Programmazione Modulo 2:
Campo minato

Il progetto consiste nella realizzazione di un programma per giocare a *Campo minato*. Nella vostra soluzione potete fare liberamente uso delle funzioni presenti nella libreria standard di C.

1 Funzionalità richieste

Il programma deve supportare le seguenti funzionalità:

- generazione di un nuovo schema, dati in input la dimensione del campo (righe e colonne) e la quantità di mine da inserire;
- lettura da file di uno schema;
- scrittura su file di uno schema;
- modalità di gioco interattiva con possibilità di annullare le ultime mosse eseguite.

Quando avviato, il programma offre un menù che permette di scegliere tra le prime due operazioni (più un'opzione per l'uscita); eseguita una di queste due operazioni, dovranno essere disponibili tutte le scelte. Le opzioni di salvataggio e di gioco si riferiscono all'ultimo schema letto/generato. Il programma dovrà terminare solo quando viene selezionata l'apposita voce dal menù.

2 Campo minato

2.1 Campo di gioco

In *Campo minato* si usa un campo rettangolare suddiviso in celle che possono contenere i seguenti elementi (cf. Figura 1):

- una mina;
- un numero che specifica la quantità di celle adiacenti (anche in diagonale) contenenti una mina;
- nessun numero, qualora nessuna delle celle adiacenti contenga una mina.

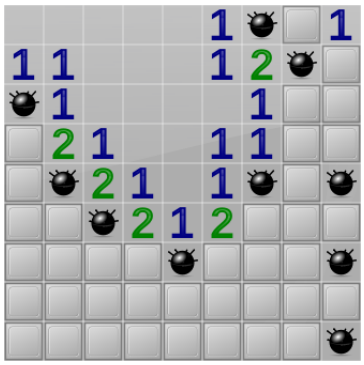


Figura 1: Un campo di gioco.

2.2 Modalità di gioco

L'obiettivo del gioco consiste nel ripulire il campo senza fare esplodere le mine. Quando il giocatore seleziona una cella coperta del campo, si possono verificare diverse azioni in base al suo contenuto:

- se contiene una mina, questa esploderà e farà terminare il gioco;
- se contiene un numero, la cella viene scoperta e il giocatore può effettuare la mossa successiva;
- se è vuota, devono essere automaticamente scoperte tutte le celle vuote adiacenti a quella selezionata, fino a quando non appaiono dei numeri o si raggiunge il bordo dello schema; in Figura 2 potete vedere la regione scoperta quando il giocatore seleziona la cella indicata dal pallino rosso.

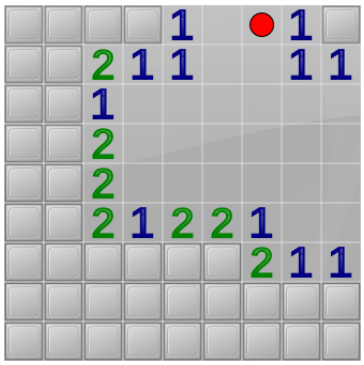


Figura 2: Esempio di selezione di una cella vuota.

Il giocatore può marcare con una bandierina le posizioni in cui crede siano presenti delle mine: una volta marcate, le celle non possono essere scoperte finché la bandierina non viene rimossa. In Figura 3 trovate un esempio di campo risolto correttamente con tutte le mine identificate da una bandierina.

Figura 3: Vittoria!

2.3 Annullamento delle mosse

Come accennato in Sezione 1, estendiamo il classico *Campo minato* con la possibilità di "tornare indietro" qualora venga scoperta una cella contenente una mina: il numero di volte in cui è possibile farlo è un intero non negativo fornito in input dall'utente prima dell'inizio della partita.

Per rendere il gioco più equo, si vuole introdurre un malus che aumenta mano a mano che viene sfruttata questa funzionalità. Se viene usato per l'n-esima volta l'annullamento, devono essere annullate le ultime n mosse eseguite (o si annullano tutte, qualora il numero di mosse effettuate sia minore di n). Pertanto al primo utilizzo verrà annullata solo l'ultima mossa, al secondo utilizzo verranno annullate le ultime due mosse e così via.

Quando una mossa viene annullata, le celle che sono state scoperte a causa dell'esecuzione della mossa devono essere nuovamente coperte. L'operazione di annullamento riguarda esclusivamente le azioni del giocatore che scoprono una cella: marcare una cella come contenente una mina o rimuovere la marcatura non sono operazioni che vengono coinvolte dall'annullamento. In Figura 4 è mostrato un esempio di annullamento delle ultime due mosse.

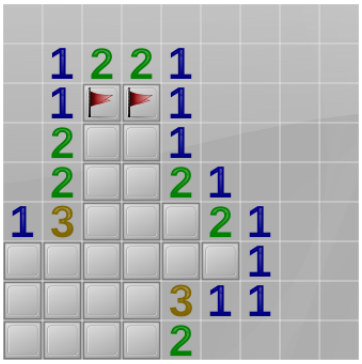
3 Modalità di gioco interattiva

La modalità di gioco interattiva deve seguire la descrizione del gioco data in Sezione 2.2 e consentire l'annullamento delle mosse. La partita si svolge sull'ultimo campo letto/generato tramite il menù principale dell'applicazione.

