Domino

Introduzione alla programmazione a.a. 2023/24

Federico Praticò, matricola 903107

Descrizione

Il progetto vuole riprodurre il gioco del domino classico utilizzando come linguaggio di programmazione il C nella versione c99.
Il programma viene eseguito da prompt di comandi.

Nel progetto è stata sviluppata la sola modalità interattiva che si svolge come segue:

- Al giocatore viene assegnato un numero di tessere variabile in base alla difficoltà selezionata (24, 60, 120); ogni tessera è composta da due valori numerici compresi in un intervallo da 1 a 6. Le possibili tessere sono le seguenti:

```
[1|1] [1|2] [1|3] [1|4] [1|5] [1|6]
[2|2] [2|3] [2|4] [2|5] [2|6]
[3|3] [3|4] [3|5] [3|6]
[4|4] [4|5] [4|6]
[5|5] [5|6]
[6|6];
```

- Durante la partita dovranno essere rispettate le regole del domino classico, ovvero sarà possibile inserire due tessere una a fianco all'altra a patto che i due lati risultanti adiacenti abbiano lo stesso valore numerico. Il punteggio viene calcolato sommando il valore delle singole tessere;
- Una volta che non vi siano più combinazioni possibili o le tessere siano state tutte inserite il gioco termina.

Struttura del progetto

Il progetto è strutturato nel seguente modo:

- domino.h: è il file d'intestazione contenete tutte le dichiarazioni delle funzioni utilizzate dal programma;
- domino.c: contiene l'implementazione delle funzioni dichiarate nel file d'intestazione;
- main.c: il file principale contente la funzione srand() e la funzione menu() all'interno della quale si sviluppa il programma.

Strutture dati

Il programma è stato implementato utilizzando delle liste concatenate semplici per la gestione della 'mano' del giocatore e del tavolo da gioco.
Ad ogni inserimento di tessera viene aggiunto un nodo alla lista implementata per il tavolo da gioco e conseguentemente viene cancellato il nodo corrispondente alla tessera selezionata dalla 'mano' del giocatore.
La singola tessera è stata implementata attraverso l'utilizzo di due struct:

- La struct più esterna denominata 'Card' identifica la tessera nella sua interezza e contiene una struct Value ed un puntatore alla struttura stessa;
- La struct 'Value' interna, contenuta nella struct Card, contiene i singoli valori della tessera stessa.

È stata utilizzata una struct ulteriore contenete le variabili utili a gestire le decisioni essenziali per lo sviluppo del gioco:

- swap: se asserita permette di ruotare la tessera;
- side: permette di selezionare il lato d'inserimento della tessera;
- confirm: se asserita conferma l'inserimento della tessera.

Si è deciso di raggruppare le tre variabili in una struct al fine di poterle gestire attraverso una funzione unica.

Makefile

```
CC = gcc

all: build/domino

build/domino: build/domino.o build/main.o
    $(CC) -o build/domino build/main.o build/domino.o

build/main.o: tools/main.c include/domino.h
    $(CC) -c tools/main.c -o build/main.o -I include

build/domino.o: src/domino.c include/domino.h
    $(CC) -c src/domino.c -o build/domino.o -I include

clean:
    rm -rf build/*.o build/domino
```

Per compilare il programma è stato creato un Makefile, permettendo così una compilazione più rapida e semplificata attraverso il comando 'make' grazie alla corretta gestione delle dipendenze tra i file che compongono il programma.

Il programma viene compilato in sistemi UNIX, quindi su Mac e Linux è sufficiente digitare il comando sopra citato.
Non possedendo sistemi Windows, non è stato possibile testare il codice e la compilazione su tale sistema. In ogni caso scaricando e installando WSL (Windows Subsystem for Linux) sarà possibile eseguire il programma.

Difficoltà riscontrate

- Durante la fase inziale di progettazione ho riscontrato difficoltà riguardo alla scelta delle corrette strutture dati da utilizzare per creare l'ossatura principale del programma; sono stato indeciso se utilizzare una lista doppiamente concatenata per gestire la 'mano' del giocatore, dal momento che pensavo avrebbe semplificato in termini di efficienza l'accesso e la conseguente eliminazione della tessera selezionata, tuttavia alla fine ho optato per una lista concatenata semplice perché la ricerca della posizione della tessera nella doppia lista concatenata sarebbe risultata nell'utilizzo di un'altra struttura di appoggio (es. array) per la mappatura delle posizioni;
- Successivamente alcune difficoltà sono state riscontrate a livello logico riguardo alla gestione di alcuni casi limite come il controllo delle possibili combinazioni durante la prima mossa, quando il tavolo da gioco ancora non contiene alcuna tessera e ciò portava alla generazione di segmentation fault.

In generale essendo ancora un principiante inesperto ho riscontrato difficoltà nell'ottimizzazione e nella scrittura di un codice leggibile ed ottimizzato.

Strumenti utilizzati

Come editor di testo principale ho utilizzato Sublime text, inoltre mi sono avvalso delle estensioni Doxygen di Visual Studio Code per la creazione dei commenti utili alla documentazione.
I test del programma sono stati effettuati su terminale.