

MINILASKA

vo duy khiem pietro pagnan

| Introduzione alla Programmazione | 30/12/2020

* *Presentazione elaborato:*

***MiniLaska*** è una variante del gioco comunemente chiamato: **Laska** (derivato dalla dama inglese e russa).

*Componenti* del gioco:

- Una scacchiera 7x7 con un totale di quarantanove caselle;

- Undici pedine assegnate ad ogni giocatore;

- Due livelli per le pedine: *soldato e ufficiale.*

Seguono ora le fasi elaborate per la realizzazione del codice sorgente:

* ***RAPPRESENTAZIONE GRAFICA***

![Immagine che contiene testo, batteria, tabellonesegnapunti

Descrizione generata automaticamente]()La prima scelta verso l’idealizzazione del nostro progetto è stata quella di rappresentare un tavolo da gioco  
dettagliato e chiaro.

* La figura proposta, presenta la tabella di gioco che ha funto da punto di   
  partenza, idealizzando la scacchiera come un array bidimensionale.

L’utilizzo di questa rappresentazione grafica ha contrastato le nostre iniziali difficoltà nella definizione degli spostamenti delle singole pedine.

Abbiamo quindi, completato i movimenti necessari apportando, inoltre un miglioramento alla qualità della scacchiera.

Immagine che contiene freccia

Descrizione generata automaticamente

* L’immagine espone il tavolo da gioco completo, cercando sempre di vedere il contenuto delle caselle come se fosse un array bidimensionale.

Stando al regolamento e ad alcune modifiche apportate al progetto “MiniLaska”, è stata creata una *struct* di tipo *struct tower*, composta da **composition[3]**.  
 [3] sta ad indicare l'altezza massima di ogni singola torre.

Specifico che in composition [0] sarà contenuto il valore della pedina che si trova nel primo livello della torre, in composition [1] il valore della pedina che sta nel secondo livello e in composition[2] il valore della pedina che sta nel terzo livello.

Nel caso in cui composizion[0] ha valore 0, ciò indica che è una pedina vuota

* Per la rappresentazione delle pedine sono stati utilizzati i numeri dall’uno al quattro:

1. SOLDATO: player1;
2. SOLDATO: player2;
3. UFFICIALE: player1;
4. UFFICIALE: player2.

* ***CONTROLLI E SPOSTAMENTI***

Riguardo alle mosse compiute da ogni giocatore, si è cercato di creare delle funzioni più generali possibili da poterle così riutilizzare ad ogni turno successivo.

Una volta che le coordinate e la destinazione della pedina che si vuole spostare sono state inserite, vengono interpellate tutte le funzioni necessarie ad effettuare i controlli per gli spostamenti.

Vediamo ora le principali funzioni:

*→ int controllopedina:* controlla che le coordinate della pedina scelta siano state inserite correttamente,

ovvero accerta:

- Il corretto inserimento delle coordinate all’interno della scacchiera;

- L’appropriato uso della casella, quindi che non sia vuota oppure di colore nero (casella non utilizzabile) o

che essa non contenga una pedina avversaria.

→ *int controllodestinazione****:*** verifica che le coordinate di destinazione siano state inserite correttamente.   
 Più precisamente, che quest’ultime non siano fuori dallo spazio prestabilito (scacchiera), che la casella

selezionata sia vuota, che il movimento della pedina non sia oltre le due caselle e che il movimento sia al rispetto delle regole di MiniLaska.

→ *int controllogrado****:*** restituisce solamente il livello più alto in cui è presente la pedina.

*→ int obbligomangiare*: controlla quali pedine possono essere vinte, in tal caso viene riferito l’obbligo di

ottenere la pedina dell’avversario. In aiuto alla prima funzione viene introdotta a seguire anche: *int*

*angolopedine.*

La conclusione dei controlli con esito positivo, comporta il successivo spostamento nel tavolo da gioco. Applichiamo così un ulteriore funzione chiamata: *void spostapedina*; Seguiamo lavorando nell’array andando a modificare le componenti interessate.

Per concludere, stampiamo il tutto a terminale per verificare che i movimenti siano avvenuti correttamente.

* ***TURNI***

Il tavolo da gioco è stato realizzato e sono stati effettuati i movimenti prestando attenzione alle regole.

Segue successivamente:

* Un controllo ai turni, per i quali è stata implementata perciò la funzione “*int turno”,* che stabilisce a quale giocatore spetti la prossima mossa.
* Un ultimo controllo è attuato dalla funzione “ int controllo\_pedine\_presenti”.   
  Si verifica così, in ogni casella, qual è la pedina che controlla la torre (comandante).  
   Se al termine del controllo, i comandanti appartengono a un unico giocatore, allora la partita è terminata e abbiamo un vincitore.