione specifica per i puntatori per lo sviluppo delle funzioni di libreria.

Suggerimenti:

- Nella funzione `display_board()` fare attenzione alle conversioni tra `int` e `unsigned int`.

Descrizione dettagliata delle funzioni:

- `random_board()`: posiziona in modo casuale `num_mines` nella matrice, facendo attenzione a non posizionare nessuna mina né nella posizione `i,j` né attorno alla posizione `i,j` . Al termine della chiamata:
  - ogni posizione della matrice deve essere o `UNKN_FREE` oppure `UNKN_MINE`,
  - ci devono essere esattamente `num_mines` posizioni con valore `UNKN_MINE`,
  - nessuna posizione attorno ad `i,j` (inclusa) deve essere `UNKN_MINE`.
- `flag_board()`: aggiunge un flag (bandiera) nella posizione `i,j`:
  - se la posizione `i,j` è già stata mostrata non fa nulla e ritorna 0,
  - se la posizione `i,j` è già "flaggata", rimuove il flag (`FLAG_FREE` diventa `UNKN_FREE` e `FLAG_MINE` diventa `UNKN_MINE`) e ritorna -1,
  - se la posizione `i,j` non è "flaggata" aggiunge un flag (`UNKN_FREE` diventa `FLAG_FREE` e `UNKN_MINE` diventa `FREE_MINE`) e ritorna 1.
- `display_board()`: mostra il contenuto nella posizione `i,j`
  - se la posizione `i,j` è già stata mostrata oppure è "flaggata" non fa nulla e ritorna 0,
  - se la posizione `i,j` non è già stata mostrata e contiene una mina, posiziona la costante `MINE` in `i,j` e ritorna -1,
  - se la posizione `i,j` non è già stata mostrata e non contiene una mina, posiziona in `i,j` la costante enumerativa che indica il numero di mine che circondano la posizione `i,j` (i.e. `C0, ..., C8`). Se la posizione `i,j` non ha mine attorno (i.e. diventa `C0`) allora tutte le posizioni non scoperte attorno ad `i,j` sono rivelate. Lo stesso algoritmo viene ripetuto fintanto che sono scoperte posizioni marcate con `C0`. La funzione ritorna il numero complessivo di posizioni rivelate.
- `expand_board()`: mostra il contenuto delle posizioni ancora non svelate attorno ad `i,j`. Il comportamento è esattamente lo stesso della `display_board()`, valore di ritorno compreso, ma applicato a tutte le celle non svelate attorno ad `i,j`. Casi in cui la funzione non fa nulla e ritorna 0:
  - se la posizione `i,j` non è già stata mostrata,
  - se la posizione `i,j` non contiene attorno un numero di celle flaggate pari al numero mostrato in `i,j`.

```c
#ifndef BOARD_H
#define BOARD_H

#include "game.h" // We need to know GAME_COLS

enum field {
  C0,          // The cell is sorrounded by 0 mines
  C1,          // The cell is sorrounded by 1 mine
  C2,          // The cell is sorrounded by 2 mines
  C3,          // The cell is sorrounded by 3 mines
  C4,          // The cell is sorrounded by 4 mines
  C5,          // The cell is sorrounded by 5 mines
  C6,          // The cell is sorrounded by 6 mines
  C7,          // The cell is sorrounded by 7 mines
  C8,          // The cell is sorrounded by 8 mines
  UNKN_FREE,   // The cell is not displayed and it is not a mine
  UNKN_MINE,   // The cell is not displayed and it is a mine
  FLAG_FREE,   // The cell is flagged and it is not a mine
  FLAG_MINE,   // The cell is flagged and it s a mine
  MINE         // The cell is a mine
};

/*
 * Fills the rows*cols board with num_mines random mines
 * leaving free the neighborhood of position i,j
 */
void random_board(int board[][GAME_COLS], unsigned int rows,
        unsigned int cols, unsigned int i, unsigned int j,
        unsigned int num_mines);

/*
 * Flags/unflags the i,j position in the board. Returns
 * - -1 if the position was flagged. Removes the flag
 * -  0 if the position is already displyed
 * -  1 if the position is not flagged and not already
 *    displayed. Puts a flag in position i,j.
 */
int flag_board(int board[][GAME_COLS], unsigned int rows,
        unsigned int cols, unsigned int i, unsigned int j);

/*
 * Displays position i,j in the board. Returns the number of
 * displayed cells or -1 if i,j contains a mine.
 */
int display_board(int board[][GAME_COLS], unsigned int rows,
        unsigned int cols, unsigned int i, unsigned int j);

/*
 * Expands all the free cells sourrounding  position i,j in
 * the board. Returns the number of expanded cells or -1 if
 * one contains a mine.
 */
int expand_board(int board[][GAME_COLS], unsigned int rows,
        unsigned int cols, unsigned int i, unsigned int j);

#endif
```