Corso di Laboratorio di Programmazione
Progetto finale – Scacchiera elettronica

Sviluppate una scacchiera elettronica in C++, capace di gestire il gioco degli scacchi. Il progetto non richiede di saper giocare, ma solo la conoscenza delle mosse che possono essere effettuate dai vari pezzi. Chi non ha familiarità con questo gioco può trovare la descrizione ai link seguenti:

https://www.scuolascacchipordenone.eu/le-regole.html
https://www.scuolascacchipordenone.eu/le-mosse-speciali-.html
https://www.scuolascacchipordenone.eu/scacco-scaccomatto-patta-.html

La scacchiera deve permettere tutte e solo le mosse previste da tali regole, incluse quelle speciali (promozione, en passant, arrocco – che deve rispettare le relative quattro condizioni). Devono essere gestite e segnalate le situazioni di scacco, scaccomatto, patta.

Giocatori e interfaccia

Esistono due tipi di giocatori:

- giocatore umano;
- computer.

Il computer effettua mosse casuali su pezzi scelti a caso, mentre al giocatore umano deve essere fornita un'interfaccia con cui è possibile indicare le mosse da effettuare. Tale interfaccia, implementata da riga di comando, accetta le indicazioni di spostamento mediante due coppie di coordinate, che rappresentano partenza e arrivo. Per esempio, data la scacchiera nella sua configurazione iniziale (vedi Figura 1), per muovere il cavallo da B1 a C3 è sufficiente indicare:

B1 C3

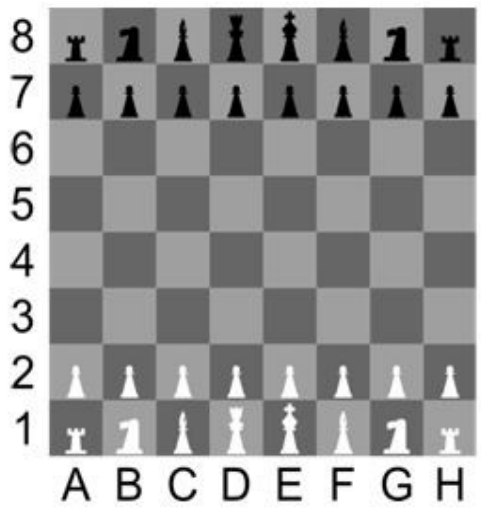

Figura 1: Scacchiera nella configurazione iniziale, con sistema di riferimento.

La scacchiera deve verificare che il comando sia dato nel formato corretto e che la mossa indicata sia lecita. Se non lo è, deve rifiutare il comando e chiederne un altro. L'interfaccia deve accettare le lettere sia minuscole che maiuscole (che hanno lo stesso valore).

Visualizzazione

Nel caso di partita con un giocatore, è possibile richiedere alla scacchiera la visualizzazione della disposizione dei pezzi usando l'interfaccia per l'inserimento delle mosse con un comando speciale:

XX XX

La visualizzazione avviene stampando un carattere per ciascuna posizione della scacchiera, più le coordinate riportate in Figura 1, separate da uno spazio dalla scacchiera. I caratteri sulla scacchiera hanno la seguente codifica:

Pezzo	**Carattere**
Re	R
Regina	D
Alfiere	A
Cavallo	C
Torre	T
Pedone	P

Se non è presente nessun pezzo, è stampato uno spazio. La regina è rappresentata con D (donna, nome alternativo usato negli scacchi) per evitare ambiguità. Usate le lettere maiuscole per i pezzi neri e quelle minuscole per i pezzi bianchi. La stampa della scacchiera nella sua configurazione iniziale sarà perciò:

```
8 TCADRACT
7 PPPPPPPP
6
5
4
3
2 pppppppp
1 tcadract

  ABCDEFGH
```

Partite

Le partite sono di due tipi:

- partite computer vs computer;
- partite giocatore vs computer.

