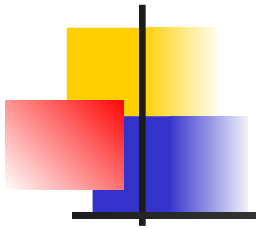


Problem-solving in linguaggio C: Gomoku

Capitolo 2: L'essenziale del linguaggio C

G. Cabodi, P. Camurati, P. Pasini, D. Patti, D. Vendraminetto

MAGGIOLI EDITORE APOGEO education 2016

Esempio: Gomoku
(esame informatica 20/1/2014)

Gioco tradizionale giapponese di allineamento.

- Due giocatori posano alternativamente pietre (bianche/nere) su una scacchiera 19x19.
- Vince il primo che riesce a mettere 5 pietre del suo colore in fila (riga, colonna o diagonale).

Realizzare programma C che permetta di

- riprendere una partita a gomoku precedentemente salvata (su file, nome argomento al main)
- portarla a termine, acquisendo e controllando le mosse dei due giocatori.

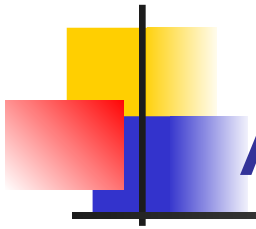
Esempio: Gomoku
(esame informatica 20/1/2014)

Le mosse sono acquisite

- ricevendo da tastiera le coordinate della casella su cui posare la propria pietra
- controllando se la mossa porta alla vittoria.
- Il programma deve inoltre verificare che la mossa sia corretta: coordinate fra 1 e 19 e casella libera.

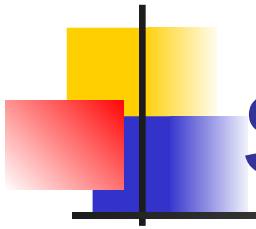
Il file che contiene la situazione del gioco è composto da

- 19 righe di 19 caratteri in cui
  - il punto “.” rappresenta una casella ancora libera
  - la “B” una pietra bianca, e la “N” una pietra nera.



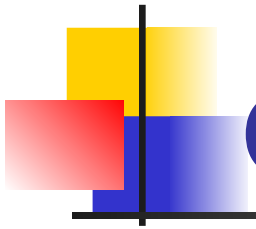
Analisi

- Problema di verifica:
  - Per ogni mossa, controllare che sia legale e controllare l'eventuale vittoria
    - quantificatore esistenziale: vittoria se esiste almeno una sequenza di 5 caselle del colore in riga, colonna, diagonale o anti-diagonale
- A questo problema principale si sovrappongono problemi minori
  - Input da file, Gestione di matrice con controlli su righe, colonne e diagonali, Modularità con funzioni



Soluzioni proposte

- Versione base: un main che fa tutto (molte parti ripetute). Problema principale: come si fanno a controllare righe, colonne e DIAGONALI !
- Versione migliorata, con funzioni per acquisizione mossa, controllo vittoria e stampa matrice (le funzioni evitano ripetizioni)
- Ulteriore versione, ancora piu' compatta (due varianti): evita 4 controlli distinti ed espliciti (riga, colonna, diagonale, anti-diagonale), sfruttando una opportuna struttura dati e un costrutto iterativo.
- Variante ulteriore, con soluzioni alternative e aggiunte non richieste dal testo.



Conclusione

- A partire da una soluzione iniziale, corretta (!), sono possibili molte migliorie.
  - In questo caso si tratta prevalentemente di programma più compatto, scritto meglio e più semplice da gestire/migliorare
  - In questo caso NON si affronta l'efficienza (o la complessità) della soluzione.
- L'obiettivo è illustrare più modi di affrontare un problema