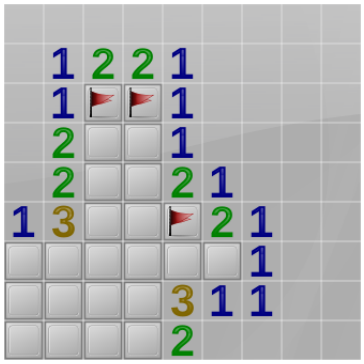
Ad ogni mossa deve essere visualizzato il campo da gioco dove tutti i tipi di celle sono facilmente distinguibili (celle coperte, celle scoperte contenenti un numero, celle marcate dall'utente, etc). Deve quindi essere offerta all'utente la possibilità di:

- scoprire una cella, data la posizione;
- marcare una cella come contenente una mina o rimuovere la marcatura;
- terminare il gioco e ritornare al menù principale.

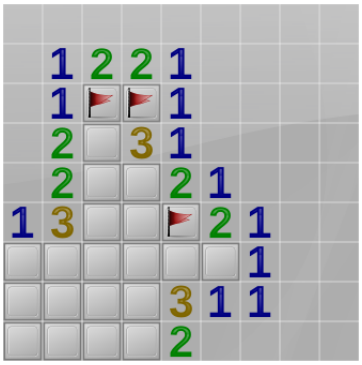
Qualora l'ultima cella scoperta contenga una mina, le scelte che devono essere offerte al giocatore sono:



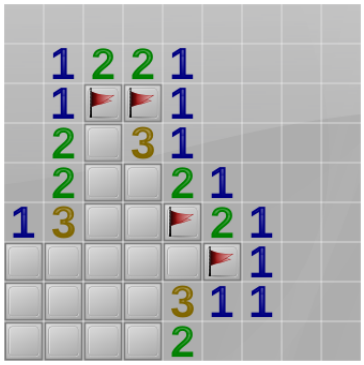
(a) Stato iniziale.



(b) Viene marcata una cella.

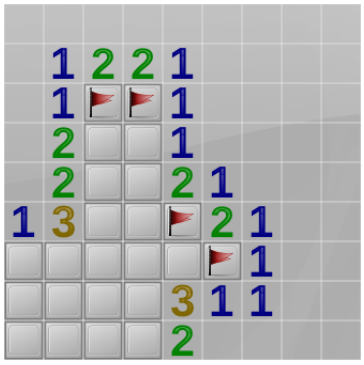


(c) Il giocatore scopre una cella.



(d) Viene marcata un'altra cella.

(e) Ops, una mina!



(f) Annullamento delle ultime due mosse.

Figura 4: Esempio di annullamento delle ultime due mosse.

- l'annullamento della mossa come descritto in Sezione 2.3, se non sono esaurite le possibilità;
- terminazione della partita.

4 Formato dei file degli schemi

Il formato dei file che devono essere letti e prodotti dalla vostra applicazione è il seguente: all'inizio è presente una riga contenente il numero di righe e di colonne dello schema, separate da virgola[1]; segue una sequenza di righe contenenti le posizioni delle mine, dove ciascuna riga contiene gli indici di riga e colonna della mina nel campo di gioco, anch'essi separati da virgola. Nel Codice 1 trovate un possibile file di input che rappresenta lo schema di Figura 1.

I file possono contenere alcune righe vuote che devono essere ignorate. Inoltre viene utilizzata la convenzione Unix dove il ritorno a capo è codificato dal carattere `\n`: su Windows, invece, il ritorno a capo è codificato dalla coppia di caratteri `\r\n`, quindi se create dei file di test su questa piattaforma assicuratevi di configurare in maniera appropriata il vostro editor di testo.

Per semplicità potete assumere che i file forniti in input alla vostra applicazione rappresentino schemi validi (le dimensioni sono positive, tutte le posizioni delle mine sono valide, etc).

Codice 1: Esempio di file di input.

```
9, 9

0, 6
1, 7
2, 0
4, 1
4, 6
4, 8
5, 2
6, 4
6, 8
8, 8
```

5 Elementi di valutazione

Sebbene il progetto sia un lavoro che può essere svolto in gruppo, la valutazione sarà individuale. Oltre alla discussione orale, saranno valutate:

- qualità del codice: oltre alla correttezza, saranno considerate l'eleganza della soluzione implementata, l'utilizzo di indentazione corretta e consistente, la suddivisione in sottoprogrammi, etc;

[1] Esistono varie tecniche per l'allocazione dinamica di array multidimensionali: consultate la pagina `http://c-faq.com/aryptr/dynmuldimary.html` per maggiori dettagli, ma ignorate la parte relativa allo standard C99.

- qualità della documentazione: codice non commentato non sarà valutato;
- eventuali funzionalità extra.[2]

6 Funzionalità extra

Siete liberi di implementare altre funzionalità, ad esempio la gestione degli errori nei file di input, l'implementazione di un'interfaccia grafica particolare per la modalità di gioco interattiva usando librerie come `ncurses` o una semplice intelligenza artificiale che cerca di giocare in maniera automatica. Non modificate le regole di base del gioco spiegate in Sezione 2. Proposte particolarmente interessanti o eleganti potrebbero convincere il Prof. Marin a darvi un bonus all'esame :)

[2] Non sono necessarie per ottenere il punteggio massimo se gli altri requisiti sono soddisfatti.