La scacchiera deve gestirle entrambe. Al giocatore umano è assegnato il colore bianco o nero a caso. Nel caso di partite tra due computer, dovete fissare un numero massimo di mosse, oltre il quale la partita è considerata nulla.

Durante ogni partita deve essere effettuato un log su file, che elenca tutte le mosse effettuate, in ordine. La scacchiera deve essere in grado di effettuare il replay di una partita dato un log su file, che deve essere un file testuale con estensione .txt.

Replay

Il progetto deve essere corredato di un modulo (costituito da un eseguibile a parte, vedi prossimo punto) per il replay della partita, effettuato leggendo il relativo log. Il replay è realizzato stampando la configurazione della scacchiera per ogni mossa effettuata. La stampa avviene a video, con un'opportuna pausa (es., 1 secondo) tra una mossa e la successiva, oppure su file – in questo caso la scrittura avviene senza pause.

Eseguibili

Il progetto deve generare due eseguibili:

- l'eseguibile della scacchiera, chiamato "scacchiera", che deve gestire gli argomenti da riga di comando, secondo il seguente schema:
  - argomento pc: partita giocatore vs computer (p sta per player, c per computer);
  - argomento cc: partita computer vs computer;
- l'eseguibile per il replay, chiamato "replay", che accetta gli argomenti da riga di comando secondo il seguente schema:
  - argomento v [nome_file_log]: stampa a video il replay del file di log indicato;
  - argomento f [nome_file_log] [nome_file_output_replay]: scrive su file il replay del file di log indicato.

Note allo sviluppo

Il progetto sviluppato deve essere gestito tramite CMake e compilabile sulla macchina virtuale Taliercio.2020. Non è necessario che sviluppiate tutto sulla macchina virtuale, ma dovete fare verifiche periodiche per essere sicuri di non aver utilizzato codice fuori standard.

Indicazioni per lo svolgimento

Il progetto deve essere suddiviso in più file sorgente. **Ogni file deve essere scritto da un solo studente.** È tuttavia possibile controllare che il codice dei propri compagni di gruppo funzioni correttamente. Un errore in un file che inficia il funzionamento del progetto potrebbe causare una penalizzazione della valutazione dell'intero gruppo. **Il nome dell'autore deve essere indicato in un commento a inizio file.** Ovviamente, è necessario e positivo che si discuta all'interno di ciascun gruppo su come realizzare il codice, ma ogni studente è responsabile della gestione del codice che deve scrivere.

Saranno valutati:

- Chiarezza e correttezza del codice;
- Corretta gestione della memoria e delle strutture dati;
- Efficienza del codice e della soluzione trovata;
- Utilizzo di strumenti appropriati.

Il software deve essere basato unicamente sulla libreria standard del C++.

Consegna

Il compito deve essere consegnato su Moodle da un componente del gruppo, caricando un archivio che include una sola directory contenente:

- Il codice sorgente (eventualmente organizzato in sottodirectory);

- Il file CMakeLists.txt necessario alla compilazione (uno solo e posto nella directory principale del progetto);
- Un file di log di una partita computer vs computer e un file di log di una partita computer vs giocatore;
- Un file readme.txt in cui riportate eventuali note che volete comunicare in fase di correzione. Questo file non è la documentazione, che deve essere inserita sotto forma di commenti nel codice, ma un elenco di note aggiuntive per chi corregge, per esempio problemi riscontrati e non risolti.

L'archivio non deve contenere l'eseguibile, perché il sorgente sarà compilato in fase di correzione. Il sistema CMake deve compilare con le opzioni di ottimizzazione attivate (-O2).

Dopo la consegna su moodle, **verificate ciò che avete consegnato** con i passi seguenti:

- Scaricate il vostro progetto da moodle in una directory diversa da quella usata per sviluppare;
- Lanciare cmake;
- Compilare il codice e verificare la corretta esecuzione.

Suggerisco di consegnare anche compiti non completi. È tuttavia necessario che il software consegnato sia compilabile ed eseguibile.